UNIVERSITÀ DEGLI STUDI DI MILANO
FACOLTÀ DI SCIENZE E TECNOLOGIE

DIPARTIMENTO DI INFORMATICA
"GIOVANNI DEGLI ANTONI"

CORSO DI DOTTORATO IN INFORMATICA
XXXVI CICLO, INF/01

TESI DI DOTTORATO DI RICERCA

PREDICTION AND CAUSAL ANALYSIS OF CHURN IN THE TELECOMMUNICATION DOMAIN

Author: Dott. Annalisa Barsotti

Supervisor:	Prof. Gabriele Gianini
Cosupervisor:	Prof. Ernesto Damiani
External Supervisor:	Dott.ssa Loredana Liverani
PhD Prog. Coordinator:	Prof. Roberto Sassi

A.A. 2022/2023

Abstract

In the Telecom context, the problem of Customer Churn Prediction (CCP) (or customer defection prediction) can be addressed by using not only well-established domain expert knowledge, but also by exploiting the potential wealth represented by customer-related data and applying Machine Learning Techniques. In this work, we used a TIM S.p.A. real-world dataset to train models that could predict which customers can defect. We compared the outcomes of a considerable number of classification algorithms on our real-world dataset. Furthermore, we applied the causal reasoning on the real dataset identifying actionable features by indirect confirmation of the domain experts. Another need of the telecom stakeholders is to understand the reasons why a customer might defect. This calls for the use of Causal Analysis: thanks to Causal Analysis – and more specifically to Causal Calculus – one can try to extract the direct and indirect causes from observational data. For such an analysis to be possible, one needs beforehand to count on the availability of a Structural Causal Model, or to extract causal graph and other information from the data. Causal discovery can be performed using a number of algorithms, based on different principles, which, when reconstructing sufficiently large graphs, can produce discordant outcomes.

In this work, we contribute a causal discovery method that takes advantage of topology-related information generated by an ensemble of causal discovery algorithms to identify which topology is closest to the ground truth.

The method relies on the results of the discovery algorithms, considered as a committee of experts, and on a supervised learning approach consisting of stacking of a multi-label classifier on the outcomes of the ensemble.

Ideally, this discovery method could also be used on Telecom data; however, since for our data the ground truth (GT) was not available, we limited ourselves to validating the method on synthetic data, generated by a specialized library. Applying our method to the synthetic data we found that we could considerably improve the accuracy of the discovery with respect to the use of individual discovery algorithms.

Contents

List of Figures

Chapter 1.

Introduction

In this study, we utilized a real-world dataset from TIM S.p.A. to anticipate customer defection, employing various machine learning algorithms. We conducted a thorough comparison of numerous classification algorithms using our dataset. While predicting churn is significant, companies are often more concerned with understanding why customers might defect and the impact of marketing strategies, such as discounts, on the likelihood of churn. In our research, we introduce a method for causal discovery that leverages topology-related insights derived from a collection of causal discovery algorithms to identify the topology closest to ground truth. This method relies on the outcomes of the discovery algorithms, treated as a committee of experts, and utilizes a supervised learning approach involving the stacking of a multi-label classifier on the ensemble results. Additionally, it involves the identification of a centroid graph. Due to the absence of ground truth (GT) for our real dataset, we validated the method using synthetic data generated by a specialized library. The causal discovery algorithms considered include GES, GOLEM, LINGAM, Notears, and PC. We framed the problem as a multi-label classification task, with each algorithm representing a label, including the CENTROID graph, resulting in six labels in total. We employed two common approaches for multi-label classification: Binary Relevance (BR) and Classifier Chain (CC). After selecting

a representative label from the predicted set based on classifier precision, we evaluated its distance from the ground truth (defined as the Hamming distance between adjacency matrices, i.e., in terms of edges). Our findings indicate a significant improvement compared to individual algorithms, particularly CENTROID. The mean distance of the algorithm selected by our method using BR and using CC are both closely aligned with the actual average minimum distance of the best-performing algorithm in the set (which is initially unknown), improving the results obtained by the CENTROID. Comparable observations apply when normalizing distances by the actual number of edges. Furthermore, we applied the causal reasoning on the real dataset identifying actionable features by indirect confirmation of the domain experts.

1.1. Motivation

In recent years, the introduction of smartphones has changed the dynamics of the telecommunications market. Based on the fact that the customer is a fundamental source of income for the company, it is vital to secure a wide customer base; to do this, companies invest a huge amount of resources into both marketing and retention programs. Especially for the retention programs it is important to focus on the customers which are likely to quit the contract. *Customer Churn*, is the term normally used to refer to customer defection from a service company. In the literature, it is also known as *customer attrition*, or *customer turnover*. The phenomenon, physiological for any service business, is of great relevance for companies. Its quantification enters into one of the typical Key Performance Indicators (KPI), the Churn Rate (CR). CR is the percentage of customers that stopped using a company's product or service during a certain period.

Customer churn has a substantial impact on all aspects of the business of a company. Losing customers entails opportunity costs; not only do company sales decrease; but it is also not easy to compensate by attracting new customers: the acquisition of a new customer is usually 5 to 6 times, but even up to 20 times [114] more expensive than keeping an existing one [119].

For this reason, companies adopt specific strategies for customer retention, and among them those that aim at *predicting* which customers are at risk of defection are of great interest.

It is common sense that a "churner" is a customer who changes from one service provider to another due to the competitive market and the large offer of service/product availability [9]. This is why *Customer Churn Prediction* (CCP) is a challenge in general and even a harder challenge in the TelCo domain: the TelCo market in the last few decades has behaved as a rapidly evolving and increasingly competitive ecosystem, where the diffusion of smartphones in the last decade has made this evolution more hectic and erratic.

In this context, the problem of the CCP cannot be addressed only by using well-established domain expert knowledge, but also by exploiting the potential wealth represented by customer-related data.

The abundance, fine-grained character, and diversity of those data, with the consequent difficulty for humans to detect characteristic patterns, call for the use of methods that go beyond simple Data Analytics and bring the challenge into the arena of Machine Learning.

Machine learning is the branch of Artificial Intelligence that uses inductive methods to learn models from the data: e.g., a predictive churn model could

be learned, in principle, by looking at a large number of historical data of customers ending in a defection or not.

Depending on the available data and the precise definition of the problem, many techniques can be applied for this purpose.

Moreover, considering the additional business requirements, many different challenges would arise. In addition to the obvious requirement of prediction accuracy, the requirement of trustworthiness and the interpretability of the learned model are important.

1.2. Research Questions

The aim of customer churn prediction naturally raises a number of research questions. Among them are the following.

Research Question 1 (RQ1) What algorithms and features can be used to optimize Customer Churn Prediction in the telecommunication sector?

Research Question 2 (RQ2) Is it possible to explain the reasons behind Customer Churning based on observational data of the telecommunication sector?

To answer the *RQs* the present thesis aims to investigate, in the first part, the prediction of customer churn by applying various Machine Learning algorithms and inspecting their performance. This part will create the baseline for answering *RQ*2, in which a **new methodology** is provided by developing a technique for ***Ensemble Discovery of Causal Graphs***. Causal graphs are Direct Acyclic Graphs (DAGs) whose nodes represent variables and whose directed links between two nodes signify that the first

variable causes the second directly. The DAG in our case contains a target variable, which reports about customers leaving or not leaving the company. Causal analysis focused on the target will be used.

1.3. Problem formalization

Companies usually have very large customer data sets, which can include millions of records. Based on the customer's information they consist of huge number of heterogeneous features, such as information regarding: the phone line (type, seniority, orders, offers), services and products associated with the line, phone traffic, and many more (see Chapter 3).

The aim of this thesis is to exploit a large telecom dataset to predict churners by applying different **Machine Learning** (ML) techniques and to perform an explanatory analysis based on **Causal Reasoning**. The **reasons why** a customer can abandon is of great interest to the company itself, because they can guide the retention strategies and make them more effective and focused.

1.4. Background: Telecommunication Business Context

In the following we will introduce the necessary background knowledge on the business and technical context of this thesis.

The telecommunication business context is rapidly changing due to the highly competitive market scenario. Customers can be overwhelmed by a wide variety of offers and can easily switch from one company to another. The customer churn has a substantial impact on all the aspects of busi-

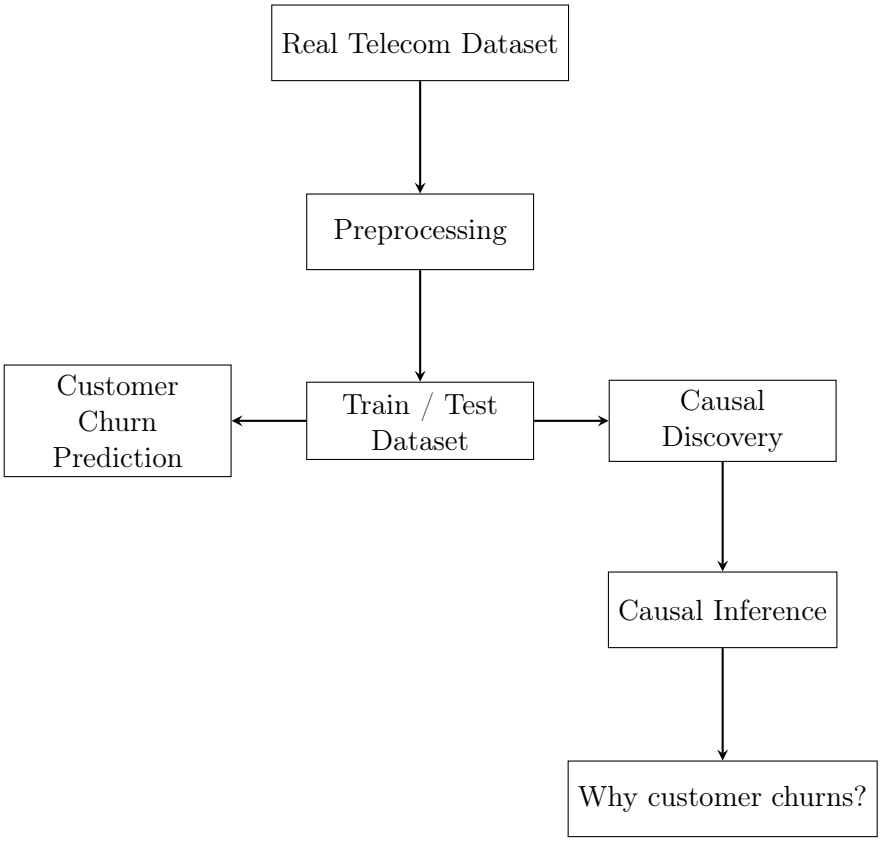


Figure 1.1.: The general pipeline.

ness of a company – e.g., telecommunication services (i.e. Offering a range of communication services), customer Acquisition and Retention (i.e. marketing and sales efforts), Customer Support (i.e. to address issues, resolve complaints, and assist with technical problems, etc.). The customers are the company's fuel, so it is important to investigate some business questions. We discussed with TIM S.p.A. **domain expert** the relevant business questions to focus on. The main ones are the following:

1.4.1. Business Question 1 (BQ1)

Given a customer, what is the likelihood that (s)he leaves the company? (**who** might leave?)

Motivation

We are trying to predict the total number of effective churners. This question is very important from the company's technicians' perspective. The more accurate the prediction, the more they can help the business lines make targeted proposals.

1.4.2. Business Question 2 (BQ2)

How can we prevent customer churn?
(**why** would the customer prefer to leave?)

1.4.3. Business Question 3 (BQ3)

What is the total amount of expenses of customers who leave from the company's perspective?

Motivation

Based on the company's business perspective one has to focus to achieve more targeted campaigns. The ability to predict that a customer may leave can help to estimate the related money losses, and may be also to save customers and increasing future profits. However, for TIM S.p.A., this point is actually less relevant than the other two BQs, mainly because it depends both on the performance of the existing models that they are using

and on the effectiveness of the actual campaigns which they have built on them.

These business questions will be considered throughout this thesis as a motivation and boundary for the technical solution to answer *RQs*. However the main focus will be on question RQ1.

1.5. Structure of the thesis

The remaining structure of the thesis is as follows. Chapter 2 describes the literature on Machine Learning methods for Customer Churn Prediction (CCP) within the scope of the Telecommunication (TelCo) domain. Chapter 3 illustrates both the experimental setup based on a **real** telecommunication dataset and shows the Predictive Analysis. Chapter 4 presents the main definitions and concepts which define the **causal reasoning**'s main tasks, in particular, the **causal discovery** and the **causal inference**. Chapter 5 proposes a new **methodology** by applying Ensemble Discovery Methods to telecommunication data. Chapter 6 describes the results. Chapter 7 illustrates the application of causal inference using the real data set. Chapter 8 provides the summary of the central contributions and discoveries provided in the thesis, including a perspective on future research directions.

In the context of this PhD thesis, the paper: *A Decade of Churn Prediction Techniques in the TelCo Domain: A survey* has to appear in Springer Nature Computer Science (SNC), and influenced Chapter 2.

Chapter 2.

Related Work

The challenge of the Customer Churn Prediction (CCP) cannot be addressed only by using well-established domain expert knowledge: there is the opportunity to exploit the potential wealth represented by customer-related data. The abundance, fine-grained character, and diversity of those data, with the consequent difficulty for humans to detect characteristic patterns, calls for the use of methods that go beyond simple data analysis and bring the challenge into the arena of Machine Learning.

Predicting churn is rarely the end goal from the perspective of the business, which is more interested in assessing the impact of a marketing action (e.g. discounts, extra-GB, etc.) on a customer's churn risk [40]. For example, knowing that long-term customers are less likely to churn than new customers is not an actionable insight because the company cannot intervene. Instead, what the company wants to know is how a treatment (or treatments) affects churn, also known as uplift [49]. In uplift modeling, only customers who react positively to the campaign are considered [32]. The aim of this chapter is to provide a survey of the literature on Machine Learning methods for CCP within the scope of the TelCo domain and in the approximate period of the last decade. In particular, it pertains to the issue of prediction. If such information is available, research papers may

also attempt to tackle other associated inquiries. The following research works described in the Section 2.2 constitute the building block for the uplift models.

2.1. Systematic Literature Review Methodology

In this Section we aim to highlight the Research Scope, furthermore knowing that the introduction and affirmation of smartphones have changed the dynamics in recent years, and therefore considering this decade is not only motivated but complementary with other surveys [69], [40], [42]. We are looking for areas not covered by current techniques to verify their effectiveness. Based on the awareness that the customer is a fundamental source of income for the company. The following questions arise: (1) What are the most effective ML algorithms in Customer Churn Prediction (according to a variety of metrics)? (2) Considering a predictive algorithm, given a customer at risk, what factors make him/her more likely to churn? (3) Is it possible to quantify the effect of possible interventions on those variables that are actionable from the point of view of the company?

The articles we considered addressed the above research questions. Other obvious questions would refer to the possible interventions by the TelCo company: after predicting the likely defection of a part of the customers, a natural step would be to adopt countermeasures aimed at their retention; typically, those measures consist of suitable offers.

Customer retention is normally addressed using Machine Learning based recommender system algorithms. However, the problem of customer retention and the use of recommender system algorithms falls outside of the scope of this work which focuses on the prediction of customer churn.

2.1.1. Contributions

The main contributions of this chapter are the following: (1) Review of Machine Learning techniques applied to the Churn prediction problem, focusing on the techniques and on the specification of data they use. (2) The ML models are presented and analyzed, describing the capabilities of the model, the list of assumptions involved in each model, and other relevant aspects. (3) Future research directions for the field are outlined and discussed. The structure of this work is illustrated in the next subsection.

2.1.2. Schema

The elements that are reported about each paper are Algorithms, Datasets, Methods, Metrics, and Results, see Figure 2.1:

- **Algorithms and Methods** – The main focus of the survey of each paper. The relation between techniques and articles is shown in the summary in Table 2.2.

- **Datasets** – Real data relating to TelCo customers are often incomplete, noisy, and unstructured. For each paper, is reported synthetically on the data used. A longer description of the data, when available, is provided in the Supplementary Material.

- **Metrics** – Several performance metrics were proposed in the literature. Some works do not consider only the usual metrics, such as precision and recall, but also cost-related metrics.

- **Results** – The results presented in each research article are discussed and compared.

A particular set of techniques both to foster customer loyalty and reten-

tion is that of recommender systems that aim to tailor the service offers according to a specific customer profile; this area of research falls outside the scope of this work.

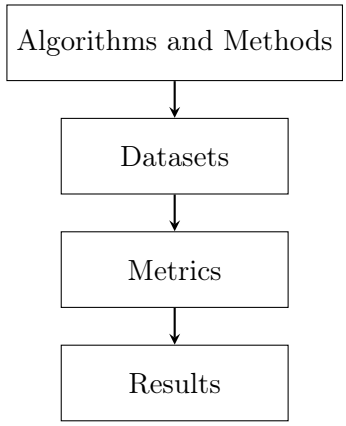


Figure 2.1.: The General Schema.

The criteria used to gather material for this chapter are reported in the following sections.

2.1.3. Inclusion Criteria

The selected papers meet the following inclusion criteria: (1) the selection of the first quartile and the second quartile journals using Scopus; (2) the classes' conferences are A++, A+, A and B; (3) studies that analyze any customer churn prediction aspect related to the TelCo domain; (4) literature review papers, which summarize the results of the previous studies; (5) papers that describe machine learning techniques; (6) English written studies; (7) studies published as Journal papers, conference proceedings, Workshop proceedings; (8) studies published in computer science venues; (9) peer-reviewed studies; (10) studies with one or two citations;

2.1.4. Exclusion Criteria

The papers discarded in our survey meet the following exclusion criteria:

(1) book chapters, work-in-progress papers, posters, Master theses, P.h.D. theses (to avoid the double count of the ones published as papers); (2) short versions of long version papers; (3) discussion papers;

2.1.5. Research Selection Process

I searched using "Google Scholar" as it offers a number of options to combine multiple search terms with Boolean operators. The basic search string used was: "customer" AND "churn" AND "prediction" AND ("telco" OR "telecom"); the search is restricted to the last 15 years. The final number of papers considered is 28.

2.1.6. Acronyms

In the following Table 2.1 is the list of the used acronyms.

Table 2.1.: The acronyms that are used in the survey.

AUC	Area Under ROC Curve
ANN	Artificial Neural Network
BN	Bayesian Network
CART	Classification And Regression Tree
CCP	Customer Churn Prediction
DL	Deep Learning
DT	Decision Tree
GBM	Gradient Boosted Machine
GP	Genetic Programming
KNN	K-Nearest Neighbors

LDA	Linear Discriminant Analysis
LR	Linear Regression
LoR	Logistic Regression
MCC	Matthews Correlation Coefficient
NB	Naïve Bayes
NN	Neural Network
NPS	Net Promoter Score
PCA	Principal Component Analysis
PCALB	Principal Component Analysis Load Balancing
POL	Polynomial Kernel
PSM	Propensity Score Matching
PSO	Particle Swarm Optimization
RA	Regression Analysis
RBF	Radial Basis Function
RF	Random Forest
SIG	Sigmoid Kernel
SRI	Single Rule Induction
SMO	Sequential Minimal Optimization
SNA	Social Network Analysis
SOM	Self-Organizing Maps
SVM	Support Vector Machine
TelCo	Telecommunication

2.2. Relevant Research Papers

Due to their increasing success, Machine Learning techniques are widely used in almost every domain such as telecommunication, finance, banking, marketing, and user behavior analytics. ML techniques are also useful and can help with the Customer Churn Prediction (CCP) problem in the TelCo domain [8]. In particular, the recognition of the customers that are going to

churn is essential for a company and can help in applying effective retention strategies that keep the customers for a long period of time.

2.2.1. Articles description

Several studies present the challenge of predicting churn as a binary classification task, distinguishing between churn and non-churn instances. Commonly employed conventional classification techniques include Decision Trees (DT), Random Forest (RF), Classification And Regression Tree (CART), Logistic Regression (LoR), Naïve Bayesian classifiers (NB), K-Nearest Neighbor clustering (KNN), and Support Vector Machines (SVM). Additionally, the application of advanced methods such as Artificial Neural Networks (ANN), Self-Organizing Maps (SOM), and Evolutionary Algorithms (EA) is also prevalent. A common issue often encountered is class imbalance, where the churn class is typically underrepresented in the available data due to obvious reasons. Numerous articles emphasize the distinct impacts of type I errors (false positives) and type II errors (false negatives) on business outcomes. The papers are sorted based on which kind of algorithms performs the best, to show the changes in the literature scenario, the related trends and the different approaches adopted. The Table 2.2 groups each paper by the algorithms used. The Table 2.3 represents the information related to telephony. In particular, the Fixed, Mobile, Mixed (i.e. Fixed and Mobile), Unspecified (i.e. if Fixed or Mobile are not indicated) and Other Fields (i.e. if the analysis are performed in other domains) according to the corresponding data set that is used in each paper.

Table 2.2.: Comparison table of algorithms including the surveys.

Algorithm	**Papers**
ANN	[113, 25, 69, 95, 106, 51, 114, 10, 6, 67, 15, 46, 29, 42, 5, 7]

Algorithm	Papers
AdaBoost	[59, 73]
BN	[71, 25, 59, 123]
CART	[82]
CatBoost	[73]
Cluster Analysis	[51]
DTs	[113, 71, 95, 114, 28, 2, 10, 106, 70, 51, 114, 2, 123, 15, 46, 73, 42, 79]
Extra Tree	[73]
GBoost Machine Tree	[2, 7]
Genetic Programming	[59]
K-Nearest Neighbor	[6, 123, 67, 15, 46, 42, 82, 5, 7]
K-means Clustering	[95, 106]
LDA	[82]
Literature Review	[51, 40, 69, 42]
LR	[95, 46, 5]
LoR	[95, 114, 28, 11, 51, 114, 11, 101, 15, 63, 73, 129, 29, 42, 79, 82, 128, 33]
Logit Boost	[5]
NB	[5, 71, 114, 59, 6, 15, 46, 73, 29, 42, 7]
PCALB	[5]
PSM	[79]
PSO	[123]
RF	[59, 2, 10, 11, 63, 73, 29, 42, 79, 33]
Regression Analysis	[114]
RotBoost	[59]
Rotation Forest	[59, 46]
Self Organizing Maps	[111]
SRI	[7]
Support Vector Machines	[111, 22, 114, 25, 10, 11, 6, 46, 63, 73, 29, 42, 5]
XGBoost	[2, 63, 73, 42, 5]

Algorithm	Papers

Table 2.3.: Information of Fixed, Mobile and Mixed (Fixed and Mobile) telephony according to the corresponding data set that is used in each paper.

Telephony	Author
Fixed (F)	[120, 71]
Mobile (Mo)	[79, 63, 1, 25, 28, 95]
Mixed (Mi)	[12, 128, 46, 58]
Unspecified (if F or Mo)	[73, 29, 78, 67, 112, 106, 9, 114, 22, 113, 5, 7]
Other Fields	[44, 15, 122, 98, 61, 101]

2.2.2. Existing Surveys

The paper by **Geiler** et al. [42] is a **survey** that provides general and practical recommendations on a churn prediction pipeline based on ensemble learning. Not only they review the recent literature, but they also evaluate and compare multiple alternatives within the machine learning churn analysis pipeline. The authors employ publicly accessible datasets pertaining to churn across various domains (Telecommunication, Music Streaming, Human Resources, Newspaper). Furthermore, by framing the churn problem as anomaly detection problem they challenge with different datasets: Bank Marketing, a Credit Card Fraud Detection and a thyroid disease, which they categorize as *churn-like*. They include semi-supervised techniques (iForest and DevNet) and supervised algorithms (K-Nearest Neighbors (k-NN), Gaussian Naïve Bayes (GNB), Logistic Regression (LR), Support Vector Machine with Radial Basis Function kernel (SVM-rbf) and without kernel

(SVM), Decision Tree (DT), Random Forest (RF), XGBoost, a feedforward neural network (NN) and Generalized Extreme Value Neural Network (GEV-NN)) in association with different undersampling, oversampling, and hybrid sampling strategies, to compensate for the class unbalance when the issue is not already addressed by the algorithm (GEV-NN, Forest, and DevNet are specifically designed for imbalance binary classification or anomaly detection). The authors focus on the association between base machine learning techniques, sampling strategies, and datasets. Lastly they focus on an ensemble method based on the four main outperforming approaches, LR, XGBoost, RF, and ANN. Based on the algorithms' performance results, they recommend using the ensemble LR, XGBoost, and RF with no sampling to analyze novel churn-like datasets. The work focuses on predicting churn and does not investigate the **causes of defection** nor discuss actionable features or interventions.

The paper of **Hashmi** et al. [51] aims to **survey** the pros and cons of renowned data mining techniques used to build predictive customer churn models in telecommunication. They highlight the use of customer call details, demographic, complaints, billing information, and contractual data for the mobile domain. Fixed-line service providers have customers' call details and billing information. Classification techniques such as Decision Trees, Neural Networks, Logistic Regression, and Cluster Analysis are mostly used to analyze continuous and qualitative data. Scientists face several challenges, such as missing or incomplete datasets, data confidentiality, and dataset size (i.e., very large datasets containing noisy or imbalanced data which affect the reliability of the predictive model). This survey **aims** to provide a **roadmap** for researchers to accumulate knowledge on data mining techniques in the TelCo domain.

The **survey** proposed by **García** et al. [40] considers the customer churn issue by taking into account the correct application of information-based

knowledge extraction in the form of business analytics. They highlight as a competitive advantage both the importance of anticipating the customer's intention to abandon the provider and the launch of retention-focused actions. For this reason the anticipation can be the result of the correct application of information-based knowledge extraction in the form of business analytics.

A great assistance for the churn management could consist in the adoption of **Intelligent Data Analysis** (IDA), (i.e. pattern recognition (PR), machine learning (ML), statistics and related approaches) or data mining (DM), for the analysis of market-surveyed information. The work aims to provide both business analysts and data scientists, with a thorough survey and review of churn analysis applications of IDA techniques.

The described DM methodology based on IDA may become a useful knowledge and information management tool for knowledge extraction from domain data. The paper has surveyed and reviewed recent literature in which the use of IDA to build predictive models has been proposed to address the churn problem, with a (non-exclusive) focus on the use of CI techniques.

A detailed survey of recent applications of business analytics to churn, with a focus on computational intelligence methods, is provided. Furthermore, authors contribute by giving an in-depth discussion of churn in the context of customer continuity management. They discuss predictive methods used: for standard techniques (Regression Analysis, Decision Trees, etc.); computational intelligence (CI) methods (Artificial neural networks, Support vector machines, Data mining by evolutionary learning and Bayesian networks (DMEL)); and for alternative ones (Semi-Markov processes, Mixture transition distribution and Goal-oriented sequential pattern). And fields of application: telecommunications, banking, and other areas of application. They also provide a discussion on the main issues of: identifying

and obtaining the best data (i.e. the relevance of adequate data gathering and attribute selection procedures); considering that the churn should be treated as a dynamic process that naturally evolves over time; developing a predictive model and validating the results. Finally they express in favour of the use of several methods for performance comparison purposes or as alternative **model combination** methods such as those of the **Ensemble Learning** family.

The study proposed by **Kamalraj** et al. [69] aims to identify the different algorithms used in the churn prediction scenario, conduct a **literature review** and focus on the implementation and understanding of existing models. The management department suggestion is to use identified churns and invest in appropriate efforts considering the customer's lifetime value in combination with the prediction of churn. Thus, to reduce the cost of an excessive retention effort in both customers who are not leaving but are considered churners and in customers who are not considered churners, but are real churners.

2.2.3. Decision Trees

In the research of **Umayaparvathi** et al. [113] data mining techniques (DT and ANN) are investigated and decision trees are observed to outperform the Neural Network model. The dataset is acquired from a PAKDD – 2006 data mining competition and aggregated for 6 months duration. A total of 24.000 customers is used, and 252 attributes (i.e. customer demography, bill and payment, call detail record, customer service, etc.). Authors use Information gain and Entropy of the attributes as there are no statistical methods applied to the selection of feature sets. Data is aggregated over six months: Customer behavior during the previous 6 months is used to predict churners during the next month (7th month).

Confusion matrix, accuracy, and error rate are calculated: for the DT accuracy is 98.88% and error rate of 1.11167% false positive of 0.93% and false negative of 2.23%. For the ANN model the predicted precision is 98.43%, the false positive is 1.26%, and the false negative is 3.40%. And the error rate is 1.5616%

The paper of **Qureshi** et al. [94] presents data mining techniques to identify customers who are going to churn. The proposed algorithms are LR, LoR, ANN, KNN, DT, CART, Chi-squared, Automatic Interaction Detector (CHAID), Exhaustive CHAID and Quick, Unbiased, and Efficient Statistical Tree (QUEST). The dataset is provided by a TelCo operator with approximately 106000 customers (active and disconnected). Traffic type (outgoing, incoming, voice, SMS (Short Message Service), data), traffic destination (on-net, competition), rate plan, loyalty, traffic behavior, etc. In particular, they have 100.264 active users (94.1%) and 6231 churners (5.9%). The dataset (106000 customers along with their usage behavior for 3 months) is divided into two subdatasets: "churn dataset1" with the traffic figures for 3 months (approximately 300000 records) and "churn dataset2" with the profile variables for each customer (rate plan, contract renewal date, status, deactivation date, value segment, etc.). `Customer_ID` is the key variable for the two subdatasets.

The customers are classified by status (active or churn). A customer is "Active" if he/she continues to use the network; "Churner" in case the contract with the network is terminated. Authors classify churners and active customers to solve the problem of class imbalance by randomly oversampling the minority class (by keeping the churners in an active ratio to 40:60 approximately). The best results are obtained with the Exhaustive CHAID algorithm (accuracy about 70%).

The following methods of evaluation are considered: precision, recall, and

F1-score. The p-value is calculated with respect to the target variable. The Spearman correlation is computed to identify variables that are closely correlated with the status of the customer (the main variables 5: Credit Score; No. of penalties for non-payment; No. of outgoing calls to rival networks; No. of Incoming SMS from rival networks; No. of days of outgoing activity).

The results show that the Exhaustive CHAID algorithm is the most accurate algorithm (about 70%), and to increase accuracy, they introduce new variables.

Recall of results based on the variables derived for active users is 76.9%, and for users it is 8.5%, which increases from the best result earlier to 68.5%. The recall in the test set for active customers is 76.3% and for churners is 60.5%. The general accuracy is 75.4%.

In the study of **Kirui** et al. [71] the development of precise and reliable predictive models is fundamental. The paper proposes a careful selection of features to improve the detection of possible churners. They considered the phone traffic figure and customer profile data and extracted a set of new features: contract-related, call pattern description, and call pattern changes description. The results of the Naïve Bayes and Bayesian networks are compared to the C4.5 decision tree. The results show improved prediction rates for all models, higher true positive rates for the probabilistic classifiers, and better overall accuracy for the decision tree. They remark to consider that the minority class of the churners skews the dataset, so higher true positive rates and lower false positive rates are better than the general precision.

The dataset used was obtained from a European TelCo company and collected in a period of three months from August to October 1997. The dataset consists of 112 attributes and 106.405 instances of which 5.6%

churners and remaining active subscribers. Additional features are added to the original dataset (consisting of two subsets: call traffic figures and customer profiles).

A complete feature set is obtained for every customer based on the customer id.

They used the C4.5 decision tree, Naïve Bayes, and Bayesian Networks to test the features and perform two experiments: In the first experiment, a modified sampled dataset is used to compare the performance of the new feature subsets. Features based on their call types: calls to competition (CMP), fixed line calls (FIXED), international calls (INTER), on-net calls (ONNET), and value-added service calls (VAS). They tested the new proposed features altogether (NEW), either the features that describe user activity (ACTIVITY + `NUM_EVENTS`) or customer profiles (CP). The information gain attribute selection technique is used to select the first 60 attributes with the highest information gain for both the original and modified datasets.

ROC curves are used, and the NEW subset (representing the proposed features) plays the most significant role. Information gain measure (select the top 60 attributes with the highest information gain). The study of **Dahiya** et al. [28] presents a new framework for the churn prediction model using WEKA (Waikato Environment for Knowledge Analysis) data mining software. A discussion of various prediction models, a comparison of quality measures such as regression analysis, and decision trees is proposed. They demonstrate that the $J-48$ decision tree technique is more efficient with respect to logistic regression analysis due to higher accuracy. They aim to use hybrid classification techniques to highlight a possible association between customer lifetime value and churn prediction. The dataset for this study is acquired from the KDD Cup 2009. It is used to analyze the

marketing tendency of customers from the large databases of the French TelCo company Orange with a total of 18000 attributes. Decision Tree and Logistic Regression are used.

The test records counts predicted correctly and incorrectly are evaluated on the performance of a classification model.

The authors use three small, medium, and large datasets with varying attributes: 10 numeric variables and 50 instances for the small dataset. There are 50 numeric variables and 200 instances for the medium dataset. For the large dataset, 100 variables (numeric variables and categorical variables) and 608 instances. The 10-fold cross-validation is used.

The paper publishes the confusion matrix and the following metrics: accuracy (number of true outcomes/total number of predictions) and error rate (Number of False Outcomes/Total Number of Predictions); Precision; Recall; F1-score; ROC Area. The J48 tree and a logistic regression classifier are applied for all dataset analysis. For the small dataset, obtain: *J*48**-tree**: Correctly classified Instances (%): 43; 94 Incorrectly classified Instances (%): 7; 6 Kappa statistic: 0.8598 Mean absolute error: 0.0846 Root mean square error: 0.2435 Relative absolute error: 19.8912% Root relative square error: 52.845% Total number of instances: 50 Accuracy (detailed by class) Precision: 0.875; 0.971 Recall: 0.933; 0.943 F1-score: 0.903; 0.957 ROC area: 0.934; 0.932

Logistic regression classifier: Correctly classified instances (Incorrectly classified instances (Kappa statistic: 0.6847; Mean absolute error: 0.2284; Root mean square error: 0.3591; Relative absolute error: 53.6768%; Root relative square error: 77.9236%; Total number of instances: 50; Precision: 0.722; 0.938; Recall: 0.867; 0.857; F1-score: 0.788; 0.896; ROC area: 0.848; 0.848.

For the medium dataset: **J48 tree**: Correctly classified instances (Incorrectly classified instances (Kappa statistic: 0.9537; Mean absolute error: 0.0237; Root mean square error: 0.1225; Relative absolute error: 5.4733%; Root relative square error: 26.3704%; Total number of instances: 200; Precision: 0.968; 0.985; Recall: 0.968; 0.985; F1-score: 0.968; 0.985; ROC area: 0.998; 0.998;

Logistic regression classifier: Correctly classified instances: 186; 93 Incorrectly classified instances (Kappa statistic: 0.8419 Mean absolute error: 0.124 Root mean square error: 0.253 Relative absolute error: 28.6819% Root relative square error: 54.4542% Total number of instances: 200 Precision: 0.855; 0.969 Recall: 0.937; 0.927 F1-score: 0.894; 0.948 ROC area: 0.94; 0.94

For the Large dataset: **J**48 **tree**: Correctly classified Instances (%): 606; 99.6711% Incorrectly classified instances (Kappa statistic: 0.9926 Mean absolute error: 0.0057 Root mean square error: 0.0586 Relative absolute error: 1.2838% Root relative square error: 12.4278% Total number of instances: 608 Precision: 0.99; 1 Recall: 1; 0.995 F1-score: 0.995; 0.998 ROC area: 0.999; 0.999

Logistic regression classifier: Correctly classified instances (Incorrectly classified instances (Kappa statistic: 0.9334 Mean absolute error: 0.0477 Root mean square error: 0.1475 Relative absolute error: 10.7272% Root relative square error: 31.2657% Total number of instances: 608 Precision: 0.956; 0.978 Recall: 0.956; 0.978 F1-score: 0.956; 0.978 ROC area: 0.978; 0.978.

In conclusion, results show that the accuracy achieved with a decision tree is much higher than the logistic regression, so a decision tree is considered to be an efficient technique.

The paper of **Ahmad** et al. [1] develops a churn prediction model of customers who are likely to quit their subscription to Syriatel and MTN.

They use Decision Tree, Random Forest, GBM and XGBOOST algorithms (on spark 2.3 framework and integrated it with ML library) on the big data platform and create a new way of feature engineering and selection.

The aim of this study is to find both the factors that increase customer abandonment and the adoption of the necessary actions to reduce it.

The dataset includes all information from prepaid customers collected over 9 months, 78 categorical features and 32 categories are considered (the first 31 most frequent and the remaining categories are replaced with a new category). Customers considered churners are defined as being in idle phase after 2 months of the investigation period.

Non-churned customers are labeled as active customers (customers acquired in the last 4 months are excluded).

The total amount of the sample is 5 million customers: 300.000 churned customers and 4.700.000 active customers.

The volume of the dataset is about 70 terabyte in the HDFS "Hadoop distributed file system" (data formats: structured, semi-structured and unstructured).

Having a large dataset is a positive aspect, on the other hand, big data poses new challenges: *volume*: 70 Terabyte; *variety*: structured, semistructured (XML-JSON) or unstructured (CSV-Text); *class unbalance*: the churning customers are about 5% of the entire dataset; *extensive number of features*: 10.000 columns before the preprocessing step (i.e., service, product, offer related to calls, SMS, MMS, Internet, personnel, and demographic information); *missing values*: e.g. not all customers have the same subscription.

To face all the mentioned challenges: a suitable big data platform is used. The dataset is aggregated to extract features for each customer. A data warehouse system (i.e., aggregates billing data, Calls/SMS/Internet, and complaints) is used to decrease the churn rate in SyriaTel. Data Mining techniques applied on top of the Data Warehouse systems do not allow to have sufficiently good results using this data.

The big social networks (15 million nodes (customers) that represent Syriatel, MTN, and Baseline numbers and more than 2.5 Billion edges (transactions)) are considered one of the fundamental components of the graphs of the big data network. Furthermore, the authors calculate all the social networks of customers and features, such as **degree centrality measures**, **similarity values**, and customer **network connectivity** for each customer. The traditional data warehouse system still suffers from deficiencies in computing the essential SNA measures on large-scale networks.

The metric used is the area under the receiver operating characteristic curve "AUC" to estimate the model performance. A predictive system is built using the Hortonworks Data Platform (HDP). In particular, they use a customized package of HDP-installed systems and tools: SYTL-BD framework (SyriaTel's big data framework).

During the feature engineering phase, the authors investigated the selection of a proper sliding window for historical data to extract statistical and Social Networking Analysis (SNA) features.

Results of a predictive model using: Statistical characteristics (extracted using the last 6 months of the raw dataset) related to different historical periods (AUC: 84%); SNA characteristics, extracted using the last four months of that dataset (AUC: 75.3%); combining SNA features with the statistical features, the results increase significantly (AUC was 93.3%).

The data imbalance problem is faced by under-sampling or oversampling without rebalancing. XGBOOST algorithm (without rebalancing) performs with an AUC value of 93.3%; Decision tree (undersampling, AUC: 83%), Random Forest (undersampling, AUC: 87.76%), Gradient Boosted Machine Tree (without rebalancing, AUC: 90.89%).

System evaluation using a new up-to-date dataset (prepaid SyriaTel customers without exception: 7.5 million customers without knowing what their status will be after 2 months). The dataset for customers who are most likely predicted to churn is divided into two datasets (Offered and NotOffered). Marketing experts act proactively to retain customers who are predicted to leave the 'Offered' dataset, and the other 'NotOffered' dataset left without action. The results of the test were compared with the customer status after two months for the two datasets. The best AUC value of SyriaTel New Data 'NotOffered' is 89% for XGBOOST and most of the cases are correctly predicted. XGBOOST (89%); GSM (85.5%); Random Forest (83.4%); Decision Tree (79.1%). The percentage of customers retained in the "Offered" dataset is 47%, which increases revenue and decreases the churn rate by about 1.5%. The performance of the model increases from 84% to 93.3% when using features of SNA. XGBoost outperforms the remaining algorithms tested.

Sniegula et al. [106] investigate which machine learning techniques are most suited to predict customer churn in the TelCo domain. They compare three machine learning techniques on a single churn dataset in the context of the telecommunication industry, in particular, K-means, Decision Tree, and Artificial Neural Network. The dataset is taken from the "bigml" database, a platform dedicated to machine learning. The file has 3333 records with 20 attributes (16 numerical attributes, one text attribute, and three boolean attributes, one of them contains information about the customer churn).

Compared different approaches: K-means method (using Java programming language, version 8), CART decision trees, and artificial neural network (Python programming language).

They aim to answer the following questions: "Is it really necessary to turn towards the complex neural networks?"; "Can satisfactory results be achieved with the use of simpler statistical approaches?"; "Which approach is the best choosing between classification or clustering?". To answer these questions they compare the K-means, DT, and ANN. The metrics used are accuracy, sensitivity, specificity, precision and F1-score. **K-means**: (four different distance metrics are tested (16 tests were conducted: 4 tests for each metric: Euclidean, Chebyshev, Manhattan, and cosine.); Results: Accuracy (60%); the F1-score<30%. Best results of the average values of the performance measures: Accuracy: 62.65%; Sensitivity (recall): 83.44%; Specificity: 70.46%; Precision: 16.73%; F1-score: 27.85%. **Decision Trees**: 28 tests are performed. The best results are achieved with the following configuration: min_samples_leaf; AreaCode; TotalEveMinutes and TotalDayCalls. Accuracy: 94.98%; Sensitivity (recall): 78.92%; Specificity: 98.77%; Precision: 90.97%; F1-score: 80.80%. Clients that did not churn are classified better. **Neural network**: 33 tests performed. Accuracy: 87.11%; Sensitivity (recall): 74.07%; Specificity: 97.81%; Precision: 68.68%; F1-score: 45.38%.

Asghar et al. [123] focus on the customer churn problem using machine learning techniques, in particular the wrapper-based feature selection approach with Particle Swarm Optimization (PSO) is used to find the best feature subset. Decision Tree (DT), Naïve Bayes, K-Nearest Neighbor, and Logistic Regression are used to evaluate the goodness of the feature subset. The use of different classifiers helps in improving the performance (wrapper-based approaches are classifier dependent). They use the customer churn prediction dataset 2020 of a TelCo company: 19 characteristics and 4250

samples by assessing a 10-fold cross-validation, the dataset is divided into 10 random (and equal) partitions. The algorithm runs 10 times using the first partition as the test set and the remaining 9 partitions as the train set. All the experiments are performed using WEKA software (Waikato Environment for Knowledge Analysis): they use the wrapper-based approach and choose PSO as a search criterion with a specific classifier (Decision Trees, K-Nearest Neighbors, Logistic Regression, and Naïve Bayes), they use the default parameters.

The PSO-DT achieves the highest accuracy (94.56%) by selecting the least number of features (only 8 features): international plan, total day minutes, total eve charge, number-vmail messages, total intl charge, total night minutes, total intl calls. The authors use the confusion matrix, dimension reduction (DR) to assess the performance of all the methods. A graphical comparison of all methods in terms of accuracy and dimension reduction achieved (PSO-DT and PSO-NN reduced the feature dimensions by more than 50% with a high accuracy of 94.56% and 89.2%). PSO-LR selected a high number of features (DR = 42.1%), relatively, achieving the lowest accuracy (87.18%). PSO-NB achieved the lowest DR by only 26.32%, although it achieved better accuracy than PSO-LR.

Simulations show that the PSO method works better with DT, identifying the best feature and improving the classification performance (highest accuracy 94.56% and lowest number of selected features).

The approach is advantageous in forecasting the churners for the exponentially increasing competition of TelCo firms.

The paper of **Gu** et al. [46] compares the accuracy and efficiency of several commonly used algorithms. Decision Tree C4.5, Decision Tree CART, K-nearest neighbor (KNN), Linear Regression, SVM (SVC, SVR), Naïve Bayes (GaussianNB, BernoulliNB), random forest, neural network. The

dataset includes 221.770 users, of which 11.282 (5%) are churn customers, with 37 features. Data include attributes, consumption data, and consumption characteristic data, such as customer age, consumption, online time, online time period, online traffic, etc.

The authors found that the Decision Tree (CART) algorithm is fast, accurate (82%), and well suited to predict customer churn.

The CART algorithm is simple and does not require huge computational resources, so it is suitable for data support personnel at all levels to quickly analyze big data for customer churn.

DecisionTree (C4.5, CART), KNN, Linear, SVM (SVC, SVR), Naïve Bayes (GaussianNB, BernoulliNB), RandomForest, Neural Network.

221.770 user data through the support of the superior company, of which 11.282 are churn customers, involving 37 features. Data include attributes, consumption data, and consumption characteristic data, such as customer age, consumption, online time, online time period, online traffic, and other 37 types of data.

The same data is used to compare the accuracy and operating efficiency of the mentioned algorithms. Python is used to develop a big data analysis program on each algorithm before and after cleaning the data.

They also discuss a quick method of cleaning abnormal values (Inter-Quartile Ranger, IQR); in order to adjust abnormal data (outliers) to the non-abnormal range.

The original dataset contains null values and extremely large values (Internet traffic and deposits), which need to be cleaned up. The data before and after data cleaning are compared to test each algorithm separately.

The train-test split is 70%-30% (221.770 pieces which were divided into 161.770 pieces of training data (around 73%) and 60.000 pieces of test data (around 27%)). Features analysis and multi-feature analysis based on experience are performed and 37 features are selected. They analyze the accuracy of the train, the accuracy of the test, the AUC, and the run time of each algorithm.

Decision Tree C4.5. The researchers first used the simpler C4.5 algorithm and the training accuracy of 85%, the test accuracy of 83%, the AUC (AUC stands for "Area under the ROC curve") of 78%, and the ROC (Receiver Operating Characteristic).

Scientists found that the Decision Tree (CART) algorithm is the fastest, most accurate (82%) and is well suited to predict customer churn. Furthermore, this algorithm is simple and does not require huge computational resources, so it is suitable for data support personnel at all levels to quickly analyze big data for customer churn.

Jain et al. [62] discuss the performance of various algorithms and build the best model for the prediction of churn in the TelCo, banking, and IT sectors. They highlight the importance of preventing customer churn. They discuss four machine learning algorithms: Random Forest (RF), logistic regression (LR), SVM, and XGBoost. In the TelCo sector, XGBoost is the best algorithm with a precision of 82.942%, RF for the banking sector with a precision of 86.312%, and logistic regression for the IT sector with a precision of 90.136%. The suggestion to stop attrition is to adopt retention strategies developed by extensive use of explanatory analysis. This study uses the Orange dataset, which is publicly available. The dataset contains 3333 subscriber entries and 21 attributes. The target attribute is “Churn”, there are 20 independent attributes and 483 churners. The experiments include several models: a hybrid model of Decision Tree and Logistic

Regression; Principal Component Analysis (PCA) with Logistic Regression and Logit Boost; Deep Learning CNN-VAE (Convolutional Neural Network with Variational Autoencoder); Logistic Regression; Logit Boost; XGBoost and Random Forest.

The scientists perform the feature engineering and selection at one place and work on improving the model performance, Feature Importance and Correlation Matrix, handling categorical and continuous features for feature extraction. This study uses multiple experiments: hybrid methods and some single techniques.

The performance of the experiments is compared before and after feature selection and with similar literature work. Every prediction model starts the process with the acquisition and processing of the dataset.

Accuracy, Precision, Recall rate, F1-score, Confusion matrix, Macro average, and Weighted average.

The study has been shown to have better results compared to previous models. Random Forest outperforms by achieving 95% accuracy and in all other experiments very good results are produced. This study states the importance of data mining techniques for a churn prediction model and proposes a very good comparison model.

The research of **Lalwani** et al. [73] proposes a six-phase methodology to predict customers that are likely to leave the company's services, in particular in the TelCo Industry. They propose a comparative study of customer churn considering the following phases: identification of most suitable data; cleaning and filtering; feature selection; development of predictive models; cross-validation; evaluation of predictive models on the test set. In particular, they use machine learning techniques such as logistic regression, Naïve Bayes, support vector machines, decision trees, random forest, XGBoost

Classifier, CatBoost Classifier, AdaBoost Classifier, and Extra Tree Classifier. The results show that Adaboost and XGboost Classifiers outperform others in terms of performance measures (accuracy (81.71% and 80.8%, respectively), precision, recall, F1-score, AUC score (84%, is achieved by both)).

In the study of **Melian** et al.[79] they determine the prediction of customer churn on a dataset of a major telecommunication company in Romania. They highlight the importance of correct identification of customers who could 'leave' the network. As a consequence, companies can propose a series of personalized offers, based on the customer's profile, to prevent abandonment. A Romanian mobile dataset is analyzed: a sample of 10715 postpaid customers (out of five million active subscribers of an anonymized database containing historical data). The number of churners is 1468 individuals (13.70% of the sample). They gather Demographic data, Information about the customer's lifecycle, financial strength, and interactions with competing TelCo clients (calls versus or received). The analysis performed determines the indicators that can best underline the attitude of churn. Clustering of k-means is applied (the set is divided into three clusters) and several algorithms are used: Logistic Regression, Decision Trees, Random Forest, Balanced Random Forest, and Propensity Score Matching (PSM).

The 1505 individuals who have not previously been contacted for one year by the TelCo service provider have been selected in the control group.

The logistic regression outlines the characteristics that influence the customer's defection: the time span the company's services (Tenure), the month's number since changing the last offer (MonthsO), the minute's number consumed outside the company (MinR), the value of the invoice paid for the services used (Invoice), the minutes received outside the network (MinR), and the value of the extra costs paid for off-network services (Ex-

traCosts). The decision tree is used to find the variables that answer the question **What are the features that influence the churn action?**. MonthsO is the main indicator (months number since the last offer was changed: change in service), the national minutes received, the additional cost and the invoice value. Customers who are not contacted by the company to change their offer, whose last service renewal is about 40 months earlier, are prone to churn. MinR: customers near completion of 12 months in the network are prone to churn. The number of minutes used to speak with other people outside the provider network is greater than 73.47% of the total number of minutes used within the network. Indicates that the customer could act as a churner in the very near future. **Customer's decision makers for churn**: MonthsO, Invoice, MinR, ExtraCosts. In particular, people without a re-offer over their contract in the last 37.3 months are prone to churn. Customers with an additional cost greater than 15.28 euros and a percentage of calls received greater than 82.06% are churners. In the event of an invoice higher than 51.21 euros, while the Extra Cost is between 15.28 euros and 37.3 euros, individuals are susceptible to follow this behavior. Exceptions: some individuals with the percentage of MinR smaller than 19.78% and the additional costs (ExtraCosts) smaller than 6.75 euros are arranged to churn. The Random Forest and Balanced Random Forest provide the variables which answer the question **"Which of the dataset variables requires the rule in the model?"** MinR (for the Random Forest) and MonthsO and MinC (in the case of the Balanced Random Forest). Random forest (higher accuracy) helps in finding those variables that influence the churning process, observing which "impose the rules" when a customer decides to churn. MinC produces the greatest influence, but also Age and Tenure. Balanced Random Forest finds out that the leading variables in churners identification are MonthsO and MinC which best discriminate between churner and nonchurner. Propensity Score Matching (PSM) determines MonthsO, invoice value (average

invoice), and tenure, as the three indicators explaining the net effect on the churn action of the applied treatment: "customers have not been contacted for 12 months". Furthermore, the 12-month contact policy, the treatment that some individuals have undergone, has affected the customers' decision to leave the network. They conclude that MonthsO, invoice, and tenure are the main indicators that explain the net effect produced by the applied treatment (the customers have not been contacted for 12 months) on the churn action.

