UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

Ismael Santana Silva

EXPLAINABILITY AND CAUSALITY FOR GENERALIZATION APPROXIMATION IN ENVIRONMENTS WITH DISTRIBUTION SHIFTS

Belo Horizonte
2024

Ismael Santana Silva

EXPLAINABILITY AND CAUSALITY FOR GENERALIZATION APPROXIMATION IN ENVIRONMENTS WITH DISTRIBUTION SHIFTS

Final Version

Dissertation presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Advisor: Adriano Alonso Veloso

Belo Horizonte
2024

Silva, Ismael Santana.

S586e Explainability and causality for generalization approximation in environments with distribution shifts [recurso eletrônico] / Ismael Santana Silva – 2024.
1 recurso online (165 f. il, color.) : pdf.

Orientador: Adriano Alonso Veloso.

Tese (Doutorado) - Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação.
Referências: f.137-148

1. Computação – Teses. 2. Aprendizagem do Computador Teses. 3. Classificação (Computadores) – Teses. 4 COVID-19 (Doença) - Teses. 5. Alzheimer, Doença de – Teses. 6. Evasão Escolar – Teses. I. Veloso, Adriano Alonso. II. Universidade Federal de Minas Gerais, Instituto de Ciência Exatas, Departamento de Computação. III.Título.

CDU 519.6*82(043)

Ficha catalográfica elaborada pela bibliotecária Irenquer Vismeg Lucas Cruz
CRB 6/819 - Universidade Federal de Minas Gerais - ICEx

UNIVERSIDADE FEDERAL DE MINAS GERAIS

EXPLAINABILITY AND CAUSALITY FOR GENERALIZATION APPROXIMATION IN ENVIRONMENTS WITH DISTRIBUTION SHIFTS

ISMAEL SANTANA SILVA

Tese defendida e aprovada pela banca examinadora constituída pelos Senhores:

Prof. Adriano Alonso Veloso - Orientador
Departamento de Ciência da Computação - UFMG

Prof. Marcos Oliveira Prates
Departamento de Estatística - UFMG

Prof. Leandro Balby Marinho
Departamento de Sistemas e Computação - UFCG

Prof. George Luiz Medeiros Teodoro
Departamento de Ciência da Computação - UFMG

Prof. Wagner Meira Júnior
Departamento de Ciência da Computação - UFMG

Prof. Anderson Almeida Ferreira
Departamento de Computação - UFOP

Belo Horizonte, 24 de outubro de 2024.

Documento assinado eletronicamente por **Anderson Almeida Ferreira**, **Usuário Externo**, em 05/02/2025, às 19:29, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Leandro Balby Marinho**, **Usuário Externo**, em 06/02/2025, às 11:22, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Marcos Oliveira Prates**, **Professor do Magistério Superior**, em 19/02/2025, às 11:22, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufmg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **3671065** e o código CRC **4EBE8F5D**.

Referência: Processo nº 23072.262203/2024-25 SEI nº 3671065

To my family, for the support, inspiration, and strength they have always given me.

Acknowledgments

I thank my parents, Samuel and Sebastiana, who raised me with love and affection. Furthermore, despite all the adversities in life, they taught me the importance of knowledge and professional qualifications. I also would like to thank my entire family, who have always been an example of unity and willpower to achieve goals. Words cannot express how grateful I am for your constant love and encouragement.

I thank my wife for always believing in me. Besides, I thank my wife for her support, love, and comprehension.

I thank all the professors and teachers who have passed through my life, as each one was important for me to reach my doctorate. I especially want to thank my professor and advisor, Adriano Veloso, for his assertive guidance and masterful way to my doctorate. I thank my colleagues and friends who supported me during my doctorate.

I want to thank the entire PPGCC team for their prompt and willing assistance with administrative issues related to my doctorate. I also thank CEFET-MG and the Brazilian government for their financial support during my doctorate.

Thank you all!

"*The only way to find out what will happen when a complex system is disturbed is to disturb the system, not merely to observe it passively.*"

(G. Mosteller and J. Tukey, 1977)

Resumo

Estudos recentes mostraram que o desempenho estimado de um modelo de classificação em um conjunto de dados de origem específico (treinamento) pode ser muito diferente do desempenho do mesmo modelo após a implantação no mundo real, ou quando avaliamos o modelo em um conjunto de dados alvo com distribuição diferente do conjunto de dados de origem. Podemos chamar esse problema de mudança de distribuição (do inglês *distribution shift*) ou mudança de conjunto de dados (do inglês *dataset shift*), e uma estratégia emergente para esse problema é estimar o desempenho do modelo de classificação em dados não rotulados com distribuição desconhecida, essas estratégias são conhecidas como Avaliação Automática do Modelo (do inglês Automatic Model Evaluation ou AutoEval). A maioria dos trabalhos recentes estudaram como a mudança de distribuição afeta os modelos de aprendizado profundo aplicados a tarefas de visão computacional (ou seja, dados não estruturados). No entanto, as mudanças na distribuição também podem afetar o desempenho de um modelo em dados tabulares/estruturados. Diante deste contexto, esta tese explorou a explicabilidade e o aprendizado de causalidade para propor novas abordagens de AutoEval para modelos aplicados a dados tabulares. Primeiramente, apresentamos o método eXplainability for Automatic Model Eval (X-Eval), um algoritmo de AutoEval baseado no uso de métricas que normalmente são usadas para explicação de modelos (por exemplo, SHAP values e confiança na previsão) para detectar padrões de predições corretas e incorretas, para estimar o desempenho do modelo. Em seguida, propusemos a abordagem Causality for Automatic Model Evaluation (C-Eval), um método de AutoEval baseado na causalidade entre os atributos do conjunto de dados. O objetivo do C-Eval é regularizar a estimativa de um estimador de desempenho (por exemplo, da validação cruzada) de acordo com as mudanças de distribuição detectadas a partir de diferenças nos gráficos causais inferidos dos dados de origem e dos dados alvo. Finalmente, conduzimos experimentos com dados do mundo real e sintéticos. Especificamente, avaliamos as abordagens propostas usando (1) seis conjuntos de dados do mundo real relacionados a três assuntos (isto é, COVID-19, doença de Alzheimer e abandono escolar) e (2) dados sintéticos simulando diferentes tipos de mudanças de distribuição. Nossos resultados indicam que os métodos propostos superaram a linha de base, alcançando até erro zero na estimativa de desempenho do modelo. Além disso, avaliamos nossas abordagens de AutoEval como indicadores para seleção de modelos na tarefa de seleção de atributos. Nesta tarefa, comparado ao CV, os algoritmos propostos obtiveram ganhos de até 77%, em relação à macro f1 no conjunto alvo. Dado o exposto, os métodos propostos podem contribuir para a avaliação contínua de um modelo de classificação em produção (isto é, em execução no mundo real). Além disso, nossas descobertas contribuem para áreas de pesquisa como Aprendizado Semi-Supervisionado, Aprendizado Ativo e Aprendizagem por Transferência, dado que os algoritmos nestas áreas frequentemente lidam com dados de diferentes distribuições e uma estimativa do desempenho do modelo mais precisa pode melhorar a eficácia dos algoritmos nessas áreas.

Palavras-chave: mudança de distribuição; mudança no conjunto de dados; mudança nas covariáveis; desvio de conceito; fora da distribuição; avaliação automática de modelos.