The paper of **Mustafa** et al. [82] investigates which variables in the Net Promoter Score (NPS rating, which is a measure of customer satisfaction and loyalty using a ten-point scale) influence directly or indirectly the customer churn. Customer churn is high for customers with a low NPS. They based their study on an NPS dataset from a Malaysian telecommunications company (gathered on September 2019 and September 2020) consisting of 7776 records with 30 fields. They develop a propensity method for customer churn comparing the following algorithms: Logistic regression (LR), linear discriminant analysis (LDA), K-nearest neighbors classifier (KNN), classification and regression trees (CART), Gaussian Naïve Bayes (NB) and support vector machines (SVM). Predictive analytics uses demographic, transactional data, and NPS: in general 33 variables (including the target churn variable). An original trait of their study is that they study the mediation effects of some factors. They identify the causing factors (treated as independent variables) that result in customer churn (the dependent variable representing the effect), but also mediator variables: churn predictors may impact customer churn directly, indirectly, or in both ways. To this purpose, in their analysis, they distinguish among "determinants" of customer churn, which may directly influence the defection, and indirect effects of NPS feedback. Another original trait of their analysis is the categorization of the customer based on the NPS score into three classes: de-

tractor (0–6), passive (7–8), promoter (9–10), and the definition of "partial defection", corresponding to a downward change of class of the customer. Specifically, we have **partial defection** if the NPS feedback rating changes from promoter to passive, and we have **total defection** if it changes from passive to detractor. This NPS feedback status is hypothesized to be one of the **mediators** of the relationship and **link** between churn predictors and customer loss. More in detail: the NPS feedback rating change partially mediates the effect of the customer churn of the variables Duration, Reply Shift, Service Request Type, Helpdesk Staff ID, and Assigned Officer. The authors find that using their model, the CART algorithm outperforms the algorithms tested with an accuracy of 98%.

2.2.4. Support Vector Machines

According to the research of **Brandusoiu** [21], an advanced methodology is proposed to predict customer churn. The predictive model implementation consists of a Support Vector Machines algorithm with four kernel functions: Radial Basis Function Kernel (RBF), Linear kernel (LIN), Polynomial Kernel (POL) and Sigmoid Kernel (SIG). They focus on the importance of understanding and preparing the dataset and then building and evaluating the models. The polynomial kernel function performs better (accuracy of 88.56%). RBF, LIN, and POL have a very good performance (around 80%). The dataset used is from the University of California, Department of Information and Computer Science, Irvine, CA, is complete, and no attributes are missing. It contains call details records and has 21 attributes for each of its 3333 subscribers. In particular, historical records of customer churn, how they turned out in hindsight, i.e., their previous behavior if it turned out that they are churners or not. For each subscriber, they have information about their corresponding inbound/outbound calls count, inbound/outbound SMS count, and voice mail. Regarding the data

preparation and model building, IBM SPSS (Statistical Product and Service Solutions) is used. The minority class is oversampled. And visualize the performance of the models on the testing set by using the confusion matrix, gain measure (models that use RBF and polynomial kernel functions perform better: 80% for RBF and POL, compared to 60% for LIN and 50% for SIG). For the prediction of churners and non-churners, the model that uses the polynomial kernel function performs best, having an overall accuracy of 88.56%. RBF, LIN, and POL have a very good performance (around 80%).

The paper of **Brandusoiu** et al. [25] proposes a new methodology to predict likely to churn subscribers in the prepaid mobile sector by applying their own neural network architecture to the call details records dataset. Model performance is evaluated using the confusion matrix. The gain measure and the ROC curve are used to evaluate the predictive model. The overall performance to predict between churner and non-churner is 99.55%. The prediction of churners on prepaid mobile devices is approximately 99%. Three types of machine learning algorithms are explored: neural networks (multilayer perceptron, MLP), support vector machines parameters are learned by means of a divide-and-conquer approach: sequential minimal optimization (SMO) algorithm and Bayesian networks learned by Iterative Parent-Child-Based Learning of Markov Blanket (IPC-MB). The University of California, Department of Information and Computer Science, Irvine, California provided the dataset. The authors investigate a dataset of pre-paid mobile call details records that consists of 3333 customers with 21 attributes each. It contains information on the usage of a mobile telecommunication system and has a total number of 3333 subscribers with 15 continuous and 5 discrete variables each, and the Churn dependent variable with two classes Yes/No. Three discrete variables (state, area code, and phone) are omitted (because of the inconsistent information contained). For each subscriber,

they found information on the inbound/outbound call count, the inbound / outbound SMS count, and voice mail (**Call Detail Record**). Gain measure and ROC curve are considered to evaluate the predictive model. Performance is evaluated using the confusion matrix. Scientists propose an advanced data mining methodology to predict churn in the prepaid mobile telecommunications industry. The sensitivity and specificity of SVM are 100%, the sensitivity for MLP and BN is approximately 99% and the specificity is 100% for MLP and 99% for BN. By evaluating the results, the models have an overall precision of 99.10% for BN, 99.55% for MLP and 99.70% for SVM. Models have very good performance (from 99% to 100%) in predicting churners.

The evaluation measures used are: Sensitivity/recall; Specificity; Precision; Accuracy; Misclassification error; F1-score; Coverage.

The Information Gain Attribute Evaluator (Feature Ranking Method) using a Weka toolkit makes it possible to select the most appropriate attributes. The authors evaluate four different algorithms for rule generation with an RST-based classification approach. A comparison of four rule generation algorithms (i.e. GA, CA, EA, LA) with RST-based classification is performed.

GA with the RST has shown a more suitable predictive capacity. LA gives the maximum accuracy which is about 0.993; however, it has a coverage of 66.8% of customers (which means that it has only classified 668 instances while 332 customers are ignored).

The covering algorithm classified 64% customers with 0.878 precision which is the least accurate among the four rule generation algorithms. Although the EA achieves less accuracy (i.e. 92.6%) as compared to LA; nevertheless, the EA method performs better than both algorithms (i.e. LA and CA) in terms of coverage, recall, and F1-score.

The work of **Apurva Sree** et al. [12] focuses on churn prediction problems using machine learning algorithms such as Support Vector Machines, Random Forest, and Logistic Regression. Different factors can be found that affect the rate of churning. The IBM Watson dataset contains 7000 customers, with 26.6% of customers which moved from one service provider to another.

The accuracy gained by each algorithm is: Logistic Regression: 80.75%; Random Forest: 80.88%; Support Vector Machine: 82%. They find the relationship between the factors influencing churn prediction. The churn rate is higher for customers which have a month-to-month contract, senior citizens churn more than younger people.

This fact allows TelCo companies to adopt effective strategies that lead to the creation of innovative applications of machine learning techniques.

In the research of **Amin** et al. [6] the Just-in-Time (JIT) approach uses datasets from across companies to address the problem of customer churn. This method requires historical data to obtain the best performance of the classification model. However, JIT can provide answers at the initial stage for a new company or for those that lack historical data archives. The prediction performance of SVM as a base classifier in the **heterogeneous ensemble** is better, compared to SVM applied as an individual classifier or homogeneous ensembles. They use publicly available datasets of two TelCo companies: Dataset Source and KDDCup (1999). **Dataset 1** No. of samples (size): 3333; No. of input variables: 21; No. of numerical features: 16; No. of discrete features: 4; No. of independent variables: 20; No. of dependent variables: 1; No. of independent variables values: 2; **Dataset 2** No. of samples (size): 5784; No. of input variables: 250; No. of numerical features: 214; No. of discrete features: 35; No. of independent variables:

249; No. of dependent variables: 1; No. of independent variables values: 2; Independent variables in dataset 1 are 20 and dataset 2 are 249.

Evaluation measures are as follows: Accuracy; Classification error; Kappa; Precision; Recall; F1-score; Sensitivity; Specificity;

The psep (parameter specifies) is the sum of the positive and negative prediction values minus one. It can be mathematically expressed as: $psep = ppv + npv - 1$ where ppv means positive prediction value and npv is a reference to negative predictive value. Standard Error (SE): it is the standard deviation of the sampling distribution of the mean, $\sigma_{x^-} = \frac{\sigma}{\sqrt{n}}$ where σ is the standard deviation of the population and n is the size of the sample. They visualize the difference in feature space of the cross-company training set, where they consider the feature space of the imbalanced training set, and it represents the features space before using the random under-sampling (RUS) method, the feature space after applying the RUS method in case of the feature space of the balanced training set. The classifier(s) is trained on enough historical data stored in the well-organized CRM of a company and applies the same with the extracted knowledge on the newly established company data. They compare the performances of the **homogeneous ensemble** method (bagging in JIT-CCP) and the **heterogeneous ensemble** method (stacking).

Classifiers used in the heterogeneous ensemble Classifier: K-nearest neighbors(kNN); Naïve Bayes (NB); Neural net (NN); Support Vector Machine (SVM). They compare the standard deviation (SD) versus accuracy(AC) and F1-score (FM) of all the applied methods. Here SD-FM means standard deviation versus F1-score, and SD-AC represents standard deviation versus accuracy. The highest accuracy, the measure F, and the lowest standard deviation represent the best model performance of individual base classifier (i.e. SVM), homogeneous, and heterogeneous ensemble methods. The re-

sults show that the JIT-CCP model without any ensemble methods achieves the best performance (i.e. accuracy: (55.3 ± 7.13)%, F1-score: (47.26 ± 11.12)%, psep: 0.117 ± 0.16 and kappa: 0.106), but with the application of a homogeneous ensemble method which has improved the performance of the JIT-CCP model by 3.94% in accuracy, 3.94% in misclassification error rate and 9.47% in the F1-score. The heterogeneous ensemble method is applied, which further improves the performance of the proposed JIT-CCP model to 18.03% in accuracy, 14.69% in the F1-score, and also reduces the misclassification error rate to 18.03%. The results reveal that the effectiveness of the heterogeneous method is more useful and practical, compared to homogeneous ensemble or individual classifier approach.

2.2.5. Artificial Neural Networks

Vafeiadis et al. [114] present a comparative study on the most popular machine learning methods: Artificial Neural Networks (ANN), Support Vector Machines (SVM), Decision Trees Learning (DTs), Naïve Bayes (NB), Regression Analysis (RA), and Logistic Regression (LR) Analysis. They first evaluate these classifiers through cross-validation. Then, test the performance with boosting. Finally, Monte Carlo simulations are performed to determine the most efficient approach. They test BPN, SVM, DT, NB, and LR models using boosting techniques, NB and LR cannot be boosted because of the lack of free parameters to be tuned. The AdaBoost.M1 algorithm is applied to BPN, SVM, and DT. The boosting technique significantly improves the classification performance. The SVM is a powerful tool compared to the other models explored. The dataset originally from the UCI Machine Learning Repository (converted to MLC++ format1), which is included in package C50 of the R language, contains 5000 samples.

Precision, recall, accuracy, and F1-score (estimated averages) for 100 Monte

Carlo realizations of BPN, SVM-RBF, SVM-POLY, DT, NB, and LR; and with boosting (BPN, SVM-RBF, SVM-POLY, DT-C5.0). Two of the best performing methods in terms of the corresponding testing error are the two-layer back propagation network with hidden units 15 and the Decision Tree classifier; both methods achieve the accuracy of 94% and the F1-score of 77%. The Support Vector Machines classifiers (RBF and POLY kernels) obtain an accuracy of about 93% and an approximate F1-score of 73%. The Naïve Bayes and logistic regression methods fail shortly with approximation accuracy 86% and an F measure of approximately 53% and 14%. Using the AdaBoost.M1 algorithm, the comparative results showed an improvement in performance for the three remaining classifiers due to boosting. Accuracy improves between 1% and 4%, while F measures between 4.5% and 15%. The best classifier is the boosted SVM (SVM-POLY with AdaBoost) with an accuracy of almost 97% and an F1-score greater than 84%.

The article of **Andrews** et al. [10] describes different kinds of machine learning techniques in particular the effectiveness of RF, SVM and KNN compared to DL models that perform similarly to conventional classifiers such as SVM and random forest. Traditionally, various types of machine learning approach like Decision trees, Random Forest, Bagging, etc., are applied to predict churned customers. Deep learning approaches have better accuracy and less processing time. The Boost calculation is proposed to prepare classifiers that show achievement in churn prediction. The dataset is certified by a TelCo Company in Belgium and contains more than 3000 information that was mimicked by 10.000. With a total of 22 factors from 10.000 customers. The test set contains around 2% churners, 200 among 10.000 customers. The entire dataset in which 29% of customers are churners.

In the study of **Shah** et al. [101] is proposed a set of generic features which can be used for most all non-subscription business settings for developing

churn prediction systems. In addition, the application of **causal reasoning** helps in predicting the possible causes that act as critical reasons for a customer to churn.

In this paper, the authors focus on **Non-contractual and continuous business**: where both the sale amount and the purchase interval can vary.

The data is about sale and payment of a dealer, per day, per month or on some other time granularity. A **dealer** is someone who acts as a trader between the business and the end user. The aim of the study is to predict the churn of these dealers.

The data contain sales and payment transactions from April 2016 to December 2018. Dealers who have not transacted from September 30^{th} are considered churned. A total of 6000 customers' sales data is analyzed. The proposed method uses a minimal set of data to generate enough features to build a basic churn prediction system. The problem is approached as typical classification problem in which training data is generated from historical sales and payment data, one sample per customer, and assign a label to each customer whether has churned or not. The sales and payment data of a business-to-business large-scale non-contractual and continuous business are used.

Generic features can be extracted from sales and payment data of almost all non-subscription-based business.

The feature set is based on RFM analysis (Recency, Frequency, Monetary is a marketing technique used to determine quantitatively which customers are the most valuable ones by examining how recently a customer has purchased (recency), how often they purchase (frequency), and how much the customer can spend (monetary)).

The network is trained for dealers with no sales transactions in the last 120 days. Due to the unavailability of the target field (churn), a novel unsupervised method is proposed.

The last transaction date of the dealer is not taken as a training feature, because the target field is derived from that date (not available during inference). The recency of the features is considered 6 months.

A 5 layers (MLP) feedforward backpropagation neural network is used, with the input layer that contains all input fields. The output layer contains the target. (Hidden layers: 200; activation function: ReLu in hidden layers and softmax in the output layer; Learning rate: 0.01; Signature cross entropy as an optimization Algorithm). TensorFlow library is used.

The metrics used are: Accuracy; Precision; Recall; F1-score.

By selecting 500 epochs and after performing a certain number of experiments, the accuracy of the test set is 79.65%. Evaluation metrics of the best configuration are favorable for the nonchurn class: Precision (77.47%); Recall (85.29%); F1-score (80.63%)

A system for **causal analysis** of churn predicts a set of causes that may have led to the customer churn and helps to derive customer retention strategies. The Number of Complaints; Salesman or point of contact changed; Orders canceled due to understock; Returns due to defective material are mentioned to be the main **causal factors**. The authors propose two models specifically for causality analysis of customer churn: **Counterfactuals-based causal model** and **cause prediction** using Bayes theorem. They experimented with both of the proposed methods using our data on two different causes of customer churn.

The counterfactuals model is used for **order blockage** (the cause). Un-

blocking orders will change the payment and sales data of those customers, which in turn changes the features used for churn prediction.

Using the "new feature" vectors to predict the churn, around 45% of customers, whose orders were blocked and were predicted as churned before unblocking, have been predicted as non-churned after unblocking.

There is an average reduction of 0.3 in the overall customer churn score using **counterfactuals** on blocked orders.

Using the Bayes method on **return due to defect** (as a cause), about 64% of customers who returned their orders due to defect and have churned have this as a possible cause of churning.

In the study of **Jolfoo** et al. [66], firstly, they review various algorithms used for customer churn prediction in the TelCo sector. Second, they propose a **hybrid** approach of Artificial Neural Network (ANN) and K-Nearest Neighbor (KNN) algorithms. The ANN model is constructed using multilayer perceptron in SPSS (Statistical Package for Social Sciences) statistics. They train the model through a backpropagation learning algorithm and adjust the synaptic weights through the descent of gradient to reduce the error through the transformation function. The churn prediction model is constructed using the KNN algorithm, as it can perform classification without prior knowledge about the data distribution. The accuracy of the prediction of churn is estimated by the ratio of correct predictions to the total number of cases evaluated. They conclude by emphasizing the importance of customer retention. Companies must concentrate on identifying the **reasons** behind the customer's behavior, actuating appropriate changes in their strategies.

Bauer et al. [15] propose a novel method for CLV customer lifetime value (CLV) prediction customized using deep learning approach based on encoder-

decoder sequence-to-sequence recurrent neural networks (RNNs) with augmented temporal convolutions. This model is then combined with gradient boosting machines (GBMs) and a set of novel features in a hybrid framework. A wrapper-based feature selection approach with Particle Swarm Optimization (PSO) is proposed. Decision Tree (DT), Naïve Bayes, KNN, and logistic regression 10 times cross-validation classifiers (PSO-DT, PSO-NN, PSO-LR, PSO-NB). Despite this work is not focusing on the churn prediction the proposed methods can be applied to the telecommunication scenario. Real-world data from a larger e-commerce company and a public dataset from the domain of online retail, the CDNOW dataset. They included the BG/NBD model (i.e. advanced RFM model) as one of the baselines in performance comparison. In the datasets used for evaluation, information about churn is also not available, which is why churn is modeled implicitly in the approach when the future CLV is predicted to be zero at some point in time. That is the way they process the available time series data. Most of the features are dynamic, that is, change over time. They do not consider the number of children of a customer as a static feature that is computed only once, but derive the feature values for every point in time considered in the past, e.g., each day or week. Features are based on: customer attributes; orders (item purchases); other item interactions. Most features are related to orders and other types of customer behavior and represent aggregated values, e.g., the number of purchases or the total profit that was made. Thus, this leads to very large feature vectors and the problem of data sparsity.

The approach is evaluated on two publicly available datasets (Children dataset) and proprietary. Public datasets that include profit or revenue information, essential for CLV prediction, are generally scarce. The dataset (European e-Commerce company selling products for children and families), having over one million customers. The corresponding purchase data is

recorded over three years. The results are validated on UKRetail (smaller dataset). A total number of 541.909 purchase records by more than 4.000 customers of a UK company are recorded for a period of about one year. For the smaller UKRetail dataset, a model based on daily data was built for the UKRetail dataset. For the large Children dataset, weekly aggregates are used instead of making the computational processes more efficient. For the same reason, only selected features of high importance are used in the RNN model for the Children dataset. To avoid bias of the results, the authors focused on regular customers and excluded wholesale customers.

Empirically-determined models are applied and capture long-term trends in purchasing behavior, for example, a steady increase in order volume over time, as well as periodic patterns.

The proposed method(s) outperform all baselines in this comparison on both measures and for both datasets, and the lowest (best) RMSE and MAE values are obtained in both cases with the stacked combination of the GBM and sequence-to-sequence RNN model (GBM-S2S).

In the paper by **Dalli** et al. [29], the authors investigate how the hyperparameters' tuning affects the performance of deep learning models. They compare several ML algorithms and techniques: traditional machine learning techniques (Random Forest, Logistic Regression, Support Vector Machines, and Naïve Bayes), Ensemble Learning, and Deep Learning. Based both on the approaches and the results in the literature, they select Deep Learning techniques as a robust and effective solution. However, while the usage of Neural Networks has been deep-routed, they notice that little investigation has been done on the impact of different hyperparameter selection, also on the Neural Network performance tuning for an improved network.

Some papers confirm analogous findings as they use the same Deep Learning methods to predict customer churn in the TelCo business. Thus, it proves that the Neural Network hyperparameters' configuration approaches are lacking when dealing with telecommunication customer defection.

In particular, due to the lack of configuration of hyperparameters in Neural Networks, they contribute by analyzing the impact of three aspects: "different combinations of activation functions on the performance of the ANN model"; "several batch sizes used on the ANN model performance"; "different optimizers on the performance of the ANN model". The applied methodology is already suited in the publication of Domingos et al. for the prediction of customer churn in the banking sector.

They consider the open-source database "Crowd Analytix", a dataset of customer churn that contains 21 characteristics (one dependent variable). The 'Churn' feature is either the churn rate or the non-churn rate of the customers (14.5% is the percentage of 'Churn', while 84.5% is 'non-churn'). In the preprocessing step, they consider several aspects, in particular: **Missing values**: they removed features with more than 95% missing values in the data considered. **Outliers**: they exclude the notably different data points. **Category variable**: binary input where a categorical variable has been safely transformed to binary variables using the dummy variable approach. There are (n–1) dummy variables at once (n is the number of distinct values in categorical variables). **Standardization**: to uniformly standardize the data, that is, **total number of calls** (thousands) and **total call time in minutes** (millions), the MAX-MIN scaling approach is used. **Class imbalance**: the churner ratio is lower than the non-churner ratio, so to balance the classes, the synthetic minority oversampling method (SMOTE) is applied to the training set. **Feature selection**: they eliminate highly correlated features (justifying that these increase the computational workload both without extra effort and without improving accu-

racy). Finally, they reduce to 10 the standardized values to use as input to the NN, and the Churn to train the model. They use the accuracy metric (the number of correct predictions made as the ratio of all predictions made). **Impact of Activation Function**: they aim to investigate the impact of the configuration of alternative activation functions (Sigmoid, ReLu, Tanh) on the extreme performance of the neural network model. The best prediction accuracy of 86.8% is derived from the combination of a hidden Rectifier (ReLu) layer and a Sigmoid Output layer. **Impact of Batch Size**: They adjust the batch size from 3 to 230, the average accuracy of 84.52% is achieved once the batch size ranges from 3 to 40 and decreases for higher values. **Impact of Optimizers**: the dataset is divided into 10 parts during the training phase. To use the k-fold cross-validation technique, the model is trained on nine flaps and tested on the tenth. The algorithms considered for optimizers are stochastic gradient descent (SGD), adaptive gradient (AdaGrad), Adadelta, root mean square propagation (RMSProp) and Adam and AdaMax. RMSProp, chosen as a training algorithm, achieves higher accuracy (86.45%). In conclusion, the authors want to generate meaningful heuristic knowledge to anticipate customer attrition. Contributing to improving the NN's hyperparameter tuning and accuracy. They suggest that a human's configuration-based hybrid architecture would autonomously pick the perfect and very best hyperparameters so as to train, test, and improve the model's performance. Investigation of the impact of sinusoidal activation functions (sine and spline) would have an effect on various types of activation functions. And finally, they suggest using the proposed framework to predict the loyalty of the customer (not loyal, loyal, or very loyal).

According to the research of **Tsai** et al. [111], a hybrid data mining technique is applied and combined with feedforward artificial Neural Networks (ANN) and self-organizing maps (SOM). In particular, they combine two

ANNs (ANN+ANN) and ANN+SOM. The first part of the hybrid model filters out unrepresentative data or outliers and evaluates a prediction model. Hybrid data mining models combine clustering and classification data mining to improve performance. This paper proposes hybrid models: ANN with artificial Neural Networks and ANN with self-organizing maps (SOM). The hybrid models outperform the single Neural Network in terms of accuracy and type I and type II errors. The ANN+ANN model performs better and is more stable. They considered a CRM dataset provided by American TelCo companies, containing 51.306 subscribers, including 34.761 churners and 16.545 non-churners, from July 2001 to January 2002. Customers which have been with the TelCo company for at least six months are defined as mature customers. Churn is calculated based on whether the subscriber left the company during the period 31–60 days after the subscriber was originally sampled.

2.2.6. Genetic Programming

The study of **Idris** et al. [59] proposes an efficient churn prediction approach, based on exploiting powerful searching capabilities of genetic programming (GP) supported by an AdaBoost-based iterative approach. It identifies customers who are ready to quit, based on Orange and Cell2Cell datasets. The Orange dataset comprises 50.000 instances where only 3276 are churners. Cell2Cell includes 40.000 instances where 20.000 are churners.

This new methodology combines genetic programming (GP) and AdaBoost (search and classify) to evolve a high-performance churn prediction system with improved churn identification capabilities. The majority class of the Orange dataset is undersampled by using Particle Swarm Optimization (PSO) (based on the undersampling method) to address the class im-

balance. This method is combined with the GP-AdaBoost algorithm and provides the ChP-GPAB churn prediction system (i.e. ChP means Churn Prediction). The algorithms used are as follows: Random forest; Rotation Forest; RotBoost and GP-AdaBoost. By applying GP-AdaBoost it is possible to identify the characteristics that represent the **reasons** for the churn behavior of customers. The final results are computed by accumulating the results over all iterations of 10 fold cross-validation. GP-AdaBoost ensemble classifier in combination with PSO-based undersampling is able to interpret customer churn behavior and achieve improved prediction performance.

Sensitivity and specificity measures and AUC and ROC curves are used. The results show that all the classifiers (Random forest; Rotation forest; RotBoost and GP-AdaBoost) show deteriorated performance on the Orange dataset (imbalanced). On the other hand, algorithms perform well on the Cell2Cell (balanced) dataset, and GP-AdaBoost achieves higher prediction performance. Ch-GPAB attains sensitivity scores of 89% (Orange) and 93% (Cell2Cell), the best prediction performance (in terms of AUC) reported in the datasets. AUC obtained by GP-AdaBoost ranges from 70% to 91% over 30 independent simulations. Ch-GPAB efficiently predicts TelCo churners and is also effective for investigating churn behavior.

The AUC is studied in the Orange dataset by making a comparison based on Ch-GPAB (0.751), gradient boost machine (0.737), decision stump-based model (0.725), decision tree-based model (0.715), Bayesian net (BN) based approach (0.714). They compare the performance of the Ch-GPAB (AUC: 0.910) and Naïve Bayes (NB) based approach (AUC 0.818) on the Cell2Cell dataset. A McNemar statistical test is performed to evaluate the confidence level of the Ch-GPAB system's prediction performance. A comparison of Ch-GPAB with Chr-PmRF (Churn Particle Swarm Random Forest) [58] and RUS-Boost (Random Under Sampling Boost) is made. McNemar's confusion matrices show: 10.200 instances are correctly predicted by the pro-

posed Ch-GPAB, but incorrectly classified by Chr-PmRF for the Cell2Cell dataset. A total of 1300 instances are correctly classified by Ch-GPAB and incorrectly predicted by Chr-PmRF for the Orange dataset. Furthermore, 450 (Cell2Cell) and 22 (Orange) are incorrectly predicted by the proposed Ch-GPAB but correctly predicted by Chr-PmRF. The proposed Ch-GPAB correctly classified 3.125 instances, while only 11 instances are correctly classified by RUS-BOOST which are incorrectly classified by CP-GPAB in the Orange dataset.

The paper by **Amin** et al. [5] proproses a novel Adaptive Customer Churn Prediction (ACCP) model with the ability to learn continuously (i.e., brain-like improving knowledge boundary). The Machine Learning at prediction time without retraining the CCP model several times with minimal data loss. In particular, an adaptive learning approach by using the NB classifier with a Genetic Algorithm (subclass of an Evolutionary Algorithm) based feature weighting approach (i.e. FWAGA: Features Weight Assignment using GA). They employ the self-learned optimum attribute weighting technique using a genetic algorithm without losing data and keeping the attribute independence. Furthermore, the model maintains good prediction accuracy. Additionally, the authors perform a literature review on the prediction of customer churn. The ACCP performance is evaluated on publicly available data sets, considered as a benchmark, BigML Telco churn, IBM Telco, and Cell2Cell, justifying that private real-world telecommunication data sets prevent reproducibility and extrapolation. The ACCP is compared to the baseline classifiers: NB with default setting, Deep-BP-ANN, CNN, NN, LR, XGBoost, KNN, Logit Boost, SVM, and PCALB. They use accuracy, precision, recall, MCC. Achieving the results of average precision, for each data set: 0.97, 0.97, 0.98, a recall rate of 0.84, 0.94, 0.97, and F1-score of 0.89, 0.96, 0.97, an MCC of 0.89, 0.96, 0.97, and accuracy 0.95, 0.97, 0.98 respectively. They conclude that the overall performance of the

ACPP approach is better by 30% in terms of average precision compared to the baseline classifier with default setting.

2.2.7. Bayesian Models

In the paper by **Amin** et al. [7] they discuss the **Cross-Company Churn Prediction** (CCCP) as an alternative handling of the bottleneck in Within Company Churn Prediction (WCCP). CCCP means that one company (the target) lacks enough data and can use the data of another company (the source) to successfully predict customer churn (i.e., training data from one company and applying it to the target data of another company). For this reason, they develop a model for CCCP using **data transformation** (DT) methods, i.e. z-score, log, rank and box-cox. The last three methods significantly improve the performance of CCCP. The authors both validate the impact of these transformation methods in CCCP and evaluate the performance of the underlying baseline classifiers: NB, KNN, GBT, SRI, and DL using publicly available datasets (i.e., Subject Dataset-1 and Subject Dataset-2). Specifically, Subject Dataset-1 is used as the target and contains 2850 churners and 483 non-churners. Subject Dataset-2 is used as the source and contains 15760 non-churners and 2240 churners. The evaluation measures used are Confusion Matrix, Probability of Detection (POD), Probability of False Alarm (POF), AUC and G-Mean (GM). The NB classifier outperforms on transformed data in terms of AUC values 0.51, 0.51, 0.513 in raw, log and Box-Cox, respectively. The DP, KNN, and GBT classifiers outperform on average, while the SRI classifier does not show significant results in terms of the commonly used evaluation measures (i.e., POD, POF, AUC, and GM). The SRI classifiers achieve the maximum performance (i.e. AUC value of 0.541) in the single DT method (i.e. Z-score) and obtain the lowest level of performance (i.e. AUC values of 0.45, 0.44, 0.357, 0.455 in raw, log, rank and box-cox). In conclusion, they suggest

that a company that lacks the necessary data for the learning purpose of the classifier can use the data of a mature company. In addition, methodological practices are provided to assess and link the significant advantage in existing and future data transformation methods in the context of telecommunication companies. A comment is made on the data requirements that need to be considered from the researcher's perspective, as the cross-firm data should be managed in an appropriate way for empirical analysis and the development of novel models. The proposed approach provides a clear picture of the expected scientific consequences of increased data normality in terms of increased predictive model performance.

2.2.8. Logistic Regression

The research of **Zhao** et al. [129] predicts high-value customer churn and identifies the potential churned customers putting forward targeted win-back strategies, it uses a logistic regression algorithm combined with TelCo big data analysis and historical information estimation of customers. The article analyzes the trends and causes of churn and answers the questions **how customer churn occurs using data mining techniques**, **what influencing factors are worth considering**, and **how enterprises can win back customers who left**. The study helps customer relationship management identify "high-risk churn" customers in advance, improves customer loyalty and viscosity, maintains "high value" customers, continues to provide customers with "value", and reduces the cost of maintaining customers. Loyal customers can help enterprises tune the expenses (cost of publicity and negotiation, etc.) and attract more new customers with a herd mentality. By analyzing the characteristics of the churn behavior, the paper identifies potential churned customers in the customer library and helps enterprises to take targeted win-back measures according to the characteristics of the potential churned customers. They consider the top 20%

of high-value customers in the dataset.

The churn **reasons** or **factors** can be price, personal, service, product, market, marketing strategy and market intervention of competitors. Correct discovery of these factors is the key to both recover the churned customers and reduce the churn rate.

The **influence factors** can be focused on three aspects: consumption-related variables (call duration and consumption amount), customer statistical variables (identity information and age, customer income and customer satisfaction), and enterprise-related variables (channel operation ability and purchase of related products). They analyze the **cause** of customer churn and use the **logistic regression** to predict the trend of churn.

The study provides a theoretical reference based on which the TelCo industry can thwart the phenomenon of customer churn, develop the winning strategy, maintain the share of users and strengthen competitiveness in the market.

The authors consider the **customer value** as an important criterion and keep in mind that not all workers are worthy of return.

They suggest focusing on a limited number of people defined as **high value customers** (golden assets, around the 20% of their dataset) and having to be the main target to be won back.

The analysis is based on historical data and the average monthly consumption of all customers. High-value customers are the key customer group to be maintained, and minimum average monthly consumption (RMB 60: threshold) is the judgment criterion. The average monthly consumption higher than the threshold is the analysis object. In particular, they take 11255 random samples (in the middle of 2020) on RMB 60 for three consecu-

tive months. They use the linear correlation between variables (correlation coefficient). The correlation between the current package value and the ARPU (0.5) is the highest.

Based on the awareness that the enterprise managers' inductive reasoning experience might be unreliable, resources invested in winning back those customers with a high possibility of churn have to be well-meditated and streamlined.

Logistic regression predicts the trend in customer churn, assists enterprises in finding the early warning signals of customer churn, and determines the tendency of a customer to churn. A positive correlation exists between ARPU and customer churn, complaint and customer churn, and negative correlation exists between discharge of usage (DOU), current package value (pack-type), convergence business (contract) and customer churn.

They use the confusion matrix, Total precision (TP), Area Under the ROC Curve (AUC), precision, sensitivity, and specificity.

The research results show that the churn rate increases when the monthly consumption increases and customer satisfaction is low. On the other hand, factors such as a high dependency on products, a high package value, and a signed bundling contract decrease the abandonment.

The price is a key factor in customer churn on the premise of the same quality of products (provided by other companies); the building of the corporate brand is still important, which will focus on customer demands, and different special products will be launched for varied market segments to improve the dependence of customers on products. The suggested approach is to consider the perspective of "the customer is always right". Based on this advice, build a series of efficient strategies and actions. In particular, the identification of "high-value" customers, their value, their index of con-

tribution, their needs, and complaints help the company to allocate more resources, improve satisfaction, and adopt the convergence business strategy that increases the dependency on the products and the transfer cost for abandoning the network. Furthermore, efficient complaints handling plays a key role in winning back customers.

The paper of **Zhang** et al. (2022) [128] focuses on producing a suitable customer segmentation to improve churn predictions based on discriminant analysis and logistic regression and investigates statistical methods, such as factor analysis, to identify the most significant factors in customer's profile description. The used data is collected from 2007 to 2018 from the three main Chinese TelCo companies (China Mobile, China Unicom and China Telecom),it includes information of 4126 customers (184 women (28.7%) and 2942 men (71.3%), aged 9 to 107 (the most common ages ranging from 20 to 60 years). Furthermore, they use **demographic** information and **business-related** information (Sex, Age, Career, Fixed and/or mobile lines, Non-fixed/Fixed monthly fee), and data on the customer's **call traffic** and **SMS/MMS activity** (Monthly minutes in local/long distance calls using mobile/fixed lines, Minutes of Usage (i.e. MOU), number of SMS/MMS). The hypothesis behind the customer's segmentation is that customers with similar consumption–expense behaviors have a similar propensity to churn. The main criterion for the segmentation is the expense-related characteristics. They assume that the following **factors** influence the customer loss: total fee receivable for the month; fixed monthly cost; local fee; roaming fee; network/company fees; fixed-line fee; total monthly caller: minutes of usage (MOU); total monthly called MOU; the total local caller MOU; SMS quantity. Starting from this set of quantities, apply the first common factor analysis, which consists of seeking the fewest factors that can account for the common variance of a set of correlated variables. Using to this purpose the Data Analysis Statistical Package for Social Sciences

(SPSS) they find by linear regression a collection of independent variables: non-monthly fixed cost (F1); monthly fixed cost (F2); the calls MOU (F3); long-distance and roaming call (F4); SMS (F5); China Unicom's MMS (F6). The Kaiser–Meyer–Olkin (KMO) and Bartlett Sphericity tests are also applied to conclude that these factors are suitable for factor analysis. Authors perform the Factor Analysis to characterize: the **Expense** (i.e. monthly fee, package type, mobile terminal price data), Call and SMS attributes. The selected **Expense-related** factors (total fee receivable for the month, fixed monthly costs, local fee, roaming fee, Unicom's network fee, China Mobile's fee and fixed line fee) are used to conduct factor analysis and determine the characteristics of the cost factors. Based on the cumulative variance and the component score coefficient matrix they concluded that common factor of non-monthly fixed costs and common factor of monthly fixed costs could characterize the expense attribute. **Customer Calls** such as data for total monthly calls, long-distance calls, and roaming are useful to understand the customer's **preferences** for the service provider (i.e., better call quality and service will positively influence customer loyalty). The **Call-related** factors such as: total monthly traffic MOU (minutes of usage); total monthly caller MOU; total monthly called MOU; total local MOU; total local called MOU; total long-distance MOU; total roaming MOU. The Kaiser–Meyer–Olkin (KMO) and Bartlett sphericity tests are applied to identify whether these factors are suitable for factor analysis. The common extracted factor values: the results are considered scientific and representative, as the loss rate for each variable is low.

Based on the cumulative variance and the matrix of the coefficient of the components score, they concluded that the common factors of the called MOU and the common factors of long-distance and roaming call characterize the **Call** attribute. **Customer SMS and MMS data** SMS-related factors: the quantity of SMS of China Unicom, the quantity of SMS of

China Mobile, the quantity of SMS of China Telecom, the quantity of MMS of China Unicom and CRBT (Caller Ring Back Tone) were used to perform factor analysis and analyze the characteristics of the cost factor.

Given those factors and the factor Gender, the authors apply first the Fisher discriminant analysis to the dimensionally reduced data. This classification technique projects high-dimensional data onto a line and performs classification in this one-dimensional space: the projection used is one that maximizes the distance between the means of the two classes while minimizing the variance within each class. The authors then use the same data to learn a classifier based on standard logistic regression. According to the results, the TelCo customer churn model constructed by logistic regression has a higher prediction accuracy (93.94%) and better results compared to Fisher discriminant equations (75%).

The summary of the literature search of this chapter is proposed in Table 2.2.

2.2.9. Uplift Models

The paper of [33] discusses the limitations of traditional customer churn prediction models and suggests the use of uplift models. They use a financial institution. It consists of records containing customer information, including a churn indicator and a variable determining whether a customer was targeted with a retention campaign. Total number of observations (i.e. $200,903$), Total number of variables (i.e. 162), Number of control group observations (i.e. $118,809$), Control group churn rate (i.e. 25.52%), Number of treatment group observations (i.e. $82,094$), Treatment group churn rate (i.e. 13.25%), Overall Uplift (12.27%). They apply random stratified sampling to the treatment and control groups to obtain training and test sets including $\frac{2}{3}$ and $\frac{1}{3}$ of records. They use Logistic Regression and

Random Forests to develop Customer Churn Prediction (CCP) and Customer Churn Uplift (CCU) models. Finally, the results evaluation is made both with profit curves according to maximum profit (MP) and maximum profit uplift (MPU) performance measures. They assess the churn rate and the corresponding lift and uplift curves. Finally, they focus on the rank correlation between different setups.

They highlight that traditional churn prediction models aim to identify customers who are likely to churn based on their historical data and predictive analytics.

A problem of these models often consists of failing to provide actionable insights for businesses because they only identify the potential churners without providing guidance on how to effectively intervene to retain them. Uplift models (i.e. persuasion modeling or incremental modeling) focus on identifying customers who are most likely to be influenced positively (i.e. persuadables) by targeted interventions, such as marketing campaigns or loyalty programs. By distinguishing between customers who are responsive to interventions and those who are not, uplift models enable businesses to allocate resources more efficiently and achieve better results in customer retention efforts.

In particular, they propose a new paradigm that improves predictive analytics by a novel, profit-driven evaluation measure called the maximum profit uplift measure.

The importance of shifting from churn prediction to uplift modeling in customer retention strategies maximize the effectiveness of marketing efforts and improve overall business outcomes.

2.3. Discussion and Conclusion

In this chapter we surveyed the literature on Machine Learning methods for the CCP within the scope of the TelCo domain in the approximate period of the last decade.

An interesting observation is that, despite numerous articles addressing this issue, they often fail to build upon each other's findings. The diverse structures of datasets, such as the inclusion or exclusion of call center contact data and details about economic offers, prevent the identification of a "best algorithm".

In the few cases where the same public dataset has been utilized, the huge variety of algorithms, variants, and hyper-parameter configurations confounds the ability to provide a definitive answer regarding algorithm fitness to the problem.

In conclusion, the **best algorithm** is **dataset dependent**. Even when considering the dataset of a single company over a sufficiently extended period, determining the "best algorithm" is challenging due to the dynamic nature of the TelCo services landscape. Thus, the quest for the best algorithm is time-specific, company-specific, and perspective-dependent, relying on the subset of company data under consideration.

However, the definition of churner is not always unambiguous: in fixed line services, there is a clear distinction of customer departure, whereas in prepaid mobile services this doesn't happen. In particular, the definitions distinguish from one another in the number of months passed since the customer used the service the last time.

Moreover, in the realm of fixed line services, additional uncertainties arise. Instances have been documented where certain companies maintain cus-

tomers on their records even after they have switched to other providers, and they agree to formally terminate the contract only after complaints have been lodged.

In addition, in the fixed line services additional ambiguities arise. Instances have been documented where certain companies maintain customers on their records [31] even after they have switched to other service providers, and they agree to formally terminate the contract only after complaints have been lodged. This situation can introduce potential distortions in churn prediction, as there may be a spike in communications just before the customer's official departure, which an algorithm might incorrectly interpret as an indicative signal for predicting churn. In reality, the customer has already left the company and is merely expressing dissatisfaction. In addition, while many studies demonstrate remarkably effective results after setting apparently reasonable definitions of churners there is a notable absence in the literature of **sensitivity studies** to assess whether these studies adopted overly conservative definitions of churners. Another aspect to take into account refers to **preprocessing issues** and **techniques**, since the prediction algorithms represent the tip of the iceberg. Setting aside common data preprocessing problems, a variety of data preparation approaches emphasize that finding a good representation of the data before using statistical learning algorithms is beneficial in several cases. The **feature selection** and **feature extraction** techniques (almost all the articles present their own) provide this representation. Additionally, Deep Learning algorithms [62] are able of learning this representation upfront, leading to considerable improvement in the outcomes.

Additional available strategies exist for enhancing prediction and target business value without trying to select the best algorithm. Prioritizing valuable customers appears to be a sensible approach, particularly because

this customer group [129], seems more straightforward to characterize for predictive purposes.

However, many aspects of the problem remain unexplored from a business perspective, often due to the absence of suitable datasets that provide relevant information.

The structure and pricing of service offerings, along with potential service bundles, play a critical role in the customer's decision-making process, yet such data is rarely publicly accessible. Even within large firms, consolidating diverse data into a cohesive repository poses a **challenge**. Additionally, customers always have to face the **dilemma** of **'stay or go'** which involves both the structure and the pricing of the service offer, as well as potential service bundles (e.g. making choices based on competitive offerings). For these reasons the acquisition of **contextual data** with accurate timing information in today's dynamic market is nearly impractical.

Focusing on value customers appears to be a sensible strategy, also because the group of those customers [129] seems easier to characterize for the purpose of prediction.

From a technical point of view, a notable gap in the literature is represented by the limited number of attempts to address the problem through **causal reasoning**. Only the work of Mustafa [82] attempts a mediation analysis approach. However, to do so, it does not use neither the **Structural Causal Models** nor the **Causal Inference**, but relies on correlation-based studies involving a ready available variable, the Net Promoter Score, which captures domain **expert knowledge** on the customers.

In summary, despite the fact that the problem has a long history and that several works address its multifaceted issues, still much room is left for investigation.

The works described in this chapter are used as the baseline for the implementation of the model and analysis pipeline discussed in Chapter 3 and Chapter 6.

Chapter 3.

Experimental Setup

3.1. Telecom Dataset Description

The work of this thesis presents a **real data-driven** modelling. The computational power of learning method is secondary compared to the data quality: the resulting model would be ineffective if the data are not informative enough [17]. For these reasons the experimental design (i.e., the data observation and the data collection) is crucial step [17].

In this scenario, we received the dataset from the Telecom Italia Mobile company **TIM S.p.A.** so we had to rely on the data collection capability of the company (i.e. observational setting) [17], [39]. Behind an observation setting, there is the strong implicit assumption that the observations are independent identically distributed (i.i.d.) samples of a stationary (i.e. invariant) stochastic process [17]. In real case scenario, this assumption can not be valid (at least not for a long time), and the **nonstationarity**, **drift** should be considered in the learning process. Furthermore, we can think about the poor causal value of inferences made in an observational setting, e.g. in situations of **sampling bias** or **non-observable variables** (e.g., in causal discovery). The purpose of using this dataset is to provide a case study to be investigated to predict the customers that leave the

company by applying machine learning techniques. Furthermore, to explain the possible reasons of the customer's abandon by using the causal inference approach. With the aim to provide to the company's technicians both an "alert" and a toolkit to apply to real big-datasets (i.e. 7/8 millions of records as mentioned by the domain expert).

3.2. Structure of the dataset

The dataset contains the information and characteristics of the **Business customers** related to the **Mobile lines**, necessary to address the research questions (dataset relevance). The entire dataset available in the **Production Department** includes more than one million of records. Due to **Privacy** concerns, in the dataset available in the **Development Department**, which provided the sample, the number of the customers which expressed consent to profilation is lower respect to the initial value.

The dataset contains five months summary of customers' activity, it comprises one entry per customer and per month, and a total of 5 months are present for the year 2022. About 255500 entries are present. The target variable, **churn**, is represented as a 1 if the customer leaves or 0 if the customer remains.

Features and Data Domains

This section is based on the summary of the unstructured interview made with the domain expert of the company. The data is provided in *.parquet* format, the dataset's structure contains 255500 rows and 87 columns. The 87 features can be grouped into categories. We denote each variable by letters corresponding to its category, in particular: 4 features (`'anag_m_'`) are included in the **Anagrafica Linea** (i.e. Registry Telephone line) domain

which contains information regarding the phone line: type, seniority, etc. 17 features (`'cns_m_'`) are included in **Consistenza ed Ordini** (i.e. Consistency and Orders) domain which contains information on orders, offers, services and products associated with the line. 12 features are included in **Traffico** (i.e. Traffic) domain which contains the line value and volume traffic indicators, incoming and outgoing (calls, data (MByte)), denormalized over the month.

8 features (`'usg_m_'`) are included in **Usage** domain which contains Data traffic usage indicators divided into the various Rating Groups (i.e. News websites, streaming websites, etc.) related to the specific "url" visited by the line. 4 features (`'esi_m_'`) are included in the **Esigenze** (i.e. Needs, or the hystorical information) domain which contains information related to needs open from the line with the Customer Relationship Management (CRM) (i.e. info requests, malfunctions: call to 187 and 191 service numbers provide a mast before talking to the operator, in which the customer is routed to a specific need type).

The customers which also have a fixed line are included in the **Business Integrato** (i.e. Integrated Business), in particular 14 features (`'anag_cli_'`) are included in the **Anagrafica Cliente** (i.e. Customer Data) domain which includes the Customer information (company) holder of the line.

3 features (`'cerved_int'`) are included in **Cerved** domain, which contains information about the Italian companies registered in the Register of Companies held by the Chambers of Commerce and the NOREA subjects present in the Cerved archives on: companies registry, reliability classes and assessments, ateco and sector performance, future scenarios, budgets.

23 features (`'jakala_int'`) are included in **Jakala** domain, which refers to the demographic and economic information of the Istat sections associated with customers (only for companies on Cerved databases), both from

data collections and estimates, both provided by Jakala. We have explored the dataset to understand its characteristics, distributions, and patterns. And visualized the key features to gain insights into the data, Figure 3.1, Figure 3.2, Figure 3.3, Figure 3.4 and Figure 3.5.

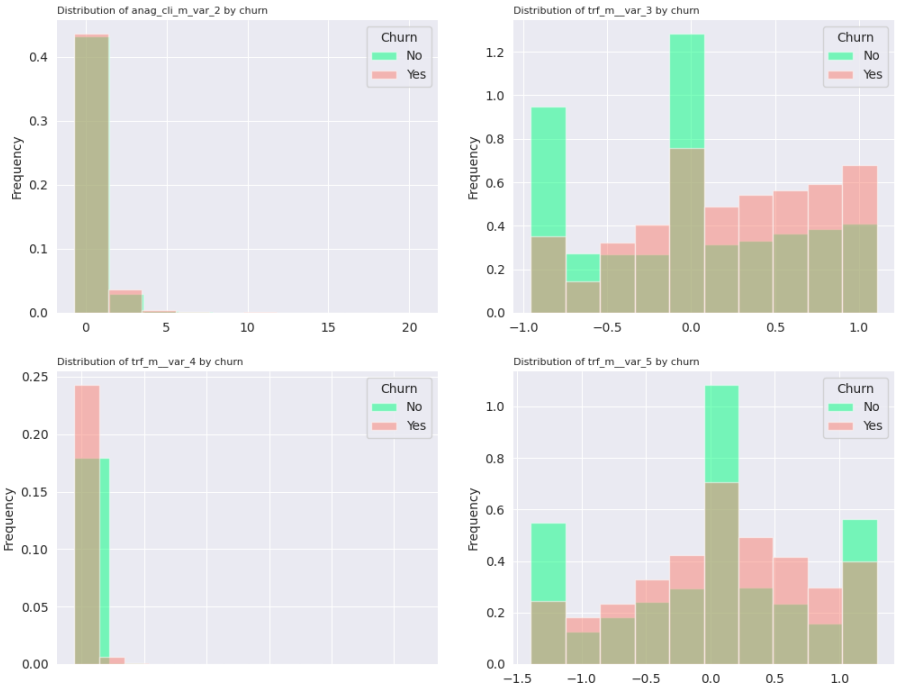


Figure 3.1.: Distribution of variables by churn: anag_cli_m_var_2, trf_m__var_3, trf_m__var_4 and trf_m__var_5.

The variable $anag\_cli\_m\_var\_2$, see Figure 3.1, presents a high peak in 0, the frequency of churners (red) is slightly higher respect to the non-churners (green), the two distributions are overlapping in the range $[0, 6]$ and it is not possible to distinguish between them. The variable $trf\_m\_\_var\_3$, see Figure 3.1, presents two distinct peaks (where the two populations overlap) when x-axis is 0 and -1, the frequency of non-churners (i.e. 0.95 and 1.35) overcomes the churners (i.e. 0.36 and 0.75). The two populations present opposite trends along the x-axis. The variable $trf\_m\_\_var\_4$, see Figure

3.1, presents a high peak in 0, the frequency of churners (i.e. 0.24) is higher respect to the non-churners (i.e. 0.17), the two distributions are overlapping in the range $[0, 6]$.

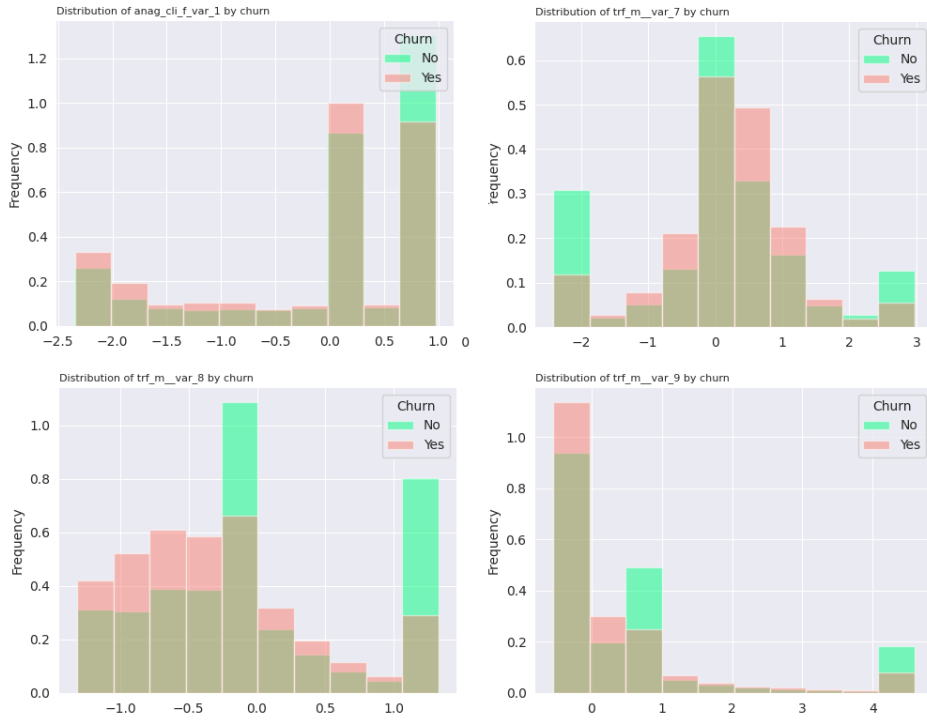


Figure 3.2.: Distribution of variables by churn: $anag\_cli\_m\_var\_1$, $trf\_m\_\_var\_7$, $trf\_m\_\_var\_8$ and $trf\_m\_\_var\_9$.

The variable $anag\_cli\_m\_var\_1$, see Figure 3.2, presents two distinct peaks in 0.25 and 0.75 respectively. The frequency of churners (red) is 1 in the first peak while for the non-churners reaches 0.85. The second peak presents a higher frequency for non-churners (i.e. 1.3) with respect to the churners which is 0.9. There seems to be a decreasing trend along the x-axis, where the churner class is more frequent than the non-churner class, except for the last peak. The variable $trf\_m\_\_var\_7$, see Figure 3.2, shows gaussian-like distribution of frequencies, except for two distinct peaks in

-2 and 3 where the non-churner class outnumbers the churning one. The variable $trf\_m\_\_var\_8$, see Figure 3.2, shows a similar bell distribution, with higher values in the churning class. We can also identify two peaks around -0.2 and 1.2, where the non churning frequency is substantially higher. The variable $trf\_m\_\_var\_9$, see Figure 3.2, shows a long tail distribution, with higher frequencies of churners around 0. We can identify two peaks of non-churners around 1 and 5.

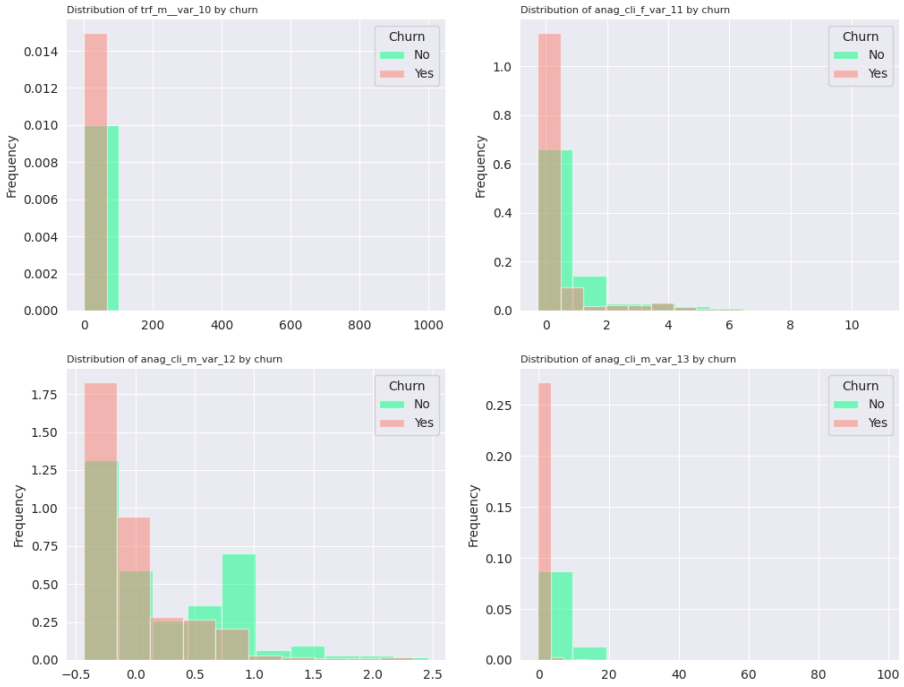


Figure 3.3.: Distribution of variables by churn: $trf\_m\_\_var\_10$, $anag\_cli\_f\_var\_11$, $anag\_cli\_m\_var\_12$ and $anag\_cli\_m\_var\_13$.

The variable $trf\_m\_\_var\_10$, see Figure 3.3, presents a high peak in 0, the frequency of churners (0.015) is higher respect to the non-churners (0.010), the two distributions are overlapping in the range $[0, 100]$, and it is possible to distinguish between them. The variable $anag\_cli\_f\_var\_11$,

see Figure 3.3, shows a peak in 0, the frequency of churners reaches 1.1, while the frequency of non-churners reaches 0.63. The churner distribution decreases following a long-tail distribution, while the non-churner distribution decreases more slowly. The variable $anag\_cli\_m\_var\_12$, see Figure 3.3, presents for the churner distribution a decreasing trend similar to a negative exponential, while the non-churners distribution presents a U-shape trend in the range $[-0.5, 1]$. he variable $trf\_m\_\_var\_13$, see Figure 3.3, exhibits a peak in 0, the frequency of the churners (0.26) is higher that the non-churners (0.08).

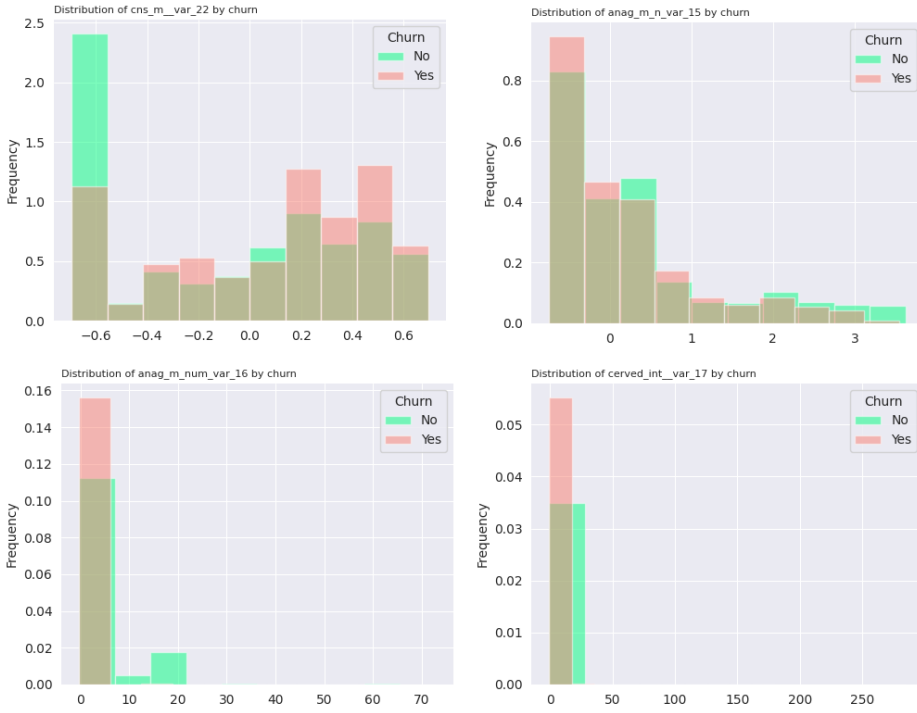


Figure 3.4.: Distribution of variables by churn: $cns\_m\_var\_22$, $anag\_cli\_m\_n\_var\_15$, $anag\_m\_num\_var\_16$ and $cerved\_int\_var\_17$.

The variable $cns\_m\_var\_22$, see Figure 3.4, shows a similar bell distribution, with higher values in the churning class. We can also identify a

distinct peak in -0.6 where the non-churners frequency is the highest. The frequency of churners (red) is higher in the range $[-0.4, 0.6]$. The variable *anag_cli_m_n_var_15*, see Figure 3.4, illustrates a peak in -1, the two distributions are overlapping and show a decreasing trend.The variable *anag_m_num_var_16*, see Figure 3.4, presents a peak in 0 in the churner class (i.e. 0.16) while non-churner class (i.e. 0.11) follows a U-shape distribution. The variable *cerved_int_var_17*, see Figure 3.4, shows a peak in 0 the churners distribution reach 0.05 while the non-churners reach 0.03, the two classes are overlapping but it is possible to distinguish between them.

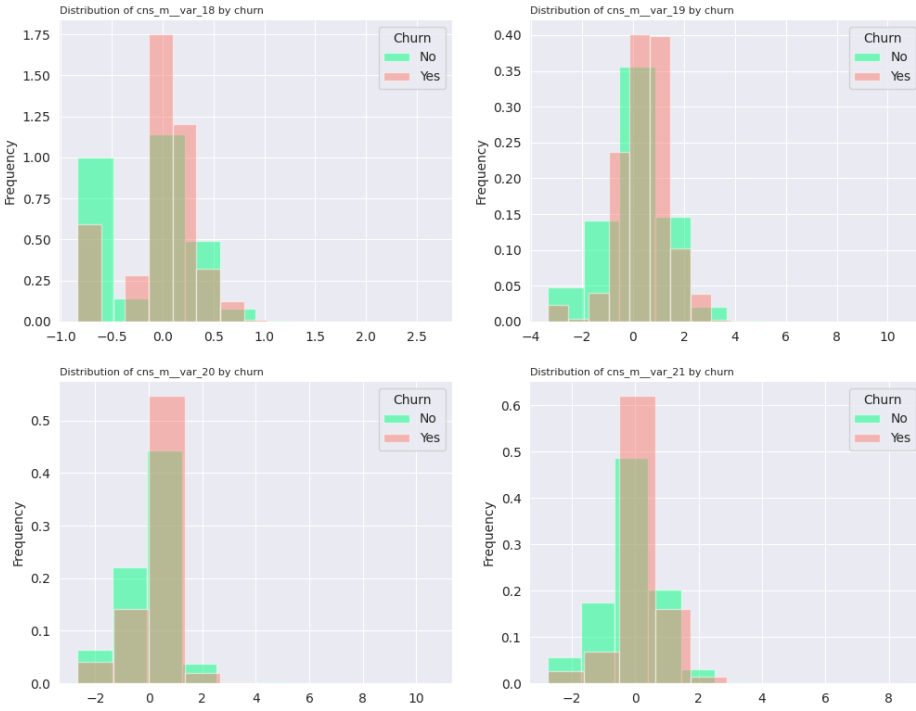


Figure 3.5.: Distribution of variables by churn: *cns_m_var_18*, *cns_m_var_19*, *cns_m_var_20*, and *cns_m_var_21*

The variable *cns_m_var_18*, see Figure 3.5, shows gaussian-like distribution of frequencies, the churner class outnumbers the non-churning one in

0, the frequency of churners (1.75) is higher respect to the non-churners (1.15). While we can distinguish a peak in -0.75 where the frequency of non-churner class (1.00) outnumbers the churning ones (0.53). The variable $cns\_m\_var\_19$, see Figure 3.5, shows gaussian-like distribution of frequencies where the churner class outnumbers the non-churning one in the range $[-4, 4]$. The variable $cns\_m\_var\_20$, see Figure 3.5, shows two left skewed-like distributions of frequencies, the churner class outnumbers the non-churning one. The variable $cns\_m\_var\_21$, see Figure 3.5, shows gaussian-like distribution of frequencies for the non-churners and a right skewed-like distribution for the churner class, which outnumbers the non-churning one.

A **more detailed description of the dataset can't be provided because it is Privacy related and subject to a non-disclosure-agreement**. The dataset has undergone some operations, which are described in the following to make it processable according to the procedures to be used.

The Table 3.1 provides some details on how this dataset (i.e. Churn) situates with respect to the public churn datasets. These datasets have diverse number of instances, number of features, and percentage of churners [41]. The number of **Features** range from 13 of SATO dataset to 230 of K2009. The number of Samples range from $1,401$ of DNS to $255,500$ of Churn dataset to Churn dataset. Finally the column $\frac{churn}{nonchurn}$ ranges from 0.03 of Orange Belgium dataset to 1 of SATO and DNS datasets.

3.3. Predictive Analysis

This section describes all the undertaken steps of **predictive analysis** applied to the real telecommunication dataset to achieve the **most signifi-**

Table 3.1.: Public datasets on churn prediction

Dataset	Features	Samples	$\frac{churn}{nonchurn}$	Citation
Churn	87	255,500	0.12	
Orange Belgium	178	11,896	0.03	[115]
K2009	230	50,000	0.08	[50]
UCI	20	5,000	0.16	[114]
TelE	19	190,776	0.19	[41]
TelC	20	7,043	0.37	[41]
SATO	13	2,000	1	[3]
DSN	15	1,401	1	[41]

cant features that will be used in the causal discovery and causal inference simulations.

As we know, the goal of a ML predictor is not (only) the **prediction**, e.g. any physician is not concerned with determining the precise probability of a heart attack, but rather on influencing subsequent decision-making, such as deciding whether a patient should take a particular drug [18].

Nowadays, the assessment and improvement in the prediction accuracy is achieved through the use of large size datasets, statistical insight and computational resources [18]. Such attitude relies on the **implicit assumption** that the more accurate the learner, the higher will be the reward of the selected action [18].

In this context, we have to be aware of several aspects: the probability estimation uncertainty's reduction; the error of prediction reduction; the use of huge amount of computational resources for training over-sized deep-learning models obtaining imperceptible accuracy improvements [18].

There is an important difference between the notion of **probability conditional on observation**, $Prob\{y \mid x = x\}$ and **probability conditional on manipulation**, $Prob\{y \mid do(x) = x\}$. These quantities are different and being overconfident (confusing them) can provide wrong outcomes that negatively affect the decision making [17].

Another aspect to consider is what can be the cost of a **false positive** or a **false negative**, and how misunderstanding these quantities can affect the company's strategies and the following incomes. An accurate prediction (whatever the network) does not imply an accurate understanding either good decision making. The data mining aim of "drowning" in data and starving for knowledge" can be reformulated as "drowning in associations and starving for causality" [17]. In general, the **idea** is to undertake a focus shift from a pure accuracy-driven (or obsessed) approach [18] to a **causal reasoning methodology**.

3.4. Problem formulation

The **problem formulation** is the preliminary and the most critical step of a learning procedure [17]. The **model designing process** consists of several phases. The choice of a particular **application domain**, e.g. the telecommunication domain. The selection of **a phenomenon of interest** that we need to study, e.g. the customer's abandon. Finally, making hypothesis that exists an **unknown dependency** (e.g. between the socio-economic status (SES) of the customer and the satisfaction (SAT) for the services provided) which is to be estimated from **experimental data** [17]. The **domain-specific knowledge** and **experience** have a key role in this phase, that's whyweasked all the specifics to the company's domain experts by unstructured interviews.

3.5. Experimental Design

As we know data is the "fuel" of Machine Learning, for this reason we have to rely on **data quality**: the resulting model would be ineffective if the data is not informative enough [17], see Chapter 3 for more details. Thus, the **experimental design** (i.e. the data observation and the data collection) is a crucial step [17]. As mentioned, **I had, to rely on the data collection capability of the company** (i.e. observational setting) [17], [39].

3.6. Classification Models

In the classification steps we compared the Decision Tree, Random Forest, Logistic Regression, K-Nearest Neighbours, Gradient Boost and Easy Ensemble classifiers described in the following sections.

3.6.1. Decision Tree Classifier

The decision tree learning is a supervised learning approach used in statistics, data mining and machine learning. Decision tree classifiers are widely used in various fields due to their simplicity, interpretability, and effectiveness in both classification and regression tasks. A classification **decision tree** is used as a predictive model to draw conclusions about a set of observations [64]. The goal of decision tree classifier is to create a model that predicts the value of a target variable based on several input variables [23]. The input space is divided into regions, and each region is associated with a specific class. The decision tree makes decisions by recursively splitting the data based on the values of input features. To make an example of classification tree we can consider a scalar outcome, Y , and a p-vector of

explanatory variables, X. By assuming that $Y \in K = \{1, 2, \ldots, k\}$. The subsets created by the splits are called **nodes**. The subsets which are not split are called **terminal nodes**. Each terminal nodes is assigned to one of the classes. A classification tree partitions the X-space and provides a predicted value, e.g. $argmax_s Pr(Y = s \mid X \in A_k)$ in each region [23].

3.6.2. Random Forest Classifier

Random Forests (RFs) [54] learn to build ensembles of decision trees that fit training data according to supervisions. In particular, RFs grow many classification trees using a probabilistic scheme [23]. They classify a new object from an input vector by putting the input vector down each of the trees in the forest. Each tree gives a classification (i.e. the tree votes for a class) [23]. Finally the forest selects the classification having the most votes over all the trees in the forest.

3.6.3. Logistic Regression Classifier

Logistic regression is a statistical model used for probabilistic prediction. A logistic regression classifier uses the predicted probability to produce a binary outcome (e.g. yes/no, true/false, success/failure, etc.) based on one or more independent variables (i.e. features or predictors). It assigns a probability to each point in the input space. The probability of an input vector $x = [x_1, \ldots, x_n]$ is calculated by applying a sigmoid function ($f(x) = \frac{1}{1+e^{-x}}$) to a linear combination of the values in x. Each variable X_i is assigned a coefficient w_i, and an intercept coefficient w_0 is also defined. The predicted probability is calculated in the Equation 3.1:

$$s(x) = \frac{1}{1 + \exp(-(w_0 + w_1 x_1 + \cdots + w_n x_n))} \tag{3.1}$$

The weights $w_0, \ldots, w_n$ can be learned by maximizing their log-likelihood given a set of learning examples using the gradient descent algorithm [100].

3.6.4. K-Nearest Neighbours Classifier

The k-nearest neighbor (KNN) algorithm is used for data classification. It assigns a label to each new data point based on the similarity (or proximity) of its k closest neighbors in a training dataset. In particular, a **k-nearest neighbours** classifier assigns the k nearest neighbours a weight $\frac{1}{k}$ and all others 0 weight [30]. This can be generalised to weighted nearest neighbour classifiers [108], [118], where the i_{th} nearest neighbour is assigned a weight w_{ni} where $\sum_{i=1}^{n} w_{ni} = 1$. KNN is a popular non-parametric, lazy learning, and instance-based method. It does not make any assumptions about the underlying data distribution. However, KNN can suffer from **high computational cost**, **curse of dimensionality**, and **sensitivity** to **noise** and **outliers**.

3.6.5. Gradient Boost Classifier

Gradient boosting techniques [55] combine predictions from multiple weak models [55], typically decision trees, to enhance the overall accuracy and generalization of the classification task. A gradient boost classifier is based on the concept of boosting, an **ensemble** method that iteratively adds new models to existing ones and assigns higher weights to instances misclassified by previous models. The gradient boost classifier is distinct from other boosting methods because it uses the negative gradient of a differentiable loss function, such as logistic loss for binary classification, as the pseudo-residuals to fit the new models. This approach enables it to optimize any arbitrary loss function that measures the discrepancy between the true labels and the predicted probabilities. The gradient boost classifier

is a versatile and potent technique that can handle various types of data and problems. However, it requires careful tuning of hyperparameters, such as the number and depth of trees, the learning rate, and the subsampling ratio.

3.6.6. Easy Ensemble Classifier

Easy Ensemble Classifier applies the idea of **ensemble** learning to deal with **imbalanced** data sets. It is a variant of the AdaBoost algorithm that creates balanced bootstrap samples [76] from the original data set and trains a weak classifier, usually a decision tree, on each sample. The final prediction is obtained by aggregating the votes of all the classifiers in the ensemble. The Easy Ensemble Classifier can enhance the classification performance of the minority class while maintaining reasonable accuracy for the majority class.

3.7. Data Preprocessing

We conducted **preprocessing** on the initial dataset to address outliers, handling missing or misshaped data, and then standardized the input data for the machine learning models.

Specific steps have been undertaken, as outlined in the procedural pipeline of the Figure 3.6.

The preprocessing pipeline is made up of the following steps: data cleaning and identification of -999 values (i.e. the customer's abandon is not happended yet), filling with NANs, Drop NANs higher than a certain threshold, NANs filling with median value, categorical values mapping, normalization and feature selection. The last step helps in reducing the complexity in

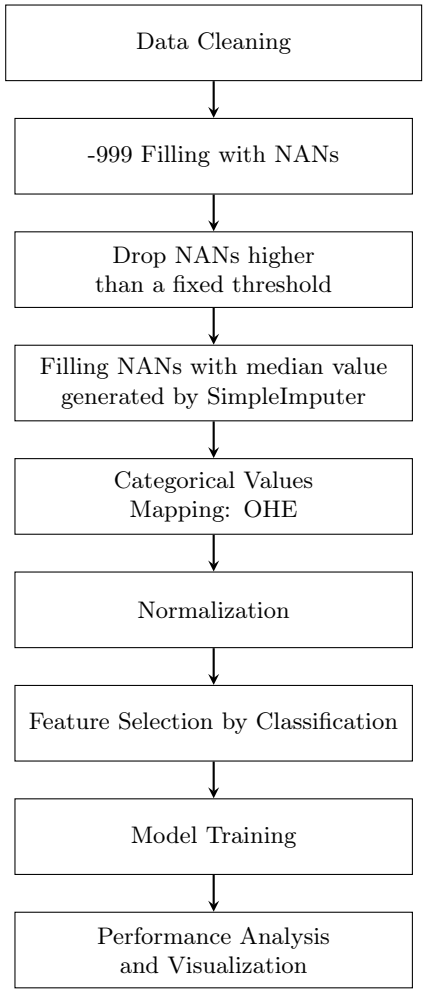


Figure 3.6.: The General Pipeline.

the analysis and also minimizes the memory needed for models to execute operations. A general scheme is described in Figure 3.6. And the pipeline implementation is shown in Algorithm 1.

Certain operations have been conducted on the dataset to eliminate uninformative features, to ensure compatibility with the machine learning procedures. The features chosen align with those used by TIM's systems

Algorithm 1: Preprocessing Pipeline implementation

```
pipeline = Pipeline([
("dataset-ridotto", ColumnTransformer([
    ("selected-features", "passthrough", selected_features)
], verbose_feature_names_out=False)),
("ct-999", ColumnTransformer([
    ("fill-999", SimpleImputer(
        missing_values=-999,
        fill_value=np.nan), fill_999)
], remainder="passthrough", verbose_feature_names_out=False)),
("drop-na-perc", DropNaPerc(threshold=70.0)),
("ct-filler", ColumnTransformer([
    ("nan_02p", Pipeline([
        ("imputer", SimpleImputer(strategy='median')),
        ("scaler", RobustScaler())
    ]), optional_column_selector(feature_numeriche)),
], remainder="passthrough", verbose_feature_names_out=False)),
("tree-feature-selection", SelectFromModel(
    ExtraTreesClassifier(n_estimators=50, random_state=42),
    max_features=60))
])
```

to address data compatibility concerns.

3.7.1. Data Cleaning and Missing data treatment

The real settings often happens to have some input values that are missing [17]. For a huge amount of data (i.e. big data) it is possible to ignore and drop the missing values higher than a certain threshold. The missing values are filled with NANs, in particular columns can contain NANs in different proportions. We found two features that contain approximately 0.2%, three features with 10%, eleven features with 20% and three features with 90%. In accordance with a threshold of 70%, all the columns with the NANs percentage higher than the given threshold are dropped.

The columns with 0.2%, 10% and 20% are filled with the median value of the column (which is more robust respect to mean value which is more prone to the outliers).

3.7.2. Categorical variables Encoding

I transformed the categorical data by using the One Hot Encoding (OHE), which is performed to prevent categorical values from being incorrectly mapped (e.g. considering the average on categorical values: if we have features which contain only values -1, 0 and 1, by applying the mean, we will question on the value 0.5 that we have to assign to which one of the two values: what does it correspond to?).

3.7.3. Normalization

Data are normalized to mean $\mu = 0$ and standard deviation $\sigma = 1$ to ensure that all features have a similar scale (Figure 3.1, Figure 3.2, Figure 3.3, Fig-

ure 3.4 and Figure 3.5), which is important for certain machine learning algorithms. When the features are on different scales, some algorithms might give more weight to features with larger magnitudes, potentially leading to biased or less accurate results. Normalizing the data helps in mitigating this issue, making the features comparable and preventing certain features from dominating the learning process.

The final dataset after the preprocessing step contains 27 features and 255500 rows.

3.7.4. Outliers Handling

Outliers are unusual data values that deviate from the majority of observations [17]. Typically, these outliers result from inaccuracies in measurement procedures, errors in storage, or malfunctions in coding [17]. The common strategies used to deal with outliers are: **outlier detection and removal** in the preprocessing step [38], [17] or by adopting robust methodologies [56], that are by design insensitive to outliers in the model identification level [17]. Here we used a LocalOutlierFactor estimator [24]. In particular, the local deviation of the density of a given sample with respect to its neighbors is measured by this method. The anomaly score is dependent on how isolated the object is with respect to the surrounding neighborhood, making it a local measure [24]. The locality is determined by k-nearest neighbors, and their distance is used to estimate the local density. By comparing the local density of a sample to the local densities of its neighbors, it is possible to identify outliers that have a substantially lower density than their neighbors. We found that the percentage of outliers though were negligible respect to the total number of customers, for this reasons we decided to maintain all the data.

3.7.5. Feature Selection through classification

This section illustrates the final step taken to achieve the most significant features (i.e. 27) that will be used in the causal discovery and causal inference simulations in Chapter 6 and Chapter 7.

The dataset contains historical information (see Section 3.2, *Esigenze* for further details) about the interaction between the customer and the company (e.g. if the customer calls the Customer Service for an issue to be solved, after solving the problem the operator of the contact center can actuate a marketing action: a new promotion, extra minutes calls, extra GB, etc. These data were available in the company's dataset but the ones that were provided were filtered by feature selection pipeline, because turned out to be not significant. these features were not used in the final analysis.

I used the $SelectFromModel()$ which is a meta-transformer for selecting features based on their importance weights [36]. In particular, an $ExtraTreesClassifier()$ meta estimator is trained on the dataset [43]. It fits a number of 50 randomized decision trees (i.e. extra-trees) on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The estimator provides a measure of the importance for each feature for the further analysis.

3.8. Model Training

The chapter is divided in two parts based on the fact that we first aim to assess the reproducibility within the company (i.e. the use case) and later within the literature scenario. In particular, the company provided dataset is already processed. The dataset was resampled to a final number of 10%

of churners, as shown in Table 3.1, while in the literature comparison we use a 50% ratio to balance the test set.

The models are trained by applying the train_test_split() function of scipy and a train_size of of 80% and test_size of 20%.

Table 3.2.: Decision Tree's parameters: param_grid

Argument Name	Values
criterion	['gini', 'entropy']
max_depth	[1, 10, 50]
min_samples_split	[2, 3, 5]

Table 3.3.: Random Forest's parameters: param_grid

Argument Name	Values
criterion	['gini']
n_estimators	[10, 50, 100]
max_depth	[1, 5, 10]
min_samples_split	[2, 3, 5]
min_samples_leaf	[1, 2, 3]

Table 3.4.: KNN's parameters: param_grid

Argument Name	Values
n_neighbors	[1, 5, 10]
weights	['uniform', 'distance']
p	[1, 2]

Grid Search Cross Validation (Grid_Search_CV) is applied to each model for hyperparameters tuning, with k-fold cross validation of 5 and optimizing

Table 3.5.: XGB's parameters: param_grid

Argument Name	Values
learning_rate	[0.1, 0.3, 0.5]
max_depth	[1, 5, 10]
min_child_weight	[1, 2]
subsample	[0.1, 0.5, 1]
n_estimators	[50, 100, 150]

Table 3.6.: Easy Ensemble's parameters: param_grid

Argument Name	Values
n_estimators	[10, 20, 50]
estimator	[AdaBoostClassifier(), LogisticRegression(max_iter=1000)]
sampling_strategy	[0.2, 0.5, 1]

for the accuracy, as illustrated in Table 3.2, Table 3.3, Table 3.4, Table 3.5 and Table 3.6 respectively for the models Decision Tree, Random Forest, K-Nearest Neighbors, XGBoost, Easy Ensamble.

3.9. Performance Analysis and Visualization: Company's case-study

In the following Table 3.7 the summary of the performance of the classification for each algorithm: accuracy, precision, F1-score and ROC curves (see Section 3.9.1) of each algorithm: Decision Tree, Random Forest, KNN, XGB and Easy Ensemble applied on the real dataset.

Table 3.7.: Summary of the metrics: accuracy, precision, recall, F1-score of the majority class "non-churner" i.e. 0 and the minority class "churner" i.e. 1 for Decision Tree, Random Forest, K-Nearest Neighbours, XGBoost and Easy Ensemble algorithms.

Algorithm	Accuracy	Class	Precision	Recall	F1-Score
DT	0.88	0	0.90	0.98	0.94
		1	0.53	0.20	0.29
RF	0.90	0	0.90	**1.00**	**0.95**
		1	**1.00**	0.15	0.27
KNN	**0.91**	0	0.90	0.98	0.94
		1	0.53	0.20	0.29
XGB	**0.91**	0	**0.91**	**1.00**	**0.95**
		1	0.94	**0.28**	**0.43**
Easy	0.88	0	0.89	0.99	0.94
		1	0.63	0.09	0.16

As shown in Table 3.7, the best performing algorithms in terms of accuracy of 0.91 are the KNN and XGB. RF achieves a the highest value in term of Precision for the class 1 (i.e. churner), while for the class 0 the XGB achieves 0.91. The Recall of class 0 is 1.0 both for the RF and XGB algorithms. It reaches for the XGB algorithm the value 0.28 for the class 1. Finally the F1-Score is 0.95 for the class 0 for the RF and XGB algorithms and 0.43 for class 1 for the XGB algorithm.

3.9.1. ROC curves

The Receiver operating characteristics (ROC) graphs help in visualizing, organizing and selecting classifiers based on their performance [37]. The ROC curves are insensitive to changes in the class distribution [37]. If the proportion of positive to negative instances in a test set changes, the ROC

curves will not change [37]. The ROC plots are two-dimensional graphs in which the true positives (tp) rate is plotted on the y-axis and the false positives (fp) rate is plotted on the x-axis. A ROC graph shows the relative tradeoffs between benefits (true positives) and costs (false positives).

The ROC curve of the DT, RF, KNN, XGBOOST and EasyEnsemble algorithms are illustrated respectively from Figure 3.7a to Figure 3.7e. The AUC score is presented in Table 3.8.

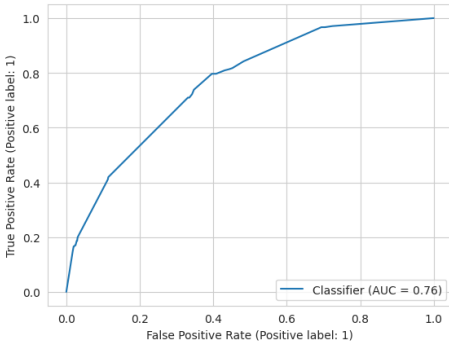
Table 3.8.: ROC curve of the DT, RF, KNN, XGBOOST and EasyEnsemble algorithms

Algorithm	AUC score	Figure
DT	0.76	3.7a
RF	0.81	3.7b
KNN	0.78	3.7c
XGB	**0.82**	3.7d
Easy	0.76	3.7e

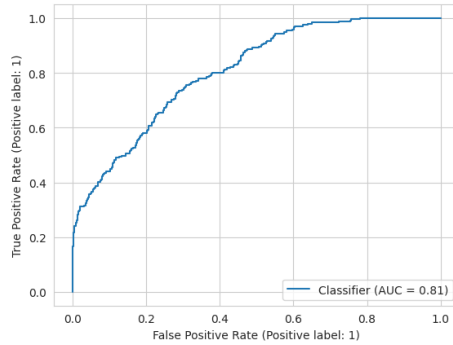
In Table 3.8, we can observe that the DT and Easy Ensemble algorithms obtain the worst result in terms of AUC score i.e. 0.76, while the XGBOOST algorithm obtains the best result of 0.82.

3.10. Performance Analysis and Visualization: Balanced Test set

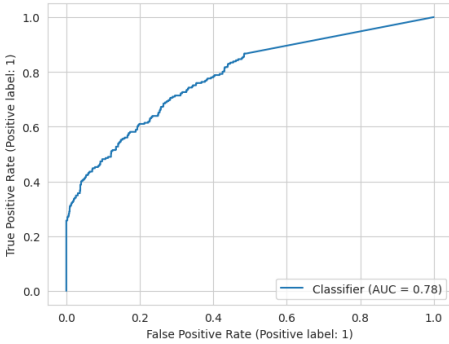
In the previous Section each algorithm presents very good accuracy values and very low Recall for class 1 (churners) instances. In the context of a literature scenario comparison, a low recall score suggests that a model fails to detect a significant number of actual churners (i.e. actual positives, identified churners, etc.). In this scenario, we can note that despite the company provided dataset was previously balanced, we still have a highly imbalanced dataset, where the negative class (non-churners) significantly outnumbers the positive class (churners), and models can achieve a low



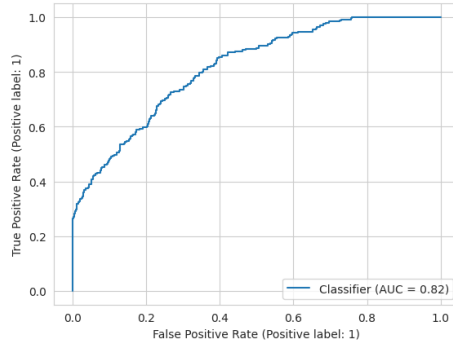
(a) ROC curve of DT classifier.



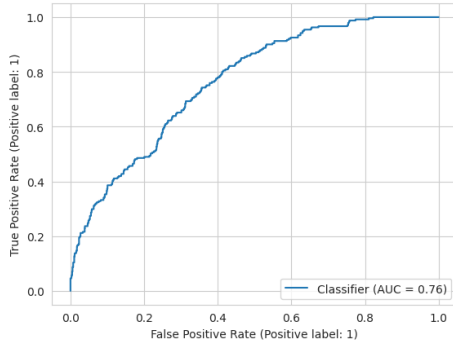
(b) ROC curve of RF classifier.



(c) ROC curve of KNN classifier.



(d) ROC curve of XGB classifier.

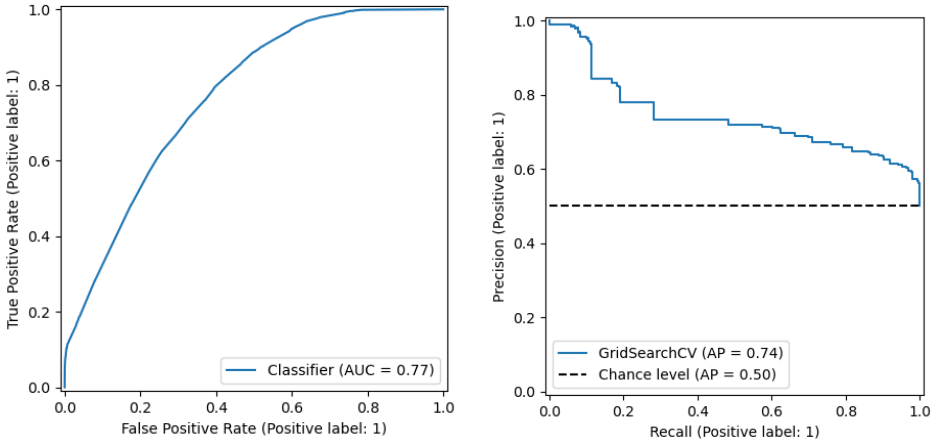


(e) ROC curve of Easy Ensemble classifier.

Figure 3.7.: The ROC curves of case-study.

FPR by simply predicting the majority class (non-churners) most of the time. As consequence models might appear to perform well in terms of

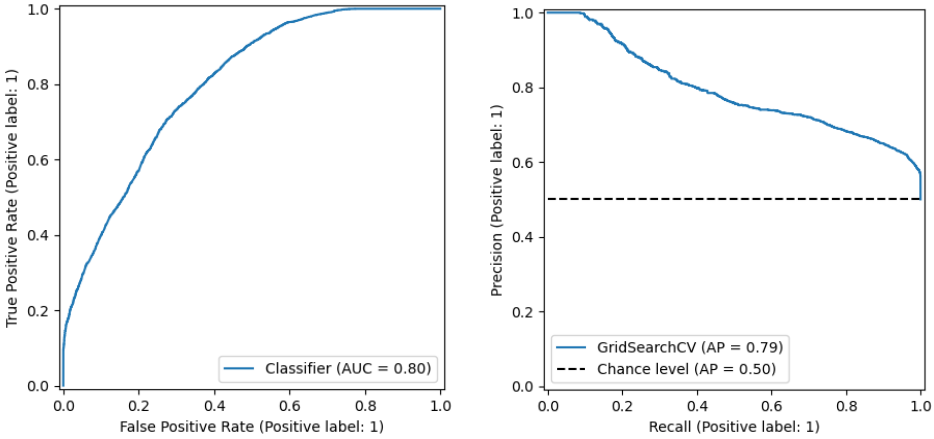
ROC AUC because a low FPR is maintained while potentially increasing the TPR, even if it's not effectively identifying the positive class. For these reasons, in the next section we investigate the performances on a **balanced** subset of the data, and inspect PR-AUC metric. As mentioned in Section 3.8, we use a 50% ratio to balance the training set.



(a) ROC curve of DT classifier.

(b) Precision-Recall curve of DT classifier.

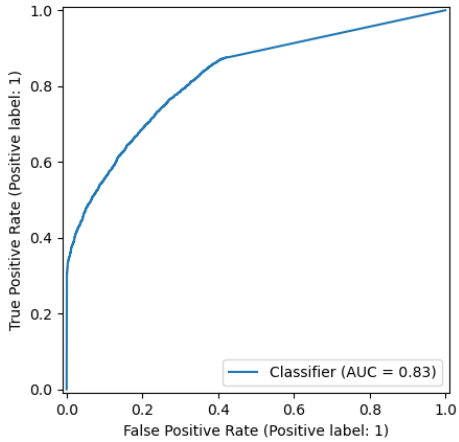
Figure 3.8.: ROC and Precision-Recall curves of DT model.



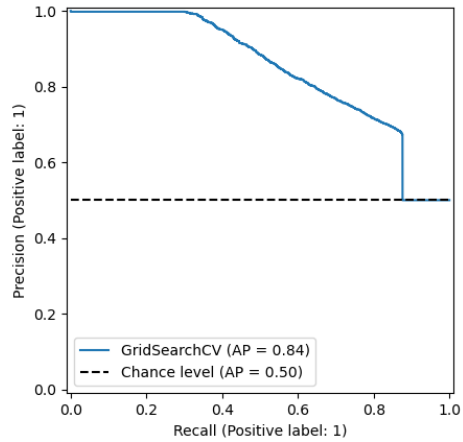
(a) ROC curve of RF classifier.

(b) Precision-Recall curve of RF classifier.

Figure 3.9.: ROC and Precision-Recall curves of RF model.

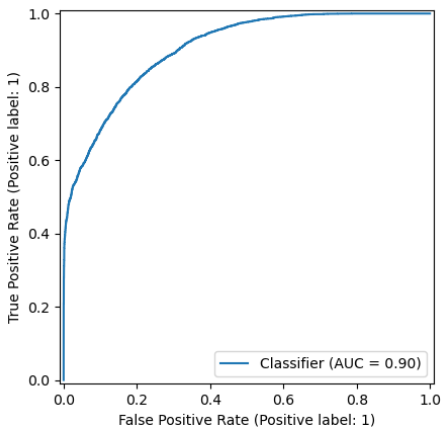


(a) ROC curve of KNN classifier.

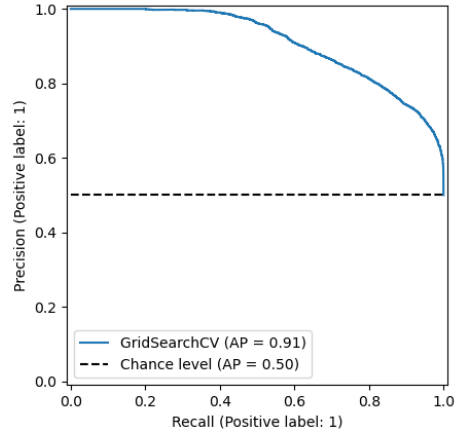


(b) Precision-Recall curve of KNN classifier.

Figure 3.10.: ROC and Precision-Recall curves of KNN model.



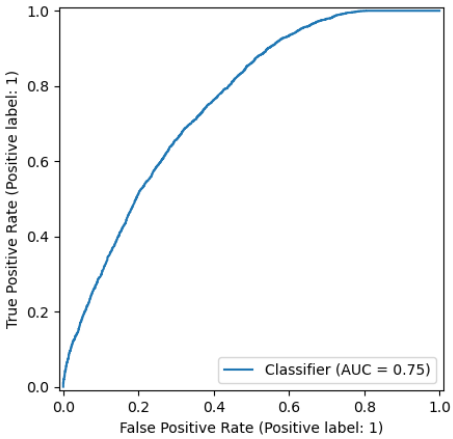
(a) ROC curve of XGB classifier.



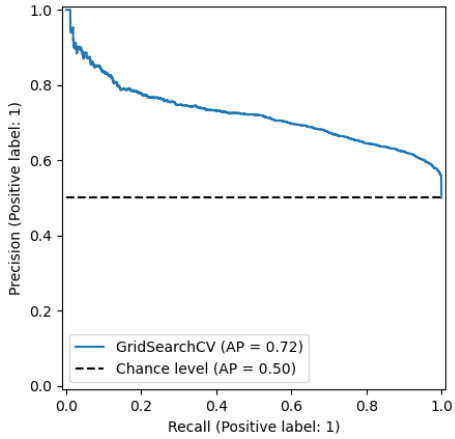
(b) Precision-Recall curve of XGB classifier.

Figure 3.11.: ROC and Precision-Recall curves of XGB model.

The ROC curve of balanced dataset used by the DT, RF, KNN, XGBOOST and EasyEnsemble algorithms are illustrated respectively from Figure 3.8 to Figure 3.12. The AUC score and the GridSearchCV(AP) are presented in Table 3.9, XGB is the best performing algorithm with AUC score of 0.90



(a) ROC curve of Easy Ensemble classifier.



(b) Precision-Recall curve of Easy Ensemble model.

Figure 3.12.: ROC and Precision-Recall curves of Easy Ensemble model.

and AP of 0.91.

Table 3.9.: ROC and Precision-Recall curves of the DT, RF, KNN, XGBOOST and EasyEnsemble algorithms

Algorithm	AUC score	AP
DT	0.77	0.74
RF	0.80	0.79
KNN	0.83	0.84
XGB	**0.90**	**0.91**
Easy	0.75	0.72

Synthesis of balanced test set performances

As we can see in Table 3.10 XGB is the best performing algorithm in terms of accuracy of 0.71, it achieves good results in terms of Precision i.e. 0.98 and Recall i.e. 0.43 for the class 1 (i.e. churner) and F1-score of 0.77 for the class 0. Despite RF achieves the highest value of Precision of 1.0 for class

Table 3.10.: Summary of metrics for the **balanced** dataset: accuracy, precision, recall, F1-score of the majority class "non-churner" i.e. 0 and the minority class "churner" i.e. 1 for Decision Tree, Random Forest, K-Nearest Neighbours, XGBoost and Easy Ensemble algorithms.

Algorithm	Accuracy	Class	Precision	Recall	F1-Score
DT	0.55	0	0.53	0.99	0.69
		1	0.95	0.10	0.19
RF	0.53	0	0.52	**1.00**	0.68
		1	**1.00**	0.06	0.12
KNN	0.68	0	0.61	0.99	0.76
		1	0.97	0.38	0.54
XGB	**0.71**	0	0.64	0.99	**0.77**
		1	**0.98**	0.43	0.60
Easy	0.68	0	**0.73**	0.57	0.64
		1	0.65	**0.79**	**0.71**

1 the accuracy is low 0.55. Easy Ensemble achieves the highest values in terms of Precision i.e. 0.73 for the class 0, both for Recall of 0.79 and F1-score of 0.71 for the class 1. Table 3.10 shows that DT and Easy Ensemble algorithms obtain the worst result in terms of ROC AUC score i.e. 0.77 and 0.75 respectively, while the XGBOOST algorithm obtains the best result of 0.90. In terms of Precision-Recall, the average precision (AP) both of DT and Easy Ensemble are the worst, i.e 0.74 and 0.72. The algorithm XGB presents the best AP value, i.e. 0.91.

Chapter 4.

Preliminary Concepts in Causality

4.1. Introduction

This chapter introduces the main definitions and concepts to define the **causal reasoning**'s main tasks: the **causal discovery** and the **causal inference**.

The causal part of the thesis is focused on discovery, one of the building blocks of churn prediction, such as counterfactuals [126] and the uplift models [116]. Moreover changing the value of a variable changes the target, and we do not have marketing actions examples in the data set that we performed the analysis on. It would be a broad topic, but out of the scope of this thesis.

Correlation indicates a general relationship, for example, two variables are correlated when show an increase or decrease in their trend [4]. The correlation between the temperature and total ice cream sales is positive: when it's hotter outside the total ice cream sales tends to be higher since more people buy ice cream when it's hot outside.

The terms **cause** and **effect** can also be used to describe **Causality** [87] where the cause is partially accountable for the effect, and the effect in turn is partially influenced by the cause [121]. Causality searches for potential cause–effect relationships in observational data [48]. For example, regular exercise leads to improved physical health. In this case, regular exercise is the cause and improved physical health is the effect.

Causal inference is the process of drawing a conclusion about a causal connection based on the conditions of the occurrence of an effect.

The main difference between **causal inference** and **inference of correlation** consists of that causal inference analyzes the **response** of the effect variable once the cause is **changed** [81]. It is well known that **"correlation does not imply causation"**.

Causal models are mathematical models representing "causal relationships within an individual system or population" [124]. **Causal relationships** entail the probabilistic (in)dependence of variables, and the effects of interventions (change on some variables) or hypothetical interventions, such as **counterfactual** claims [84].

4.2. Basic Definitions and Notations

In this section we define the basic notions, definitions and notations used throughout the thesis.

Independence: the variables X and Y, are **independent** when our knowledge about X does not change our knowledge about Y (and vice-versa) [80]. In terms of probability distributions, see Equation 4.1:

$$P(Y) = P(Y \mid X) \tag{4.1}$$

the **marginal probability** of Y is the same as the **conditional probability** of Y given X. The Equation 4.2:

$$P(X) = P(X \mid Y) \tag{4.2}$$

represents the **marginal probability** of X is the same as the **conditional probability** of X given Y.

The **independence**, see Equation 4.3 between two variables is that their **joint distribution factorizes** into the **product of the marginals** [77]:

$$P(X, Y) = P(X)P(Y) \tag{4.3}$$

In general, we can refer to Equation 4.4:

$$X \perp\!\!\!\perp Y \tag{4.4}$$

Conditional Independence: is the **generalization** of **independence**, in particular, in Equation 4.5: X and Y are **conditionally independent** given Z, when assuming that we observed Z, X does not give us any new information about Y [77].

$$P(X, Y \mid Z) = P(X \mid Z)P(Y \mid Z) \tag{4.5}$$

The joint distribution of X and Y (given Z) is factorized into a product of two simple conditionals ($X|Z$ and $Y|Z$) given the property $P(X, Y) = P(X)P(Y)$ [80]. In general, we can refer to Equation 4.6:

$$X \perp\!\!\!\perp Y \mid Z \tag{4.6}$$

Graphs Independence: the following notations are used in thesis:

- Independence in the **distribution**: $\perp\!\!\!\perp_P$
- Independence in the **graph**: $\perp\!\!\!\perp_G$

In particular, X and Y are independent in their **distributions**: $X \perp\!\!\!\perp_P Y$; conversely X and Y are independent in the **graph**: $X \perp\!\!\!\perp_G Y$ [80].

The key point is to find a **mapping** between the **independence** in the **graph** and in the **distribution** [80].

- **causal inference** determines the causal impact of one set of variables on another and relies on the fundamental connection between graphical and statistical characteristics (e.g. by using the information about the data-generating process, which can be encoded as a (causal) graph: i.e. confounders can be accurately controlled) [80].
- **causal discovery**, aims at recreating the **causal graph** from **observational** and/or **interventional** data [80].

Based on the previous definition of conditional independence: the **independence/dependence in a graph** can be expressed as a function of open paths between nodes [80].

In particular, two nodes (i.e. X and Y) in a graph can be considered **unconditionally** (or **marginally**) **independent** if there's no open direct or indirect connecting path [80].

Otherwise, X and Y, are **conditionally independent** given (a set of) node(s) Z when Z blocks all open paths that connect X and Y [80].