Abstract

Recent studies have showed that the estimated performance of a classification model in a specific source (training) dataset can be very different from the performance of the same model after the deployment in the real world, or when we evaluate the model in a target dataset with different distribution from the source dataset. We can call this problem of distribution shift or dataset shift, and an emerging strategy for this problem is to estimate the performance of the classification model in unlabeled data with unknown distribution (i.e., aka AutoEval approaches). Most recent works have studied how distribution shift affects Deep Learning Models applied to Computer Vision tasks (i.e., unstructured data). However, distribution shifts can also affect the performance of a model on tabular/structured data. This thesis explored explainability and causality learning to propose novel AutoEval approaches on tabular data. First, we presented the method eXplainability for Automatic Model Evaluation (X-Eval), an AutoEval algorithm based on the use of metrics which are typically used as model explanations (e.g., SHAP values, and prediction confidence) to detect patterns of correct and incorrect predictions to estimate model performance. Next, we proposed the Causality for Automatic Model Evaluation (C-Eval) approach, an AutoEval method based on causality among the features. The C-Eval goal is to regularize the estimation of a performance estimator (e.g., 10-fold cross-validation) according to distribution shifts detected from differences in the causal graphs inferred from the source and the target data. Finally, we conducted experiments with real-world and synthetic data. We evaluated the proposed approaches using (1) six real-world datasets related to three subjects (i.e., COVID-19, Alzheimer's disease, and School dropout), and (2) synthetic data simulating different types of distribution shifts. Our results indicated that our proposed methods outperform the baseline, with a reduction of up to 100% in the gap between the estimated and the true performance, compared with standard 10-fold cross-validation error estimation. Besides, we evaluated our AutoEval approaches as indicators for model selection in the feature selection task. In this task, compared to CV, the proposed algorithms achieved gains up to 77%, regarding the macro f1 in the target set. Given this context, our proposed methods can contribute to the continuous evaluation of a classification model in production environments (i.e., executing a task in the real-world). Furthermore, our findings contribute to research areas such as Semi-Supervised learning, Active Learning, and Transfer Learning. Given that the algorithms in these areas often deal with data from different distributions, the improved model performance estimation can improve the efficacy of these algorithms.

Keywords: distribution shift; dataset shift; covariate shift; concept drift; out-of-distribution; automatic model evaluation.

List of Figures

List of Tables

List of Abbreviations and Acronyms

AI	*Artificial Intelligence*
AutoEval	*Automatic Model Evaluation*
BORUTASHAP	*Boruta Feature Selection with SHAP*
C-Eval	*Causality for Automatic Model Evaluation*
CNN	*Convolutional Neural Network*
COVID-19	*Coronavirus Disease 2019*
CV	*Cross-Validation*
DAG	*Directed Acyclic Graph*
HBP	*Hospital Beneficência Portuguesa de São Paulo (COVID-19 dataset)*
HSL	*Hospital Sírio Líbanes (COVID-19 dataset)*
IJCNN	*International Joint Conference on Neural Networks*
LiNGAM	*Linear Non-Gaussian Acyclic Model*
Notears	*No Tears Causal Discovery Algorithm*
PC	*Peter-Clark Algorithm*
PDF	*Probability Density Function*
SHAP	*SHapley Additive exPlanations*
X-Eval	*eXplainability for Automatic Model Evaluation*
UFMG	Universidade Federal de Minas Gerais
Students2008	*School Dropout Dataset - 2008*
Students2014	*School Dropout Dataset - 2014*
Geriatrics	*Alzheimer Dataset - Geriatrics Department*
Neurology	*Alzheimer Dataset - Neurology Department*

Contents

Chapter 1

Introduction

How can we estimate the performance of a classification model on an unlabeled dataset with unknown distribution? Usually, there is a source and a target dataset in the automatic classification process. We build (or train) a classification model using a source dataset and we apply this model to predict labels for the target data (also known as a test dataset). If the source dataset and the target dataset come from the same distribution, we can use a strategy such as k-fold cross-validation (CV) to estimate the model performance [46]. However, if the source and target dataset come from different distributions, the k-fold cross-validation (CV) cannot be adequate to estimate the model performance [61, 54]. This is because a pre-requirement of the CV is that the source and the target datasets are independent and identically distributed (*iid*) [61].

The situation in which the distribution of the target dataset is different from the distribution of the source dataset is known as out-of-distribution, distribution shift or dataset shift [69, 54, 25, 33, 31]. In this thesis, we will use the term distribution shift. Figure 1.1 illustrates the problem of estimating the model performance in an environment where we expect distribution shifts.

Distribution shifts are very common when we deploy a model in the real-world [21, 67, 25, 33]. Since, in the real world, a classification model may face data with different distributions from the source dataset in which it was built because the data can change in different ways. Suppose that a hospital wants to use an X-ray classifier trained on data from another hospital. In that case, distribution shifts can occur because of (1) differences between the X-ray devices or (2) differences in the age of patients treated at each hospital [33]. Thus, several studies have shown that this distribution shift (aka dataset shift) can significantly deviate the estimated performance of a classification model from its true performance [21, 67, 25, 33, 11].

This difference between the estimated performance and the true performance can have severe impacts when the model is applied in the real world. For instance, we can have the prerequisite that a model achieves a minimum performance for this model to be suitable for a given real-world task. However, suppose the model achieves this minimum performance on the source dataset, but the same model does not achieve the performance in the target data. In that case, the mistakes of the model may cause severe consequences

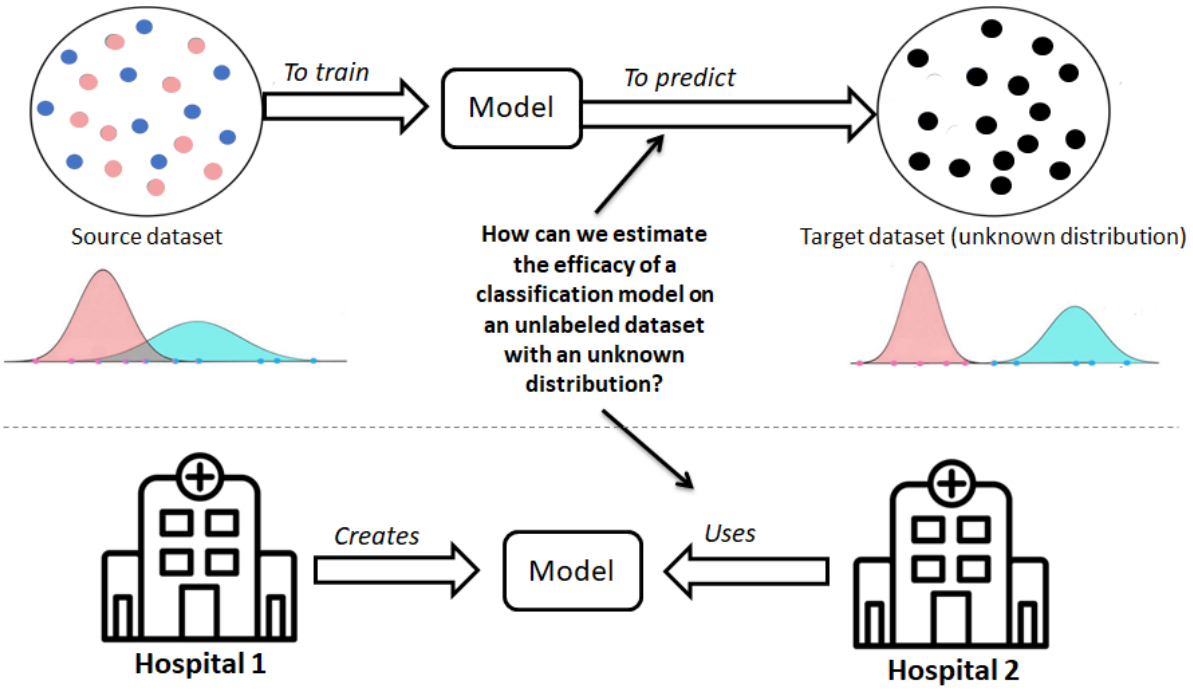
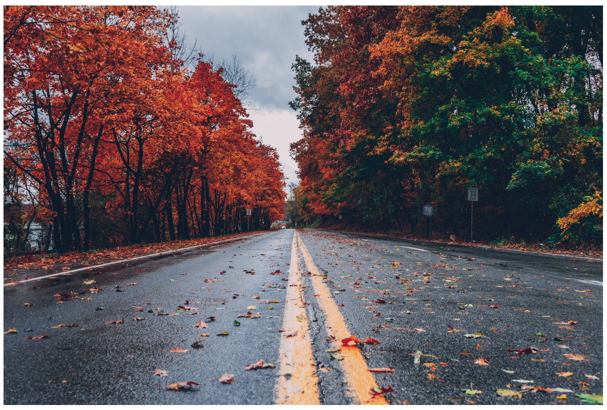
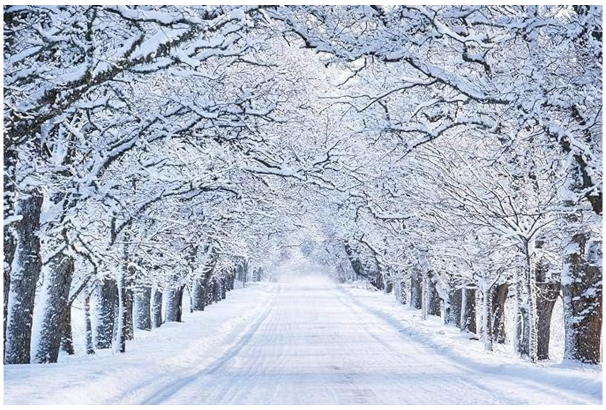