Basic Conditional Independence structures:

Chain: sequence of causally-linked events, i.e. $A \to B \to C$,

Fork: the edge between nodes A and B is reversed compared to the chain structure, i.e. $A \leftarrow B \to C$, B is the common cause of nodes A and C.

Collider: (or immorality) $A \to B \leftarrow C$, A and C are unconditionally independent, when control for B, they become dependent,

4.3. Graphs

In the following the graph definitions used in this thesis.

Directed Graphs: A directed graph [20] $G = \{V, E\}$ consists of a set $V = \{X_1 \dots X_n\}$ of vertices (also called nodes, actors, variables, etc.), each associated with a random variable [80]. And a set $\mathbb{E} \subseteq V \times V$ of directed edges (also called links, ties, marks [84], labels, etc.).

If we define an edge from a vertex i to another vertex j as $i \to j$ we can define V_i a **parent** of V_j and we call V_j a **child** of V_i [80]. We define any directed path $i \to \cdots \to j$, with $k \in \mathbb{Z}_0^+$ (non-negative) vertices between i and j, we call V_i an **ancestor** of V_j and we call V_j a **descendant** of V_i. For $k = 0$, parents are a special case of ancestors and children are a special case of descendants.

For a directed edge $e = (u, v) \in E$, v is called the head (source, parent, ancestor, etc.) of e, u is called its tail (target, child, descendant, etc.), and v is said to be adjacent to u.

A (s,t)-*path* defined in Equation 4.7, is a path from a source $s \in V$ to a target $t \in V$, i.e. is an alternating sequence of vertices and edges:

$$s, (s, v_1), v_1, (v_1, v_2), v_2, \ldots, (v_k, t), t \tag{4.7}$$

starting with s and ending with t. The **length** of an (s, t)-path is the number of edges it contains, and the **distance**, $dist(s,t)$, from s to t is defined as the minimum length of any (s,t)-path if one exists, and undefined otherwise (if s = t implies dist(s, t) = 0).

$\sigma(s,t)$ is the **geodesics**: number of shortest $(s,t) - paths$

$\sigma(s,t \mid v)$ be the number of shortest $(s,t) - paths$ passing through a vertex v other than s,t.

If $s = t$, then $\sigma(s,t) = 1$, and if $v \in \{s,t\}$, then $\sigma(s,t \mid v) = 0$. The shortest-path **betweenness** $c_B(v)$ of a vertex $v \in V$ is defined in Equation 4.8:

$$c_B(v) = \sum_{s,t \in V} \frac{\sigma(s,t \mid v)}{\sigma(s,t)} \tag{4.8}$$

where $0/0 = 0$ by convention. $P(Y \mid X)$ is defined as the **observational** conditional probability, which is the probability of how Y would change if X is observed (e.g. cross-sectional study).

Direct Acyclic Graphs A Direct Acyclic Graph (DAG) is a completely abstract mathematical object [99] sets of probability. DAGs are given two distinct functions [107] representing either **probability distributions** or **causal structures**.

The probability distributions is given by the **Markov condition**, mentioned in the following paragraphs, which (in DAGs) is equivalent to the **d-separation** (see next paragraphs) [85] (i.e. three disjoint sets of vertices in a directed graph, are d-separated whether a set of vertices Z blocks all connections of a certain type between X and Y in the graph G. If so, then X and Y are d-separated by Z in G).

Giving DAGs a **causal interpretation**, **d-separation** is the correct connection between a causal DAG and probability distributions [107].

Often exist many distinct DAGs that represent exactly the same set of independence relations, and thus the same set of distributions [107].

In general, we want both a **d-separation** procedure for any graph, and an **algorithm** that computes all the DAGs that represent a given set of independence relations.

Causal Markov condition, (i.e. stronger assumption than the **Markov condition**), states that when DAGs are interpreted causally the **Markov condition** and **d-separation** are the correct connection between **causal structure** and **probabilistic independence** [107]. DAGs causally interpreted are causal DAGs.

4.4. Causal Reasoning

Causality [92], [109], [87], can be defined as the influence by which an event contributes to the production of other events, in particular, one event or phenomenon known as the cause, brings about or leads to another event or phenomenon, known as the effect [84].

4.4.1. Causal Questions

In general causal questions such as: **what are the reasons why a customer can churn?**. **What is the impact of a marketing campaign?**. **What is the impact of a new product feature?** [102].

In this context, answering the question: "**What is the right model?**" is one of the biggest challenges is translating both the domain/expert knowledge into a causal graph (i.e. modeling assumptions) and the implications of these assumptions for causal identification and estimation [102]. Moreover, causal tasks frequently lack of the **ground truth** for comparison, unlike supervised machine learning models which can be assessed using separate test data, thus there is a shift in verification and testing of the practical aspects [102].

Therefore, **checking** these assumptions, the robustness and applying sensitivity tests is critical to gain confidence in results [102].

It is important to have clear in mind both the causal questions and the following tips:

- **Causal Discovery**: infers a causal graph given both the data and domain knowledge (if available) [16].
- **Causal Reasoning**: infers, given a causal graph, the **quantitative causal insights** [16]. In particular:
  - **Graph Validation**: validates the model assumptions, i.e. graph structure's rejection or types' validation of the assigned causal mechanisms [16].
  - **Attribution**: attributes observed effects to their root causes [16]. In particular, includes identification in the causal graph of the upstream nodes responsible for outcomes such as **outliers**, **causal influences**, or identification in the causal graph [16].
  - **Effect Estimation**: the causal effect of past treatment on a target variable of interest is estimated [16].

Broadly speaking, it is advisable to consider a system in its entirety, taking into account all its components and interactions, not just the effect of one variable on another [16]. In addition, we have to consider the models' **scalability** which depends on the algorithms' inference complexity, the causal graph's variables number, the sample size, and the causal graph' structure[16]. For instance, by modeling **cause-effect relationships** between all the **relevant variables** into a **causal graph**. Then, by learning **causal mechanisms** modeled by probabilistic models. And finally, use the trained probabilistic models to answer different causal questions [16].

4.4.2. Causal Discovery

Given a set of observational data, the aim of **causal discovery** [125] is to infer the **causal relationship** among the different variables in the dataset [84]. In particular, any relationship across involved variables is assumed, but it is inferred directly from a set of data [84]. Causal discovery analyzes and creates models that illustrate the relationships inherent in the data [84].

Causal discovery is still in the early stages of development within the scientific realm and has not yet achieved **robust** solutions for addressing general use cases [16].

4.4.3. Causal Inference

Every decision-makers is asking a **what-if** question. **Data-driven answers** to such questions require **understanding** the **causes** of an event and **how** to **take action** to improve future outcomes. It is reasonable to understand the sentence: "*... we are not about trying to magically pull causal rabbits out of a statistical hat.*", Scheines, Peter Spirtes and Clark Glymour [99]. In this context, this thesis aims at using this powerful tool to give hints and trace a direction for the company's technicians of the analytics department and for the business department.

Causal inference focuses on testing whether two variables are related and assessing the impact of one on the other [84]. A relationship among variables is assumed and a test and quantification of the actual relationship in the available data is performed [84]. Finally, causal inference aims to inspect the possible effects of altering a given system [121], [84].

Causal Inference aims to estimate the impact deriving from a change of a certain variable over an outcome of interest. In particular, trying to test whether two variables are related. And, trying to assess the impact of one on the other. Causal inference assumes a relationship among variables and tries to test and quantify the actual relationship in the available data. Causal inference aims to study the possible effects of altering a given system [121].

4.4.4. Causal Connectivity

The **causal Markov condition** and the **causal faithfulness condition** together imply a correspondence between the (conditional) independences in the probability distribution and the **causal connectivity** relations within the graph G [35]. Causal connectivity is defined both with the **d-separation** and the **d-connection**.

The **d-connection** is defined as the path p that connects X and Y, given

a conditioning set $C \subseteq V \setminus \{X, Y\}$ if and only if both all colliders on p are in C or have a descendent in C and no non-colliders of p are in C.

The **d-separation** or $\perp$, is defined if and only if d-connecting paths between X and Y are missing, in general there are paths between sets of nodes blocked by another set of nodes [80]. The **d-separation** allows to control the information's flow (i.e non-directional, which is closely related to the notions of correlation and confounding) in a graph [80]. Two nodes in a directed acyclic graph (DAG) G are d-separated when all paths between them are blocked (e.g. there is a collider on a path between them or if there's a fork or a chain that contains another variable that we control for, or a descendant of such a variable).

The (conditional) **d-separation** corresponds to (conditional) **probabilistic independence** given the validity of both **causal Markov** and the **causal faithfulness assumptions**: $X \perp Y \mid C \leftrightarrow X \perp\!\!\!\perp Y \mid C$

Definition of d-separation: is all about blocking paths between (sets of) nodes in a DAG. Paths in a graph can be blocked by using the fundamental logic of the tools that we already have in our toolbox, the three basic conditional independence structures – chains, forks, and colliders (or immoralities). The d-separation allows us to control the flow of information in a graph which is non-directed and strictly connected to correlation and confounding. The **d-separation effectively enables us to build estimands**.

D-separation

The **directional separation** or "**d-separation**" of two set of nodes in a DAG G can be defined as: two sets of nodes X and Y are **d-separated** by a third set of nodes Z, where X, Y and Z are pairwise disjoint, if Z is blocking all the paths between nodes in X and $X \perp\!\!\!\perp_G Y \mid Z$.

We first define the "d"-separation of a trail and then we will define the "d"-separation of two nodes in terms of that. Let P be a trail from node u to v. A trail is a loop-free, undirected (i.e. all edge directions are ignored) path between two nodes. Then P is said to be d-separated by a set of nodes Z if any of the following conditions holds: P contains (but does not need to be entirely) a directed chain, $u \cdots \leftarrow m \leftarrow \ldots v$ or $u \cdots \rightarrow m \rightarrow \ldots v$,

such that the middle node m is in Z, P contains a fork, $u \cdots \leftarrow m \rightarrow \ldots v$, such that the middle node m is in Z, or P contains an inverted fork (or collider), $u \cdots \rightarrow m \leftarrow \ldots v$ such that the middle node m is not in Z and no descendant of m is in Z. The nodes u and v are d-separated by Z if all trails between them are d-separated. If u and v are not d-separated, they are d-connected. X is a Bayesian network with respect to G if, for any two nodes u, v: $X_u \perp\!\!\!\perp X_v \mid X_Z$ where Z is a set which d-separates u and v. (The Markov blanket is the minimal set of nodes which d-separates node v from all other nodes).

Causal networks Although Bayesian networks are often used to represent causal relationships, this need not be the case: a directed edge from u to v does not require that X_v be causally dependent on X_u. This is demonstrated by the fact that Bayesian networks on the graphs are equivalent and impose the same conditional independence requirements: $a \leftarrow b \leftarrow c$ and $a \rightarrow b \rightarrow c$

The basic goal of **causal inference** is to estimate the causal effect of one set of variables on another [121], [84]. In particular, (in)dependence in the graph is a function of open paths between nodes in the graph, which is a mapping between graphical and statistical properties (i.e. mapping graphical (conditional) independencies into statistical (conditional) independencies). Moving from statistics and using a (causal) graph, which contains the information about the data-generating process, it is possible to accurately control for variables (e.g. confounders). On the other hand **causal discovery** aims at creating the causal graph from observational and/or interventional data, which translates from statistical to graphical properties [80].

Causal Markov condition The **Causal Markov condition** (i.e. causal Markov assumption or (local) Markov property) is expressed in Equation 4.9. The node, V_i, is independent (i.e. there are not open paths) of all other nodes in the **directed acyclic graph (DAG)**, $G(V, E)$, excluding the *descendants* and *parents* of this node, given its *parents* [80].

$$V_i \perp\!\!\!\perp_G V_j \mid PA(V_i) \forall_{j \neq i \in G(V,E) \setminus \{DE(Vi), PA(Vi)\}} \tag{4.9}$$

Global Markov property In general, controlling for the *parent* of a node removes the association between the two nodes. When the **causal Markov**

condition is respected, we can define the **global Markov property**, illustrated in Equation 4.10: X and Y are independent in the graph given Z, then they are also statistically independent given Z [80].

$$X \perp\!\!\!\perp_G Y \mid Z \Rightarrow X \perp\!\!\!\perp_P Y \mid Z \tag{4.10}$$

The global Markov property, the local Markov property, and the Markov factorization property are equivalent [80], [74], [92]. But we can have uncertainty regarding the graph's **true structure** (i.e. there can be multiple graphs that represent the same distribution) we want to retrieve because it can be ambiguous.

Faithfulness assumption The **Faithfulness** assumption states that if X and Y are independent in the distribution given Z, they are independent in the graph given Z [80], see in Equation 4.11.

$$X \perp\!\!\!\perp_P Y \mid Z \Rightarrow X \perp\!\!\!\perp_G Y \mid Z \tag{4.11}$$

Causal Minimality Condition The **causal minimality assumption** defines when a DAG G is minimal to distribution, P, if and only if G induces P, but no proper sub-graph of G induces P. In other words, if graph G induces P, removing any edge from G should result in a distribution that is different than P. Although the assumption is usually perceived as a form of Ockham's razor, its implications have practical significance for constraint-based causal discovery methods and their ability to recover correct causal structures [80].

Causal Sufficiency The **Causal Sufficiency** (i.e. **No hidden confounding**) [84], [80] asserts that all the shared causes of a pair of nodes are measured. While many techniques depend on this assumption, it may not always be feasible to fulfill. Due to this, some methods model the existence of **latent variables** [84]. In particular, for a pair of observed variables X and Y, all their common causes must also be observed in the data and modeled in a graph G [84]. In **Bayesian networks** the edges depict **conditional probabilities** based on **observations**, rather than implying **causal effects** through **interventions**. A variable X is considered to have a causal effect on Y when altering X results in a change in the distribution

of Y. The **Causal Bayesian networks** expands upon the factorization formula, in Equation 4.12, to account for do-interventions:

$$P(X \mid \mathbf{do}(W = w) = \prod_{X \in X \setminus W} P(X \mid Pa(X)) \mathbb{1}_{W=w} \tag{4.12}$$

Causal sufficiency and the causal Markov condition are related [107], [99]. The **Markov equivalence class (MEC)** of a set of DAGs,

$$D = \{G_0(V, E_0)\}, \ldots, G_n\{(V, E_n)\} \tag{4.13}$$

is Markov equivalent if and only if all DAGs have the same skeleton and the same set of colliders [117]. The **causal effect rule**: given a graph, G, and a set of variables, Pa (the (causal) parents of X), the causal effect of X on Y is given by the following Equation4.14 [89]:

$$P(Y = y \mid \mathbf{do}(X = x)) = \sum_{z} P(Y = y \mid X = x, Pa = z) P(Pa = z) \tag{4.14}$$

Estimates, Estimators, Estimands

Estimates can be imagined as our best guesses regarding some quantities of interest given (finite) data [80]. **Estimators** are "computational devices" or "procedures" that allow us to map between a given (finite) data sample and an estimate of interest [80]. For example, a customer wants to **estimate** how much time he/she spends calling in a week and takes note in a list of the calls' duration over the week. Therefore, it is possible to apply the **arithmetic mean**: the **estimate** of the true average calls duration. Instead it is possible to fit a distribution to this data or apply Bayesian statistics, i.e. compute a point-wise **estimate**. The arithmetic mean and the parameters of the distribution are called **Estimates**. We can define **estimator** the procedure (i.e. random forest, linear regression, neural network models) used to achieve the estimates.

Finally, in the causal inference context, an **estimand** is a quantity that we want to estimate.

If an **estimator** corresponds to **how**, an **estimand** corresponds to **what**.

For example, in Figure 4.1 we can consider the variables: bandwidth connection (BWC), customer service calls (CSC) and churn (CHU).

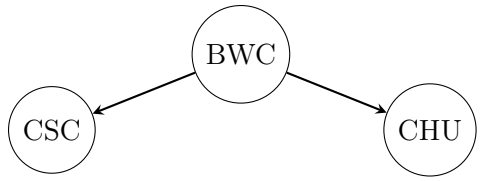


Figure 4.1.: Graphical representation of the example, see [80].

The CSC is not causally related to CHU, we can define it as **spurious** relationship which is a result of undirected (unconstrained) flow of information (in the graph). The **d-separation deconfounds** the relationship among these variables. The example can be as $CSC \sim CHU$. To estimate the causal effect of CSC on CHU, we have to understand what the change would be in CHU if we **intervene** on CSC as shown in Equation 4.15.

$$P(CHU = chu \mid do(CSC = csc)) \tag{4.15}$$

By applying a Naïve model, the estimand in Equation 4.16:

$$P(CHU = chu \mid do(CSC = csc)) = P(CHU = chu \mid CSC = csc) \tag{4.16}$$

which is incorrect because the relation between CSC and CHU is **spurious**. The correct **estimate** of CSC on CHU, is when we control for BWC, as in Equation 4.17:

$$CHU \sim CSC + BWC \tag{4.17}$$

And the $P(CHU = chu \mid do(CSC = csc))$ can be expressed as in Equation 4.18:

$$\sum_{bwc} P(CHU = chu \mid CSC = csc, BWC = bwc)P(BWC = bwc) \tag{4.18}$$

The previous formula can be written as shown in Equation 4.19:

$$P(CHU \mid do(CSC)) = \sum_{bwc} P(CHU \mid CSC, BWC)P(BWC) \tag{4.19}$$

The notation can be simplified using the **causal effect rule** [89]: given the graph G and the set of (causal) parents (Pa) of X, the causal effect of X on Y is shown in Equation 4.20: 4.20:

$$P(Y = y \mid do(X = x)) = \sum_{z} P(Y = y \mid X = x, Pa = z)P(Pa = z) \quad (4.20)$$

This example is coherent with the **d-separation** definition, if there's a **fork** between the two variables, X and Y, we need to **control** for the **middle node** (in the fork) to block a path between X and Y. BWC is the middle node in the **fork** between CSC and CHU. Thus, controlling for BWC **blocks** the **non-causal path** between CSC and CHU. This provides the model's correct **estimand** (in some cases, the same model can have multiple correct estimands).

Finding an **estimand** allows to compute **unbiased causal effects** from **observational data** [80]. Estimands are different from estimators and estimates [80]. Building a correct **estimand** shows both how it is related to a more general **causal effect rule** and the **links** between **estimands**, **d-separation**, and **confounding**.

The **do-operator** highlights the fact that we are focusing on **interventional** distribution rather than **observational** distribution. Sometimes the interventional and observational distributions might be the same: i.e. if the **true causal graph** is $X \rightarrow Y$, then $P(Y = y \mid do(X = x)) = P(Y = y \mid X = x)$. Once a **confounding** appears, we need to **adjust** for its **effects** by controlling for additional variables.

Back-Door Criterion:

Definition: The **back-door criterion** aims at estimating the causal effect of a treatment variable on an outcome while controlling for observed **confounding** factors that influence both the treatment and the outcome (i.e. blocking **spurious** paths between the treatment and the outcome nodes). It seeks to isolate the direct effect of the treatment.

Interpretation: The result of the back-door criterion answers, **"What is the causal effect of the treatment on the outcome while taking into account observed variables that may confound the relationship?"**. It represents the direct causal effect. A set of variables, Z, satisfies the **back-door criterion**, if considering a graph G, and a pair of variables, any node in Z is a descendant of X, and Z blocks all the paths, between

X and Y, including an arrow into X [89]. If exists a direct path from X to Y, the path can be direct or pass through other nodes ($X \to \cdots \to Y$). In the aforementioned example, all the paths are blocked between CSC and CHU that contains an arrow into CSC. The node BWC is not a descendant of CSC, and the condition that no new spurious paths are opened is respected.

If it is sufficient to control for one of the variables (X or Y) to obtain the **correct estimand** $X \to Y$, we can essentially estimate the causal effect of X on Y even if one of the variables remains **unobserved** [80].

Equivalent Estimand:

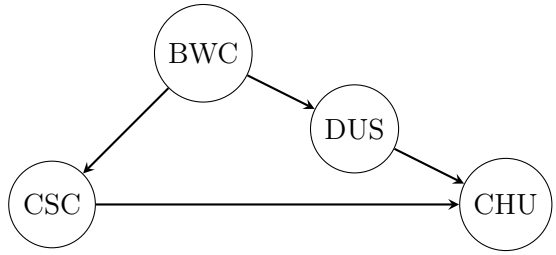


Figure 4.2.: The graph presents a confounding pattern, see [80].

In Figure 4.2, is shown an example of **confounding** pattern. To estimate the causal effect of CSC on CHU we need to look for the nodes we want to control for. By blocking all the paths that contain an arrow into CSC, we do not control for any descendants of CSC nor open any new paths. We can fulfill these conditions in three different ways by controlling for: BWC; DUS; both BWC and DUS. These options give us **different**, but **equivalent estimands**. $P(CHU \mid do(CSC))$ is equivalent to shown in Equation 4.21, Equation 4.22 and Equation 4.23:

$$\sum_{bwc} P(CHU \mid CSC, BWC)P(BWC) \tag{4.21}$$

$$\sum_{dus} P(CHU \mid CSC, DUS)P(DUS) \tag{4.22}$$

$$\sum_{bwc,dus} P(CHU \mid CSC, BWC, DUS)P(BWC)P(DUS) \tag{4.23}$$

In conclusion, it is possible to estimate the causal effect of CSC on CHU, even if there is one **unobserved** variable, only by controlling for **one** of the variables BWC or DUS, to obtain the correct estimand for $CSC \rightarrow CHU$.

In the case, we need to find two or more equivalent estimands, the computed estimates might differ slightly.

If the sample size is big enough then the differences should be negligible. But if there are big differences then this suggests that the estimand might be erroneous, or there is a lack of model convergence, or even errors in the model code. In Figure 4.3, the BWC variable is **unobserved**.

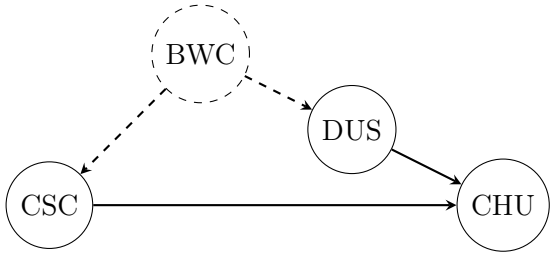


Figure 4.3.: The graph presents an unobserved variable and a confounding pattern, see [80].

Assuming that the global causal structure is known, if we consider BWC as an **unobserved** variable and two edges (i.e. $BWC \rightarrow DUS$ and $BWC \rightarrow CSC$, in dashed line) are all unobserved. We know nothing about BWC or what the functional form of BWC's influence on CSC or DUS could be. By assuming that BWC exists and has no other edges than the ones pointing to CSC and DUS.

The model's estimand, see Equation 4.24, becomes:

$$P(CHU \mid do(CSC)) = \sum_{dus} P(CHU \mid CSC, DUS)P(DUS) \tag{4.24}$$

It is worth noticing, that if the recording of a variable (e.g. CSC) is the most expensive part of data collection process. By the back-door criterion it is possible to skip this variable's recording.

In addition, we have to be sure that the causal structure is valid, if the

estimand is valid. For example, by changing the structure a bit by **adding** a **direct edge** from BWC to CHU, the preceding estimand would lose its validity, in fact we added a **brige**. Finally, by completely removing BWC and all its edges from the model the estimand would remain valid.

By controlling for DUS removes BWC's influence on CSC and CHU. Removing BWC from the graph, changes anything (up to noise) in the estimate of the relationship strength between CSC and CHU. In a graph with a removed node (i.e. BWC), controlling for another node (i.e. DUS) is irrelevant.

Front-Door Criterion:

Definition: The **front-door criterion** seeks to estimate the causal effect of a treatment variable on an outcome by using **mediators** a set of variables that lie on the causal path between the treatment and the outcome. It accounts for the **indirect effects** of the treatment and allows to obtain a valid causal estimand, when in (some) cases the back-door criterion fails [80] (i.e. the **confounder** is **unobserved** in the models is impossible to deconfound it).

Interpretation: The result of a **front-door criterion** provides an **estimate** of the treatment effect on the outcome through the specified **mediators**, while **controlling** for potential **confounders**. It helps answer the question: **"What is the effect of the treatment on the outcome that is mediated through these variables?"**. In particular, the influence of one variable X on another Y is mediated by a third variable, Z (or a set of variables, Z), when at least one path from X to Y goes through Z [80]. We can say that Z **fully mediates** the relationship between X and Y when the only path from X to Y goes through Z. If there are paths from X to Y that do not pass through Z, the **mediation** is **partial** [80].

In Figure 4.4, assuming that the DUS **fully mediates** the effects of CSC on CHU. We can assume that BWC only affects DUS **only indirectly** through CSC. If BWC would be able to influence DUS directly, **front-door** would be of no help. The assumption that BWC cannot directly change the DUS appears plausible.

The **back-door** and **front-door** criteria are **special cases** of a more general and complete framework: the **do-calculus** [87]. If there is an **identifiable causal effect** in a given DAG, G, it can be found using **do-calculus**'s

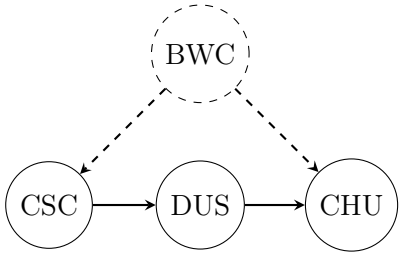


Figure 4.4.: The graph presents a mediator, see [80].

rules [105].

Instrumental Variable (IV) Estimand: **Definition**: The **IV estimand** belongs to the family of **deconfounding techniques** and is used when an **instrument variable** (i.e. Z) affects the treatment (X, which is a mediator) but is not directly related to the outcome (Y). In particular, the effect of interest is the causal effect of X on Y.

Interpretation: The result of an **IV estimand** provides an estimate of the causal effect of the treatment on the outcome. The instrument variable Z needs to meet the following three conditions [52]: Z, is associated with X (i.e. **association** rather than **causation**); Z, does not affect Y in any way except through X; no common causes of Z and Y [80].

The **IV estimand** answers the **"What is the causal effect of the treatment when we use this instrument variable to isolate the exogenous variation in the treatment?"**

In Figure 4.4 is shown an instrumental variable:

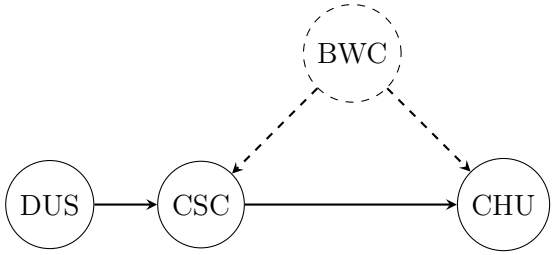


Figure 4.5.: The graph presents an instrumental variable, see [80].

the variable DUS is associated with CSC, it does not affect CHU other

than through CSC, and there are no common causes of DUS and CHU; therefore, DUS meets all of the criteria of being an **instrument**. Moreover, DAG contains the relationship between DUS and CSC is causal and direct, which allows us to approximate the exact causal effect (as opposed to just bounds). To calculate the causal effect of CSC on CHU in a linear case, all we need to do is fit two linear regression models and compute the ratio of their coefficients. The two models are as follows: $Y \sim Z$, $X \sim Z$; by dividing the coefficient for the first model by the coefficient from the second model.

4.5. Causal interpretation

The used techniques take statistical data and output sets of directed graphs. Everyone observed the process in action, but it is challenging to analyze the extra assumptions needed to attribute a causal meaning to the results. In other words, understanding how the systems would react to interventions based on the output is not easy [99].

4.6. Sensitivity Analysis

Based on the fact that in the real dataset provided by the TIM S.p.A we don't know which features are **confounders**. For example, in Figure 4.6, considering the underlying causal model: X is the treatment, Z is the set of **measured** features which are confounders, U is the set of **unmeasured** features which are confounders, and Y is the outcome. The **sensitivity analysis** helps in assessing the **robustness** of the **results** with respect to **unmeasured confounders** that can exist. For instance, if we can identify confounders that do not have an impact on the treatment X, if their effect is lower than a fixed threshold α, they can be considered less influential or may not substantially affect the results.

For example in Figure 4.7, if we consider a telecommunications company offering various monthly plans to its customers. A plan denoted as GLD (representing the **treatment**), is characterized as "gold" and is very rich in services but comes with a high cost. We define a **measured confounder** as Z. In this case, Z represents the customer's **socio-economic status**

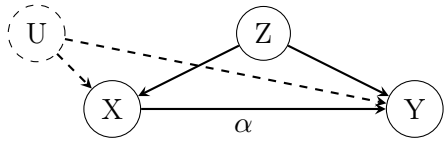


Figure 4.6.: Sensitivity analysis with measured (Z) and unmeasured (U) confounders, see [80].

(SES).

The main objective is to evaluate the impact of the premium monthly plan (GLD) on customer satisfaction. We can hypothesize that customers subscribing to the premium plan (GLD) will exhibit higher satisfaction levels (SAT). In particular, SES whic is a measured confounder, **might influence** both the choice of the "gold" plan (GLD) and the customer's **satisfaction** SAT, consequently unsatisfied customers can churn. However, there could be additional **unmeasured** factors i.e. U, such as **personal preferences** or **lifestyle**, **usage patterns**, to mention a few, that might impact the customer's satisfaction SAT within the "gold" plan.

For this reasons, the **sensitivity analysis** becomes a valuable tool and helps in **assessing** the **robustness** of **results** drawn from a study comparing the satisfaction levels (Y) of customers on the "gold" plan (GLD) by considering the potential unmeasured confounders (U). In particular, we can **measure the effect** and find a positive effect of GLD on customer satisfaction. The SES might not be the only factor influencing satisfaction, infact we have to **Identify Unmeasured Confounders** U that can affect both the choice of GLD and SAT. Then the for unmeasured confounders we set a threshold α (e.g. 15%): if an unmeasured variable's impact on SAT is less than 15%, it is less likely to affect the results. Futher, we perform the **Sensitivity Analysis**: by exploring hypothetical scenarios where unmeasured confounders have **different degrees of impact** on SAT.

Finally we **assess robustness**: if the results are **robust** it means that the conclusions about the positive effect of GLD on satisfaction hold even under varying scenarios of unmeasured confounders, it strengthens our confidence in the study's findings. By applying sensitivity analysis, we aim to ensure that our conclusion about the positive impact of GLD on satisfaction is robust and not prone to unmeasured confounders.

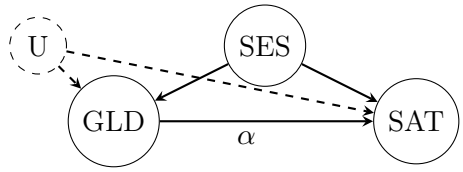


Figure 4.7.: Sensitivity analysis with measured (SES) and unmeasured (U) confounders, see [80].

4.6.1. Sensitivity assumption and problem statement

Assuming that n units $(X_i, Y_i(0), Y_i(1), Z_i, U_i)$ are sampled **independently** from the distribution P_{full}.

We define the vector of **confounders**: $(X_i, U_i) \in \mathbb{R}^d \times \mathbb{R}^k$; the real-valued potential outcomes: $(Y_i(0), Y_i(1))$ and the binary treatment $Z_i \in \{0, 1\}$.

Only the vector (X_i, Y_i, Z_i), where $Y_i = Y_i(Z_i)$, is **observed**. The **confounders** U_i are also **never observed**, and their dimension k may be unknown.

The P to denotes the distribution of the **observables**, $(X_i, Z_i, Y_i) \sim P$ are **independent** and **identically distributed**. We omit the subscript i to denote a generic draw from P_{full} or P.

The use of **observational data** helps to draw **inferences** about the **average treatment effect**, in Equation 4.25

$$\psi_{ATE}(P_{full}) = \mathbb{E}_{P_{full}}[Y(1) - Y(0)] \tag{4.25}$$

the counterfactual means, see Equation 4.26:

$$\psi_z(P_{full}) = \mathbb{E}_{P_{full}}[Y(z)], z \in \{0, 1\} \tag{4.26}$$

The **unmeasured confounders** [34], are not functions of P alone (i.e. point-identified) and cannot be consistently estimated. Nevertheless, they may be **bounded** by imposing on the unmeasured confounders U_i the assumption that measuring unobserved confounders could not change the odds of treatment by more than the factor $\Lambda \geq 1$. Formally, this restricts the distribution P_{full} to the **marginal sensitivity model, MSM** [110].

Definition 1 (Marginal sensitivity model).

Given an observed-data distribution P and an odds ratio bound $\Lambda \geq 1$, the marginal sensitivity model MSM (P, Λ) is the set of distributions P_{full} on $(X, Y(0), Y(1), Z, U)$ with $U \in \mathbb{R}^k$ for some k satisfying the following properties, see the Equation 4.27 for **Unconfoundedness** and Equation 4.28 for the **Bounded odds ratio**, where measuring U does not change the odds of treatment by more than a factor of Λ:

$$(Y(0), Y(1))Z \mid (X, U) under P_{full} \tag{4.27}$$

$$\Lambda^{-1} \leq P_{full}(Z = 1 \mid X, U) \frac{\frac{P_{full}(Z=1|X,U)}{P_{full}(Z=0|X,U)}}{\frac{P_{full}(Z=1|X)}{P_{full}(Z=0|X)}} \leq \Lambda \tag{4.28}$$

4.6.2. Causal effect and ATE

Individual Treatment Effect models the effect of treatment on the unit i: $ITE_i = Y_{1,i} - Y_{0,i}$. The **Average Treatment Effect** is the expectation of the ITEs, as shown in Equation 4.29:

$$ATE = E[Y_1 - Y_0] \tag{4.29}$$

Assuming a binary treatment variable T in Equation 4.30:

$$ATE = E[Y \mid \mathbf{do}(T = 1)] - E[Y \mid \mathbf{do}(T = 0) \tag{4.30}$$

Average Treatment Effect on Treated is the ATE restricted on the treated units is depicted in Equation 4.31:

$$ATT = E[Y_1 - Y_0 \mid \mathbf{do}(T = 1)] \tag{4.31}$$

Considering any experimental setup we denote that: "correlation does not imply causation", which means that **causal relations** can be identified. For example, in a randomized controlled trial, each individual in the experiment is randomly assigned to either the treatment or the control group and

the effect on certain variable is measured [35]. It is worth to understand the underlying causal relations which can support predictions about how a system will behave when a certain intervention is performed [35]. **Causal Discovery** focuses on obtaining **causal knowledge** directly from **observational data** trying to infer the causal relationship across the different variables in the dataset. Causal discovery does not assume any relationship among involved variables; rather, they are inferred directly from a set of data. Causal discovery is responsible for analyzing and creating models that illustrate the relationships inherent in the data. In general we would like to understand "what is the source of causal graphs in the real world? [80]. For this reason, it worth to distinguish the different ways of obtaining causal graphs. For instance, by both the **Causal discovery** and **causal structure learning** used to uncover causal structure from **observational** or **interventional** data. Secondly, by the **Expert knowledge** that helps in defining or disambiguate the causal relations between two or more variables [80]. Finally, by a **combination of both**, some causal discovery algorithms can easily incorporate the expert knowledge as a priority (e.g. by freezing certain edges in the graph or by suggesting their existence or a certain direction) [80].

4.7. Causal Discovery Methods

The following Section shows the Taxonomy of Causal Discovery Methods in particular:

Constraint-based methods: or independence-based causal discovery methods focus on the graph's independencies. This family includes algorithms such as: PC-algorithm (PC) [75], Inductive Causation (IC), FCI Algorithm (Fast Causal Inference) [27], etc.

Score-based methods: focus on the Bayesian network approach. The development of the **Greedy Equivalence Search** (GES) algorithm [26] is one of the classic examples of a score-based method, that are applied to restrict the number of candidates. In particular, these algorithms assign a relevance score to candidate graphs through some adjustment measures (i.e. Bayesian Information Criterion (BIC)). These algorithms are computationally expensive since they have to enumerate (and score) every possible graph among the given variables.

Functional causal discovery: it is inspired by **Independent Component Analysis (ICA)** [57] and developed by [104]. Algorithms in this family leverage various aspects of functional forms of the relationships between variables in order to determine the causal direction. A classic example representing this family is the Linear Non-Gaussian Acyclic Model (LiNGAM) algorithm [103], DirectLingam [104]. LiNGAM, the non-linear additive noise (ANM) model, and the post-nonlinear (PNL) causal model.

Gradient-based methods: the idea is treating the graph space search as a continuous optimization problem. Typically Gradient Descent is used for optimization purposes: NOTEARS algorithm [130].

4.7.1. PC Algorithm

The PC algorithm [109] is a classic causal discovery algorithm which is based on conditional independence tests [80]. It recovers structural information from observational data and leverages the independence structure to recover the graph.

The PC algorithm steps are presented in the following: it starts with a fully connected graph, then the edges between unconditionally independent variables are deleted. Further, it iterates over all remaining variable pairs (A, B). If $A \perp\!\!\!\perp B \mid C$, given the conditioning set C, which contains only one variable, the edge between A and B is deleted. This step is iterated by increasing by 1 the size of C for all the remaining pairs.

We denote with the symbol " an undirected edge and A–B–C given A and C not adjacent. Edges are oriented as $A \rightarrow B \leftarrow C$ whenever $A \not\perp\!\!\!\perp C|B$ (i.e. **collider**). Edges are oriented as $A \rightarrow B$–C, where A and C are not adjacent, orient the edge between B and C as $B \rightarrow C$ (i.e. **orientation propagation**) [109].

PC-algorithm is computationally feasible for sparse problems with many nodes (i.e. variables) achieving high computational efficiency as a function of sparseness of the true underlying DAG [68]. PC-algorithm is insensitive to the choice of α, its single tuning parameter [68].

The **limitations** of Constraint-based algorithms [17]: two graphs that are **I-equivalent** (if they have the same associated set of independencies) can-

not be distinguished by constraint-based methods without resorting to manipulative experimentation or temporal information [17]. The use of **conditional independence** tests requires an assumption on the dependence (e.g. linear (assuming a Normal distribution) or nonlinear). In the nonlinear case, it may be particularly expensive to have recourse to conditional tests. **Finite-sample error propagation**, as for the sequential nature of these methods, suffers from an error propagation (false positive) which is difficult to monitor and control [17]. The **asymptotic results of correctness**: if the conditional independence decisions are correct in the large sample limit, the PC algorithm converges to the true Markov Equivalence Class in the large sample limit, assuming i.i.d. samples and the Markov, Faithfulness, Sufficiency assumptions [17]. Finally the **dimensionality** problem: the exponential complexity nature of the algorithm makes those algorithms inadequate in large dimensional settings [17].

4.7.2. GES Algorithm

GES is a greedy algorithm that searches through equivalence classes of DAGs [26]. Greedy search (generally) proceeds at each step by evaluating each neighbor of the current state, moving to the highest scoring one if it improves the score. The set of neighbors of each state in the search defines the connectivity of the search space. GES consists of two phases. In the first phase, a greedy search is performed over equivalence classes using a given connectivity between equivalence classes. Once a local maximum is reached, a second phase starts from the previous local maximum using a second connectivity. This equivalence class is returned as the solution when the second phase reaches a local maximum. We denote $\boldsymbol{\varepsilon}^{+}(\varepsilon)$ to denote the neighbors of state ε during the first phase of GES. This corresponds to say that an equivalence class ε' is in $\boldsymbol{\varepsilon}^{+}(\varepsilon)$ if and only if there is some DAG $G \in \varepsilon$ to which we can add a single edge that results in a DAG $G \in \varepsilon'$. In other words, G is any DAG in ε and G' is any DAG in ε'. Then $\varepsilon' \in \boldsymbol{\varepsilon}^{+}(\varepsilon)$ if and only if there exists a sequence of covered edge reversals followed by a single edge addition followed by another sequence of covered edge reversals that transform G into G'. We use $\boldsymbol{\varepsilon}^{-}(\varepsilon)$ to denote the neighbors of state E during the second phase of GES. The definition of $\boldsymbol{\varepsilon}^{-}(\varepsilon)$ is completely analogous to that of $\boldsymbol{\varepsilon}^{+}(\varepsilon)$, and contains equivalence classes that are obtained by deleting a single edge from DAGs in ε, [26]. In Figure 4.8 shows a DAG. Figure 4.9 illustrates all the members $\boldsymbol{\varepsilon} = \varepsilon(G)$. Figure 4.10 shows all the DAGs reachable by a single edge addition to a member of ε. Finally, in Figure

4.11 we show all DAGs reachable by a single edge deletion from a member of ε. The union of the two corresponding equivalence classes constitutes $\boldsymbol{\varepsilon}^{-}(\varepsilon)$ [26].

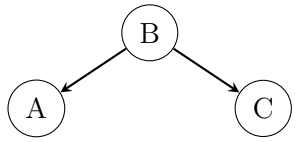


Figure 4.8.: DAG G, see [26].

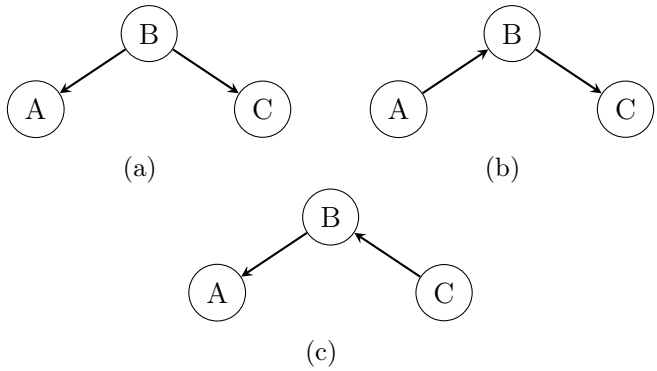


Figure 4.9.: The $\boldsymbol{\varepsilon} = \varepsilon(G)$

GES is optimal in the **limit** of large datasets [26]. The feasibility of applying any search algorithm depends on the complexity of both the algorithm and the search space. The use of greedy search algorithm over edges shows that the total number of search states visited by GES in a domain of n variables can never exceed $n(n-1)$ and the number of states visited generally grows linearly with n. Another problem is that the complexity of the search space: for each state visited by the algorithm we need to generate and evaluate all states that are reachable by the application of a single operator. In particular, if the number of the neighbor states grows very large, the evaluation time increases largely. Another problem, is that learning the optimal structure using the Bayesian scoring criterion is **NP-hard** [26]; this means that in the **worst case**, the connectivity of the search space that the algorithm encounters is a problem [26]. In real-world problems the portion of the search space traversed by GES is sparse [26]. A possible solution provided by Chickering et al. is that the worst-case scenario will not occur, and that for real-world problems the portion of the search space

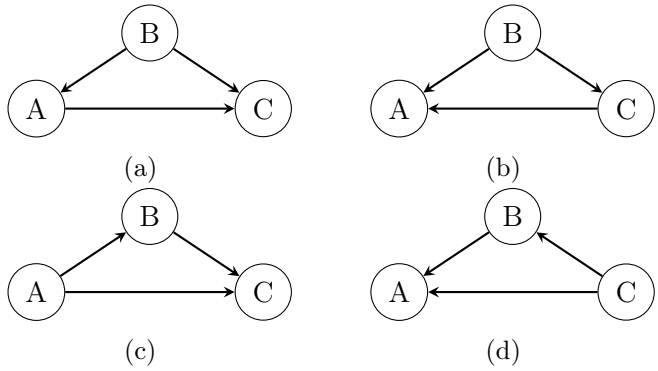


Figure 4.10.: The single member of $\boldsymbol{\varepsilon}^{+}(\varepsilon)$

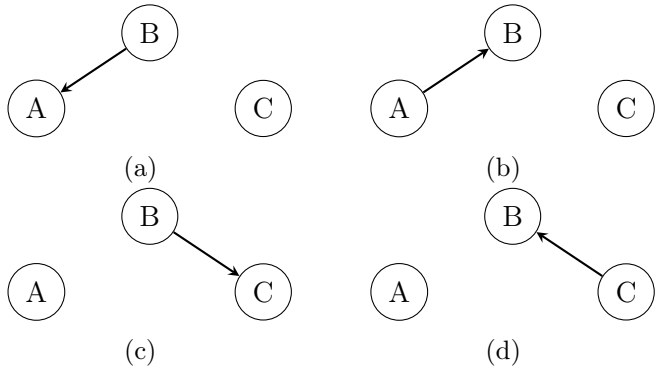


Figure 4.11.: The two members of $\boldsymbol{\varepsilon}^{-}(\varepsilon)$

traversed by GES will be sparse. Due to the fact that portions of the search space can be too dense to search efficiently, it may be beneficial to consider only a heuristically-selected subset of the candidate neighbors at each step. However, this comes at the cost of losing the large-sample optimality guarantee.

GES converts the PDAGs (Partially Direct Acyclic Graphs) to completed PDAGs takes time $O(|E| * k^2)$ in the worst case—where $|E|$ is the number of edges in the PDAG and k is the maximum number of parents per node—which could potentially be a problem for domains with a large number of variables.

4.7.3. DirectLINGAM

The DirectLINGAM is a **function-based** causal discovery method [104]. It is based on the Linear Non-Gaussian Acyclic Model (LiNGAM) method which is a non-Gaussian variant of Structural equation models (SEM) and Bayesian networks (BN). LiNGAM estimates a causal ordering and connection strengths based on non-Gaussianity. A problem arises when, most ICA algorithms including FastICA [57] and gradient-based algorithms [26] may not converge to a correct solution in a finite number of steps if the initially guessed state is badly chosen [53] or if the step size is not suitably selected for those gradient-based methods[103]. The appropriate selection of parameters is tricky.

DirectLINGAM converges to the right solution in a fixed number of steps i.e.equal to the number of variables if the data strictly respect the hypothesis of: Linearity, non-Gaussianity, Acyclicity, absence of hidden Confounders and Causal Sufficiency. Additionally for small datasets, the algorithms' permutations are **not scale-invariant** it means a different or wrong order of variables depending on scales or standard deviations of variables, especially if they have a wide range of scales.

DirectLINGAM aims to estimate a causal ordering and the connection strengths in the LiNGAM [104] .

The complexity of the ICA-LiNGAM is presumed to be $O(np^3 + p^4)$ [103]. However, in real-world scenario these strict assumptions could not be satisfied. DirectLiNGAM requires no prior knowledge on the structure. In the case of some prior knowledge is provided (i.e. the number of causal orders and connection strengths to be estimated gets smaller), DirectLiNGAM is more efficient in learning. The cost of DirectLiNGAM would be larger than that of ICA-LiNGAM especially when the dateset size is large.

4.7.4. NOTEARS

The NOTEARS algorithm is *Non-combinatorial Optimization via Trace Exponential and Augmented lagRangian for Structure* learning. This method formulates the structure learning problem as a continuous constrained optimization task, leveraging an algebraic characterization of DAGs [130], by adopting the least squares objective, which is related to but does not

directly maximize the data likelihood. NOTEARS learns DAGs as a continuous optimization task [130] leveraging an algebraic characterization of DAGs via exponential function. It adopts a regression-based objective, i.e., the least squares loss, with $l1$ penalty and a hard DAG constraint. The constrained optimization problem is then solved using the augmented Lagrangian method, followed by a thresholding step on the estimated edge weights [130]. The main advantage of NOTEARS is smooth, global search (delegated to standard numerical solvers), as opposed to combinatorial, local search [130]. It has numerical difficulties and illconditioning issues as the penalty coefficient to reach infinity to enforce acyclicity [83]. Notears performs well in small-scale experiments solving the original problem [130]. It returns a local minimizer, and the obtained solution often is very close to the global minimizer. On real datasets NOTEARS shows advantages over existing equivalence search algorithms. By performing global updates (e.g. all parameters at once) instead of local updates (e.g. one edge at a time) in each iteration, it avoids relying on assumptions about the local structure of the graph. A limit is that the equality constrained program ($min_{W \in R^{d \times d} F(W)}$, subject to $h(W) = 0$ and that $W : h(W) = 0$ is a nonconvex constraint [130]. Important advantage of NOTEARS is smooth, global search, as opposed to combinatorial, local search [130]. NOTEARS is specifically developed for linear DAGs, and has been extended to handle nonlinear cases via neural networks [83].Furthermore, the search is delegated to standard numerical solvers and the evaluation of the adjacency matrix is exponential i.e. $O(d^3)$, the computational complexity of this method is cubic in the number of nodes [130], although the constant is small for sparse matrices (used). Finally, NOTEARS outperforms existing methods (PC, GES, LiNGAM, etc.) when the in-degree (i.e the number of edges in input in a certain node) is large [130].

4.7.5. GOLEM

This section describes a continuous unconstrained optimization method, called *Gradient-based Optimization of dag-penalized Likelihood for learning linEar dag Models* (GOLEM) [83]. It is likelihood-based with soft sparsity and DAG constraints asymptotically returns a DAG equivalent to the ground truth DAG, under mild assumptions [83]. Unlike NOTEARS [130] that requires a hard DAG constraint, GOLEM treat it as a soft constraint, based on the hypothesis that ground truth is acyclic (under mild assumptions) the estimated graph (asymptotically) is a DAG [83]. An issue arises

due to finite samples and nonconvexity [83], in particular the local solution obtained may contain several edges with weights near zero and may not be exactly acyclic. The the computational complexity is $O(d^3)$. GOLEM is robust to weight scaling (i.e. Structural Hamming Distance (SHD), Structural Intervention Distance (SID) are consistently low). By using a GPU, the computational acceleration enables to easily handle thousands of nodes.

4.7.6. DoWhy

DoWhy for causal inference uses graph-based criteria, do-calculus and supports explicit modeling and testing of causal assumptions [102] and the identification of non-parametric causal effect [102]. DoWhy unifies **graphical causal models** (GCM) [87], [16] and **potential outcomes** (PO) [60], **graph-based criteria** and **do-calculus** for **modeling assumptions** and **identifying** a **non-parametric causal effect**. For the effect estimation (i.e. **effect inference**) follows a four step procedure: creating a causal model from the data and a given graph (i.e. models a **causal model**); identifying the causal effect and returns target estimands (i.e. **identification**); estimating the target estimand using a statistical method (i.e. **estimation**) and refuting the obtained estimate using multiple robustness checks (i.e. **refutation**).

Modeling Data alone is not enough for causal inference, we need **domain knowledge** and **assumptions**. Since there is **no ground-truth test dataset available** that an estimate can be compared to, causal inference requires a series of principled steps to achieve a good estimator. The modeling phase is quite sensitive: it encodes domain knowledge into a causal model (i.e. graph).

The input assumptions largely affect the final outcome of a causal inference analysis. The causal effect's estimation, involves specifying two types of variables: **Confounders** (i.e. common_causes): variables that cause both the action (i.e. treatment) and the outcome (i.e. target). Any observed correlation between the treatment and the target may simply be due to the confounder variables, and not due to any causal relationship between them. **Instrumental Variables** (instruments): variables that reduce the bias, in particular cause the action, but do not directly affect the outcome. They are not affected by any variable that affects the outcome.