To train
Model
To predict
Source dataset
How can we estimate the efficacy of a classification model on an unlabeled dataset with an unknown distribution?
Target dataset (unknown distribution)
Creates
Model
Uses
Hospital 1
Hospital 2

Figure 1.1: Illustration of the problem of estimate the model performance in an environment where distribution shifts can occur.



(a) Source example [75]



(b) Target example [38]

Figure 1.2: Examples of possible images in the source and target dataset that could cause a distribution shift and degrade the performance of a classification model.

in the real world. For example, an image classification model in an autonomous car may cause severe accidents whether the true performance deviates too much from the estimated performance [11]. Figure 1.2 shows an example in which this situation of distribution shift could occur, where a model may have been trained with images of sunny streets, but this model may face images of streets with snow in the real world. In addition, a wrong estimation may provoke the selection of an unsuitable model to be applied in the real world. A recent approach to deal with the distribution shift problem is to estimate the classification model performance on unlabeled data with unknown distribution [25, 33].

The task of estimating the performance of a classification model on unlabeled data with unknown distribution is called Automatic Model Evaluation (or AutoEval) [25]. AutoEval is particularly appealing when a distribution shift may occur, and the acquisition cost of new label data is prohibited. In this case, unlabeled data is explored in specific ways to measure the performance of the classification model [33].

The majority of recent studies on this topic have applied AutoEval to Deep Learning Models in Computer Vision tasks (i.e., unstructured data) [25, 33]. However, distribution shifts can also affect the performance of models built from tabular data [61], e.g., disease prescription data may have different distributions depending on the devices used for performing laboratory tests or different laboratories. Besides, tabular data is the most commonly used data type in real-world machine learning applications [18, 4].

1.1 Goals and Methodology

Considering the previous context, this work aims to advance the Machine Learning community knowledge regarding the applicability of AutoEval approaches in the context of tabular data. To achieve this goal, we have proposed AutoEval approaches based on explainability and causality learning to estimate the performance of classification models built from tabular data in situations where a distribution shift is expected to occur to some degree. Specifically, we proposed two novel AutoEval methods named eXplainability for Automatic Model Evaluation (X-Eval) [90] and Causality for Automatic Model Evaluation (C-Eval).

In the X-Eval, we proposed the use of explanations (e.g., local feature importance SHAP values) to AutoEval. Once we build a classification model using the source dataset, we can extract the explanations from the source and the target datasets in an unsupervised way. Intuitively, changes in the explanations can provide patterns of correct and incorrect predictions on the source and target datasets. So, we can estimate the model performance on a particular target dataset with these patterns. We introduced the core X-Eval idea in [90].

Specifically, the X-Eval uses explanations because they represent the model from different perspectives. Thus, we investigated the applicability of these perspectives to provide information to model evaluation (i.e., to approximate the estimated performance to the true performance). Thus, we investigated the hypothesis that model representation with explanations can provide patterns of correct and incorrect predictions on environments with distribution shifts.

In turn, the C-Eval employs causality learning [17] to infer causal graphs of the source (training) and target (test) datasets. A causal graph is a directed acyclic graph (DAG), in which the vertices are the features of a dataset and an edge is a causal relation between two features [103]. Then, the C-Eval computes the similarity among inferred causal graphs and uses this similarity to regularize the model performance estimated by the standard k-fold cross-validation or the X-Eval. Consequently, this regularization yields an estimative of the model performance in a specific target dataset (i.e., yields an estimation of the model performance in a possible out-of-distribution target dataset).

Thus, in the C-Eval, from the hypothesis that the causality relationships are stronger than the correlations and that causal relations should not change from one dataset to another, we used causality learning in the C-Eval to investigate the hypotheses that (1) differences between the causal graphs of the source and target datasets indicate distribution shifts [94] and (2) these differences can be used to regularize the estimates of the X-Eval and the k-fold cross-validation.

We evaluated the proposed algorithms using real-world and synthetic data. In our experiments with real-world data, we used six datasets related to three subjects (i.e., COVID-19, Alzheimer's disease, and Dropout School) to evaluate the proposed algorithms on natural distribution shifts. On the other hand, we used synthetic data to evaluate our proposed algorithms on controlled distribution shift types and intensities.

In our experiments, we estimated the performance of two types of models: (1) decision tree ensembles (i.e., XGBoost, LightGBM, and CatBoost) [15] since they are state-of-the-art representative to deal with tabular data; and (2) the Deep Learning model TabNet since Arik and Pfister [4] demonstrated that the TabNet can outperform state-of-the-art classifiers on tabular datasets. We emphasize that our work did not intend to improve model performance, but rather to refine the estimation of model performance.

In the first type of experiment, we evaluated the error between the estimated and the true performance (aka estimation error). In this experiment, our proposed methods outperformed the standard 10-fold cross-validation in the majority of experiments. Besides, in the second type of experiment, we evaluated the usage of the proposed AutoEval methods as indicators within the feature selection task. In this task, our proposed methods achieved gains up to 77% over the standard 10-fold cross-validation.

1.2 Contributions

In terms of contributions, the proposed AutoEval algorithms and our insights can improve the effectiveness of the classification model in several Machine Learning areas, such as Semi-Supervised Learning, Active Learning, Transfer Learning, and Continue

Learning [79], since estimating the model performance in this kind of area is very important.

For instance, a Semi-Supervised algorithm could evaluate the model performance after inserting a new example with a pseudo-label in the training set. Besides, Semi-Supervised learning is an approach to performing Continual Learning [44]. Moreover, researchers have studied the advantages of incorporating AutoEval approaches in Active Learning algorithms [59].

Additionally, the proposed algorithms can contribute to a continuous evaluation of a classification model in the production (i.e., in the real-world execution). The continuous evaluation of a model is a challenge to apply a model in a real-world task [80, 69, 25, 33].

1.3 Thesis Statement

In different situations, the source dataset in which a classification model was built can come from a different distribution of the target dataset. However, the traditional approaches to estimate the model performance (e.g., the k-fold cross-validation) assume that the source and the target dataset come from the same distribution. This requirement is hardly ever attended in the real world because the data distribution can change in an uncontrolled way. Thus, this thesis aims to explore explainability and causality learning to propose approaches to estimate the model performance on an unlabeled dataset with unknown distribution.

1.4 Thesis Structure

The remainder of the thesis is organized as follows: Chapter 2 presents relevant related work. Chapter 3 presents the main concepts used in this thesis. Chapter 4 describes our proposed approaches for Automatic Model Evaluation. Chapter 5 presents the experimental design, in which we describe the main decisions to perform the experimental evaluation of our proposed approaches. In Chapter 6 and Chapter 7, we report the results of experiments with real-world and synthetic datasets with different distribution shifts. Chapter 8 presents a discussion of our results. Finally, Chapter 9 presents conclusions and future works.

Chapter 2

Theoretical Background

This chapter introduces the fundamental concept used in our work. Additionally, we present the background that supports our proposed algorithms. Next, we start by presenting the concept and the different types of distribution shifts.

2.1 Distribution Shift

Distribution shift occurs when the source dataset in which a model was trained differs from the target data that the model is used to predict the label [69]. For example, when we train a model with images of sunny streets and the model is applied to classify images of snowy streets. In this situation, the model probably could fail to predict labels in the snowy streets.