Identification A causal graph reflects the ways of conveying domain knowledge (either through named variable sets of confounders and instrumental variables, or through a causal graph). DoWhy supports the following identification algorithms: Backdoor, Frontdoor, Instrumental variable, ID algorithm, Mediation (direct and indirect effects) [102].

Estimation In general, the problem that an analyst faces during the estimation step is that (s)he should have **already** figured out **how** to **build a reasonable causal model** from **data** and **domain knowledge** and **identified** the **correct estimand** [102].

Furthermore, the analyst is expected to have already assessed the verification and robustness of causal analyses. For this reason, the libraries provide no guidance on their own, and that makes things more challenging [102].

DoWhy provides estimation methods compatible with the identification strategy for estimating the average causal effect [102]. In particular, we can find: Regression-based methods, Distance-based matching, Propensity-based methods and Do-sampler.

- Distance-based matching is based on **matching confounders**' values based on distance metrics;
- Propensity-based Stratification, Propensity Score Matching, Inverse Propensity Weighting are based on **estimating the treatment assignment**;
- Linear Regression, Generalized Linear Models (e.g. logistic regression, etc.) are based on **estimating the outcome model**;
- Binary Instrument/Wald Estimator, Regression discontinuity are based on the **instrumental variable** equation;
- Two-stage linear regression is specific for **front-door** criterion and general **mediation**.

Refutation Checking for the estimate's correctness is not possible [102], thus it is possible to reject it: if the estimator fails the **refutation test** (p-value is < 0.05), we simply can understand that there are some issues with the estimator [102].

As the **estimate is not affected by data changes**, any **estimator with a significant variation** between the original data and the modified data **fails the test**. Furthermore, after any **changing in the data**, the causal true estimate is zero, and any significant estimator's changes from zero let it fails the test.

4.8. Relation between Causal Inference and Machine Learning

Machine learning helps in building causal effect **estimators**, and **causal reasoning** can be help in building more **robust** machine learning **models**.

The experimental and observational information fusion can synthesize an estimate of the desired causal relation [13]. In other words, it is important to establish a necessary and sufficient condition to decide **when** causal effects in the target domain are estimable from both the statistical information available and the causal information **transferred** from the experiments [13].

When we perform experiments a group of subjects, e.g. the telecommunication's customers, the issue arises whether the conclusions are applicable to a different but somewhat related group [13].

Artificial Intelligence holds a distinct advantage in addressing this issue in a systematic manner.

A causal graph [86], [109], [87], [72], [13] constitutes the syntactic representation of the the **distinction** between **statistical** and **causal knowledge**. Graphical models offer a means of expressing distinctions and similarities within various domains, environments, and populations [88].

The **do-calculus** [86], [72] is especially well-suited for integrating these two aspects into a unified framework and creating efficient algorithms for **transferring knowledge** [13].

4.8.1. Transferability

Transferring causal knowledge between two environments $\prod$ and $\prod^*$.

In environment $\prod$, experiments can be performed and causal knowledge gathered. In $\prod^*$, potentially different from $\prod$, only passive observations can be collected but no experiments conducted. The problem is to **infer** a **causal relationship R** in $\prod^*$ using knowledge obtained in $\prod$. If nothing is known about the relationship between $\prod$ and $\prod^*$, the problem is unsolvable.

Non-transportability requires the construction of two models agreeing on $\langle P, I, P^* \rangle$, while non-identifiability requires the two models to agree solely on the observational distribution P. P represents the true distribution of the data. I denotes the set of interventions or treatments considered. P^* is the distribution of potential outcomes under those interventions.

The simplest non-transportable structure is an extension of the famous 'bow arc' graph named here 's-bow arc', see Figure 4.12. The s-bow arc has two **endogenous** nodes: X, and its child Y, sharing a **hidden exogenous** parent U, and a S-node pointing to Y. This and similar structures that **prevent transportability** will be useful in our proof of completeness, which requires a demonstration that whenever the algorithm fails to transport a causal relation, the relation is indeed non-transportable. The challenge of transportability lies in transferring two solutions, posing a hindrance. Regarding the addressed issue, the robustness of the conclusions can be enhanced by quantifying their strength through **sensitivity analysis**. The example is useful to understand that it is possible to transfer two predictions ignoring which is transferable, by understanding the underlying causal model. Because it is impractical to directly confirm the presence or absence of unmeasured confounding variables for each generated model, especially when there is suspicion of their existence, sensitivity analysis is employed [13].

Transportability presupposes having **sufficient structural understanding** of both domains (i.e. statistical and causal) to support the creation of their respective causal diagrams. If there is a lack of domain knowledge, **causal discovery algorithms** can be used to infer diagrams from available data [87], [90], [109].

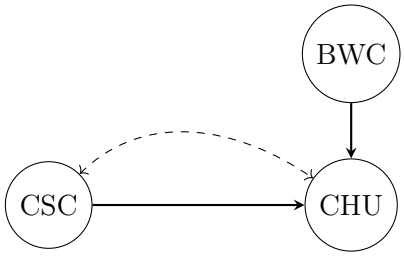


Figure 4.12.: Example of non-transportability.

The study of **transportability** or **external validity** aims to identify conditions under which causal information learned from experiments can be reused in a different environment where only **passive observations** can be collected [13], **which is the case study of this thesis.** For Bernoulli distributions, which model binary outcomes (e.g., success/failure), the probability mass function is often denoted by $\mathbf{P()}$, representing the probability of a particular outcome. Example: P(X = 1) represents the probability of success. For other types of distributions, especially those that represent **continuous** or **non-binary random** variables, the expectation, $\mathbf{E()}$ operator is often used and represents the **average** value of a random variable.

In the following example Figure 4.13 shows the transfer [13] of experimental results between two locations i.e. Milan (MI) and Rome (RO) and estimate the causal effect of treatment CSC on outcome CHU. BWC = bwc, denoted P(CHU|do(CSC), BWC).

We want to generalize the results to the population of Rome i.e. RO, but we find the distribution $P(CSC, CHU, BWC)$ in Milan, i.e. MI to be different from the one in RO (call the latter $P^*(CSC, CHU, BWC)$, because the distribution is changed). In particular, the **average** of CSC in RO is significantly higher than that in MI [13]. How can we estimate the causal effect of CSC on CHU in RO, denoted

$$R = P^*(CHU|do(CSC))$$

? Given the Equation 4.32:

$$P_i(CHU = 1 \mid do(CSC), BWC) \tag{4.32}$$

Overall causal effect in RO is given by the Equation 4.33:

$$R = \sum_{BWC} P^*(CHU|do(CSC), BWC)P^*(BWC) = \tag{4.33}$$

$$= \sum_{BWC} P(CHU|do(CSC), BWC)P(BWC) \tag{4.34}$$

The combination of **experimental results** from MI, $P(CHU|do(CSC), BWC)$, with **observational aspects** of RO population, $P^*(z)$, allows to obtain an **experimental claim** $P^*(CHU|do(CSC))$ about RO.

The transport formula is shown in Equation 4.35

$$\sum_{BWC} P(CHU|do(CSC), BWC)P(BWC) \tag{4.35}$$

In the following example 4.13: we want to know **what is the impact** of the treatment, i.e. CSC, on the customer churn CHU, in particular, the customer's survival in five to ten years. **What is the effect of** CSC **respect to** CHU**?** The BWC is the common cause, if we find something about the influence of CSC on CHU by applying the **Backdoor Criterion** by taking into account the country or the city where the study is performed. If CSC is higher than a certain threshold and CHU is higher than e.g. 5 years. The customer which maintains the line "survives" i.e. $CHU = 0$.

Given the Equation 4.36, in Milan we have:

$$\sum_{BWC} P(BWC)E[CHU \mid do(CSC = 1), BWC] - E[CHU \mid do(CSC = 0), BWC] \tag{4.36}$$

Given the Equation 4.37, in Rome we have:

$$\sum_{BWC} P^*(BWC)E[CHU \mid do(CSC = 1), BWC] - E[CHU \mid do(CSC = 0), BWC] \tag{4.37}$$

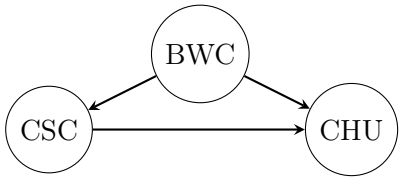


Figure 4.13.: Example of results transportability [80].

Chapter 5.

Experimental setup for Causal Discovery

The aim of this thesis is to propose a new **Ensemble Discovery** method for telecommunication data. However, in this real telecom data set scenario, we do not have the **Ground Truth** (GT). For this reason, I performed experiments starting with **synthetic datasets** generated by the IIDSimulation tool using gCastle [127]. The Ground Truth consists of a Direct Acyclic Graph (DAG) generated by a **topology generator** described in Section 5.2.

In particular, N represents the number of random datasets and for each of them a pool of $k = 5$ algorithms (i.e., PC, GES, LINGAM, GOLEM, and Notears) is applied. For each synthetic dataset, we apply the k algorithms, and each of them produces a graph. Ideally, this should be a DAG, however sometimes some ambiguities are left in the form of undirected edges.

We calculated the distances between the graph generated by each algorithm and the GT. We are interested in a method for selecting the closest graph(s), i.e., the one(s) that better approximates the ground truth, in some sense which will be specified.

A relevant concept that supports the selection of the minimum distance graph is the one of **centroid graph**: the graph that has the minimum distance with respect to the other graphs.

5.1. Outline of our study

The general structure of our study is the following:

- Generation of Ground Truth Synthetic Datasets by means of a **Topology Generator**
- Causal Discovery **Algorithm Selection and Setup.**
- Models training with the **Synthetic Datasets**.
- For each of the applied algorithms an **Algorithm Specific Graph** is generated.
- Graph selection based on the **Ensemble** method.
- Performance analysis and visualization.

The first four steps are illustrated in this chapter (Section 5.2).The others are described in the next chapter.

5.2. The Topology Generator

The procedure relies on the generation of a high number of DAG topologic structures (i.e. ground truth DAGs) and the simulation [19]. I have created, by using the *gCastle* tool, multiple **causal networks**, corresponding to multiple configurations in terms of the number of nodes, the number of edges, topologies, non-dipendent variable distributions, and a noise (both Gaussian and not Gaussian).

I selected 10 combinations (n, e) [n:number of nodes in the graph in $N = [n\_min, \ldots, n\_max]$ and e: number of arcs in $E = [e\_min, \ldots, e\_max]$] to generate the **ground truth**, that is, the adjacent matrix of the graph. We call the generic combination

$$(n, e)$$

as c in $C = [(n\_min, e\_min), \ldots, (n\_max, e\_max)]$.

I chose the combinations so as to have some variability in terms of graph size

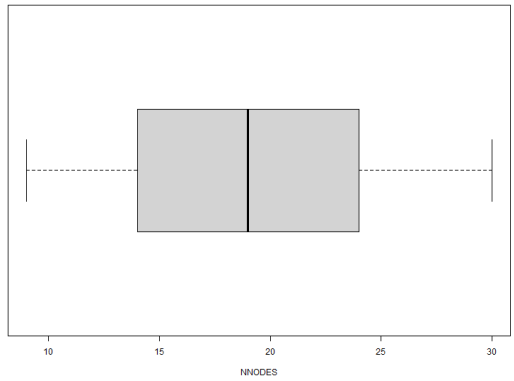


Figure 5.1.: Boxplot of the number of nodes NNODES.

and its graph density, so I generated graphs from a minimum of $n_{min} = 9$ nodes to a maximum of $n_{max} = 30$ nodes, and with a number of arcs between $m_{min} = 21$ and $m_{max} = 435$.[1]

The distributions of the number of nodes and the number of arcs are shown in Figure 5.1, and Figure 5.2.

For each specific combination c, 5 topologies were produced: those were scale-free graphs. For each topology 10 variants were generated (using IIDSimulation) each with a different probability distribution for exogenous variables (among the distributions available in gCastle).

In this way, we created $10 \times 5 \times 10 = 500$ datasets.

Due to a temporary limitation of the computational resources to be used in the subsequent phase, we randomly chose a set of 100 datasets on which to run the discovery algorithms. Then we ran each of the 5 algorithms under study, that is $ALGO \in \{GES, GOLEM, LINGAM, Notears, PC\}$.

The dataset generator creates 100 scale-free adjacency matrices i.e. 100 Ground Truths.

At this point, we ran the following discovery algorithms: `PC` in Section 4.7.1, `DirectLiNGAM` in Section 4.7.3, `GES` in Section 4.7.2, `GOLEM` in Section

[1] Specifically the 10 (n, e) pairs were: $(9, 21)$, $(9, 36)$, $(14, 48)$, $(14, 91)$, $(19, 86)$, $(19, 171)$, $(24, 136)$, $(24, 276)$, $(30, 210)$, $(30, 435)$.

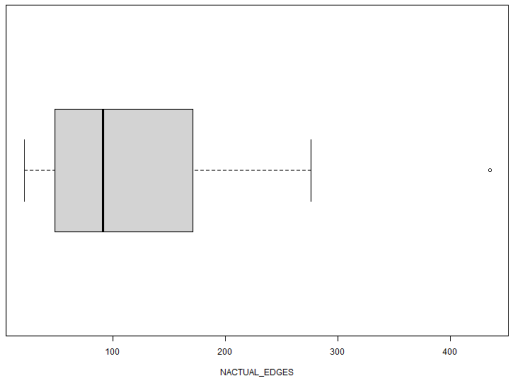


Figure 5.2.: Boxplot of the number of nodes NACTUAL_EDGES.

4.7.5 and `Notears` in Section 4.7.4, see Chapter 4. I used these algorithms to create the predicted matrices to compare with the generated ground truth.

For each discovery algorithm an **Algorithm Specific Graph** was generated. The model is trained for each dataset and the resulting causal matrix is saved for the next analysis steps.

A number of times, some algorithms failed converging. Keeping only the runs where all the 5 algorithms had converged, we were left with 78 datasets, with the respective GT and the respective reconstructed graphs.

To each dataset was thus associated a record. Those records were the statistical units on which we performed the analysis described in the next chapter.

5.3. Simulations: Comparison heatmaps

The models are applied in conjunction with linear dependencies, and to adhere to the framework suggested by gCastle, the distributions are categorized using the `sem_type` method. The `sem_type` parameter allows you to specify the type of distribution to be used in the structural equation model. Depending on whether the model is linear or nonlinear, it is possi-

ble to choose from the appropriate set of distributions [127].

sem_type: str gauss, exp, gumbel, uniform, logistic (linear); mlp, mim, gp, gp-add, quadratic (nonlinear).

Figure 5.3.: The `.sem_type()` method.

For details of linear and nonlinear distributions see Appendix C.

5.4. A simple example of GT with fixed number of nodes

Linear and nonlinear datasets, for the details see the Appendix C, are generated with `IIDSimulation` with the aim to compare the ground truth and the predicted graph heatmaps. Visual inspection is useful in the cases where we have small DAGs, for higher dimensions the comparison becomes challenging [80].

Thus, it is preferable to use the `GraphDAG` tool, as it is possible to plot and compare the heatmaps [80]. Each learned adjacency matrix generated by the PC, GES, LINGAM, NOTEARS and GOLEM algorithms is compared with the ground truth by visual inspection. The resulting heatmaps are binary: the value 1 indicates that the link is present and the value 0 indicates the link absence.

5.4.1. The model evaluation metrics

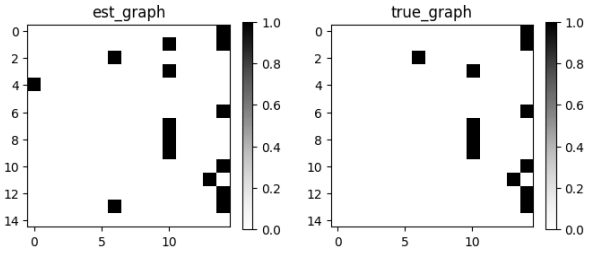
There are several available metrics in `MetricsDAG`, further described in Appendix D:

```
from castle.metrics import MetricsDAG
metrics = MetricsDAG(
    B_est=pred_dag,
    B_true=adj_matrix
)
```

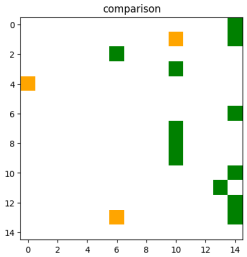
5.5. Predicted matrices and Ground Truth Comparison

In case of small dimensions graphs, the visual comparison is useful. A valid alternative consists of the **heatmaps**. As the adjacency matrix is unweighted, the heatmap is binary (0 or 1 values only). In the following, from Figure 5.4 to Figure 5.53 are presented the heatmaps generated by each algorithm (PC, GES, LINGAM, GOLEM, NOTEARS) using a Linear Gaussian distribution representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

5.6. Current Dataset: Linear Gaussian



(a) Predicted graph by PC and Ground Truth.



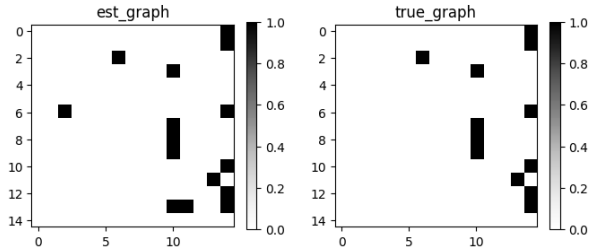
(b) Comparison of PC using Linear Gaussian distribution with Ground Truth.

Figure 5.4.: Comparing PC using Linear Gaussian distribution.

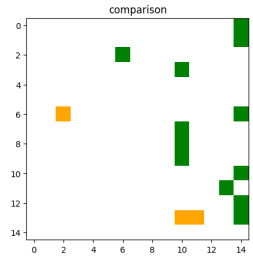
Heatmaps PC vs GT In Figure 5.4a, heatmaps generated by algorithm using a PC representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.4b, a visual com-

parison of the heatmaps generated by the algorithm using a PC representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

The colors refer to the comparison between the links in the *true_dag* and the *pred_dag*: the **white** color means that the link is **absent** from the dags; the **red** color means that the link is present in the *true_dag*, but missing in the *pred_dag*; the **orange** color means that the link is present in the *pred_dag*, but missing in the *true_dag*, i.e. **spurious**; the **green** color means that the link is present in both the dags.



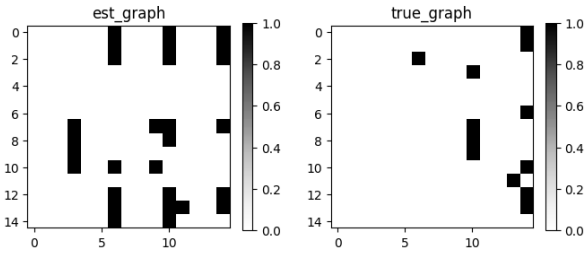
(a) Predicted graph by GES and Ground Truth.



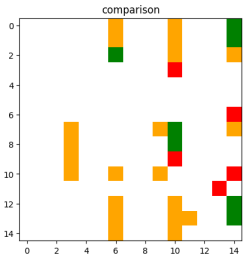
(b) Comparison of GES using Linear Gaussian distribution with Ground Truth.

Figure 5.5.: Comparing GES using Linear Gaussian distribution.

Heatmaps GES vs GT In Figure 5.5a, heatmaps generated by algorithm using a GES representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.5b, a visual comparison of the heatmaps generated by the algorithm using a GES representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).



(a) Predicted graph by DIRECTLingam and Ground Truth.



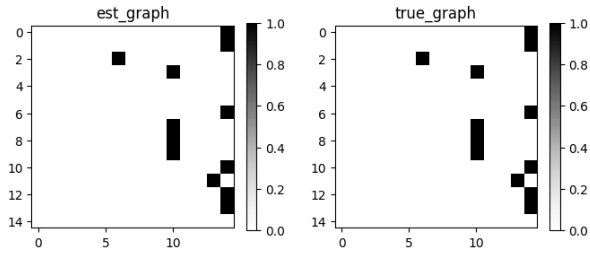
(b) Comparison of DIRECTLingam using Linear Gaussian distribution with Ground Truth.

Figure 5.6.: Comparing DIRECTLingam using Linear Gaussian distribution.

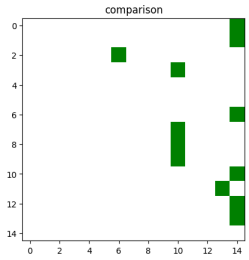
Heatmaps DIRECTLingam vs GT In Figure 5.6a, heatmaps generated by algorithm using a DIRECTLingam representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.6b, a visual comparison of the heatmaps generated by the algorithm using a DIRECTLingam representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps Notears vs GT In Figure 5.7a, heatmaps generated by algorithm using a Notears representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.7b, a visual comparison of the heatmaps generated by the algorithm using a Notears representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps Golem vs GT In Figure 5.8a, heatmaps generated by algorithm using a Golem representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.8b, a visual compar-



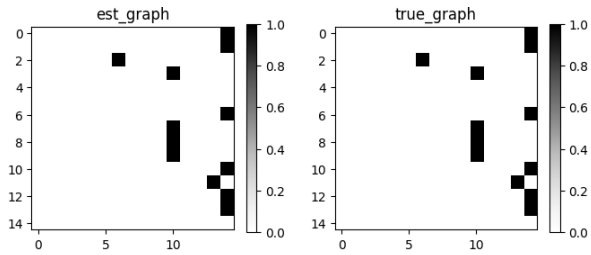
(a) Predicted graph by Notears and Ground Truth.



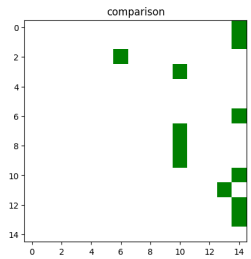
(b) Comparison of Notears using Linear Gaussian distribution with Ground Truth.

Figure 5.7.: Comparing Notears using Linear Gaussian distribution.

ison of the heatmaps generated by the algorithm using a Golem representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).



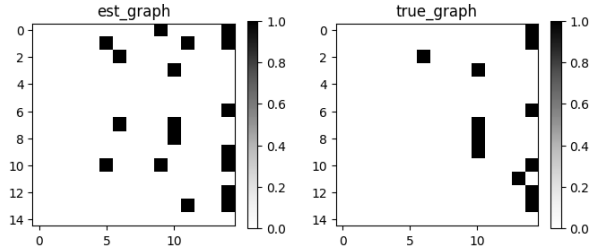
(a) Predicted graph by Golem and Ground Truth.



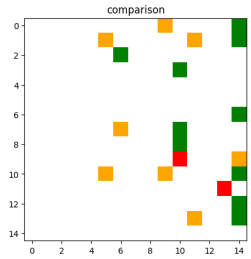
(b) Comparison of Golem using Linear Gaussian distribution with Ground Truth.

Figure 5.8.: Comparing Golem using Linear Gaussian distribution.

5.7. Current Dataset: Linear Exponential



(a) Predicted graph by PC and Ground Truth.



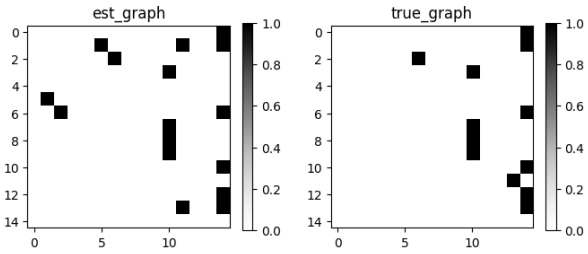
(b) Comparison of PC using Linear Exponential distribution with Ground Truth.

Figure 5.9.: Comparing PC using Linear Exponential distribution.

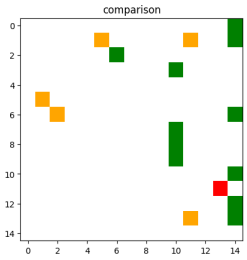
Heatmaps PC vs GT In Figure 5.9a, heatmaps generated by algorithm using a PC representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.9b, a visual comparison of the heatmaps generated by the algorithm using a PC representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps GES vs GT In Figure 5.10a, heatmaps generated by algorithm using a GES representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.10b, a visual comparison of the heatmaps generated by the algorithm using a GES representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps DIRECTLingam vs GT In Figure 5.11a, heatmaps generated by algorithm using a DIRECTLingam representing the learned ad-



(a) Predicted graph by GES and Ground Truth.



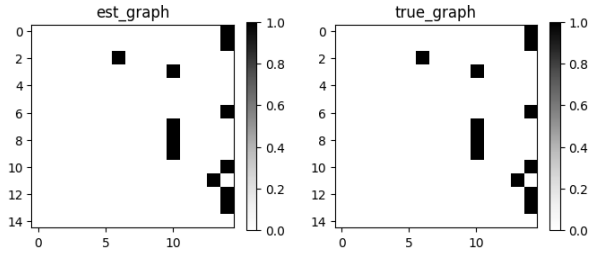
(b) Comparison of GES using Linear Exponential distribution with Ground Truth.

Figure 5.10.: Comparing GES using Linear Exponential distribution.

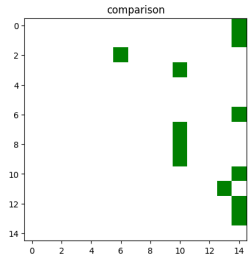
jacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.11b, a visual comparison of the heatmaps generated by the algorithm using a DIRECTLingam representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps Notears vs GT In Figure 5.12a, heatmaps generated by algorithm using a Notears representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.12b, a visual comparison of the heatmaps generated by the algorithm using a Notears representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps Golem vs GT In Figure 5.13a, heatmaps generated by algorithm using a Golem representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.13b, a visual comparison of the heatmaps generated by the algorithm using a Golem representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

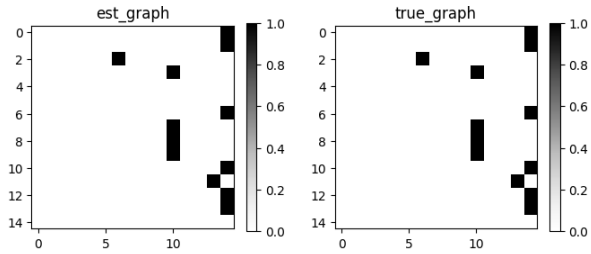


(a) Predicted graph by DIRECTLingam and Ground Truth.

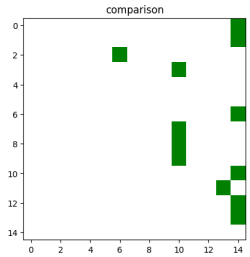


(b) Comparison of DIRECTLingam using Linear Exponential distribution with Ground Truth.

Figure 5.11.: Comparing DIRECTLingam using Linear Exponential distribution.

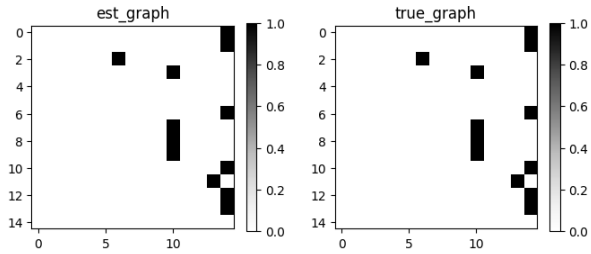


(a) Predicted graph by Notears and Ground Truth.

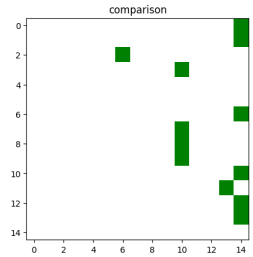


(b) Comparison of Notears using Linear Exponential distribution with Ground Truth.

Figure 5.12.: Comparing Notears using Linear Exponential distribution.



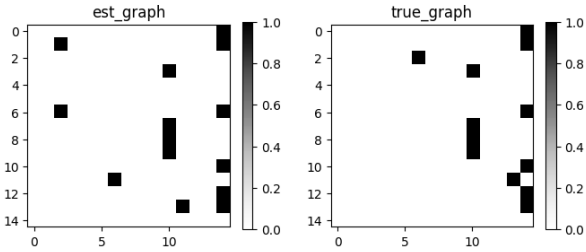
(a) Predicted graph by Golem and Ground Truth.



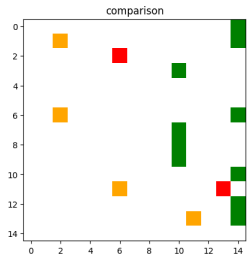
(b) Comparison of Golem using Linear Exponential distribution with Ground Truth.

Figure 5.13.: Comparing Golem using Linear Exponential distribution.

5.8. Current Dataset: Linear Uniform



(a) Predicted graph by PC and Ground Truth.



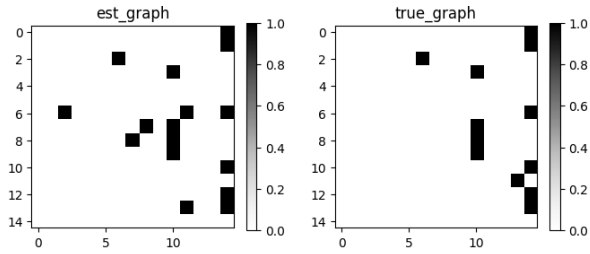
(b) Comparison of PC using Linear Uniform distribution with Ground Truth.

Figure 5.14.: Comparing PC using Linear Uniform distribution.

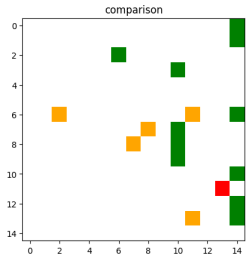
Heatmaps PC vs GT In Figure 5.14a, heatmaps generated by algorithm using a PC representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.14b, a visual comparison of the heatmaps generated by the algorithm using a PC representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps GES vs GT In Figure 5.15a, heatmaps generated by algorithm using a GES representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.15b, a visual comparison of the heatmaps generated by the algorithm using a GES representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps DIRECTLingam vs GT In Figure 5.16a, heatmaps generated by algorithm using a DIRECTLingam representing the learned ad-



(a) Predicted graph by GES and Ground Truth.



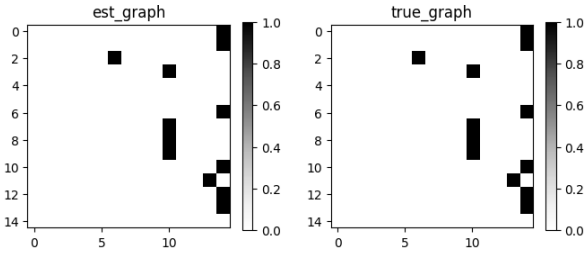
(b) Comparison of GES using Linear Uniform distribution with Ground Truth.

Figure 5.15.: Comparing GES using Linear Uniform distribution.

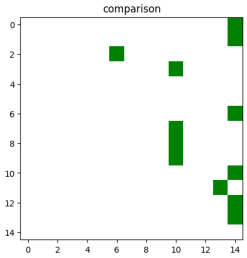
jacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.16b, a visual comparison of the heatmaps generated by the algorithm using a DIRECTLingam representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps Notears vs GT In Figure 5.17a, heatmaps generated by algorithm using a Notears representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.17b, a visual comparison of the heatmaps generated by the algorithm using a Notears representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps Golem vs GT In Figure 5.18a, heatmaps generated by algorithm using a Golem representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.18b, a visual comparison of the heatmaps generated by the algorithm using a Golem representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

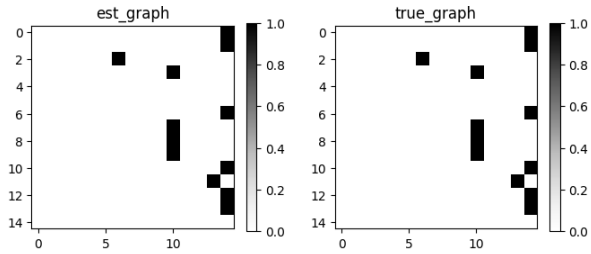


(a) Predicted graph by DIRECTLingam and Ground Truth.

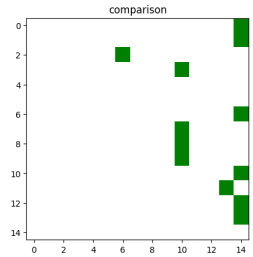


(b) Comparison of DIRECTLingam using Linear Uniform distribution with Ground Truth.

Figure 5.16.: Comparing DIRECTLingam using Linear Uniform distribution.

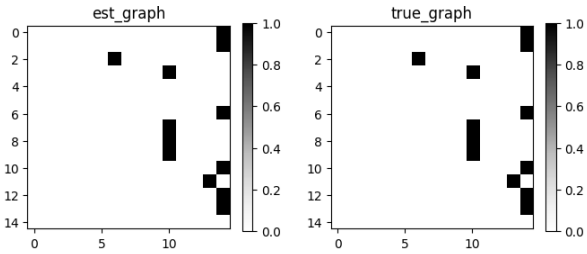


(a) Predicted graph by Notears and Ground Truth.

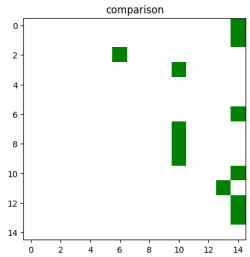


(b) Comparison of Notears using Linear Uniform distribution with Ground Truth.

Figure 5.17.: Comparing Notears using Linear Uniform distribution.



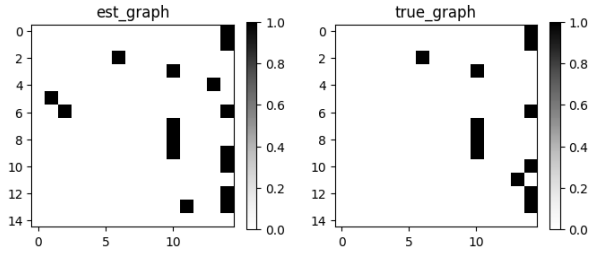
(a) Predicted graph by Golem and Ground Truth.



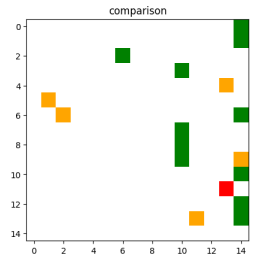
(b) Comparison of Golem using Linear Uniform distribution with Ground Truth.

Figure 5.18.: Comparing Golem using Linear Uniform distribution.

5.9. Current Dataset: Linear Gumbel



(a) Predicted graph by PC and Ground Truth.



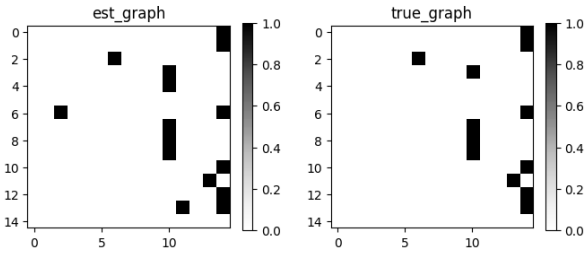
(b) Comparison of PC using Linear Gumbel distribution with Ground Truth.

Figure 5.19.: Comparing PC using Linear Gumbel distribution.

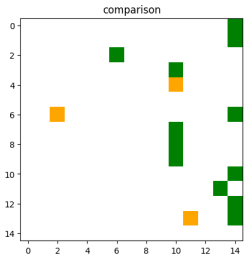
Heatmaps PC vs GT In Figure 5.19a, heatmaps generated by algorithm using a PC representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.19b, a visual comparison of the heatmaps generated by the algorithm using a PC representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps GES vs GT In Figure 5.20a, heatmaps generated by algorithm using a GES representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.20b, a visual comparison of the heatmaps generated by the algorithm using a GES representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps DIRECTLingam vs GT In Figure 5.21a, heatmaps generated by algorithm using a DIRECTLingam representing the learned ad-



(a) Predicted graph by GES and Ground Truth.



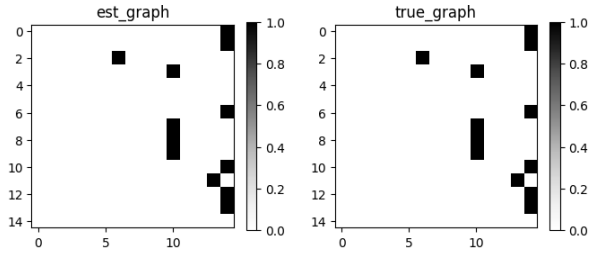
(b) Comparison of GES using Linear Gumbel distribution with Ground Truth.

Figure 5.20.: Comparing GES using Linear Gumbel distribution.

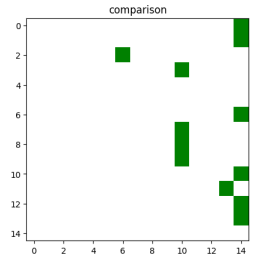
jacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.21b, a visual comparison of the heatmaps generated by the algorithm using a DIRECTLingam representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps Notears vs GT In Figure 5.22a, heatmaps generated by algorithm using a Notears representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.22b, a visual comparison of the heatmaps generated by the algorithm using a Notears representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps Golem vs GT In Figure 5.23a, heatmaps generated by algorithm using a Golem representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.23b, a visual comparison of the heatmaps generated by the algorithm using a Golem representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

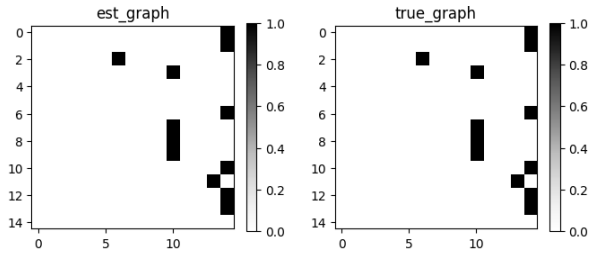


(a) Predicted graph by DIRECTLingam and Ground Truth.

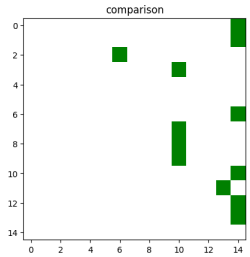


(b) Comparison of DIRECTLingam using Linear Gumbel distribution with Ground Truth.

Figure 5.21.: Comparing DIRECTLingam using Linear Gumbel distribution.

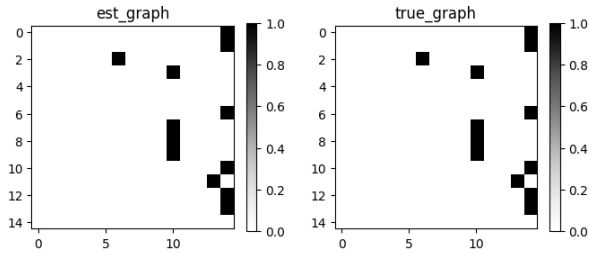


(a) Predicted graph by Notears and Ground Truth.

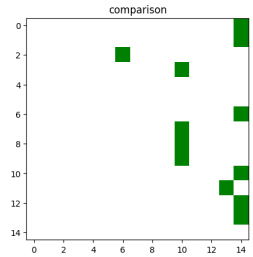


(b) Comparison of Notears using Linear Gumbel distribution with Ground Truth.

Figure 5.22.: Comparing Notears using Linear Gumbel distribution.



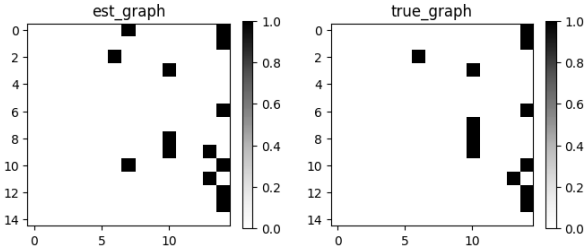
(a) Predicted graph by Golem and Ground Truth.



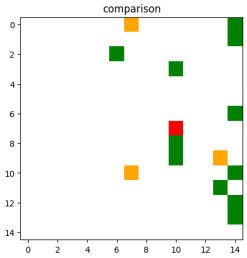
(b) Comparison of Golem using Linear Gumbel distribution with Ground Truth.

Figure 5.23.: Comparing Golem using Linear Gumbel distribution.

5.10. Current Dataset: Linear Logistic



(a) Predicted graph by PC and Ground Truth.



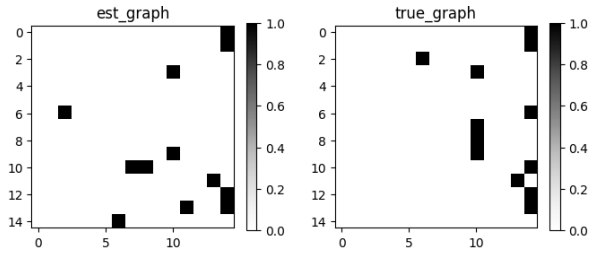
(b) Comparison of PC using Linear Logistic distribution with Ground Truth.

Figure 5.24.: Comparing PC using Linear Logistic distribution.

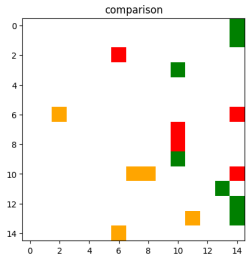
Heatmaps PC vs GT In Figure 5.24a, heatmaps generated by algorithm using a PC representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.24b, a visual comparison of the heatmaps generated by the algorithm using a PC representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps GES vs GT In Figure 5.25a, heatmaps generated by algorithm using a GES representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.25b, a visual comparison of the heatmaps generated by the algorithm using a GES representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps DIRECTLingam vs GT In Figure 5.26a, heatmaps generated by algorithm using a DIRECTLingam representing the learned ad-



(a) Predicted graph by GES and Ground Truth.



(b) Comparison of GES using Linear Logistic distribution with Ground Truth.

Figure 5.25.: Comparing GES using Linear Logistic distribution.

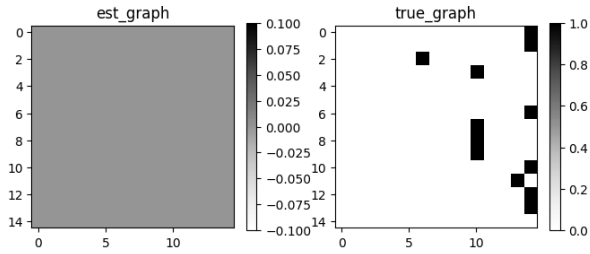
jacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.26b, a visual comparison of the heatmaps generated by the algorithm using a DIRECTLingam representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

The result is obviously not correct because the algorithm did not perform properly.

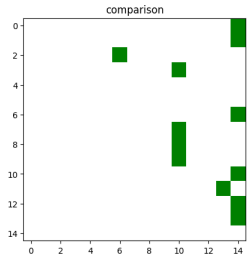
Heatmaps Notears vs GT In Figure 5.27a, heatmaps generated by algorithm using a Notears representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.27b, a visual comparison of the heatmaps generated by the algorithm using a Notears representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

The result is obviously not correct because the algorithm did not perform properly.

Heatmaps Golem vs GT In Figure 5.28a, heatmaps generated by algo-



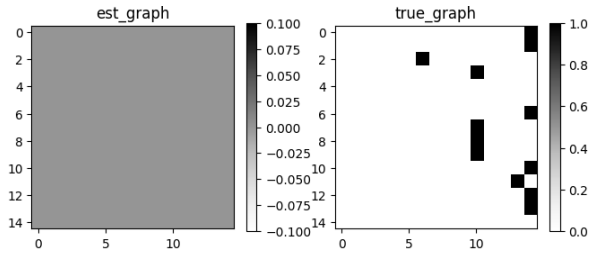
(a) Predicted graph by DIRECTLingam and Ground Truth.



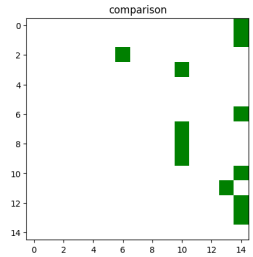
(b) Comparison of DIRECTLingam using Linear Logistic distribution with Ground Truth.

Figure 5.26.: Comparing DIRECTLingam using Linear Logistic distribution.

rithm using a Golem representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.28b, a visual comparison of the heatmaps generated by the algorithm using a Golem representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

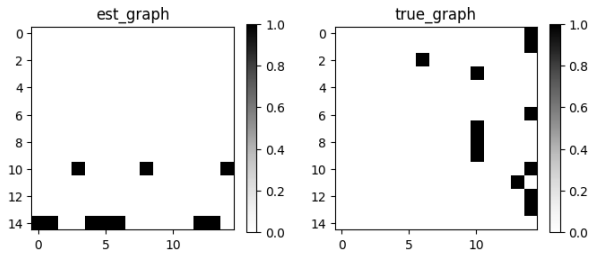


(a) Predicted graph by Notears and Ground Truth.

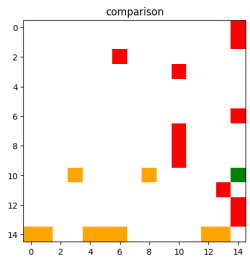


(b) Comparison of Notears using Linear Logistic distribution with Ground Truth.

Figure 5.27.: Comparing Notears using Linear Logistic distribution.



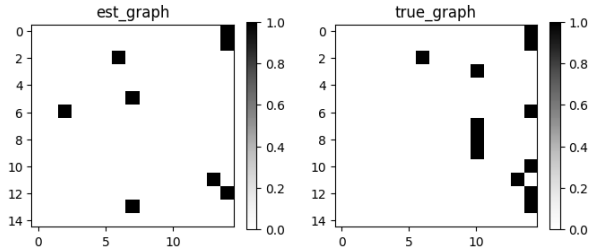
(a) Predicted graph by Golem and Ground Truth.



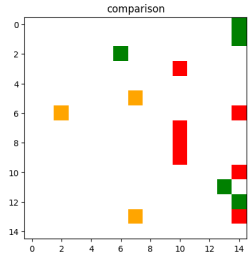
(b) Comparison of Golem using Linear Logistic distribution with Ground Truth.

Figure 5.28.: Comparing Golem using Linear Logistic distribution.

5.11. Current Dataset: Quadratic



(a) Predicted graph by PC and Ground Truth.



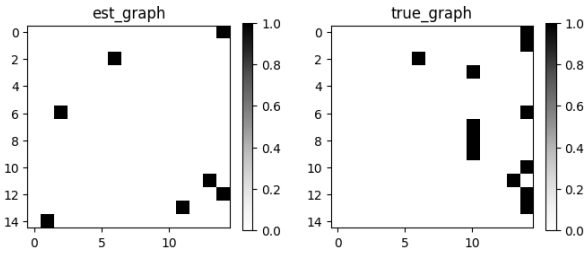
(b) Comparison of PC using Quadratic distribution with Ground Truth.

Figure 5.29.: Comparing PC using Quadratic distribution.

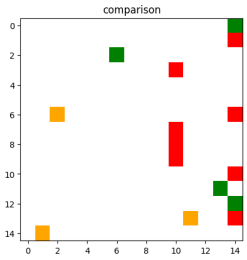
Heatmaps PC vs GT In Figure 5.29a, heatmaps generated by algorithm using a PC representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.29b, a visual comparison of the heatmaps generated by the algorithm using a PC representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps GES vs GT In Figure 5.30a, heatmaps generated by algorithm using a GES representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.30b, a visual comparison of the heatmaps generated by the algorithm using a GES representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps DIRECTLingam vs GT In Figure 5.31a, heatmaps generated by algorithm using a DIRECTLingam representing the learned ad-



(a) Predicted graph by GES and Ground Truth.



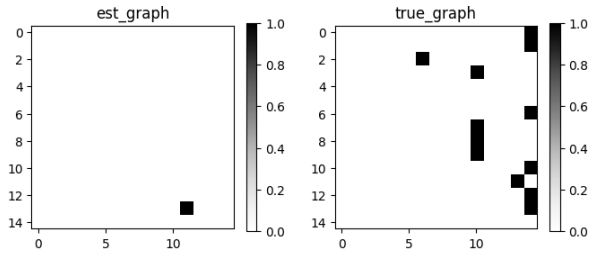
(b) Comparison of GES using Quadratic distribution with Ground Truth.

Figure 5.30.: Comparing GES using Quadratic distribution.

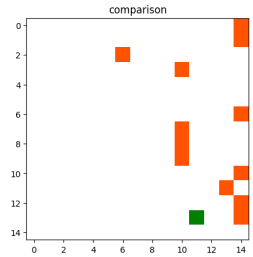
jacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.31b, a visual comparison of the heatmaps generated by the algorithm using a DIRECTLingam representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps Notears vs GT In Figure 5.32a, heatmaps generated by algorithm using a Notears representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.32b, a visual comparison of the heatmaps generated by the algorithm using a Notears representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps Golem vs GT In Figure 5.33a, heatmaps generated by algorithm using a Golem representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.33b, a visual comparison of the heatmaps generated by the algorithm using a Golem representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

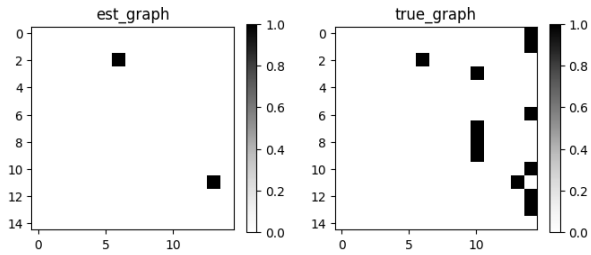


(a) Predicted graph by DIRECTLingam and Ground Truth.

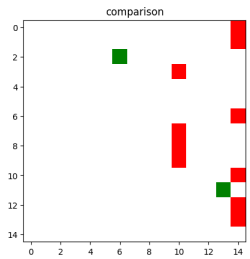


(b) Comparison of DIRECTLingam using Quadratic distribution with Ground Truth.

Figure 5.31.: Comparing DIRECTLingam using Quadratic distribution.

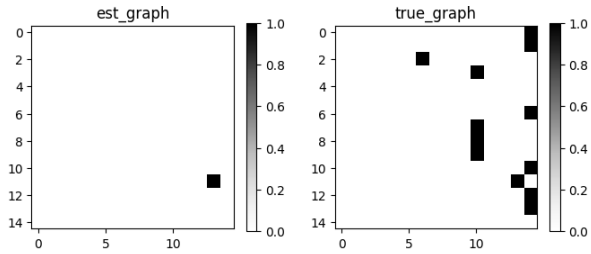


(a) Predicted graph by Notears and Ground Truth.

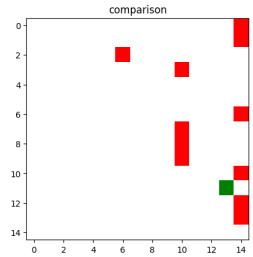


(b) Comparison of Notears using Quadratic distribution with Ground Truth.

Figure 5.32.: Comparing Notears using Quadratic distribution.



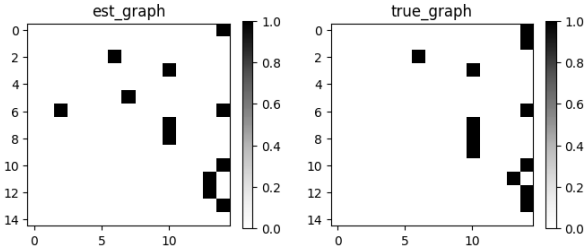
(a) Predicted graph by Golem and Ground Truth.



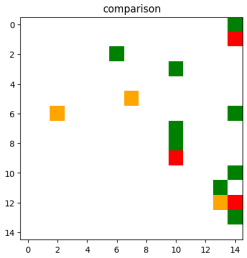
(b) Comparison of Golem using Quadratic distribution with Ground Truth.

Figure 5.33.: Comparing Golem using Quadratic distribution.

5.12. Current Dataset: Multilayer Perceptron



(a) Predicted graph by PC and Ground Truth.



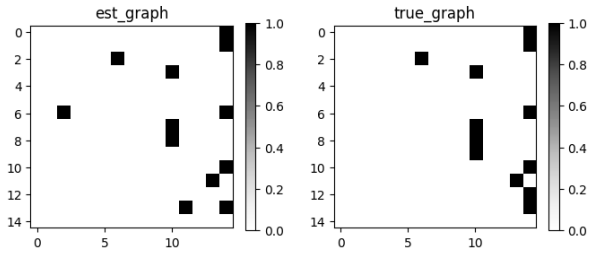
(b) Comparison of PC using Multilayer Perceptron distribution with Ground Truth.

Figure 5.34.: Comparing PC using Multilayer Perceptron distribution.

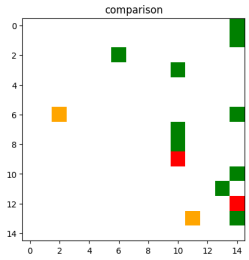
Heatmaps PC vs GT In Figure 5.34a, heatmaps generated by algorithm using a PC representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.34b, a visual comparison of the heatmaps generated by the algorithm using a PC representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps GES vs GT In Figure 5.35a, heatmaps generated by algorithm using a GES representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.35b, a visual comparison of the heatmaps generated by the algorithm using a GES representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps DIRECTLingam vs GT In Figure 5.36a, heatmaps generated by algorithm using a DIRECTLingam representing the learned ad-



(a) Predicted graph by GES and Ground Truth.



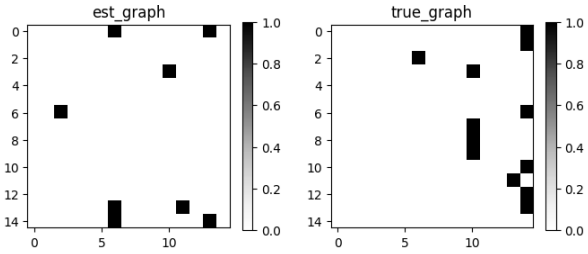
(b) Comparison of GES using Multilayer Perceptron distribution with Ground Truth.

Figure 5.35.: Comparing GES using Multilayer Perceptron distribution.

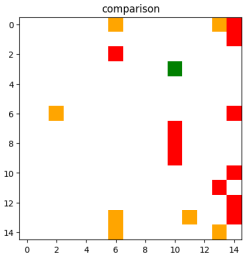
jacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.36b, a visual comparison of the heatmaps generated by the algorithm using a DIRECTLingam representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps Notears vs GT In Figure 5.37a, heatmaps generated by algorithm using a Notears representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.37b, a visual comparison of the heatmaps generated by the algorithm using a Notears representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps Golem vs GT In Figure 5.38a, heatmaps generated by algorithm using a Golem representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.38b, a visual comparison of the heatmaps generated by the algorithm using a Golem representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

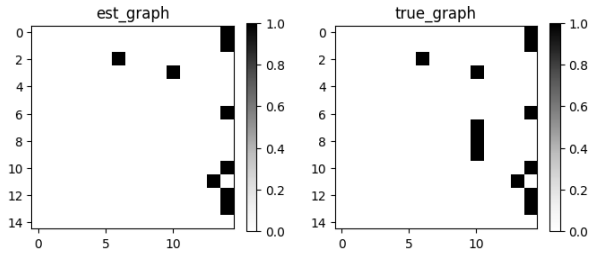


(a) Predicted graph by DIRECTLingam and Ground Truth.

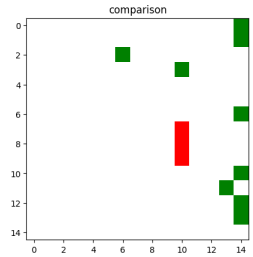


(b) Comparison of DIRECTLingam using Multilayer Perceptron distribution with Ground Truth.

Figure 5.36.: Comparing DIRECTLingam using Multilayer Perceptron distribution.

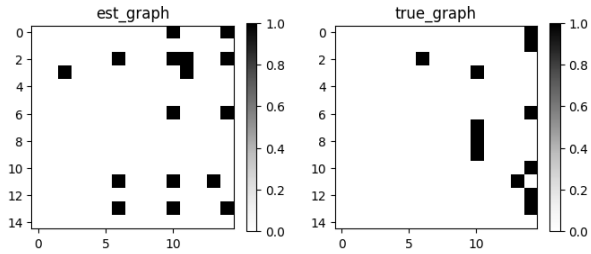


(a) Predicted graph by Notears and Ground Truth.

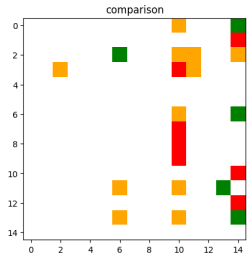


(b) Comparison of Notears using Multilayer Perceptron distribution with Ground Truth.

Figure 5.37.: Comparing Notears using Multilayer Perceptron distribution.



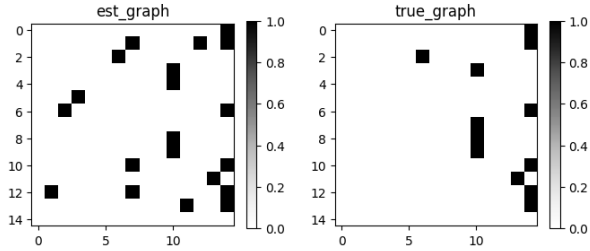
(a) Predicted graph by Golem and Ground Truth.



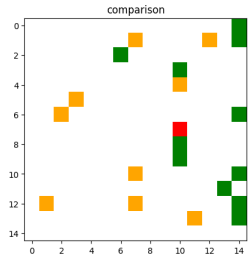
(b) Comparison of Golem using Multilayer Perceptron distribution with Ground Truth.

Figure 5.38.: Comparing Golem using Multilayer Perceptron distribution.

5.13. Current Dataset: Multiple Instance Learning



(a) Predicted graph by PC and Ground Truth.



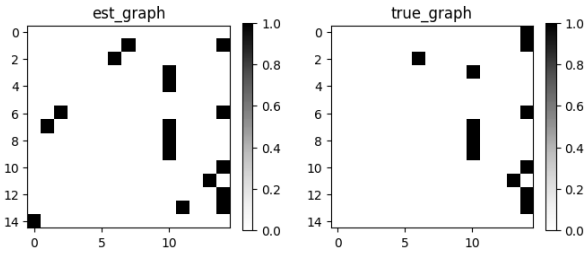
(b) Comparison of PC using MIM distribution with Ground Truth.

Figure 5.39.: Comparing PC using MIM distribution.

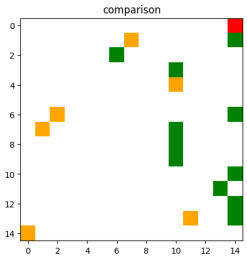
Heatmaps PC vs GT In Figure 5.39a, heatmaps generated by algorithm using a PC representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.39b, a visual comparison of the heatmaps generated by the algorithm using a PC representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps GES vs GT In Figure 5.40a, heatmaps generated by algorithm using a GES representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.40b, a visual comparison of the heatmaps generated by the algorithm using a GES representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps DIRECTLingam vs GT In Figure 5.41a, heatmaps generated by algorithm using a DIRECTLingam representing the learned ad-



(a) Predicted graph by GES and Ground Truth.



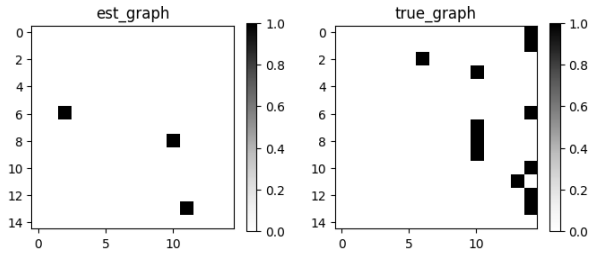
(b) Comparison of GES using MIM distribution with Ground Truth.

Figure 5.40.: Comparing GES using MIM distribution.

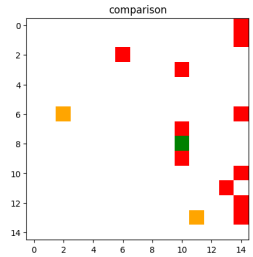
jacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.41b, a visual comparison of the heatmaps generated by the algorithm using a DIRECTLingam representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps Notears vs GT In Figure 5.42a, heatmaps generated by algorithm using a Notears representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.42b, a visual comparison of the heatmaps generated by the algorithm using a Notears representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps Golem vs GT In Figure 5.43a, heatmaps generated by algorithm using a Golem representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.43b, a visual comparison of the heatmaps generated by the algorithm using a Golem representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

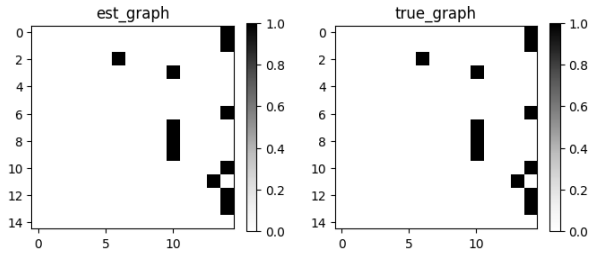


(a) Predicted graph by DIRECTLingam and Ground Truth.

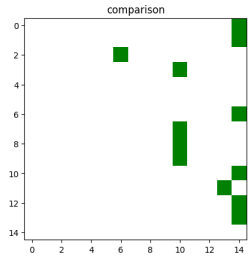


(b) Comparison of DIRECTLingam using MIM distribution with Ground Truth.

Figure 5.41.: Comparing DIRECTLingam using MIM distribution.

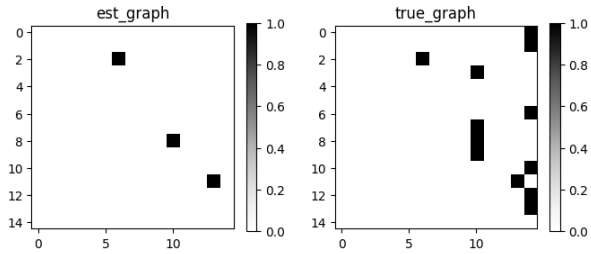


(a) Predicted graph by Notears and Ground Truth.

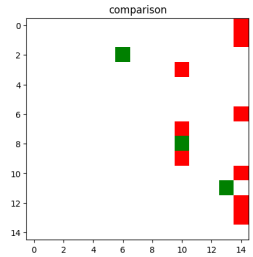


(b) Comparison of Notears using MIM distribution with Ground Truth.

Figure 5.42.: Comparing Notears using MIM distribution.



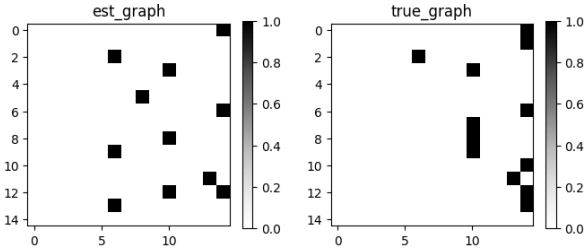
(a) Predicted graph by Golem and Ground Truth.



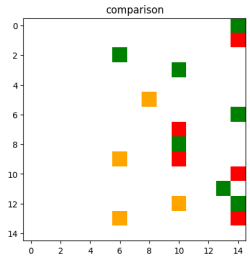
(b) Comparison of Golem using MIM distribution with Ground Truth.

Figure 5.43.: Comparing Golem using MIM distribution.

5.14. Current Dataset: Gaussian Process



(a) Predicted graph by PC and Ground Truth.



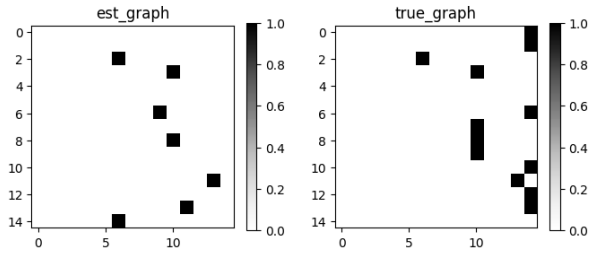
(b) Comparison of PC using GP distribution with Ground Truth.

Figure 5.44.: Comparing PC using GP distribution.

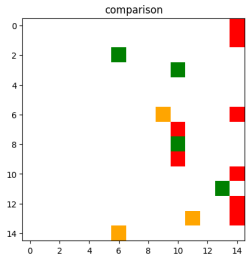
Heatmaps PC vs GT In Figure 5.44a, heatmaps generated by algorithm using a PC representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.44b, a visual comparison of the heatmaps generated by the algorithm using a PC representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps GES vs GT In Figure 5.45a, heatmaps generated by algorithm using a GES representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.45b, a visual comparison of the heatmaps generated by the algorithm using a GES representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps DIRECTLingam vs GT In Figure 5.46a, heatmaps generated by algorithm using a DIRECTLingam representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In



(a) Predicted graph by GES and Ground Truth.



(b) Comparison of GES using GP distribution with Ground Truth.

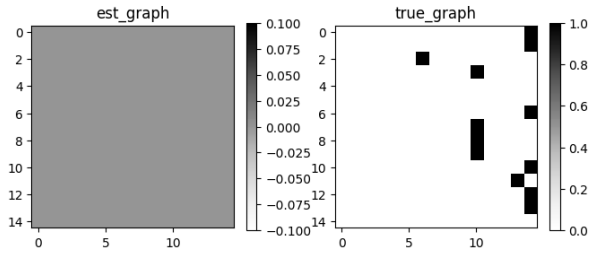
Figure 5.45.: Comparing GES using GP distribution.

Figure 5.46b, a visual comparison of the heatmaps generated by the algorithm using a DIRECTLingam representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

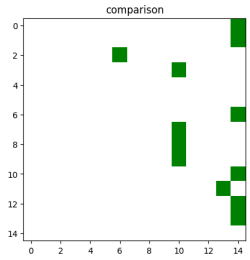
The result is obviously not correct because the algorithm did not perform properly.

Heatmaps Notears vs GT In Figure 5.47a, heatmaps generated by algorithm using a Notears representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.47b, a visual comparison of the heatmaps generated by the algorithm using a Notears representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps Golem vs GT In Figure 5.48a, heatmaps generated by algorithm using a Golem representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.48b, a visual comparison of the heatmaps generated by the algorithm using a Golem rep-



(a) Predicted graph by DIRECTLingam and Ground Truth.

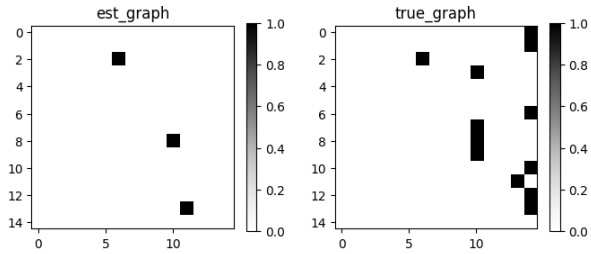


(b) Comparison of DIRECTLingam using GP distribution with Ground Truth.

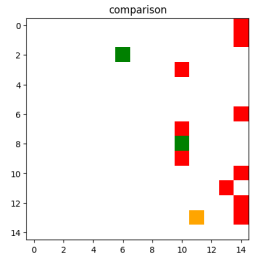
Figure 5.46.: Comparing DIRECTLingam using GP distribution.

resenting the learned adjacency matrix (est_graph) and the ground truth (true_graph).

The result is obviously not correct because the algorithm did not perform properly.

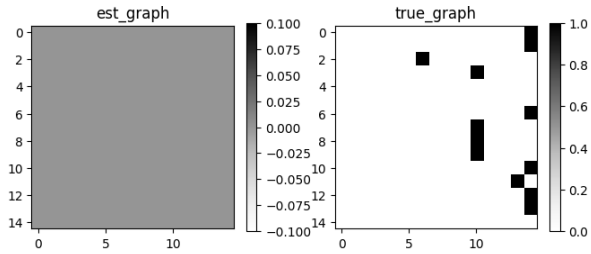


(a) Predicted graph by Notears and Ground Truth.

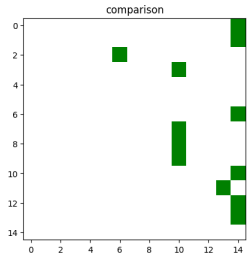


(b) Comparison of Notears using GP distribution with Ground Truth.

Figure 5.47.: Comparing Notears using GP distribution.



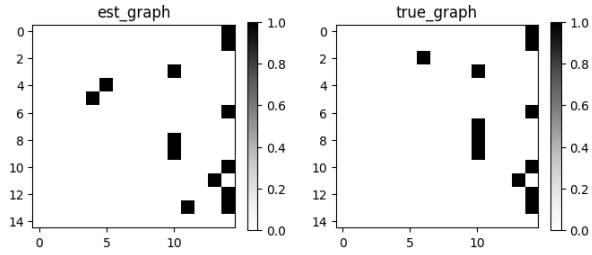
(a) Predicted graph by Golem and Ground Truth.



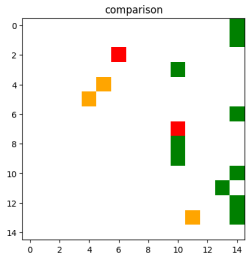
(b) Comparison of Golem using GP distribution with Ground Truth.

Figure 5.48.: Comparing Golem using GP distribution.

5.15. Current Dataset: Gaussian Process for Additive Models



(a) Predicted graph by PC and Ground Truth.



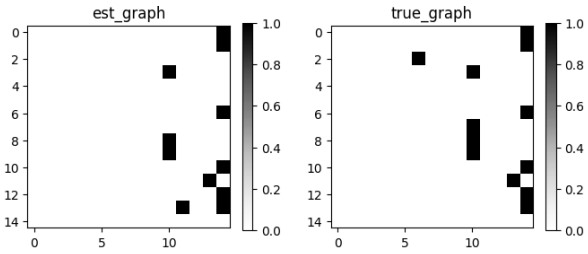
(b) Comparison of PC using GP-Add distribution with Ground Truth.

Figure 5.49.: Comparing PC using GP-Add distribution.

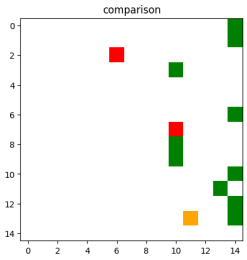
Heatmaps PC vs GT In Figure 5.49a, heatmaps generated by algorithm using a PC representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.49b, a visual comparison of the heatmaps generated by the algorithm using a PC representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps GES vs GT In Figure 5.50a, heatmaps generated by algorithm using a GES representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.50b, a visual comparison of the heatmaps generated by the algorithm using a GES representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps DIRECTLingam vs GT In Figure 5.51a, heatmaps generated



(a) Predicted graph by GES and Ground Truth.



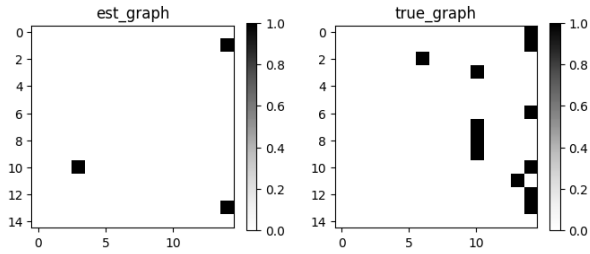
(b) Comparison of GES using GP-Add distribution with Ground Truth.

Figure 5.50.: Comparing GES using GP-Add distribution.

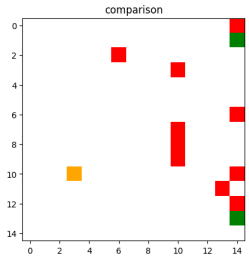
by algorithm using a DIRECTLingam representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.51b, a visual comparison of the heatmaps generated by the algorithm using a DIRECTLingam representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps Notears vs GT In Figure 5.52a, heatmaps generated by algorithm using a Notears representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.52b, a visual comparison of the heatmaps generated by the algorithm using a Notears representing the learned adjacency matrix (est_graph) and the ground truth (true_graph).

Heatmaps Golem vs GT In Figure 5.53a, heatmaps generated by algorithm using a Golem representing the learned adjacency matrix (est_graph) and the ground truth (true_graph). In Figure 5.53b, a visual comparison of the heatmaps generated by the algorithm using a Golem representing the learned adjacency matrix (est_graph) and the ground



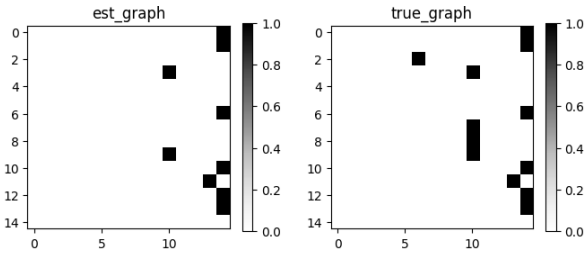
(a) Predicted graph by DIRECTLingam and Ground Truth.



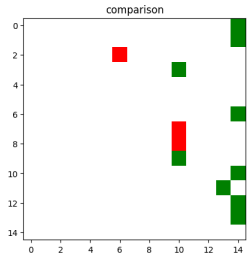
(b) Comparison of DIRECTLingam using GP-Add distribution with Ground Truth.

Figure 5.51.: Comparing DIRECTLingam using GP-Add distribution.

truth (true_graph).

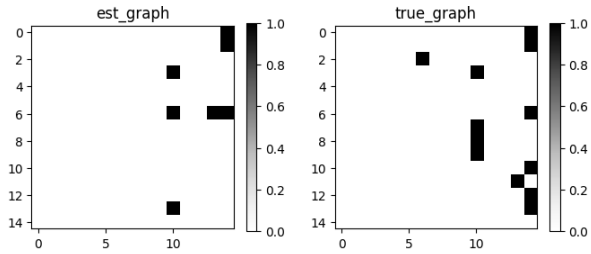


(a) Predicted graph by Notears and Ground Truth.

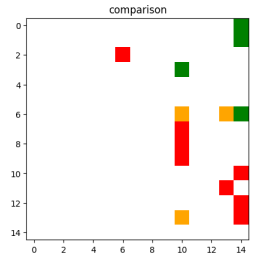


(b) Comparison of Notears using GP-Add distribution with Ground Truth.

Figure 5.52.: Comparing Notears using GP-Add distribution.



(a) Predicted graph by Golem and Ground Truth.



(b) Comparison of Golem using GP-Add distribution with Ground Truth.

Figure 5.53.: Comparing Golem using GP-Add distribution.

Chapter 6.

Ensemble Based Causal Discovery

We aim at performing causal structure learning based on the in principle discordant outcomes of several causal discovery algorithms.

Our causal discovery analysis is based on a data set of 78 records, the statistical units of our analysis. Each unit consists of a ground truth (GT) causal model represented by a Direct Acyclic Graph (DAG) and of the 5 DAGs reconstructed by as many causal discovery algorithms: GES, GOLEM, LINGAM, Notears and PC.

The data had been obtained by generating a large number of examples using a GT DAG topology with different numbers of nodes (from 9 to 30) and edges (from 7 to 435), with various functional dependencies between parent and child nodes (including linear and quadratic), and with various probability distributions of the exogenous variables, i.e., parentless (such as Linear Gaussian, Linear Exponential, Linear Uniform, Linear Gumbel, Linear Logistic, Quadratic, Multilayer Perceptron, Multiple Instance Learning, Gaussian Process and Gaussian Process for Additive Models).

6.1. The Centroid

The definition of a centroid graph and the distance from the reconstructed graphs from it was one of the elements useful in helping to select the graph closest to the GT.

For each DAG_i we computed the sum of the d-separation bitset distances between it and the other DAGs, $s_i = \sum_j d_{ij}$. The centroid DAG has been

defined as the one with the lowest sum of distances $\min_i s_i$. The DAGs corresponding to the centroid could be one or more. The dispersion of the collection of DAGs has been evaluated using both the maximum of s_i and the total sum of distances $t = \sum_i s_i$.

Thus, each record of those mentioned in the previous section was enriched with the annotation about which algorithm (or which algorithm) would play the role of centroid(s). To the record also the distances of all the algorithm specific DAGs were added.

6.2. A causality oriented definition of distance

Since we investigate the performance of causal discovery algorithms a more appropriate definition of agreement between two DAGs should be based on causal concepts, not only adjacency matrix relationships: we defined a distance based on the pair of nodes **d-separation**: given a DAG, each pair of nodes has been evaluated for d-separation, and each DAG with n nodes has been associated to a bitset of $n(n-1)/2$ bits, where each bit was set to one in case the two nodes of that pair were d-separated. The bitset were used as signatures of the DAG for pairwise comparing the DAGs. The distance $d_{ij} = d(DAG_i, DAG_j)$ between two DAGs i and j has been defined as the **Hamming distance** between their d-separation bitsets.

The distance based on the bit-set of the d-separated pairs is much more meaningful than the Hamming distance between the adjacency matrices. Indeed, reversing a single edge in one of two identical dags creates a difference of exactly two elements of the adjacency matrix, but in terms of d-separation can completely change the causal structure of the graph, affecting the d-separation of many pairs (as an illustrative example, consider two large subgraphs connected by a sequence pattern: reversing the second edge of the sequence transforms the sequence into a collider and d-separates all the pairs with one node in the first subgraph and one node in the second).

Thus, in the input data later used for training, we took the adjacency matrices information, transformed it into d-separation information for all the node pairs of a DAG, and computed the Hamming distance between bitsets. However, notice that when comparing the predictions of the DAGs

to the GT we used the Hamming distance between adjacency matrices, which informs directly about the topology of the graphs (i.e., measured the difference in terms of edge difference count).

6.3. Learning form the DAGs relationships to select the best performing DAG

In this work, we contribute a causal discovery method that takes advantage of the synthesis of topology-related information about the relationships among graphs of an ensemble captured by a suitably defined centroid graph to identify which topology is closest to the truth of the ground.

We will focus on answering the following questions.

Question 1 Which algorithm or which algorithms have correctly predicted the GT DAG?

Question 2 Which algorithm is the closest to the GT DAG?

We will find a method for addressing **Question 2**, and then look at the candidate performance thus selected to answer to **Question 1**.

To this purpose, we use a supervised learning method.

Notice that the DAGs generated by the discovery algorithms can sometimes coincide with one another, whether they correspond to the ground truth or not: thus, there is not necessarily a single correct answer to the question of which is the ground truth or which is the algorithm closest to the ground truth. As a consequence, we formulate the problem as a **multilabel classification** task (each algorithm plays the role of a label, there are also 6 labels, counting the CENTROID graph). Then we use two common approaches to address this type of classification: the **Binary Relevance** (BR) approach and the **Classifier Chain** (CC) approach. After choosing a representative from the set of predicted labels, using a classifier-precision-based criterion, we used its distance from the GT (defined as the Hamming distance between adjacency matrices, i.e., in terms of edges). Notice that since we are using a distance based selection methodology, there might be

ties between several solutions. This implies that we are not dealing with a classification problem, where clear classes are defined, but we have to guess a topologic structure.

6.4. Features Description

6.4.1. Dataset generation related variables

The variables characterizing each dataset were those described in the following; only the ones with a name not ending by an underscore were available to the training algorithms. Number of nodes ($NNODES$, integer), see Figure 5.1, Number of possible pairs ($NPOSSIBLE\_PAIRS$, integer), Number of actual edges ($NACTUAL\_EDGES$, integer), see Figure 5.2, Linear or nonlinear functional dependence ($Linear\_$, possible values 'linear' and 'nonlinear'), statistical distribution used for extrinsic variables (distribution).

6.4.2. Unlabeled part of the dataset

This part of the dataset contains, for each statistical unit, those variables describing the **relationships among the DAGs** as reconstructed by the five causal discovery algorithms (at least one of which is identified as **centroid**).

The variables describing those DAG distances and their quality of centroid are the following (notice that [ALGO] can take the values GES, GOLEM, LINGAM, Notears, PC):

```
[ALGO]_SUM_DIST (integer): sum of the hamming distances
    between the DAG yielded by the [ALGO] algorithm and the
    other DAGs.
[ALGO]_SUM_DIST_STAND (real): previous variable divided
    by NPOSSIBLE_PAIRS.
[ALGO]_IS_CENTROID (Boolean): whether the DAG
    is the centroid.
```

See Figure 6.1 and Figure 6.2.

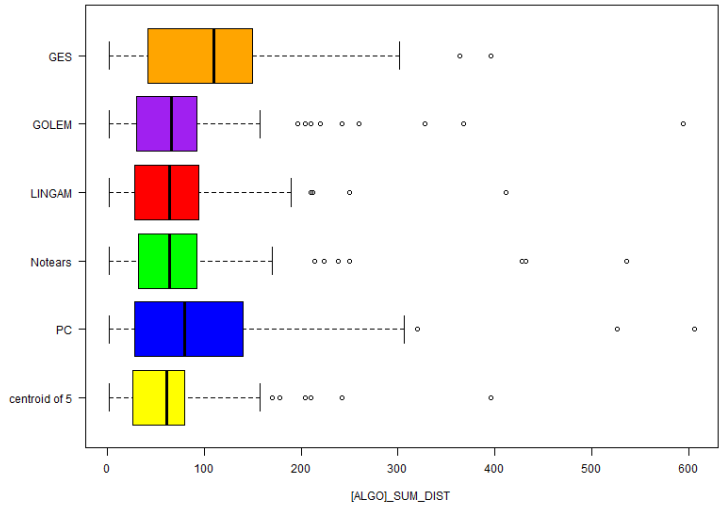


Figure 6.1.: Boxplot of [ALGO]_SUM_DIST.

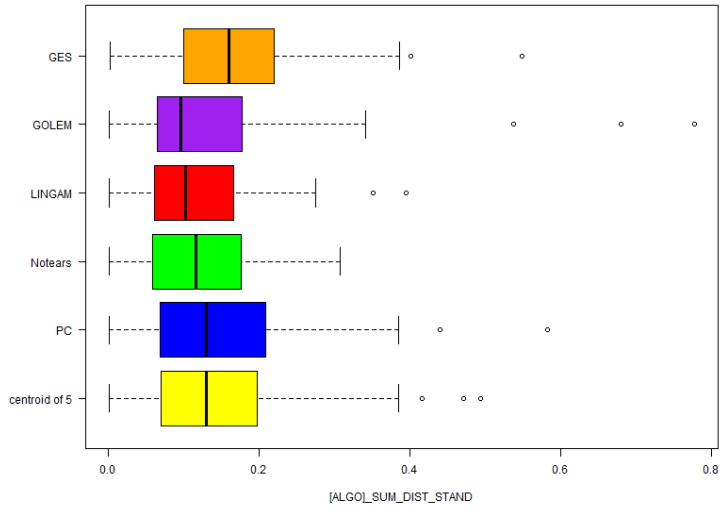


Figure 6.2.: Boxplot of [ALGO]_SUM_DIST_STAND.

Then we had variables describing the ensemble of causal discovery DAGs:

```
CENTROIDS_COUNT (positive integer): number of algorithms
    taking the centroid role (from 1 to 5)
MAX_SUM_DIST (integer): maximum of the sums of distances
    across DAGs
MAX_SUM_DIST_STAND (real): previous variable divided by
    NPOSSIBLE_PAIRS
CENTROID_SUM_DIST (integer): sums of distances for the
    centroid
CENTROID_SUM_DIST_STAND (real): previous variable divided
    by NPOSSIBLE_PAIRS
```

6.4.3. Labels and GT related variables

A number of variables were used to describe the **relationship of the individual algorithms DAGs to the GT DAG**. We recall that the distances were computed as the Hamming distance between the adjacency matrices respectively of [ALGO] and GT. Notice that here the standardization is performed using the actual number of edges present in the GT DAG, no longer using the number of possible node pairs. Here the values taken by [ALGO] were the 5 algorithms in the set GES, GOLEM, LINGAM, Notears, PC plus CENTROID. Typically, one or more of these variables, which are not available to the learned predictor, are used as predictor targets.

```
[ALGO]_GT_DIST_ (integer): see above
[ALGO]_GT_DIST_STAND_ (real): previous variable divided
    by NACTUAL_EDGES
[ALGO]_IS_GT_ (Boolean): whether [ALGO] has made a
    correct prediction
```

Then we have variables describing the ensemble of causal discovery DAGs

```
NUM_OF_CORRECT_ (integer): n. of correct predictions
    by the set of 5 algorithms
MIN_DIST_TO_GT_ (integer): minimum distance, when 0 the
    prediction was correct
MIN_DIST_TO_GT_STAND_ (real): previous variable divided
    by NACTUAL_EDGES
[ALGO]_IS_MIN (Boolean): whether [ALGO]'s prediction
```

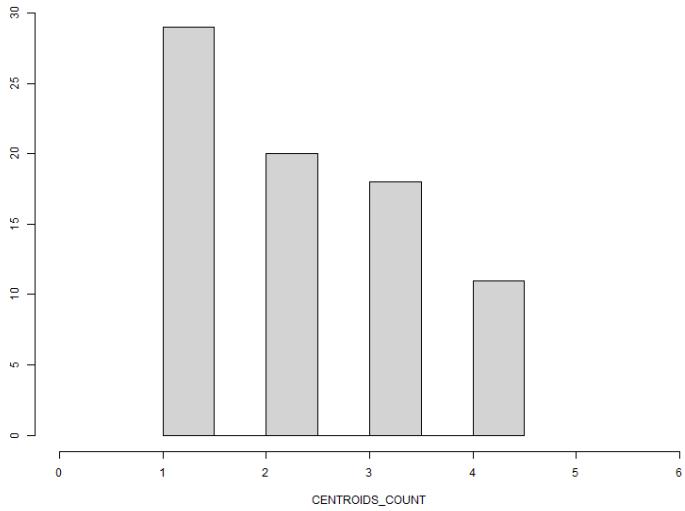


Figure 6.3.: Histogram of CENTROID_COUNT.

```
    corresponds to the minimum distance from GT
MAX_DIST_TO_GT_ (integer): maximum distance
MAX_DIST_TO_GT_STAND_ (real): previous variable divided
    by NACTUAL_EDGES
```

We want to find whether the knowledge from the ensemble can be extracted so as to obtain **ensemble predictions** which are better than the individual algorithms' predictions, and in affirmative case we want to find out what level of improvement can be achieved.

We will check in a preliminary analysis that at least an algorithm (out of the 5 considered) made a DAG prediction coinciding with the GT: the answer will turn out to be positive in about two thirds of the times.

6.4.4. Relationships among DAGs, not considering the Ground Truth

Here are the percentages of the datasets for which the algorithm is the CENTROID: GES 15%, GOLEM 63%, LINGAM 63%, Notears 45%, PC 28%. There is often more than one centroid as shown in Figure 6.3.

6.4.5. Features of the Ensemble relative to the Ground Truth

Here we report about the relationship between the DAGs issued by the different algorithms and the GT DAGs. This helps contextualize each question: from the data it is clear that there are potentialities in the exploitation of the knowledge of the ensemble.

6.4.5.1. Correct/incorrect reconstruction of the GT DAG

The different algorithm provides the perfectly correct solution with the following percentages. In the same table is reported the number of times at least one algorithm makes a perfectly correct prediction (for convenience the rightmost column anticipates also the results of our method, which will be discussed later). See Table 6.1.

Table 6.1.: Correct/incorrect reconstruction of the GT DAG.

% [ALGO] IS GT	GES	GOLEM	LINGAM	Notears	PC	CENTROID	At least 1 correctly predicts GT	Algorithm BR selected	Algorithm CC selected
n. times True	8	37	35	28	11	38	50	48	49
n. cases	78	78	78	78	78	78	78	78	78
% True	10.26	47.44	44.87	35.90	14.10	48.72	64.10	61.54	62.82
stderr	3.55	5.84	5.82	5.61	4.07	5.85	5.61	5.69	5.66
Lower 95% interval	3.30	35.98	33.46	24.89	6.12	37.25	53.10	50.38	51.73
Upper 95% interval	17.22	58.89	56.28	46.90	22.09	60.18	75.11	72.70	73.91

It is possible to observe that in terms of the number of correct guesses (i.e., in terms of precision) the CENTROID prevails over the others, but the difference w.r.t. the second best, here GOLEM, and w.r.t LINGAM and Notears, **is not statistically significant**. GES and PC are significantly less performing than the others.

No individual algorithm among the best – GOLEM, LINGAM and Notears – outperforms overall the others, nor selecting the centroid DAG is in average significantly better that the individual algorithms.

Nonetheless, from the Table 6.1 it is possible to see that in principle there is room for improving the results, by a suitable case by case selection of the algorithm. If an oracle were able to point to the best algorithm(s) for a specific dataset one would achieve 64.10% **precision. Thus there is room for approximately** 15% **improvement.**

6.4.5.2. Distance variables

The picture becomes more precise as we pass from the Boolean variable $[ALGO]\_IS\_GT\_$ to the distance variables $[ALGO\_GT\_DIST\_]$.

Looking at the box plot of $[ALGO]\_GT\_DIST$ in Figure 6.4 it is possible to observe that GOLEM, LINGAM and CENTROID are again the best performing algorithms. In particular, the CENTROID seems to slightly outperform the others.

The numerical summary of the variables is the following (for convenience the two rightmost columns anticipates also the results of our method, which will be discussed later).

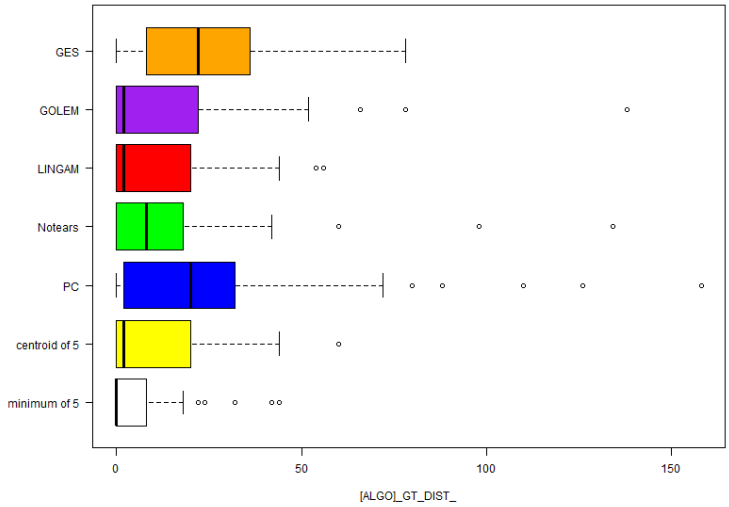


Figure 6.4.: Boxplot of [ALGO]_GT_DIST.

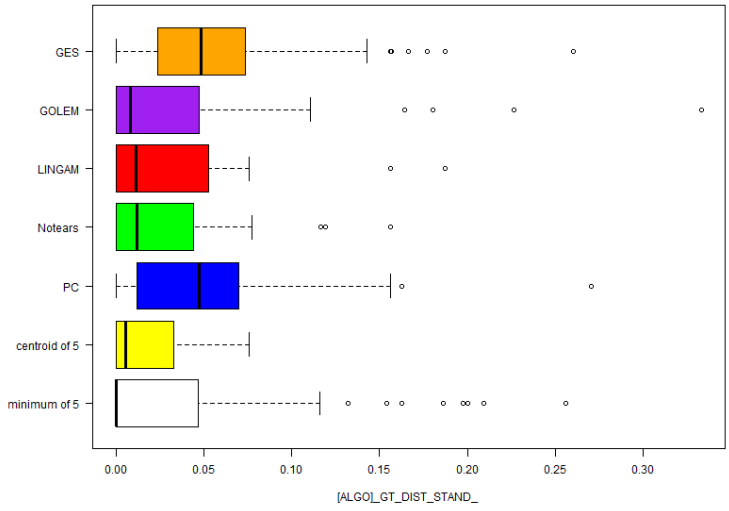


Figure 6.5.: Boxplot of [ALGO]_GT_DIST_STAND.

Table 6.2.: Distance variables

[ALGO] GT DIST	GES	GOLEM	LINGAM	Notears	PC	CENTROID	Distance from GT of the closest algorithm	Distance from GT of BR selected	Distance from GT of CC selected
Min:	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Ist Qu.:	8.50	0.00	0.00	0.00	2.50	0.00	0.00	0.00	0.00
Median :	22.00	2.00	2.00	8.00	20.00	2.00	0.00	0.00	0.00
Mean :	24.49	13.41	10.59	13.08	25.62	10.08	5.38	6.62	6.36
3rd Qu.:	36.00	21.50	20.00	18.00	32.00	19.00	8.00	9.50	8.00
Max:	78.00	138.00	56.00	134.00	158.00	60.00	44.00	44.00	44.00
Std:	19.50	22.57	14.76	20.84	29.21	14.34	10.36	11.40	11.36
Stderr:			1.67			1.62	1.17	1.29	1.32
Lower end 95% interval			7.31			6.89	3.09	4.09	3.78
Upper end 95% interval			13.87			13.26	7.68	9.14	8.94

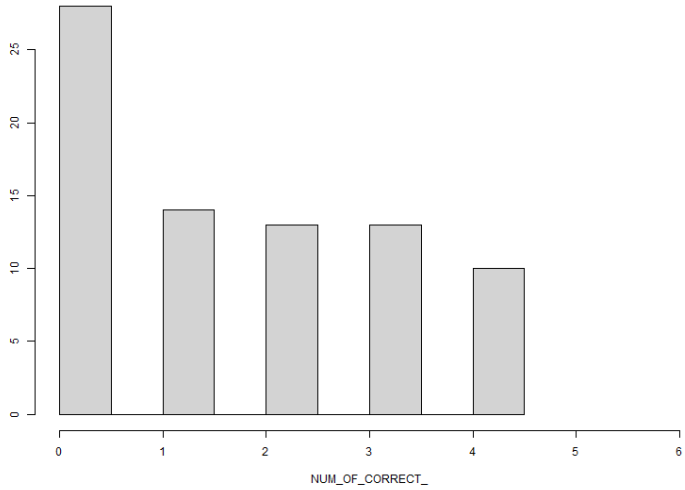


Figure 6.6.: Histogram of the number of correct predictions.

The difference in means between CENTROID and the second best **is not statistically significant** already from the fact that the two means lie largely inside each other's 95% confidence intervals. Performing a one-tailed Student t-Test we find out that the two means are not significantly different from one another at the 0.05 confidence level: indeed, the p-value turns out to be 0.4130. So we cannot reject the hypothesis that the observed prevalence of CENTROID over the second best is just due to chance.

Similar considerations hold for the variable $[ALGO]\_GT\_DIST\_STAND\_$, obtained from $[ALGO]\_GT\_DIST\_$ dividing by the number of actual edges (for the sake of clarity, we are not reporting the standardized distance data here). The boxplots for the variable $[ALGO]\_GT\_DIST\_STAND\_$ are shown in Figure 6.5.

Therefore, neither individual algorithm outperforms in average the others, nor a simple rule such as the one of selecting the centroid DAG is in average significantly better than the individual algorithms.

However, one can see that if one were able to select case by case the algorithm with the minimum distance from the GT, the performance would be significantly improved, see Figures 6.4 and 6.5.

$[ALGO]\_IS\_MIN\_$	GES	GOLEM	LINGAM	Notears	PC	CENTROID
n. of times True	10	44	43	**49**	12	48
n. of cases	78	78	78	**78**	78	78
Percentage True	**12.82**	**56.42**	**55.13**	**62.82**	**15.38**	**61.54 %**
stderr	4.25	6.30	6.32	**6.14**	4.58	6.18

Table 6.3.: The algorithms performance.

Looking at the number of times a specific algorithm turns out to be the closest to GT sheds further light on the performance of the candidates. LINGAM, Notears, and CENTROID are the best performing, see Table 6.3.

Despite the prevalence of the three mentioned algorithms, appears to be not a simple strategy to select a case-by-case algorithm so as to **minimize the average distance** of the selected algorithm from the GT: for instance choosing always Notears, would get the minimum distance about 63% of the times, but in the other 37% of the times the resulting distance could be far from minimal, indeed (from the distance table) we know that the average distance of Notears would be 13.10, i.e., far above the theoretically allowed minimum of 5.38.

For these reasons, address the task of selecting **case by case** which algorithm is the closest to GT given only the input variables by using a supervised ML approach.

6.5. The ML task

Given the input variables that describe the relative distances based on d-separation between pairs of DAGs, we want to select the algorithms/DAGs closest to the ground truth (possibly coinciding with it). Several DAGs at the same time have the same distance from the GT (or even coincide with it), thus several labels can coexist.

The problem can be cast in a multi-label classification problem, with the peculiarity that the prediction can be considered right if even just one of the correct labels is designated by the predictor.

To address the task, we set up two approaches: One built as a variant of **Binary Relevance** (BR) The other as a variant of **Classifier Chain** (CC)

BR works by training a separate binary classifier – sometimes called detector - for each label. Each classifier predicts whether the corresponding label applies or not.

CC works by training a separate binary classifier for each label, similar to BR, but also taking into account the predictions of previous classifiers in the chain: for each label in the chain, it trains a binary classifier using the input features and the binary labels of all preceding labels in the chain as additional input features. For example, when training the classifier for the third label in the chain, the input features would include the original input features as well as the binary predictions (0 or 1) of the first and second labels in the chain.

BR has the drawback that it does not incorporate possible label dependencies: in our case, if two algorithms yield the same value of $[ALGO]\_SUM\_DIST$ their DAGS coincide (in the d-separation sense); despite this objective condition the two independently trained detectors could in principle predict inconsistent results (one predicting that the first label applies, the other that the second does not apply, despite both do).

CC has some issues too: Error Propagation and Order Sensitivity. If a classifier makes a mistake in predicting a label early in the chain, it can lead to cascading errors in the predictions of subsequent labels, potentially reducing overall performance. The choice of label ordering can impact the predictive performance of the model, while finding the optimal ordering can be computationally expensive (in our case 6 possible labels would entail 720 possible orderings). We address the latter point by ordering the classifier chain in **decreasing order of accuracy** (the accuracy was determined empirically when running BR, and we used the decreasing order of precision in case of break-even).

Both BR and CC produce for each case a set of labels that apply to that statistical unit. The variant we use in both cases consists in the final

choice of **a single representative of the predicted label set**. In fact, eventually, we want to take the output of the BR algorithm and choose exactly one of the labels (then take its distance to answer **Question 2**, and see whether it coincides with GT to answer **Question 1**).

The chosen criterion selects the [*ALGO*] among the candidates predicted by the individual detectors to be closest to the GT is based on precision and is described below.

6.5.1. The AutoML (Automated Machine Learning) approach

To choose an ideally optimal binary classifier, we adopted an **AutoML** approach, for model selection and hyperparameter optimization, using the *Pycaret* Python libraries (pycaret.org). This library automates the entire learning pipeline, from data pre-processing to model deployment, making it easier and faster to build high-performance classification models.

We decided to keep the default optimization strategy, aimed at maximizing the accuracy through Bayesian Optimization (BO). BO uses probabilistic modeling of the solution space to efficiently find the optimum of a costly black-box objective function (in our case the ML model test performance) by iteratively selecting the most promising candidates to evaluate (in our case the data subsets) while balancing exploration and exploitation.

6.5.2. Selection criterion

The selection of [ALGO] among the candidate labels predicted by the detectors works as follows. We start looking at the [ALGO] whose detector has the highest test precision as evaluated by the optimizer (in case of break-even, we use decreasing order of cardinality of the true positives), and if that [ALGO] is predicted True, we assign that label to the statistical unit; otherwise, we look at the [ALGO] with second best precision, and so on.

If no [ALGO] detector predicts TRUE (i.e., if $COUNT\_MIN\_PRED = 0$), we adopt as default the [ALGO] that performs best when all the other detectors predict FALSE. In our case, the Notears detector significantly stands out as the best with respect to this performance criterion: in the 17 cases where a single [ALGO] is predicted to be at the minimum distance to

GT, where [ALGO] corresponds to Notears 10 times, Lingam 5, Golem and CENTROID 1, GES and PC 0. Thus, we adopt Notears as default. This choice turns out to be the best 4 times out of 5 (5 are statistical units where no detector predicts that its [ALGO] is the minimum distance one).

6.6. Outcomes

6.6.1. Outcomes of Binary Relevance (BR) approach

We searched for the optimal binary classifier detectors of each [ALGO] using the AutoML approach described above.

The base set of input variables is in the following:

```
X ={'NNODES', 'NPOSSIBLE_PAIRS', 'GES_SUM_DIST',
'GES_SUM_DIST_STAND', 'GES_IS_CENTROID', 'GOLEM_SUM_DIST',
'GOLEM_SUM_DIST_STAND', 'GOLEM_IS_CENTROID', 'LINGAM_SUM_DIST',
'LINGAM_SUM_DIST_STAND', 'LINGAM_IS_CENTROID', 'Notears_SUM_DIST',
'Notears_SUM_DIST_STAND', 'Notears_IS_CENTROID', 'PC_SUM_DIST',
'PC_SUM_DIST_STAND', 'PC_IS_CENTROID', 'CENTROIDS_COUNT',
'MAX_SUM_DIST', 'MAX_SUM_DIST_STAND', 'CENTROID_SUM_DIST',
'CENTROID_SUM_DIST_STAND', 'AVE_SUM_DIST_of_5', 'AVE_SUM_DIST_STAND_of_5',
'STD_SUM_DIST_of_5', 'STD_SUM_DIST_STAND_of_5', 'SKW_SUM_DIST_of_5'}
```

The highest accuracy configurations issued by the process were the following.

```
Input variables X
Target variable CENTROID_IS_MIN_
Output variable CENTROID_IS_MIN_PRED

RidgeClassifier(alpha=1.0,
                class_weight=None,
                copy_X=True,
                fit_intercept=True,
                max_iter=None,
                positive=False,
                random_state=7906,
                solver='auto',
                tol=0.0001)
```

```
Input variables X
Target variable GES_IS_MIN_
Output variable GES_IS_MIN_PRED
XGBClassifier(base_score=None,
              booster='gbtree',
              callbacks=None,
              colsample_bylevel=None,
              colsample_bynode=None,
              colsample_bytree=None,
              device='cpu',
              early_stopping_rounds=None,
              enable_categorical=False,
              eval_metric=None,
              feature_types=None,
              gamma=None,
              grow_policy=None,
              importance_type=None,
              interaction_constraints=None,
              learning_rate=None,
              max_bin=None,
              max_cat_threshold=None,
              max_cat_to_onehot=None,
              max_delta_step=None,
              max_depth=None,
              max_leaves=None,
              min_child_weight=None,
              missing=nan,
              monotone_constraints=None,
              multi_strategy=None,
              n_estimators=None,
              n_jobs=-1,
              num_parallel_tree=None,
              objective='binary:logistic',
              ...)

Input variables X
Target variable GOLEM_IS_MIN_
Output variable GOLEM_IS_MIN_PRED

ExtraTreesClassifier(bootstrap=False,
                     ccp_alpha=0.0,
                     class_weight=None,
                     criterion='gini',
                     max_depth=None,
                     max_features='sqrt',
                     max_leaf_nodes=None,
                     max_samples=None,
                     min_impurity_decrease=0.0,
                     min_samples_leaf=1,
```

```
                    min_samples_split=2,
                    min_weight_fraction_leaf=0.0,
                    n_estimators=100,
                    n_jobs=-1,
                    oob_score=False,
                    random_state=4223,
                    verbose=0,
                    warm_start=False)

Input variables X
Target variable LINGAM_IS_MIN_
Output variable LINGAM_IS_MIN_PRED

ExtraTreesClassifier(bootstrap=False,
                    ccp_alpha=0.0,
                    class_weight=None,
                    criterion='gini',
                    max_depth=None,
                    max_features='sqrt',
                    max_leaf_nodes=None,
                    max_samples=None,
                    min_impurity_decrease=0.0,
                    min_samples_leaf=1,
                    min_samples_split=2,
                    min_weight_fraction_leaf=0.0,
                    n_estimators=100,
                    n_jobs=-1,
                    oob_score=False,
                    random_state=2990,
                    verbose=0,
                    warm_start=False)

Input variables X
Target variable Notears_IS_MIN_
Output variable Notears_IS_MIN_PRED

ExtraTreesClassifier(bootstrap=False,
                    ccp_alpha=0.0,
                    class_weight=None,
                    criterion='gini',
                    max_depth=None,
                    max_features='sqrt',
                    max_leaf_nodes=None,
                    max_samples=None,
                    min_impurity_decrease=0.0,
                    min_samples_leaf=1,
                    min_samples_split=2,
                    min_weight_fraction_leaf=0.0,
                    n_estimators=100,
```

```
                      n_jobs=-1,
                      oob_score=False,
                      random_state=3789,
                      verbose=0,
                      warm_start=False)

Input variables X
Target variable PC_IS_MIN_
Output variable PC_IS_MIN_PRED

DecisionTreeClassifier(ccp_alpha=0.0,
                       class_weight=None,
                       criterion='gini',
                       max_depth=None,
                       max_features=None,
                       max_leaf_nodes=None,
                       min_impurity_decrease=0.0,
                       min_samples_leaf=1,
                       min_samples_split=2,
                       min_weight_fraction_leaf=0.0,
                       random_state=7051, splitter='best)
```

In Table 6.4 the test precision and recall reported are averaged in the positive and negative classes.

Table 6.4.: Correct/incorrect reconstruction of the DAG.

Target_IS_MIN	Model	Accuracy	AUC	Recall	Precision	F1	Kappa	MCC
CENTROID	Ridge Classifier	0.8700	0.0000	**0.8750**	**0.9300**	**0.8860**	**0.7296**	**0.7659**
GES	Extreme Gradient Boosting	0.9033	0.6800	0.4000	0.4000	0.4000	nan	0.4000
GOLEM	Extra Trees Classifier	0.8400	0.8028	0.8667	0.8500	0.8514	0.6782	0.6914
LINGAM	Extra Trees Classifier	0.8000	0.8000	0.8667	0.8183	0.8312	0.5879	0.6100
Notears	Extra Trees Classifier	0.7233	**0.8125**	0.8250	0.7717	0.7785	0.3815	0.4061
PC	Decision Tree Classifier	**0.9833**	0.7900	0.8000	0.7500	0.7667	nan	0.7632

Individual detectors have shown the following performances on the full data set (number of instances 78).

Table 6.5.: Performance of Binary Relevance detectors

Binary Relevance	CENTROID	GES	GOLEM	LINGAM	Notears	PC
TP	43	7	42	39	45	10
FP	15	0	4	4	2	0
FN	5	3	2	4	4	4
TN	25	68	30	31	27	66
Precision %	89.58	**100.00**	91.30	90.70	95.74	**100.00**
Recall %	89.58	70.00	**95.45**	90.70	91.84	83.33
Accuracy %	87.18	96.15	92.31	89.74	92.31	**97.44**

We took the predictions from the binary classifier detectors and selected [ALGO] as described above. The ranking in precision used by the aggregator function was PC, GES, Notears, GOLEM, Lingam, and CENTROID. The results shown also in Table 6.1 and Table 6.2 are the following.

Although the lowest mean individual [ALGO]'s distance was CENTROID's **10.08** the mean of the selected [ALGO]'s distance using BR was 6.62, very close to the actual average minimum distance **5.38**: this result represents a significant improvement as shown by the standard errors and 95% confidence intervals reported in Table 6.2: comparing the selected algorithm's and CENTROID's distance through a one-tailed t-Student test we find that the former distance is lower than the latter with a **p-value** of **0.0487**, hence below the conventional 95% ($\alpha = 0.05$) confidence level threshold.

Expectedly, the selection of [ALGO] according to the above method improves also the prediction of which [ALGO] perfectly corresponds to the GT. The results are summarized in Table 6.2. Although the single most effective individual [ALGO] in correctly predicting GT was CENTROID with correct predictions 48.72%, the [ALGO] selected using BR is correct 61.54% of the time, very close to the actual percentage of times when at least one algorithm in the group guesses correctly, which is 64.10%.

6.6.2. Outcomes of Classifiers Chain (CC) approach

Using the incremental approach to multilabel classification provided by CC (ordering of targets in decreasing accuracy of their individual detectors:

```
PC_IS_MIN_, GES_IS_MIN_, Notears_IS_MIN_,
GOLEM_IS_MIN_,LINGAM_IS_MIN_, CENTROID_IS_MIN_),
```

the configurations of the highest accuracy issued by the AutoML process turned out to be the following.

```
Input variables X
Target variable PC_IS_MIN_
Output variable PC_IS_MIN_PRED
RidgeClassifier(alpha=1.0,
                class_weight=None,
                copy_X=True, f
                it_intercept=True,
```

```
                  max_iter=None,
                  positive=False,
                  random_state=7467,
                  solver='auto',
                  tol=0.0001)

Input variables {X, PC_IS_MIN_PRED},
Target variable GES_IS_MIN_
Output variable GES_IS_MIN_PRED
XGBClassifier(base_score=None,
              booster='gbtree',
              callbacks=None,
              colsample_bylevel=None,
              colsample_bynode=None,
              colsample_bytree=None,
              device='cpu',
              early_stopping_rounds=None,
              enable_categorical=False,
              eval_metric=None,
              feature_types=None,
              gamma=None,
              grow_policy=None,
              importance_type=None,
              interaction_constraints=None,
              learning_rate=None,
              max_bin=None,
              max_cat_threshold=None,
              max_cat_to_onehot=None,
              max_delta_step=None,
              max_depth=None,
              max_leaves=None,
              min_child_weight=None,
              missing=nan,
              monotone_constraints=None,
              multi_strategy=None,
              n_estimators=None,
              n_jobs=-1,
              num_parallel_tree=None,
              objective='binary:logistic',
              ...)

Input variables {X, PC_IS_MIN_PRED, GES_IS_MIN_ PRED}
Target variable Notears_IS_MIN_
Output variable Notears_IS_MIN_PRED
```

```
RandomForestClassifier(bootstrap=True,
                       ccp_alpha=0.0,
                       class_weight=None,
                       criterion='gini',
                       max_depth=None,
                       max_features='sqrt',
                       max_leaf_nodes=None,
                       max_samples=None,
                       min_impurity_decrease=0.0,
                       min_samples_leaf=1,
                       min_samples_split=2,
                       min_weight_fraction_leaf=0.0,
                       n_estimators=100,
                       n_jobs=-1,
                       oob_score=False,
                       random_state=3346,
                       verbose=0,
                       arm_start=False)

Input variables {X, PC_IS_MIN_PRED,
                    GES_IS_MIN_PRED,
                    Notears_IS_MIN_PRED}
Target variable GOLEM_IS_MIN_
Output variable GOLEM_IS_MIN_PRED
ExtraTreesClassifier(bootstrap=False,
                     ccp_alpha=0.0,
                     class_weight=None,
                     criterion='gini',
                     max_depth=None,
                     max_features='sqrt',
                     max_leaf_nodes=None,
                     max_samples=None,
                     min_impurity_decrease=0.0,
                     min_samples_leaf=1,
                     min_samples_split=2,
                     min_weight_fraction_leaf=0.0,
                     n_estimators=100,
                     n_jobs=-1,
                     oob_score=False,
                     random_state=1692,
                     verbose=0,
                     warm_start=False)


Input variables {X, PC_IS_MIN_PRED,
```

```
                    GES_IS_MIN_PRED,
                    Notears_IS_MIN_PRED,
                    GOLEM_IS_MIN_PRED }
Target variable LINGAM_IS_MIN_
Output variable LINGAM_IS_MIN_PRED

ExtraTreesClassifier(bootstrap=False,
ccp_alpha=0.0, class_weight=None,
                     criterion='gini',
                     max_depth=None,
                     max_features='sqrt',
                     max_leaf_nodes=None,
                     max_samples=None,
                     min_impurity_decrease=0.0,
                     min_samples_leaf=1,
                     min_samples_split=2,
                     min_weight_fraction_leaf=0.0,
                     n_estimators=100,
                     n_jobs=-1,
                     oob_score=False,
                     random_state=8154,
                     verbose=0,
                     warm_start=False)

Input variables {X,
                 PC_IS_MIN_PRED,
                 GES_IS_MIN_PRED,
                 Notears_IS_MIN_PRED,
                 GOLEM_IS_MIN_PRED,
                 LINGAM_IS_MIN_PRED}
Target variable CENTROID_IS_MIN_
Output variable CENTROID_IS_MIN_PRED

RidgeClassifier(alpha=1.0,
                 class_weight=None,
                 copy_X=True,
                 fit_intercept=True,
                 max_iter=None,
                 positive=False,
                 random_state=7184,
                 solver='auto',
                 tol=0.0001)
```

The individual detectors have shown the following performances on the full

data set (number of instances 78).

We took the predictions of the incrementally trained binary classifiers and selected the [ALGO] as described above. The precision-based ranking used by the selector function was GES, Notears, GOLEM, PC, Lingam, CENTROID. The outcomes displayed also in Table 6.1 and 6.2 are the following.

We recall that the lowest mean individual [ALGO]'s distance was CENTROID's **10.08**. The mean of the selected [ALGO]'s distance using CC was **6.36**, very close to the actual average minimum distance **5.38**: this result represents a significant improvement as shown by the standard errors and 95% confidence intervals reported in Table 6.2: comparing the selected algorithm's and CENTROID's distance through a one-tailed t-Student test we find that the former distance is lower than the latter with a p-value of 0.0387, hence below the conventional 95% ($\alpha = 0.05$) confidence level threshold. The CC result was slightly better than the BR result (6.62) but was not significantly different from it.

As with BR, the selection of the [ALGO] according to CC improves also the prediction of which [ALGO] perfectly corresponds to the GT. The results are summarized in the Table 6.2. While the single most effective individual [ALGO] in correctly guessing the GT was CENTROID with **48.72%** correct predictions, the [ALGO] selected using BR is correct **62.82%** of the time, very close to the actual percentage of times when at least one algorithm in the group guesses correctly, which is **64.10%**. Also in this respect the result from CC was slightly better than the result by BR (**61.54**) but not significantly different from it.

Chapter 7.

Causal Discovery on Real Data

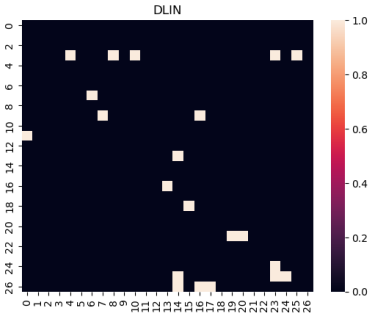
This chapter uses the real data set to apply causal inference. We refer to the preprocessing steps to the Algorithm 1. We finally obtained 27 features. Each model PC, DirectLiNGAM, GES, GOLEM and Notears was trained on the preprocessed data thus to obtain each corresponding adjacency matrix. From Figure 7.1b to Figure 7.1e it is possible to see the heatmap of the adjacency matrix obtained for each model.

For each graph we mapped the nodes with the same labels, the features of the dataset. For each graph, we calculated the d-separation matrix iterating on every couple of nodes. We then calculate a distance matrix between each graph by comparing their d-separation matrices and measuring their distance element-wise. We choose as the central graph, the one with the minimum distance from the others.

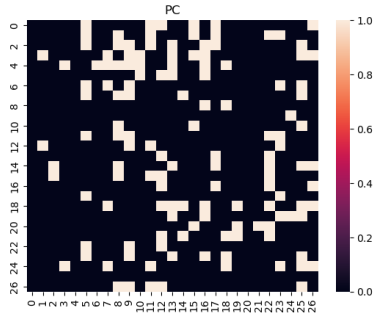
7.1. Centroid Graph and Causal Inference by using Real Data

Based on the central graph given by the GES algorithm we built the Causal Model object. And we used the DoWhy tool to automatically find estimands for the graph. Then the estimands can be identified based on the provided DoWhy methods: Back-door; Front-door; Instrumental variable. The `.identify_effect()` method provides the estimands. we then obtained the estimates based on `.estimate_effect()` and calculate the value of the Estimate of the causal effect.

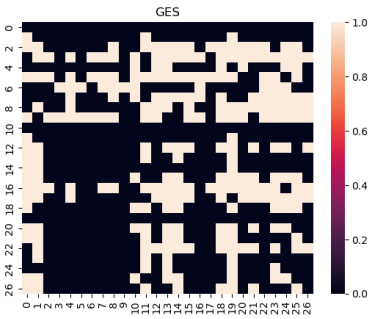
In the following results the variables' discussion by integrating the **domain**



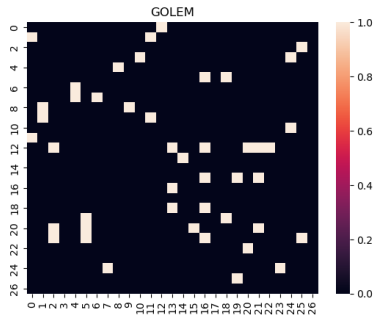
(a) DLIN algorithm.



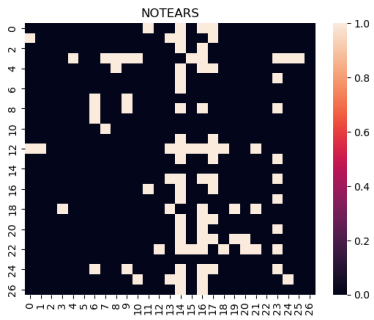
(b) PC algorithm.



(c) GES algorithm.



(d) GOLEM algorithm.



(e) NOTEARS algorithm.

Figure 7.1.: The Heatmaps representing the adjacency matrices obtained by DLIN, PC, GES, GOLEM and NOTEARS algorithms applied on real data.

expert knowledge. we will describe step by step the simulator's outputs of the variables starting from showing the code snippets and commenting each single part.

7.1.1. The analysis walkthrough of the $trf\_m\_\_var\_57$ variable.

In the following the DoWhy output code snippet related to the variable $trf\_m\_\_var\_57$.

```
Estimand type: EstimandType.NONPARAMETRIC_ATE
### Estimand : 1
Estimand name: backdoor
Estimand expression:
```

$$\frac{d}{d[trf\_m\_\_var\_57]}\left(E[\text{target}]\right)$$

Estimand assumption 1, Unconfoundedness:
If $U \rightarrow trf\_m\_\_var\_57$ and $U \rightarrow target$ then
$P(target \mid trf\_m\_\_var\_57, , U)$
$= P(target \mid trf\_m\_\_var\_57,)$

```
### Estimand : 2
Estimand name: iv
No such variable(s) found!
```

```
### Estimand : 3
Estimand name: frontdoor
Estimand expression:
```

$$E[\frac{d}{d[cns\_m\_\_var\_17]}\left(target\right) \cdot \frac{d}{d[trf\_m\_\_var\_57]}\left([cns\_m\_\_var\_17]\right)]$$

7.1.1.1. Estimand 1: Backdoor.

```
Estimand assumption 1, Unconfoundedness:
```

If $U \rightarrow TRF$ and $U \rightarrow target$ then $P(target \mid TRF, U) = P(target \mid TRF)$

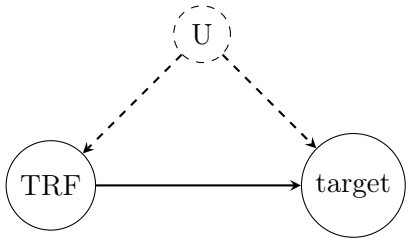


Figure 7.2.: The graph presents the backdoor with unobserved variable (U) that points both the treatment and the target.

7.1.1.2. Frontdoor Estimand

```
Estimand assumption 1,
Full-mediation: cns_m__var_17 intercepts
(blocks) all directed paths from trf_m__var_57 to target.
```

```
Estimand assumption 2, First-stage-unconfoundedness:
```

If $U \rightarrow trf\_m\_\_var\_57$ and
$U \rightarrow cns\_m\_\_var\_17$ then
$P(cns\_m\_\_var\_17 \mid trf\_m\_\_var\_57, U) =$
$P(cns\_m\_\_var\_17 \mid trf\_m\_\_var\_57)$

```
Estimand assumption 3, Second-stage-unconfoundedness:
```

If $U \rightarrow cns\_m\_\_var\_17$ and $U \rightarrow target$ then
$P(target \mid cns\_m\_\_var\_17, trf\_m\_\_var\_57, U) =$
$P(target \mid cns\_m\_\_var\_17, trf\_m\_\_var\_57)$

To provide a more detailed insight of this sentence:

```
Estimand assumption 1,
Full-mediation:
cns_m__var_17 intercepts (blocks) all directed paths
from trf_m__var_57 to target
```

If $U \rightarrow cns\_m\_\_var\_17$ and $U \rightarrow target$ then
$P(target \mid cns\_m\_\_var\_17, trf\_m\_\_var\_57, U) =$
$P(target \mid cns\_m\_\_var\_17, trf\_m\_\_var\_57)$

`The Estimand Assumption 1:` means **what** we are trying to estimate or measure.
`Full-Mediation:` means that the variable `cns_m__var_17` **fully mediates** or **completely explains** the relationship between the `trf_m__var_57` and `target`.
For the sake of graphical aspect we rename `trf_m__var_57` to TRF and `cns_m__var_17` to CNS.

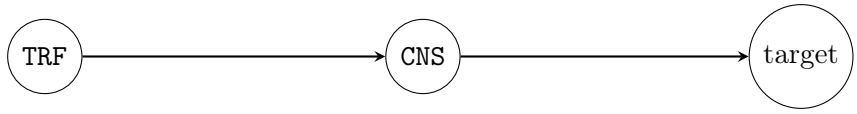


Figure 7.3.: The graph presents a Full-mediation.

As to provide an explanation of the sentence: CNS intercepts (blocks) all directed paths from TRF to target, see Figure 7.3. CNS: is a variable or a factor in the analysis. Intercepts (blocks): means that CNS is acting as a barrier or stopping point in the relationship between TRF and target. All directed paths: based on the definition in statistical analysis, a path is a sequence of connections between variables. Directed paths have a specific direction, and imply a cause-and-effect relationship. From TRF to target: this specifies the direction of the paths being blocked. In simpler terms, CNS stops or blocks all the connections between TRF and the target, the ultimate outcome or variable of interest. Finally, CNS fully explains the relationship between TRF and target by blocking or intercepting all the paths between them. This variable is a crucial link that influences the relationship between the "treatment" and the "target".

```
Estimand assumption 2,
First-stage-unconfoundedness:
```

If $U \rightarrow trf\_m\_\_var\_57$ and
$U \rightarrow cns\_m\_\_var\_17$ then
$P(cns\_m\_\_var\_17 \mid trf\_m\_\_var\_57, U)$
$= P(cns\_m\_\_var\_17 \mid trf\_m\_\_var\_57)$

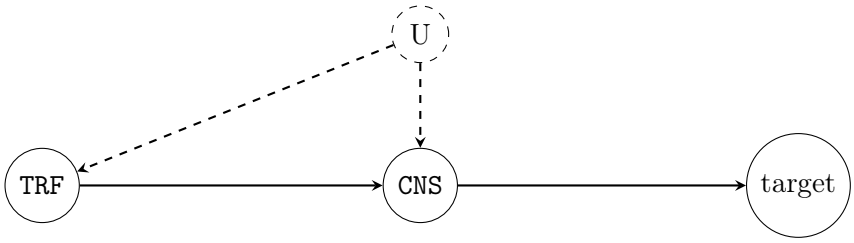


Figure 7.4.: The graph presents mediation with unobserved variable (U) that points both to the treatment (TRF) and the mediator (CNS).

U: represents some **unobserved** or **hidden variable**. $U \rightarrow TRF$: means that U influences or causes variations in the variable TRF. $U \rightarrow CNS$: means that U influences or causes variations in the variable CNS. $P(...)$: is the probability. $P(CNS \mid TRF, U)$: is the probability of CNS given TRF and the hidden variable U, equals, $P(CNS \mid TRF)$: the probability of CNS given TRF without considering the hidden variable U. This assumption expresses that if there are **unobserved** factors (U) that influence both the treatment (TRF) and the **mediating** variable (CNS), then the probability of the **mediating variable** given the treatment (TRF) and the unobserved factors is the same as the probability of the mediating variable given only the treatment (TRF). The unobserved factors don't introduce any additional variation when considering the relationship between the treatment and the mediating variable.

```
Estimand Assumption 3, Second-stage-unconfoundedness:
```

If $U \rightarrow CNS$ and $U \rightarrow target$ then
$P(target \mid CNS, TRF, U) = P(target \mid CNS, TRF)$

If $U \rightarrow TRF$ and $U \rightarrow target$, U: is unobserved or hidden variable. $U \rightarrow TRF$: means that U influences or causes variations in the variable TRF.

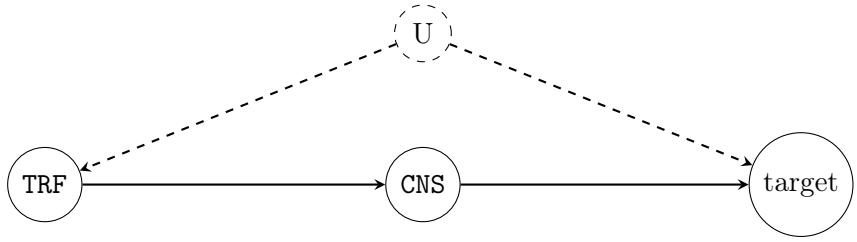


Figure 7.5.: The graph presents a Full-mediation with unobserved variable (U) that points both to the treatment (TRF) and target.

$U \rightarrow target$: means that U influences or causes variations in the target. Then $P(target \mid CNS, TRF, U) = P(target \mid CNS, TRF)$. $P(...)$: is the probability. $P(target \mid ...)$: is the probability of the target variable given certain conditions. $P(target \mid CNS, TRF, U)$: is the probability of the target variable given the mediating variable CNS, the treatment TRF, and the hidden variable U. $P(target \mid CNS, TRF)$: is the probability of the target variable given the **mediating** variable and the treatment, without considering the hidden variable U. If we have unobserved factors (U) that influence both the mediating variable CNS and the target variable, then the probability of the target variable given the **mediating** variable, the starting point, and the unobserved factors is the same as the probability of the target variable given only the mediating variable and the treatment. In conclusion, unobserved factors don't introduce additional variation when considering the relationship between the mediating variable, the treatment and the target variable.

Estimate of the causal effect: 0.007097120779912439

7.1.2. The variable `cns_m__var_17`

`Estimand type: EstimandType.NONPARAMETRIC\_ATE`

```
### Estimand : 1
Estimand name: backdoor
Estimand expression:
```

$$\frac{d}{d[cns\_m\_\_var\_17]}\left(E[\text{target}]\right)$$

```
Estimand assumption 1, Unconfoundedness:
```

If $U \rightarrow cns\_m\_\_var\_17$ and $U \rightarrow target$ then
$P(target \mid cns\_m\_\_var\_17, , U) =$
$= P(target \mid cns\_m\_\_var\_17,)$

7.1.2.1. Instrumental Variable

```
### Estimand : 2
Estimand name: iv
Estimand expression:
```

For the sake of easy visualization of the formula:
usg_m__var_60 is usg60,
trf_m__var_57 is trf57,
usg_m__var_59 is usg59 and
anag_cli_f_var_9 is anag9

$$E[\left(\frac{d(\text{target})}{d[\text{usg60,trf57,usg59,anag9}]}\right)\left(\frac{d[\text{cns17}]}{d[\text{usg60,trf,usg59,anag9}]}\right)^{-1}]$$

```
Estimand assumption 1
As-if-random:
```

If $U \to\to target$ then
$\neg(U \to\to usg\_m\_\_var\_60, trf\_m\_\_var\_57, usg\_m\_\_var\_59,$
$anag\_cli\_f\_var\_9)$

```
Estimand assumption 2
Exclusion: If we remove
```

$usg\_m\_\_var\_60,$
$trf\_m\_\_var\_57, usg\_m\_\_var\_59,$
$anag\_cli\_f\_var\_9 \to cns\_m\_\_var\_17$ then
$\neg(usg\_m\_\_var\_60, trf\_m\_\_var\_57, usg\_m\_\_var\_59,$
$anag\_cli\_f\_var\_9 \to target)$

```
### Estimand : 3
Estimand name: frontdoor
No such variable(s) found!
```

Estimate of the causal effect: 0.06635127533149945

An instrumental variable is a variable that is used to estimate the causal relationship between an independent variable (treatment) and a dependent variable (target, that we want to explain or predict).

`Estimand Assumption 1: As-if-random` This assumption refers to the relationship between the instrumental variable (iv) and the target variable.

The instrumental variable is assumed to be unrelated to any other factors that might affect the studied outcome, except through its impact on the independent variable (i.e. iv is treated as if it is randomly assigned).

As-if-random: If $U \to\to target$ then
$\neg(U \to\to usg\_m\_\_var\_60, trf\_m\_\_var\_57, usg\_m\_\_var\_59,$
$anag\_cli\_f\_var\_9)$

"If $U \to\to target$" means if the instrumental variable (U) **affects** the target variable.
"$\neg(U \to\to usg\_m\_\_var\_60, trf\_m\_\_var\_57, usg\_m\_\_var\_59,$
$anag\_cli\_f\_var\_9)$" means that the instrumental variable is **not allowed** to have a **direct impact** on any of the remaining variables
($usg\_m\_\_var\_60, trf\_m\_\_var\_57, usg\_m\_\_var\_59,$
$anag\_cli\_f\_var\_9$).

The instrumental variable's influence on the outcome should not come through any other paths or variables.

In summary, the statement is emphasizing that the instrumental variable **should only impact** the target variable and not have any other direct influences on other variables related to the analysis. This is important for ensuring that the **estimated causal relationship is not confounded by other factors**.

`Estimand Assumption 2, Exclusion:`

The "Exclusion" assumption is about removing the impact of certain nodes to see the real effect of target.

Exclusion: If we remove
$usg\_m\_\_var\_60$,$trf\_m\_\_var\_57$,$usg\_m\_\_var\_59$,
$anag\_cli\_f\_var\_9 \to cns$, then $\neg(usg\_m\_\_var\_60, trf\_m\_\_var\_57,$
$usg\_m\_\_var\_59, anag\_cli\_f\_var\_9 \to target)$:
removing the influence of $usg\_m\_\_var\_60, trf\_m\_\_var\_57, usg\_m\_\_var\_59,$
$anag\_cli\_f\_var\_9$ on cns then the relationship between
$usg\_m\_\_var\_60$, $trf\_m\_\_var\_57$, $usg\_m\_\_var\_59$,
$anag\_cli\_f\_var\_9$ and the target is not maintained true. If we ignore the impact of certain factors on one thing, then a certain relationship won't be true anymore.

Exploring the impact of $usg\_m\_\_var\_60$, $trf\_m\_\_var\_57$, $usg\_m\_\_var\_59$, $anag\_cli\_f\_var\_9$ on the target, and the exclusion of these nodes' impact on cns, then a specific relationship with the target doesn't hold.

7.1.3. The variable `anag_cli_f_var_1`

```
### Estimand : 1
Estimand name: backdoor
Estimand expression:
```

$$\frac{d}{d[anag\_cli\_f\_var\_1]}\left(E[\text{target}]\right)$$

```
### Estimand : 2
Estimand name: iv
No such variable(s) found!
```

```
### Estimand : 3
Estimand name: frontdoor
No such variable(s) found!
```

Estimate of the causal effect: -0.013978754618979392

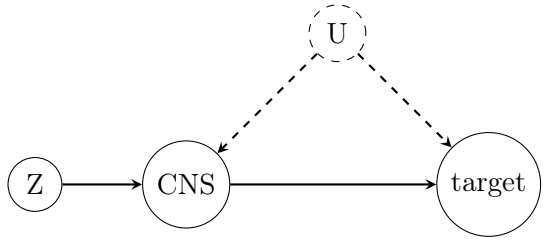


Figure 7.6.: The graph presents an instrumental variable.

7.1.4. The variable `anag_cli_f_var_3`

```
Estimand type: EstimandType.NONPARAMETRIC_ATE
```

```
### Estimand : 1
Estimand name: backdoor
Estimand expression:
```

$$\frac{d}{d[anag\_cli\_f\_var\_3]}\left(E[\text{target}]\right)$$

```
Estimand assumption 1, Unconfoundedness:
```

If $U \rightarrow anag\_cli\_f\_var\_3$ and $U \rightarrow target$ then
$P(target \mid anag\_cli\_f\_var\_3, , U) = P(target \mid anag\_cli\_f\_var\_3,)$

```
### Estimand : 2
Estimand name: iv
No such variable(s) found!
```

```
### Estimand : 3
Estimand name: frontdoor
No such variable(s) found!
```

Estimate of the causal effect: -0.028959988210801715

7.1.5. The variable `anag_cli_f_var_7`

```
anag_cli_f_var_7
No directed path from ['anag_cli_f_var_7'] to ['target'] in
the causal graph.
```

Causal effect is zero. 0

7.1.6. The variable `trf_m__var_49`

```
trf_m__var_49
No directed path from ['trf_m__var_49'] to ['target'] in the
causal graph.
```

Causal effect is zero. 0

7.1.7. The variable `trf_m__var_51`

```
trf_m__var_51
No directed path from ['trf_m__var_51'] to ['target'] in the
causal graph.

Causal effect is zero. 0
```

7.1.8. The variable `trf_m__var_52`

```
trf_m__var_52
No directed path from ['trf_m__var_52'] to ['target'] in the
causal graph.

Causal effect is zero. 0
```

7.1.9. The variable `trf_m__var_53`

```
trf_m__var_53
No directed path from ['trf_m__var_53'] to ['target'] in the
causal graph.

Causal effect is zero. 0
```

7.1.10. The variable `trf_m__var_54`

```
trf_m__var_54
No directed path from ['trf_m__var_54'] to ['target'] in the
causal graph.

Causal effect is zero. 0
```

7.1.11. The variable `trf_m__var_55`

```
trf_m__var_55
 No directed path from ['trf_m__var_55'] to ['target'] in the
causal graph.

Causal effect is zero. 0
```

7.1.12. The variable `trf_m__var_56`

```
trf_m__var_56
No directed path from ['trf_m__var_56'] to
['target'] in the causal graph.

Causal effect is zero. 0
```

7.1.13. The variable `anag_cli_f_var_0`

```
Estimand type: EstimandType.NONPARAMETRIC_ATE
```

```
### Estimand: 1
Estimand name: backdoor
Estimand expression:
```

$$\frac{d}{d[anag\_cli\_f\_var\_0]}\left(E[\text{target}]\right)$$

```
Estimand assumption 1, Unconfoundedness:
```

If $U \rightarrow anag\_cli\_f\_var\_0$ and $U \rightarrow target$ then
$P(target \mid anag\_cli\_f\_var\_0, , U) =$
$P(target \mid anag\_cli\_f\_var\_0,)$

```
### Estimand : 2
Estimand name: iv
No such variable(s) found!
```

```
### Estimand : 3
Estimand name: frontdoor
No such variable(s) found!
```

Estimate of the causal effect: -0.029293209329663714

7.1.14. The variable anag_cli_f_var_5

```
anag_cli_f_var_5
No directed path from ['anag_cli_f_var_5'] to
['target'] in the causal graph.

Causal effect is zero. 0
```

The $anag\_cli\_f\_var\_5$ variable is very **relevant**, for example if the customer joined in the very recent period of time: "young status customer" it is more probable that can churn respect to both a customer which is already in the company from an intermediate period of time (i.e. "intermediate status customer") and an historical customer (i.e. "veteran status customer") which can be considered "lazier", and the leaving is more difficult (e.g. they have been there for 20 years).

7.1.15. The variable anag_cli_f_var_9

```
Estimand type: EstimandType.NONPARAMETRIC_ATE
```

```
### Estimand : 1
Estimand name: backdoor
Estimand expression:
```

$$\frac{d}{d[anag\_cli\_f\_var\_9]}\left(E[\text{target}]\right)$$

```
Estimand assumption 1, Unconfoundedness:
```

If $U \rightarrow anag\_cli\_f\_var\_9$ and $U \rightarrow target$ then $P(target \mid anag\_cli\_f\_var\_9,,U) = P(target \mid anag\_cli\_f\_var\_9,)$

The anag_cli_f_var_9 variable highlights the fact that the line is customer-

level hooked. The customers that have multiple hooked lines remain more faithful and therefore less prone to churn. For example, the customer is a small or medium size company gaining 10 lines, all of them are under 1 single contract, for this reason the customer is less prone to cease all the lines all together. In the case of a small dealer with a single headed line which becomes a commodity, it is easier to be attracted by more bearish offers and so to leave.

```
### Estimand : 2
Estimand name: iv
No such variable(s) found!
```

```
### Estimand : 3
Estimand name: frontdoor
Estimand expression:
```

$$E[\frac{d}{d[cns\_m\_\_var\_17]}(target) \cdot \frac{d}{d[anag\_cli\_f\_var\_9]}([cns\_m\_\_var\_17])]$$

```
Estimand assumption 1, Full-mediation: cns_m__var_17
intercepts (blocks) all directed paths from anag_cli_f_var_9
to target.

Estimand assumption 2, First-stage-unconfoundedness:
If $Urightarrow {anag_cli_f_var_9} $ and
$ Urightarrow {cns_m__var_17} $ then
$ P(cns_m__var_17mid anag_cli_f_var_9,U) = $
$ P(cns_m__var_17mid anag_cli_f_var_9)$

Estimand assumption 3, Second-stage-unconfoundedness:
```

If $U \rightarrow cns\_m\_\_var\_17$ and $U \rightarrow target$ then
$P(target \mid cns\_m\_\_var\_17, anag\_cli\_f\_var\_9, U) =$
$P(target \mid cns\_m\_\_var\_17, anag\_cli\_f\_var\_9)$

Estimate of the causal effect: -0.008481039893568285.

The variable cns_m__var_17 is influenced by another SIM card, and it treats the count of master flags as a parameter. The particular line it focuses on is not random; it pertains to an individual, whereas the line associated with a burglar alarm is supplementary. Neglecting to recharge the alarm SIM card within one year results in its deactivation.

7.1.16. The variable anag_cli_f_var_10

```
Estimand type: EstimandType.NONPARAMETRIC_ATE
```

```
### Estimand : 1
Estimand name: backdoor
Estimand expression:
```

$$\frac{d}{d[anag\_cli\_f\_var\_10]}\left(E[\text{target}]\right)$$

```
Estimand assumption 1, Unconfoundedness:
```

If $U \rightarrow anag\_cli\_f\_var\_10$ and $U \rightarrow target$ then
$P(target \mid anag\_cli\_f\_var\_10, , U) =$
$P(target \mid anag\_cli\_f\_var\_10,)$

```
### Estimand : 2
Estimand name: iv
No such variable(s) found!
```

```
### Estimand : 3
Estimand name: frontdoor
No such variable(s) found!
```

Estimate of the causal effect: -0.006650086650659068

7.1.17. The variable anag_cli_f_var_13

```
anag_cli_f_var_13
No directed path from ['anag_cli_f_var_13'] to ['target'] in
the causal graph.
```

Causal effect is zero. 0

7.1.18. The variable anag_cli_f_var_14

```
anag_cli_f_var_14
No directed path from['anag_cli_f_var_14'] to ['target'] in
the causal graph.

Causal effect is zero. 0
```

7.1.19. The variable cerved_int__var_15

```
cerved_int__var_15
No directed path from ['cerved_int__var_15'] to ['target'] in
the causal graph.

Causal effect is zero. 0
```

This variable has no causal effect.

7.1.20. The variable cns_m__var_18

```
Estimand type: EstimandType.NONPARAMETRIC_ATE
```

```
### Estimand : 1
Estimand name: backdoor
Estimand expression:
```

$$\frac{d}{d[cns\_m\_\_var\_18]}\left(E[\text{target}]\right)$$

```
Estimand assumption 1, Unconfoundedness:
```

If $U \rightarrow cns\_m\_\_var\_18$ and $U \rightarrow target$ then
$P(target \mid cns\_m\_\_var\_18, , U) = P(target \mid cns\_m\_\_var\_18,)$

The variable cns_m__var_18 can provide the customer's line profile. For example, a mobile line with multiple service cards reveals the complete customer profile. Typically, the customer subscribes to a basic service and has the option to activate additional services, such as an answering machine or call message removal. Therefore, if a customer has numerous active

services, it could indicate substantial usage of that line.

In the current landscape, unlimited data plans are rare due to various reasons: they can be costly for service providers (resulting in the adoption of tiered data plans to manage costs and revenue), and the implementation of data caps efficiently controls network traffic. A limited number of users with unlimited data could potentially strain the network and compromise service quality. The restriction on data usage enhances the profitability of service providers' plans by offering diverse tiers of data plans at various price points to cater to different user needs. The structure of data plans often adapts to market trends and competition. If the prevailing trend involves tiered plans, service providers are likely to adopt a similar approach to remain competitive.

As the demand for high-bandwidth applications like streaming and gaming continues to rise, data usage patterns evolve. It's worth noting that while unlimited data plans may not be as prevalent as before, some providers and plans still offer them, especially in specific regions or for certain services. Consumer preferences, technological advancements, and market dynamics collectively influence the availability and structure of data plans.

For a line with a basic profile, customers are less inclined to add unnecessary services to their service card. Adding service recalls, unlimited Gigas, or an additional 50 Gigas indicates increased usage, reducing the likelihood of churn. This reflects satisfaction, active services, and an engaged customer. Conversely, having extra Gigas in services that end up being discontinued may indicate customer dissatisfaction and the potential for churn.

```
### Estimand : 2
Estimand name: iv
No such variable(s) found!
```

```
### Estimand : 3
Estimand name: frontdoor
No such variable(s) found!
```

```
Estimate of the causal effect: 0.025520130920288556
```

7.1.21. The variable cns_m__var_19

```
cns_m__var_19
No directed path from ['cns_m__var_19'] to
['target'] in the causal graph.

Causal effect is zero. 0
```

The variable cns_m__var_19 shares similarities with the previously mentioned variable cns_m__var_17.

7.1.22. The variable cns_m__var_20

```
Estimand type: EstimandType.NONPARAMETRIC_ATE
```

```
### Estimand : 1
Estimand name: backdoor
Estimand expression:

$$\frac{d}{d[cns\_m\_\_var\_20]}\left(E[\text{target}]\right)$$

Estimand assumption 1, Unconfoundedness:

If $U \rightarrow cns\_m\_\_var\_20$ and
$U \rightarrow target$ then
$P(target \mid cns\_m\_\_var\_20, , U) = P(target \mid cns\_m\_\_var\_20, )$
```

```
### Estimand : 2
Estimand name: iv
No such variable(s) found!
```

```
### Estimand : 3
Estimand name: frontdoor
No such variable(s) found!
```

```
Estimate of the causal effect: 0.019379429938147855
```

7.1.23. The variable `cns_m__var_21`

```
cns_m__var_21
No directed path from ['cns_m__var_21'] to ['target'] in the
causal graph.

Causal effect is zero. 0
```

The cns_m__var_21 variable refers to the average percentage discount on the fee i.e. the amount to pay (i.e. charged or scaled). Averaged percentage discount is not on everything but on service card on average the discount of the line's benefits.

7.1.24. The variable `usg_m__var_59`

```
Estimand type: EstimandType.NONPARAMETRIC_ATE
```

```
### Estimand : 1
Estimand name: backdoor
Estimand expression:
```

$$\frac{d}{d[usg\_m\_\_var\_59]}\left(E[\text{target}]\right)$$

```
Estimand assumption 1, Unconfoundedness:
```

If $U \rightarrow usg\_m\_\_var\_59$ and $U \rightarrow target$ then
$P(target \mid usg\_m\_\_var\_59, , U) = P(target \mid usg\_m\_\_var\_59,)$

The variable usg_m__var_59 refers to the rating group of the navigation category (i.e. web), the customers that fall into this cluster are more prone to churn.

```
### Estimand : 2
Estimand name: iv
No such variable(s) found!
```

```
### Estimand : 3
Estimand name: frontdoor
Estimand expression:
```

7.1.25. The variable cns_m__var_17

```
Estimand assumption 1,
Full-mediation: cns_m__var_17 intercepts (blocks) all directed
paths from usg_m__var_59 to target.
```

The variable usg_m__var_59 arises the following issues: a **qualitative issue** which refers to the most used navigation cluster by those who have master lines; a **quantitative issue** which refers to the cluster in which customers have higher traffic.

```
Estimand assumption 2,
First-stage-unconfoundedness:
```

If $U \to usg\_m\_\_var\_59$ and
$U \to cns\_m\_\_var\_17$ then
$P(cns\_m\_\_var\_17 \mid usg\_m\_\_var\_59, U)$
$=P(cns\_m\_\_var\_17 \mid usg\_m\_\_var\_59)$

```
Estimand assumption 3,
Second-stage-unconfoundedness:
```

If $U \to cns\_m\_\_var\_17$ and $U \to target$ then
$P(target \mid cns\_m\_\_var\_17, usg\_m\_\_var\_59, U) =$
$P(target \mid cns\_m\_\_var\_17, usg\_m\_\_var\_59)$

```
Estimate of the causal effect: 0.0033793960406430096
```

7.1.26. The variable usg_m__var_60

```
Estimand type: EstimandType.NONPARAMETRIC_ATE
```

```
### Estimand : 1
Estimand name: backdoor
Estimand expression:
```

$$\frac{d}{d[usg\_m\_\_var\_60]}\left(E[\text{target}]\right)$$

```
Estimand assumption 1, Unconfoundedness:
```

If $U \rightarrow usg\_m\_\_var\_60$ and $U \rightarrow target$ then $P(target \mid usg\_m\_\_var\_60, , U)$
$= P(target \mid usg\_m\_\_var\_60,)$

```
### Estimand : 2
Estimand name: iv
No such variable(s) found!
```

```
### Estimand : 3
Estimand name: frontdoor
Estimand expression:
```

$$E[\frac{d}{d[cns\_m\_\_var\_17]}\left(target\right) \cdot \frac{d}{d[usg\_m\_\_var\_60]}\left([cns\_m\_\_var\_17]\right)]$$

The aforementioned variables reflect both the cutomer's **personal** characteristics (i.e. line type, age, etc.) and the **behavioral** characteristics (i.e. traffic which tells what a person does). Those characteristics are related, for example by analogy we can consider the people playing the "playstation", there can be an anagraphic correlation, so here the correlation is between the line type and the cluster of web navigations, semantic clusters, etc.

```
Estimand assumption 1, Full-mediation:
cns_m__var_17 intercepts (blocks) all directed paths from
usg_m__var_60 to target.

Estimand assumption 2, First-stage-unconfoundedness:
```

If $U \to usg\_m\_\_var\_60$ and
$U \to cns\_m\_\_var\_17$ then
$P(cns\_m\_\_var\_17 \mid usg\_m\_\_var\_60, U)$
$= P(cns\_m\_\_var\_17 \mid usg\_m\_\_var\_60)$

```
Estimand assumption 3, Second-stage-unconfoundedness:
```

If $U \to cns\_m\_\_var\_17$ and $U \to target$ then
$P(target \mid cns\_m\_\_var\_17,$
$usg\_m\_\_var\_60, U)$
$= P(target \mid cns\_m\_\_var\_17$

```
Estimate of the causal effect: 0.01635061546326745
```

7.1.27. The variable usg_m__var_61

```
Estimand type: EstimandType.NONPARAMETRIC_ATE
```

```
### Estimand : 1
Estimand name: backdoor
Estimand expression:
```

$$\frac{d}{d[usg\_m\_\_var\_61]}\left(E[\text{target}]\right)$$

```
Estimand assumption 1, Unconfoundedness:
```

If $U \to usg\_m\_\_var\_61$ and $U \to target$ then
$P(target \mid usg\_m\_\_var\_61, , U)$
$= P(target \mid usg\_m\_\_var\_61,)$

```
### Estimand : 2
Estimand name: iv
No such variable(s) found!
```

```
### Estimand : 3
Estimand name: frontdoor
No such variable(s) found!
```

Estimate of the causal effect: 0.0540562791062031

The variable pertains to the usg_m__var_61 rating groups, offering a distinct behavioral profile indicating whether the customer is likely to churn or not. Specifically, those most likely to churn exhibit similar behavior in terms of phone use, particularly in the qualitative thematic area related to the sum of phone use. The correlation is quantitative, with low values impacting high values in a reverse correlation type.

Chapter 8.

Conclusions

8.1. Contributions

The methodological results and contributions of this thesis are summarized in the following.

8.1.1. Churn Prediction

The first contribution is a solution to churn prediction based on a real data set by applying a variety of algorithms. The performance of the models was evaluated through a standard qualitative metric and compared with the internal solution of TIM S.p.A. The results obtained are an improvement over the state of the art, covering the Research Question *RQ1*. The results will improve the internal processes of the company, solving the Business Question *BQ1*. Furthermore, the solutions obtained will constitute the foundational guidelines for new processes for the handling of customer churn in the company.

8.1.2. Causal Discovery

The second contribution is a novel ensemble-based methodology for causal discovery. The model aims to identify the algorithm that can most accurately reconstruct the underlying causal structure by using ground-truth data as a benchmark for training. This methodology capitalizes on the strengths of multiple algorithms and introduces a novel way to improve the **reliability** of causal discovery processes. The causal discovery algorithms

that we considered were as follows: GES, GOLEM, LINGAM, Notears, and PC.

Our method improves the learning of causal structures utilizing the combined results of multiple causal discovery algorithms. We analyze the distances between Directed Acyclic Graphs (DAGs) generated by these algorithms and a centroid DAG, as well as the distances among the DAGs themselves. Based on this analysis, we propose a machine learning model that can predict the quality of directed acyclic graphs (DAGs) in new, unseen data.

The Direct Acyclic Graphs (DAGs) generated by the discovery algorithms can sometimes coincide with one another, whether they correspond to the ground truth or not: thus, there is not necessarily a single correct answer to the question of which is the ground truth or which is the algorithm closest to the ground truth. As a consequence, we formulate the problem as a multi-label classification task (each algorithm plays the role of a label, there are also 6 labels, counting the CENTROID graph). Then we use two common approaches to address this type of classification: the Binary Relevance (BR) approach and the Classifier Chain (CC) approach. After choosing a representative from the set of predicted labels, using a classifier-precision-based criterion, we used its distance from the GT (defined as the Hamming distance between adjacency matrices, i.e., in terms of edges) and found that we could considerably improve with respect to the use of individual algorithms. Although the best individual algorithm was CENTROID (although close to GOLEM and LINGAM) with a mean of 10.08 edges from the GT, the mean distance of the algorithm selected by our method was 6.62 edges using BR and 6.36 edges using CC, both very close to the actual average minimum distance of the algorithm of the set that performs the best (but is a priori unknown), which was of 5.38 edges. Similar considerations hold if we divide the distances by the actual number of edges.

The method relies on the use of existing discovery algorithms, considered as a committee of experts, and on a supervised learning approach consisting of stacking of a multi-label classifier on the outcomes of the ensemble.

8.1.3. Causal Inference

The third contribution solves the Business Question *BQ2*, in particular, providing advice on focusing the efforts on those customers that are most likely to churn and exhibit similar behavior in terms of phone usage, especially in the qualitative thematic area related to the phone usage.

8.2. Future research directions

The potential for the proposed novel ensemble-based methodology for causal discovery is to significantly enhance the field of causal inference for further exploration and application of these techniques to a wide variety of scenarios. For example, in the health assessment of patients such as in Emergency Rooms; in the bank sector, credit card fraud detection; agricultural environment for the optimization of crops growth.

The real-dataset availability is very scarce and limited to company's boundaries. This creates a huge gap between the research fields both in the academic and in the industry. Finally, the presented methodology may be improved by combining it with other methodologies available in the literature.

Appendix A.

Probability Notations

The probability function P_z describes the uncertainty of z, the use of quantity is often impractical since it requires to deal with as many values as the size of Z. It is more convenient to compute a functional (i.e. a function of a function) of P_z. The measure of central tendency:
Definition 1 (Expected value). The expected value of a discrete random variable z is:

$$E[z] = \mu = \sum_{z \in Z} z P_z(z)$$

It minimizes the mean squared error:

$$\mu = argmin_m E[(z - m)^2]$$

Appendix B.

Metrics

The evaluation of the model performance is fundamental, in the following table B.1 some of the metrics.

A confusion matrix is a summary of the prediction results on a classification problem [29]. The number of correct and incorrect predictions is summarized with the count values for each class, as shown in Table 2.

The confusion matrix shows how the classification model is confused when it makes predictions. It gives us insight not only into the errors made by a classifier but, more importantly, into the types of error.

We can explain TP, FP, TN, and FN as follows:

- True positive (TP): if the predicted "churn customer" is actually "churn customer," the prediction is TP
- False positive (FP): if the predicted "churn customer" is real news, the prediction is FP
- True negative (TN): if the predicted "no-churn customer" is actually "no-churn customer," the prediction is TN
- False negative (FN): if the predicted "no-churn customer" is actually "churn customer," the prediction is FN

Confusion matrix: It is a table with two rows and two columns that reports the number of false positives (FP), false negatives (FN), true positives (TP), and REFERENCE true negatives (TN). Provides the information required to analyze the accuracy of the prediction of churn in terms of false. Accu-

racy: The accuracy of the given prediction model is defined below.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

"Precision recall statistics are useful to compare algorithms that make predictions with a confidence score. Using these statistics, performance of an algorithms given a set threshold (confidence score) can be approximated."

Precision is defined by:

$$Precision = \frac{TP}{TP + FP}$$

and directly denotes the total classification accuracy given a confidence threshold.

On the other hand, Recall is defined by:

$$Recall = \frac{TP}{TP + FN}$$

and denotes misclassification given a threshold.

F1-score: defined below

$$F1 = \frac{2 * TP}{2 * TP + FP + FN}$$

AUC curve: in contrast to other metrics, the AUC is not influenced by any threshold value, as it takes into account all possible thresholds on the predicted probabilities. "Area under the precision-recall curve, as well as the coordinates of the precision recall curve are computed, using the scikit-learn library tools. Note that unlike the AUROC metric, this metric does not account for class imbalance." Decile Lift or Lift curve: The top decile lift focuses exclusively on the most critical group of customers and their churn risk.

Table B.1.: Metrics.

Classification	**Regression**	**Clustering**
Accuracy Score	Mean Absolute Error	Adjusted Rank Index
Confusion Matrix	Mean Squared Error	Homogeneity
Precision	R^2 Score	V-measure
Recall	Mean Average Precision	Mean Reciprocal Rank
F1-score		Variation Mean Reciprocal Rank
Specificity		

Appendix C.

Distributions

```
#sem\_type: str
#        gauss, exp, gumbel, uniform, logistic (linear);
#        mlp, mim, gp, gp-add, quadratic (nonlinear).
```

C.1. Probability Distributions and Densities

C.1.0.1. Gaussian (Normal) Distribution

'gauss': Gaussian (Normal) Distribution – This is a commonly used distribution for linear models when the assumption of normality is appropriate.

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

C.1.0.2. Exponential Distribution

'exp': Exponential Distribution – Used for modeling the time between events in a Poisson process.

$$f(x) = \lambda e^{-\lambda x}$$

for $x > 0$ and 0 elsewhere. β is the scale parameter, which is the inverse of the rate parameter $\lambda = \frac{1}{\beta}$. The rate parameter is an alternative, widely used parameterization of the exponential distribution [91].

C.1.0.3. Gumbel Distribution

'gumbel': Gumbel Distribution – Often used in extreme value theory and can be associated with maximum or minimum values in a distributionc[47], [97].

The Gumbel (or Smallest Extreme Value (SEV) or the Smallest Extreme Value Type I) distribution is one of a class of Generalized Extreme Value (GEV) distributions used in modeling extreme value problems. The Gumbel is a special case of the Extreme Value Type I distribution for maximums from distributions with "exponential-like" tails [47], [97].

The probability density for the Gumbel distribution is

$$f(x) = \frac{1}{\beta} e^{-\frac{x-\mu}{\beta}} e^{-e^{-\frac{x-\mu}{\beta}}}$$

where μ is the mode, a location parameter, and β is the scale parameter.

C.1.0.4. Uniform Distribution

'uniform': Uniform Distribution – Assumes that all values within a given range are equally likely. The probability density function of the uniform distribution is

$$f(x) = \frac{1}{b-a}$$

anywhere within the interval $[a, b)$, and zero elsewhere [65].

'logistic': Logistic Distribution – Frequently used in logistic regression and other modeling scenarios.

The Logistic distribution is used in Extreme Value problems where it can act as a mixture of Gumbel distributions, in Epidemiology, and by the World Chess Federation (FIDE) where it is used in the Elo ranking system, assuming the performance of each player is a logistically distributed random variable [97]

The probability density function is

$$f(x) = \frac{e^{-(x-\mu)/s}}{s(1+e^{-(x-\mu)/s})^2}$$

where μ = location and s = scale.

C.2. Nonlinear Models

In the following the function used in gcastle to generate the nonlinear models [96]

```
@staticmethod
def _simulate_nonlinear_sem(W, n, sem_type, noise_scale):
"""
Simulate samples from nonlinear SEM.

Parameters
----------
B: np.ndarray
    [d, d] binary adj matrix of DAG.
n: int
    Number of samples.
sem_type: str
    mlp, mim, gp, gp-add, or quadratic.
noise_scale: float
    Scale parameter of noise distribution
    in linear SEM.

Return
------
X: np.ndarray
    [n, d] sample matrix
"""
if sem_type == 'quadratic':
    return IIDSimulation._simulate_quad_sem(
        W, n, noise_scale)

def _simulate_single_equation(X, scale):
    """X: [n, num of parents], x: [n]"""
    z = np.random.normal(scale=scale, size=n)
    pa_size = X.shape[1]
    if pa_size == 0:
        return z
    if sem_type == 'mlp':
        hidden = 100
```

```
        W1 = np.random.uniform(low=0.5, high=2.0,
        size=[pa_size, hidden])
        W1[np.random.rand(*W1.shape) < 0.5] *= -1
        W2 = np.random.uniform(
            low=0.5, high=2.0, size=hidden)
        W2[np.random.rand(hidden) < 0.5] *= -1
        x = sigmoid(X @ W1) @ W2 + z
    elif sem_type == 'mim':
        w1 = np.random.uniform(
            low=0.5, high=2.0, size=pa_size)
        w1[np.random.rand(pa_size) < 0.5] *= -1
        w2 = np.random.uniform(
            low=0.5, high=2.0, size=pa_size)
        w2[np.random.rand(pa_size) < 0.5] *= -1
        w3 = np.random.uniform(
            low=0.5, high=2.0, size=pa_size)
        w3[np.random.rand(pa_size) < 0.5] *= -1
        x = np.tanh(X @ w1) + np.cos(X @ w2) +
            np.sin(X @ w3) + z
    elif sem_type == 'gp':
        from sklearn.gaussian_process import
            GaussianProcessRegressor
        gp = GaussianProcessRegressor()
        x = gp.sample_y(
            X, random_state=None).flatten() + z
    elif sem_type == 'gp-add':
        from sklearn.gaussian_process import
            GaussianProcessRegressor
        gp = GaussianProcessRegressor()
        x = sum([gp.sample_y(
                X[:, i, None], random_state=None).flatten()
                for i in range(X.shape[1])]) + z
    else:
        raise ValueError('Unknown sem type.
                         In a nonlinear model,
                         the options are as follows:
                         mlp, mim, gp, gp-add,
                         or quadratic.')
    return x
```

```
B = (W != 0).astype(int)
d = B.shape[0]
if noise_scale is None:
    scale_vec = np.ones(d)
elif np.isscalar(noise_scale):
    scale_vec = noise_scale * np.ones(d)
else:
    if len(noise_scale) != d:
        raise ValueError('noise scale must be
        a scalar or has length d')
    scale_vec = noise_scale

X = np.zeros([n, d])
G_nx = nx.DiGraph(B)
ordered_vertices = list(nx.topological_sort(G_nx))
assert len(ordered_vertices) == d
for j in ordered_vertices:
    parents = list(G_nx.predecessors(j))
    X[:, j] = _simulate_single_equation(
    X[:, parents], scale_vec[j] )
return X
```

For Nonlinear Models:

C.2.0.1. Nonlinear Models

'mlp': Multi-Layer Perceptron (MLP)

C.2.1. Multiple Indicators Multiple Causes (MIMIC)

'mim': Multiple Indicators Multiple Causes (MIMIC)

C.2.2. Gaussian Process

'gp': Gaussian Process

C.2.3. Additive Gaussian Process

'gp-add': Additive Gaussian Process

C.2.4. Quadratic Distribution

'quadratic': Quadratic Distribution – Represents a quadratic relationship between variables.

Appendix D.

Metrics in gCastle

The class "MetricsDAG" computes various accuracy metrics for estimated DAG. In particular, we can define:

- true positive (TP): an edge estimated with correct direction.
- true negative (TN): an edge that is neither in estimated graph nor in true graph.
- false positive (FP): an edge that is in estimated graph but not in the true graph.
- false negative (FN): an edge that is not in estimated graph but in the true graph.
- reverse: an edge estimated with reversed direction.

The following definitions are:

- **False Discovery Rate**

$$fdr = \frac{reverse + FP}{TP + FP}$$

- **True Positive Rate**

$$tpr = \frac{TP}{TP + FN}$$

- **False Positive Rate**

$$fpr = \frac{reverse + FP}{TN + FP}$$

- Structural Hamming Distance is the No. of edge additions, flips, or deletions to get from the predicted graph to the true one

$$shd = undirectedextra + undirectedmissing + reverse$$

- **No. of Non-Negative Entries**

$$nnz = TP + FP$$

- **Precision**

$$precision = \frac{TP}{TP + FP}$$

- **Recall**

$$recall = \frac{TP}{TP + FN}$$

- **F1-Score**

$$F1 = \frac{2 * (recall * precision)}{(recall + precision)}$$

- **G-Score** ranges from 0 to 1

$$gscore = max(0, \frac{TP - FP}{TP + FN})$$

Appendix E.

Simulated values of Ground Truth and Central

Table E.1.: Table that describes the relationship between the GT and the algorithms

Run	GES	GOLEM	LINGAM	Notears	PC
(GT__19.171.0, nonlinear, quadratic)_GT	36	8	8	8	26
(GT__19.86.0, linear, logistic)_GT	14	22	22	22	24
(GT__9.21.0, linear, exp)_GT	0	0	0	0	0
(GT__19.86.0, linear, uniform)_GT	22	0	0	20	22
(GT__19.86.0, linear, gauss)_GT	26	0	16	20	20
(GT__24.136.0, linear, exp)_GT	32	0	0	8	32
(GT__14.91.0, nonlinear, quadratic)_GT	20	20	22	14	20
(GT__24.276.0, nonlinear, gp)_GT	56	34	34	32	52
(GT__19.86.0, nonlinear, mlp)_GT	22	38	26	0	36
(GT__19.86.0, linear, gumbel)_GT	26	0	0	18	20
(GT__19.171.0, linear, exp)_GT	12	0	0	12	6
(GT__19.86.0, nonlinear, mim)_GT	24	20	20	16	24
(GT__24.276.0, nonlinear, gp-add)_GT	40	48	34	22	72
(GT__30.210.0, nonlinear, gp-add)_GT	60	138	44	60	126
(GT__9.36.0, nonlinear, gp)_GT	8	0	0	0	2
(GT__9.21.0, nonlinear, gp)_GT	0	0	0	0	0
(GT__24.276.0, linear, gumbel)_GT	38	52	0	26	36

Run	GES	GOLEM	LINGAM	Notears	PC
(GT__14.91.0, linear, exp)_GT	6	0	0	14	18
(GT__30.210.0, nonlinear, mlp)_GT	44	40	56	24	42
(GT__24.136.0, nonlinear, gp)_GT	0	0	0	0	2
(GT__9.21.0, linear, logistic)_GT	0	28	0	0	0
(GT__14.48.0, nonlinear, gp)_GT	30	2	4	2	26
(GT__9.21.0, linear, uniform)_GT	0	0	0	0	0
(GT__9.36.0, nonlinear, gp-add)_GT	0	2	0	0	2
(GT__24.136.0, linear, gauss)_GT	40	0	10	18	6
(GT__30.210.0, nonlinear, mim)_GT	78	42	54	14	62
(GT__19.171.0, nonlinear, gp)_GT	10	8	8	8	8
(GT__24.136.0, nonlinear, gp-add)_GT	38	0	0	0	0
(GT__9.36.0, linear, exp)_GT	6	0	0	0	0
(GT__9.21.0, linear, gumbel)_GT	0	0	0	0	0
(GT__14.48.0, nonlinear, gp-add)_GT	10	4	2	4	52
(GT__24.136.0, linear, gumbel)_GT	42	0	0	30	28
(GT__19.171.0, linear, uniform)_GT	14	0	0	4	8
(GT__24.136.0, nonlinear, mlp)_GT	24	0	0	0	44
(GT__19.171.0, linear, gauss)_GT	16	0	4	10	8
(GT__14.48.0, nonlinear, mlp)_GT	6	0	36	0	2

Run	GES	GOLEM	LINGAM	Notears	PC
(GT__14.48.0, linear, exp)_GT	34	0	0	2	2
(GT__14.91.0, linear, logistic)_GT	14	26	22	22	20
(GT__19.171.0, linear, gumbel)_GT	18	0	0	4	30
(GT__24.136.0, nonlinear, mim)_GT	4	0	0	0	80
(GT__14.91.0, linear, uniform)_GT	20	0	0	22	28
(GT__14.48.0, nonlinear, mim)_GT	2	2	4	0	2
(GT__19.86.0, nonlinear, quadratic)_GT	28	78	22	16	26
(GT__24.276.0, nonlinear, mlp)_GT	36	28	36	8	52
(GT__14.91.0, linear, gumbel)_GT	16	2	0	18	22
(GT__9.36.0, linear, logistic)_GT	2	26	2	0	2
(GT__30.435.0, nonlinear, quadratic)_GT	0	0	0	134	24
(GT__9.36.0, linear, uniform)_GT	2	0	0	0	0
(GT__9.21.0, linear, gauss)_GT	0	0	6	0	0
(GT__9.36.0, linear, gauss)_GT	4	0	0	0	0
(GT__9.21.0, nonlinear, gp-add)_GT	2	2	0	0	4
(GT__24.276.0, nonlinear, mim)_GT	36	34	34	4	34
(GT__30.210.0, nonlinear, quadratic)_GT	72	44	44	98	88
(GT__14.48.0, linear, logistic)_GT	50	2	2	2	26
(GT__9.36.0, nonlinear, mlp)_GT	4	0	2	0	4

Run	GES	GOLEM	LINGAM	Notears	PC
(GT__14.48.0, linear, uniform)_GT	36	0	0	0	30
(GT__9.21.0, nonlinear, mlp)_GT	0	4	0	0	0
(GT__9.36.0, linear, gumbel)_GT	12	0	0	0	0
(GT__14.48.0, linear, gauss)_GT	32	0	30	2	2
(GT__19.171.0, nonlinear, gp-add)_GT	50	8	8	0	14
(GT__14.91.0, nonlinear, gp)_GT	22	22	24	18	24
(GT__14.48.0, linear, gumbel)_GT	30	0	0	2	0
(GT__9.36.0, nonlinear, mim)_GT	0	0	6	0	10
(GT__9.21.0, nonlinear, mim)_GT	12	0	2	0	0
(GT__19.171.0, nonlinear, mlp)_GT	16	18	8	0	22
(GT__14.91.0, linear, gauss)_GT	18	0	14	16	20
(GT__14.91.0, nonlinear, gp-add)_GT	18	14	20	12	34
(GT__19.86.0, nonlinear, gp)_GT	54	24	22	22	56
(GT__19.171.0, nonlinear, mim)_GT	32	8	8	4	44
(GT__14.91.0, nonlinear, mlp)_GT	14	14	22	10	18
(GT__19.86.0, nonlinear, gp-add)_GT	42	20	18	8	20
(GT__30.435.0, nonlinear, gp)_GT	2	0	0	0	0
(GT__14.91.0, nonlinear, mim)_GT	18	18	20	12	32
(GT__19.86.0, linear, exp)_GT	28	0	0	20	20

Run	GES	GOLEM	LINGAM	Notears	PC
(GT__30.210.0, nonlinear, gp)_GT	54	42	44	42	110
(GT__30.435.0, nonlinear, gp-add)_GT	76	6	2	14	158
(GT__24.276.0, linear, uniform)_GT	36	30	0	34	36
(GT__9.36.0, nonlinear, quadratic)_GT	0	0	0	0	0
(GT__30.435.0, nonlinear, mlp)_GT	60	0	0	0	8
(GT__9.21.0, nonlinear, quadratic)_GT	0	0	0	10	2
(GT__24.276.0, linear, exp)_GT	36	66	0	32	32
(GT__24.136.0, nonlinear, quadratic)_GT	42	0	0	0	16
(GT__14.48.0, nonlinear, quadratic)_GT	26	2	4	30	20
(GT__30.435.0, nonlinear, mim)_GT	0	0	0	0	0

Table E.2.: Table that describes the relationship between the Central and the algorithms

Run	GES	GOLEM	LINGAM	Notears	PC
(GT__19.171.0, nonlinear, quadratic)_Central	0	1	1	1	0
(GT__19.86.0, linear, logistic)_Central	0	1	1	1	0
(GT__9.21.0, linear, exp)_Central	1	1	1	1	1
(GT__19.86.0, linear, uniform)_Central	0	1	1	0	0
(GT__19.86.0, linear, gauss)_Central	0	1	0	0	0
(GT__24.136.0, linear, exp)_Central	0	1	1	0	0
(GT__14.91.0, nonlinear, quadratic)_Central	0	0	1	0	0
(GT__24.276.0, nonlinear, gp)_Central	0	1	1	0	0
(GT__19.86.0, nonlinear, mlp)_Central	0	0	1	0	0
(GT__19.86.0, linear, gumbel)_Central	0	1	1	0	0
(GT__19.171.0, linear, exp)_Central	0	0	0	0	1
(GT__19.86.0, nonlinear, mim)_Central	0	0	1	0	0
(GT__24.276.0, nonlinear, gp-add)_Central	0	0	1	0	0
(GT__30.210.0, nonlinear, gp-add)_Central	1	0	0	0	0
(GT__9.36.0, nonlinear, gp)_Central	0	1	1	1	0
(GT__9.21.0, nonlinear, gp)_Central	1	1	1	1	1
(GT__24.276.0, linear, gumbel)_Central	0	0	0	0	1

Run	GES	GOLEM	LINGAM	Notears	PC
(GT__14.91.0, linear, exp)_Central	0	1	1	0	0
(GT__30.210.0, nonlinear, mlp)_Central	0	0	0	0	1
(GT__24.136.0, nonlinear, gp)_Central	1	1	1	1	0
(GT__9.21.0, linear, logistic)_Central	1	0	1	1	1
(GT__14.48.0, nonlinear, gp)_Central	0	0	1	0	0
(GT__9.21.0, linear, uniform)_Central	1	1	1	1	1
(GT__9.36.0, nonlinear, gp-add)_Central	1	0	1	1	0
(GT__24.136.0, linear, gauss)_Central	0	1	0	0	0
(GT__30.210.0, nonlinear, mim)_Central	0	1	0	0	0
(GT__19.171.0, nonlinear, gp)_Central	0	1	1	1	1
(GT__24.136.0, nonlinear, gp-add)_Central	0	1	1	1	1
(GT__9.36.0, linear, exp)_Central	0	1	1	1	1
(GT__9.21.0, linear, gumbel)_Central	1	1	1	1	1
(GT__14.48.0, nonlinear, gp-add)_Central	0	0	1	0	0
(GT__24.136.0, linear, gumbel)_Central	0	1	1	0	0
(GT__19.171.0, linear, uniform)_Central	0	1	1	0	0
(GT__24.136.0, nonlinear, mlp)_Central	0	1	1	1	0
(GT__19.171.0, linear, gauss)_Central	0	0	0	0	1
(GT__14.48.0, nonlinear, mlp)_Central	0	1	0	1	0

Run	GES	GOLEM	LINGAM	Notears	PC
(GT__14.48.0, linear, exp)_Central	0	1	1	0	0
(GT__14.91.0, linear, logistic)_Central	0	0	1	1	0
(GT__19.171.0, linear, gumbel)_Central	0	1	1	1	0
(GT__24.136.0, nonlinear, mim)_Central	0	1	1	1	0
(GT__14.91.0, linear, uniform)_Central	0	1	1	0	0
(GT__14.48.0, nonlinear, mim)_Central	1	1	0	0	1
(GT__19.86.0, nonlinear, quadratic)_Central	0	0	0	1	0
(GT__24.276.0, nonlinear, mlp)_Central	0	0	0	1	0
(GT__14.91.0, linear, gumbel)_Central	1	0	0	0	0
(GT__9.36.0, linear, logistic)_Central	1	0	0	0	1
(GT__30.435.0, nonlinear, quadratic)_Central	1	1	1	0	0
(GT__9.36.0, linear, uniform)_Central	0	1	1	1	1
(GT__9.21.0, linear, gauss)_Central	1	1	0	1	1
(GT__9.36.0, linear, gauss)_Central	0	1	1	1	1
(GT__9.21.0, nonlinear, gp-add)_Central	0	0	1	1	0
(GT__24.276.0, nonlinear, mim)_Central	0	1	1	0	1
(GT__30.210.0, nonlinear, quadratic)_Central	0	1	1	0	0
(GT__14.48.0, linear, logistic)_Central	0	1	1	1	0
(GT__9.36.0, nonlinear, mlp)_Central	0	1	0	1	0

Run	GES	GOLEM	LINGAM	Notears	PC
(GT__14.48.0, linear, uniform)_Central	0	1	1	1	0
(GT__9.21.0, nonlinear, mlp)_Central	1	0	1	1	1
(GT__9.36.0, linear, gumbel)_Central	0	1	1	1	1
(GT__14.48.0, linear, gauss)_Central	0	1	0	0	0
(GT__19.171.0, nonlinear, gp-add)_Central	0	1	1	0	0
(GT__14.91.0, nonlinear, gp)_Central	0	0	1	0	0
(GT__14.48.0, linear, gumbel)_Central	0	1	1	0	1
(GT__9.36.0, nonlinear, mim)_Central	1	1	0	1	0
(GT__9.21.0, nonlinear, mim)_Central	0	1	0	1	1
(GT__19.171.0, nonlinear, mlp)_Central	0	0	1	0	0
(GT__14.91.0, linear, gauss)_Central	0	0	0	0	1
(GT__14.91.0, nonlinear, gp-add)_Central	0	0	0	1	0
(GT__19.86.0, nonlinear, gp)_Central	0	1	0	0	0
(GT__19.171.0, nonlinear, mim)_Central	0	1	1	0	0
(GT__14.91.0, nonlinear, mlp)_Central	0	1	0	1	0
(GT__19.86.0, nonlinear, gp-add)_Central	0	0	0	0	1
(GT__30.435.0, nonlinear, gp)_Central	0	1	1	1	1
(GT__14.91.0, nonlinear, mim)_Central	0	0	0	1	0
(GT__19.86.0, linear, exp)_Central	0	1	1	0	0

Run	GES	GOLEM	LINGAM	Notears	PC
(GT__30.210.0, nonlinear, gp)_Central	0	1	0	1	0
(GT__30.435.0, nonlinear, gp-add)_Central	0	1	0	0	0
(GT__24.276.0, linear, uniform)_Central	0	0	0	1	0
(GT__9.36.0, nonlinear, quadratic)_Central	1	1	1	1	1
(GT__30.435.0, nonlinear, mlp)_Central	0	1	1	1	0
(GT__9.21.0, nonlinear, quadratic)_Central	1	1	1	0	0
(GT__24.276.0, linear, exp)_Central	0	0	0	0	1
(GT__24.136.0, nonlinear, quadratic)_Central	0	1	1	1	0
(GT__14.48.0, nonlinear, quadratic)_Central	0	0	1	0	0
(GT__30.435.0, nonlinear, mim)_Central	1	1	1	1	1

Appendix F.

Publications

[14]

Title: A Decade of Churn Prediction Techniques in the TelCo Domain: a Survey
Authors: Annalisa Barsotti, Gabriele Gianini, Corrado Mio, Jianyi Lin, Himanshi Babbar, Aman Singh, Fatma Taher, Ernesto Damiani
Journal: SN Computer Science, (to appear)
Year: 2024
Abstract: This work surveys the research contributions of the last decade to the prediction of customer churn and adds a perspective toward what is yet to be reached. The main objective of this article is to report on (1) the methods and algorithms studied, the evaluation metrics adopted, and the results achieved, (2) the data used, and (3) the issues and limitations identified. Furthermore, the work highlights the gaps in the current literature and suggests a direction for future research.

[93]

Title: Retinex by Autoencoders
Authors: Claudio Pezzoni, Corrado Mio, Annalisa Barsotti and Gabriele Gianini
Book: 2022 16th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), pp. 140–147
Year: 2022
Abstract: The Retinex algorithms find wide applications as image enhancers, for their capability of preserving edges, while at the same time attenuating smooth gradients and chromatic dominants. They are characterized by the fact that the output chromatic intensity of a pixel is not determined in isolation (or looking only at the contiguous pixels) but

through an operation of comparison to different local and remote areas of the image. This local/global comparison implies also a high computational cost for the algorithms: their complexity is not linear with the number of pixels; furthermore, the more systematic the comparison, the higher the complexity. Thus, most Retinex algorithms are unfit for real-time processing. The recent development of efficient Machine Learning architectures for Image Processing has raised the question of whether one of the Retinex "transforms" could be efficiently learned by training a feed-forward Artificial Neural Network, thus creating a model characterized by short processing time. Selecting a variant of the Random Spray Retinex model – FuzzyRSR – as representative of the Retinex family, and choosing suitably structured autoencoder neural networks, we found that we could accurately reproduce the Retinex effects. The computational cost of the training phase was moderate, while that of the inference phases was linear in the number of pixels, and three orders of magnitude lower than the one of FuzzyRSR, thus making the ANN implementation of Retinex suitable for real-time processing.

[45]

Title: Heterogeneous Transfer Learning from a Partial Information Decomposition perspective

Authors: Gabriele Gianini, Annalisa Barsotti, Corrado Mio and Jianyi Lin

Book: International Conference on Management of Digital, pp. 133–146

Year: 2023

Abstract:Transfer Learning (TL) encompasses a number of Machine Learning Techniques that take a pre-trained model aimed at solving a task in a Source Domain and try to reuse it to improve the performance of a related task in a Target Domain An important issue in TL is that the effectiveness of those techniques is strongly dataset-dependent. In this work, we investigate the possible structural causes of the varying performance of Heterogeneous Transfer Learning (HTL) across domains characterized by different, but overlapping feature sets (this naturally determine a partition of the features into Source Domain specific sub- set, Target Domain specific subset, and shared subset). To this purpose, we use the Partial Information Decomposition (PID) framework, which breaks down the multivariate information that input variables hold about an output variable into three kinds of components: Unique, Synergistic, and Redundant. We consider that each domain can hold the PID com- ponents in implicit form: this restricts the information directly accessible to each domain. Based on the

relative PID structure of the above men- tioned feature subsets, the framework is able to tell, in principle: 1) which kind of information components are lost in passing from one domain to the other, 2) which kind of information components are at least implic- itly available to a domain, and 3) what kind information components could be recovered through the bridge of the shared features. We show an example of a bridging scenario based on synthetic data.

Bibliography

[1] Abdelrahim Kasem Ahmad, Assef Jafar, and Kadan Aljoumaa. Customer churn prediction in telecom using machine learning in big data platform. *Journal of Big Data*, 6(1):1–24, 2019.

[2] K. Ahmad, A. K., Jafar, A., Aljoumaa. Customer churn prediction in telecom using machine learning in big data platform. *Journal of Big Data*, 6(1):1–24, 2019.

[3] Mahreen Ahmed, Hammad Afzal, Imran Siddiqi, Muhammad Faisal Amjad, and Khawar Khurshid. Exploring nested ensemble learners using overproduction and choose approach for churn prediction in telecom industry. *Neural Computing and Applications*, 32:3237–3251, 2020.

[4] Naomi Altman and Martin Krzywinski. Points of significance: Association, correlation and causation. *Nature methods*, 12(10), 2015.

[5] Adnan Amin, Awais Adnan, and Sajid Anwar. An adaptive learning approach for customer churn prediction in the telecommunication industry using evolutionary computation and naïve bayes. *Applied Soft Computing*, 137:110103, 2023.

[6] Adnan Amin, Feras Al-Obeidat, Babar Shah, May Al Tae, Changez Khan, Hamood Ur Rehman Durrani, and Sajid Anwar. Just-in-time customer churn prediction in the telecommunication sector. *The Journal of Supercomputing*, 76(6):3924–3948, 2020.

[7] Adnan Amin, Babar Shah, Asad Masood Khattak, Fernando Joaquim Lopes Moreira, Gohar Ali, Alvaro Rocha, and Sajid Anwar. Cross-company customer churn prediction in telecommunication: A comparison of data transformation methods. *International Journal of Information Management*, 46:304–319, 2019.

[8] S. Amin, A., Al-Obeidat, F., Shah, B., Adnan, A., Loo, J., Anwar. Customer churn prediction in telecommunication industry under uncertain situation. *Journal of Business Research*, 94:290–301, 2019.

[9] K. Amin, A., Anwar, S., Adnan, A., Nawaz, M., Alawfi, K., Hussain, A., Huang. Customer churn prediction in the telecommunication sector using a rough set approach. *Neurocomputing*, 237:242–254, 2017.

[10] M. M. Andrews, R., Zacharias, R., Antony, S., James. Churn prediction in telecom sector using machine learning. *International Journal of Information*, 8(2), 2019.

[11] J. Apurva Sree, G., Ashika, S., Karthi, S., Sathesh, V., Shankar, M., Pamina. Churn prediction in telecom using classification algorithms. *International Journal of Scientific Research and Engineering Development*, 5:19–28, 2019.

[12] G ApurvaSree, S Ashika, S Karthi, V Sathesh, M Shankar, and J Pamina. Churn prediction in telecom using classification algorithms. *International Journal of Scientific Research and Engineering Development*, 5:19–28, 2019.

[13] Elias Bareinboim and Judea Pearl. Transportability of causal effects: Completeness results. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, pages 698–704, 2012.

[14] Annalisa Barsotti, Gabriele Gianini, Corrado Mio, Jianyi Lin, Himanshi Babbar, Aman Singh, Fatma Taher, and Ernesto Damiani. A decade of churn prediction techniques in the telco domain: a survey. *SN Computer Science*, (to appear), 2024.

[15] Josef Bauer and Dietmar Jannach. Improved customer lifetime value prediction with sequence-to-sequence learning and feature-based models. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15(5):1–37, 2021.

[16] Patrick Blöbaum, Peter Götz, Kailash Budhathoki, Atalanti A. Mastakouri, and Dominik Janzing. Dowhy-gcm: An extension of dowhy for causal inference in graphical causal models. *arXiv preprint arXiv:2206.06821*, 2022.

[17] Gianluca Bontempi. 'statistical foundations of machine learning' handbook. 2021.

[18] Gianluca Bontempi. Between accurate prediction and poor decision making: the ai/ml gap, 2023.

[19] Gianluca Bontempi and Maxime Flauder. From dependency to causality: a machine learning approach. *J. Mach. Learn. Res.*, 16(1):2437–2457, 2015.

[20] Ulrik Brandes. On variants of shortest-path betweenness centrality and their generic computation. *Social networks*, 30(2):136–145, 2008.

[21] Ionut Brandusoiu and Gavril Toderean. Churn prediction in the telecommunications sector using support vector machines. *Margin*, 1:x1, 2013.

[22] G. Brandusoiu, I., Toderean. Churn prediction in the telecommunications sector using support vector machines. *Margin*, 1, 2013.

[23] Leo Breiman. *Classification and regression trees*. Routledge, 2017.

[24] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.

[25] H. Brânduşoiu, I., Toderean, G., Beleiu. Methods for churn prediction in the pre-paid mobile telecommunications industry. *2016 International conference on communications (COMM)*, pages 97–100, 2016.

[26] David Maxwell Chickering. Optimal structure identification with greedy search. *Journal of machine learning research*, 3(Nov):507–554, 2002.

[27] Diego Colombo, Marloes H Maathuis, Markus Kalisch, and Thomas S Richardson. Learning high-dimensional directed acyclic graphs with latent and selection variables. *The Annals of Statistics*, pages 294–321, 2012.

[28] S. Dahiya, K., Bhatia. Customer churn analysis in telecom industry. *In 2015 4th International Conference on Reliability, Infocom Tech-*

nologies and Optimization (ICRITO)(Trends and Future Directions), pages 1–6, 2015.

[29] Anouar Dalli. Impact of hyperparameters on deep learning model for customer churn prediction in telecommunication sector. *Mathematical Problems in Engineering*, 2022, 2022.

[30] Belur V Dasarathy. *Nearest neighbour norms*. IEEE Computer Society Press, Los Alamitos, CA, December 1991.

[31] Sophia Daskalaki, Ioannis Kopanas, M Goudara, and N Avouris. Data mining for decision support on customer insolvency in telecommunications business. *European Journal of Operational Research*, 145(2):239–255, 2003.

[32] Matthieu Defrance. *Lessons learned from empirical and theoretical research*. PhD thesis, Katholieke Universiteit Leuven, 2024.

[33] Floris Devriendt, Jeroen Berrevoets, and Wouter Verbeke. Why you should stop predicting customer churn and start using uplift models. *Information Sciences*, 548:497–515, 2021.

[34] Jacob Dorn, Kevin Guo, and Nathan Kallus. Doubly-valid/doubly-sharp sensitivity analysis for causal inference with unmeasured confounding. *arXiv preprint arXiv:2112.11449*, 2021.

[35] Frederick Eberhardt. Introduction to the foundations of causal discovery. *International Journal of Data Science and Analytics*, 3:81–91, 2017.

[36] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. 2004.

[37] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[38] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. The kdd process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27–34, 1996.

[39] Valerii Vadimovich Fedorov. *Theory of optimal experiments*. Elsevier, 2013.

[40] David L García, Àngela Nebot, and Alfredo Vellido. Intelligent data analysis approaches to churn as a business problem: a survey. *Knowledge and Information Systems*, 51(3):719–774, 2017.

[41] Louis Geiler, Séverine Affeldt, and Mohamed Nadif. An effective strategy for churn prediction and customer profiling. *Data & Knowledge Engineering*, 142:102100, 2022.

[42] Louis Geiler, Séverine Affeldt, and Mohamed Nadif. A survey on machine learning methods for churn prediction. *International Journal of Data Science and Analytics*, pages 1–26, 2022.

[43] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63:3–42, 2006.

[44] Nada Ghanem, Stephan Leitner, and Dietmar Jannach. Balancing consumer and business value of recommender systems: A simulation-based analysis. *arXiv preprint arXiv:2203.05952*, 2022.

[45] Gabriele Gianini, Annalisa Barsotti, Corrado Mio, and Jianyi Lin. Heterogeneous transfer learning from a partial information decomposition perspective. In *International Conference on Management of Digital*, pages 133–146. Springer, 2023.

[46] Yuanhu Gu, Thelma Domingo Palaoag, and Josephine S Dela Cruz. Comparison of main algorithms in big data analysis of telecom customer retention. In *IOP Conference Series: Materials Science and Engineering*, volume 1077, page 012045. IOP Publishing, 2021.

[47] Emil Julius Gumbel. *Statistics of extremes*. Columbia university press, 1958.

[48] Ruocheng Guo, Lu Cheng, Jundong Li, P Richard Hahn, and Huan Liu. A survey of learning causality with data: Problems and methods. *ACM Computing Surveys (CSUR)*, 53(4):1–37, 2020.

[49] Pierre Gutierrez and Jean-Yves Gérardy. Causal inference and uplift modelling: A review of the literature. In *International conference on predictive applications and APIs*, pages 1–13. PMLR, 2017.

[50] Isabelle Guyon, Vincent Lemaire, Marc Boullé, Gideon Dror, and

David Vogel. Analysis of the kdd cup 2009: Fast scoring on a large orange customer database. In *KDD-Cup 2009 Competition*, pages 1–22. PMLR, 2009.

[51] Nabgha Hashmi, Naveed Anwer Butt, and Muddesar Iqbal. Customer churn prediction in telecommunication a decade review and classification. *International Journal of Computer Science Issues (IJCSI)*, 10(5):271, 2013.

[52] Miguel A Hernán and James M Robins. Causal inference, 2010.

[53] Johan Himberg, Aapo Hyvärinen, and Fabrizio Esposito. Validating the independent components of neuroimaging time series via clustering and visualization. *Neuroimage*, 22(3):1214–1222, 2004.

[54] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

[55] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[56] PJ Huber and EM Ronchetti. Robust statistics, wiley: New york, 1981.

[57] Aapo Hyvarinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE transactions on Neural Networks*, 10(3):626–634, 1999.

[58] Adnan Idris, Muhammad Rizwan, and Asifullah Khan. Churn prediction in telecom using random forest and pso based data balancing in combination with various feature selection strategies. *Computers & Electrical Engineering*, 38(6):1808–1819, 2012.

[59] Z. Idris, A., Iftikhar, A., ur Rehman. Intelligent churn prediction for telecom using GP-AdaBoost learning and PSO undersampling. *Cluster Computing*, 22(3):7241–7255, 2019.

[60] Guido W Imbens and Donald B Rubin. *Causal inference in statistics, social, and biomedical sciences*. Cambridge University Press, 2015.

[61] Muhammad Ali Imron and Budi Prasetyo. Improving algorithm accuracy k-nearest neighbor using z-score normalization and particle swarm optimization to predict customer churn. *Journal of Soft Computing Exploration*, 1(1):56–62, 2020.

[62] Hemlata Jain, Ajay Khunteta, and Sumit Private Shrivastav. Telecom churn prediction using seven machine learning experiments integrating features engineering and normalization. 2021.

[63] R. Jain, H., Yadav, G., Manoov. Churn prediction and retention in banking, telecom and IT sectors using machine learning techniques. *In Advances in Machine Learning and Computational Intelligence*, 137-156, 2021.

[64] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*. Springer texts in statistics. Springer, New York, NY, 1 edition, June 2013.

[65] Mark R Jerrum, Leslie G Valiant, and Vijay V Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical computer science*, 43:169–188, 1986.

[66] Muhammad BA Joolfoo, Rameshwar A Jugumauth, and Khalid MBA Joolfoo. A systematic review of algorithms applied for telecom churn prediction. In *2020 3rd International Conference on Emerging Trends in Electrical, Electronic and Communications Engineering (ELECOM)*, pages 136–140. IEEE, 2020.

[67] K. M. Joolfoo, M. B., Jugumauth, R. A., Joolfoo. A systematic review of algorithms applied for telecom churn prediction. *In 2020 3rd International Conference on Emerging Trends in Electrical, Electronic and Communications Engineering (ELECOM)*, pages 136–140, 2020.

[68] Markus Kalisch and Peter Bühlman. Estimating high-dimensional directed acyclic graphs with the pc-algorithm. *Journal of Machine Learning Research*, 8(3), 2007.

[69] N Kamalraj and A Malathi. A survey on churn prediction techniques in communication sector. *International Journal of Computer Applications*, 64(5):39–42, 2013.

[70] V Kavitha, G Hemanth Kumar, G Hemanth Kumar, and M Harish. Churn prediction of customer in telecom industry using machine learning algorithms. *Int. J. Eng. Res. Technol.(IJERT)*, 9(5):181–184, 2020.

[71] Kirui H. Kirui C., Hong, L. Cheruiyot, W. Predicting customer churn in mobile telephony industry using probabilistic classifiers in data mining. *International Journal of Computer Science Issues (IJCSI)*, 10(2 Part 1):165, 2013.

[72] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[73] Praveen Lalwani, Manas Kumar Mishra, Jasroop Singh Chadha, and Pratyush Sethi. Customer churn prediction system: a machine learning approach. *Computing*, 104(2):271–294, 2022.

[74] Steffen L Lauritzen, A Philip Dawid, Birgitte N Larsen, and H-G Leimer. Independence properties of directed markov fields. *Networks*, 20(5):491–505, 1990.

[75] Thuc Duy Le, Tao Hoang, Jiuyong Li, Lin Liu, Huawen Liu, and Shu Hu. A fast pc algorithm for high dimensional causal discovery with multi-core pcs. *IEEE/ACM transactions on computational biology and bioinformatics*, 16(5):1483–1495, 2016.

[76] Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):539–550, 2008.

[77] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, New York, NY, 1 edition, August 2006.

[78] S Mahalakshmi and M Hemalatha. Customer churns prediction in telecom using adaptive logitboost vs peergrading regression learning technique. *IJITEE*, 9(6):1025–1037, 2020.

[79] Denisa Maria Melian, Andreea Dumitrache, Stelian Stancu, and Alexandra Nastu. Customer churn prediction in telecommunication industry. a data analysis techniques approach. *Postmodern Openings*,

13(1 Sup1):78–104, 2022.

[80] Aleksander Molak. *Causal Inference and Discovery in Python*. Packt Publishing, Birmingham, England, May 2023.

[81] Stephen L Morgan and Christopher Winship. *Counterfactuals and causal inference*. Cambridge University Press, 2015.

[82] Nurulhuda Mustafa, Lew Sook Ling, and Siti Fatimah Abdul Razak. Customer churn prediction for telecommunication industry: A malaysian case study. *F1000Research*, 10, 2021.

[83] Ignavier Ng, AmirEmad Ghassami, and Kun Zhang. On the role of sparsity and dag constraints for learning linear dags. *Advances in Neural Information Processing Systems*, 33:17943–17954, 2020.

[84] Ana Rita Nogueira, Andrea Pugnana, Salvatore Ruggieri, Dino Pedreschi, and João Gama. Methods and tools for causal discovery and causal inference. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 12(2):e1449, 2022.

[85] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan kaufmann, 1988.

[86] Judea Pearl. Causal diagrams for empirical research. *Biometrika*, 82(4):669–688, 1995.

[87] Judea Pearl. *Causality*. Cambridge university press, 2009.

[88] Judea Pearl and Elias Bareinboim. Transportability of causal and statistical relations: A formal approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, pages 247–254, 2011.

[89] Judea Pearl, Madelyn Glymour, and Nicholas P Jewell. *Causal inference in statistics: A primer*. John Wiley & Sons, 2016.

[90] Judea Pearl and Thomas S Verma. A theory of inferred causation. In *Studies in Logic and the Foundations of Mathematics*, volume 134, pages 789–811. Elsevier, 1995.

[91] Peyton Z Peebles Jr et al. *Probability Random variables, and random signal principles*. 1987.

[92] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

[93] Claudio Pezzoni, Corrado Mio, Annalisa Barsotti, and Gabriele Gianini. Retinex by autoencoders. In *2022 16th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, pages 140–147. IEEE, 2022.

[94] Saad Ahmed Qureshi, Ammar Saleem Rehman, Ali Mustafa Qamar, Aatif Kamal, and Ahsan Rehman. Telecommunication subscribers' churn prediction model using machine learning. In *Eighth international conference on digital information management (ICDIM 2013)*, pages 131–136. IEEE, 2013.

[95] A. Qureshi, S. A., Rehman, A. S., Qamar, A. M., Kamal, A., Rehman. Telecommunication subscribers' churn prediction model using machine learning. *In Eighth international conference on digital information management (ICDIM 2013)*, pages 131–136, 2013.

[96] Hans Reichenbach. *The direction of time*, volume 65. Univ of California Press, 1956.

[97] Rolf-Dieter Reiss, Michael Thomas, and RD Reiss. *Statistical analysis of extreme values*, volume 2. Springer, 1997.

[98] Afifah Ratna Safitri and Much Aziz Muslim. Improved accuracy of naive bayes classifier for determination of customer churn uses smote and genetic algorithms. *Journal of Soft Computing Exploration*, 1(1):70–75, 2020.

[99] Richard Scheines. An introduction to causal inference. 1997.

[100] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162:83–112, 2017.

[101] Muzaffar Shah, Darshan Adiga, Shabir Bhat, and Viveka Vyeth. Prediction and causality analysis of churn using deep learning. *Comput. Sci. Inf. Technol*, 9(13):153–165, 2019.

[102] Amit Sharma and Emre Kiciman. Dowhy: An end-to-end library for causal inference. *arXiv preprint arXiv:2011.04216*, 2020.

[103] Shohei Shimizu. Lingam: Non-gaussian methods for estimating causal structures. *Behaviormetrika*, 41:65–98, 2014.

[104] Shohei Shimizu, Takanori Inazumi, Yasuhiro Sogawa, Aapo Hyvarinen, Yoshinobu Kawahara, Takashi Washio, Patrik O Hoyer, Kenneth Bollen, and Patrik Hoyer. Directlingam: A direct method for learning a linear non-gaussian structural equation model. *Journal of Machine Learning Research-JMLR*, 12(Apr):1225–1248, 2011.

[105] Ilya Shpitser and Judea Pearl. Identification of joint interventional distributions in recursive semi-markovian causal models. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page 1219. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[106] Anna Śniegula, Aneta Poniszewska-Marańda, and Milan Popović. Study of machine learning methods for customer churn prediction in telecommunication company. In *Proceedings of the 21st International Conference on Information Integration and Web-based Applications Services*, pages 640–644, 2019.

[107] Michael E Sobel. An introduction to causal inference. *Sociological Methods & Research*, 24(3):353–379, 1996.

[108] Yang Song, Jian Huang, Ding Zhou, Hongyuan Zha, and C Lee Giles. Iknn: Informative k-nearest neighbor pattern classification. In *European conference on principles of data mining and knowledge discovery*, pages 248–264. Springer, 2007.

[109] Peter Spirtes, Clark N Glymour, and Richard Scheines. *Causation, prediction, and search*. MIT press, 2000.

[110] Zhiqiang Tan. A distributional approach for causal inference using propensity scores. *Journal of the American Statistical Association*, 101(476):1619–1637, 2006.

[111] Y. H. Tsai, C. F., Lu. Customer churn prediction by hybrid neural networks. *Expert Systems with Applications*, 36(10):12547–12553,

2009.

[112] Irfan Ullah, Basit Raza, Ahmad Kamran Malik, Muhammad Imran, Saif Ul Islam, and Sung Won Kim. A churn prediction model using random forest: analysis of machine learning techniques for churn prediction and factor identification in telecom sector. *IEEE access*, 7:60134–60149, 2019.

[113] V Umayaparvathi and K Iyakutti. Applications of data mining techniques in telecom churn prediction. *International Journal of Computer Applications*, 42(20):5–9, 2012.

[114] K. C. Vafeiadis, T., Diamantaras, K. I., Sarigiannidis, G., Chatzisavvas. A comparison of machine learning techniques for customer churn prediction. *Simulation Modelling Practice and Theory*, 55:1–9, 2015.

[115] Théo Verhelst, Olivier Caelen, Jean-Christophe Dewitte, Bertrand Lebichot, and Gianluca Bontempi. Understanding telecom customer churn with machine learning: from prediction to causal inference. In *Artificial Intelligence and Machine Learning: 31st Benelux AI Conference, BNAIC 2019, and 28th Belgian-Dutch Machine Learning Conference, BENELEARN 2019, Brussels, Belgium, November 6-8, 2019, Revised Selected Papers 28*, pages 182–200. Springer, 2020.

[116] Théo Verhelst, Denis Mercier, Jeevan Shrestha, and Gianluca Bontempi. Partial counterfactual identification and uplift modeling: theoretical results and real-world assessment. *Machine Learning*, pages 1–25, 2023.

[117] Thomas S Verma and Judea Pearl. Equivalence and synthesis of causal models. In *Probabilistic and causal inference: The works of Judea Pearl*, pages 221–236. 2022.

[118] Jaime Vitola, Francesc Pozo, Diego A Tibaduiza, and Maribel Anaya. A sensor data fusion system based on k-nearest neighbor pattern classification for structural health monitoring applications. *Sensors*, 17(2):417, 2017.

[119] W. D. Xia, G. E., Jin. Model of customer churn prediction on support vector machine. *Systems Engineering-Theory Practice*, 28(1):71–77, 2008.

[120] Hong Xu et al. Analysis and comparison of forecasting algorithms for telecom customer churn. In *Journal of Physics: Conference Series*, volume 1881, page 032061. IOP Publishing, 2021.

[121] Liuyi Yao, Zhixuan Chu, Sheng Li, Yaliang Li, Jing Gao, and Aidong Zhang. A survey on causal inference. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15(5):1–46, 2021.

[122] A. (2021) Yaseen. Next-wave of e-commerce: Mobile customers churn prediction using machine learning. *LGURJCSIT*, 5(2):62–72, 2021.

[123] Asif Yaseen. Next-wave of e-commerce: Mobile customers churn prediction using machine learning. *Lahore Garrison University Research Journal of Computer Science and Information Technology*, 5(2):62–72, 2021.

[124] Edward N Zalta, Uri Nodelman, Colin Allen, and John Perry. Stanford encyclopedia of philosophy, 1995.

[125] Alessio Zanga, Elif Ozkirimli, and Fabio Stella. A survey on causal discovery: theory and practice. *International Journal of Approximate Reasoning*, 151:101–129, 2022.

[126] Guozhen Zhang, Jinwei Zeng, Zhengyue Zhao, Depeng Jin, and Yong Li. A counterfactual modeling framework for churn prediction. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 1424–1432, 2022.

[127] Keli Zhang, Shengyu Zhu, Marcus Kalander, Ignavier Ng, Junjian Ye, Zhitang Chen, and Lujia Pan. gcastle: A python toolbox for causal discovery. *arXiv preprint arXiv:2111.15155*, 2021.

[128] Tianyuan Zhang, Sérgio Moro, and Ricardo F Ramos. A data-driven approach to improve customer churn prediction based on telecom customer segmentation. *Future Internet*, 14(3):94, 2022.

[129] Ming Zhao, Qingjun Zeng, Ming Chang, Qian Tong, and Jiafu Su. A prediction model of customer churn considering customer value: an empirical research of telecom industry in china. *Discrete Dynamics in Nature and Society*, 2021, 2021.

[130] Xun Zheng, Bryon Aragam, Pradeep Ravikumar, and Eric P. Xing. DAGs with NO TEARS: Continuous Optimization for Structure Learning. In *Advances in Neural Information Processing Systems*, 2018.