There are different categories of distribution shifts [79]. However, the literature commonly categorizes the distribution shift into the following types:

1. Covariate shift: when $p(\mathcal{X})$ changes between the source and target datasets, but $p(y|\mathcal{X})$ does not change [79];

2. Label shift: when $p(y)$ changes between the source and target datasets, but $p(\mathcal{X}|y)$ does not change [79];

3. Concept drift: when $p(y|\mathcal{X})$ changes between the source and target datasets [107].

Where $\mathcal{X} = \{x_1, x_2, .., x_n\}$ and each $x_i \in \mathcal{X}$ is a set of features or covariates. y is an array with the labels of each $x \in \mathcal{X}$.

Once covariate shift occurs when $p(\mathcal{X})$ changes between source and target dataset, but $p(y|\mathcal{X})$ does not change (i.e., the distribution of the feature changes between the source and the target dataset, but the relationship between the features and the labels does not change), may be counter-intuitive think that this impacts the model performance, since the model learns the function $p(y|\mathcal{X})$ [79]. However, suppose the population in the target dataset increases exactly in a range of feature values in which the model performance is low. In that case, the estimated model performance can deviate too much from the true performance.

For instance, consider a scenario where a model has a high performance in predicting a disease in older individuals but has less efficiency when predicting the same disease in younger people. If we evaluate this model in a hospital where most of the population comprises old individuals, the model performance might be high. However, if we apply the same model in another hospital where the patients are predominantly young, the model performance may be comparatively lower. That is, given a patient, the probability of the disease remains the same (i.e., $p(y|x)$), but there is a distribution shift of patients by age (i.e., $p(\mathcal{X})$).

For example, consider a linear model built from the source data exemplified in Figure 2.1. In this situation, we can get the function represented by the dashed line. However, the solid line represents a model that better fits the target data. Then, from Figure 2.1, we easily see that the estimated model performance could deviate from the true performance because of the position where the target data are localized (i.e., $p(\mathcal{X})$). Besides, this example shows that covariate shifts can impact the model selection process, since an inappropriate model can be selected.

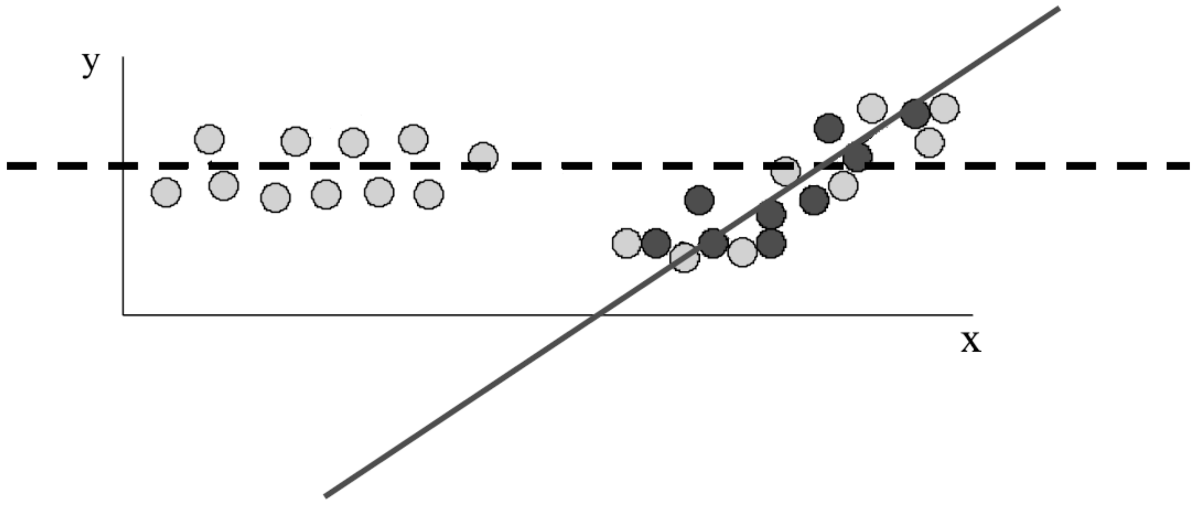


Figure 2.1: Example of covariate shift. We represented the source examples with the light gray color and the target examples with the dark gray color. For example, a linear model has a low performance in the target data using all the source data to build the model (dashed line). However, a linear model focused on the region associated with the target data distribution (solid line) could have a better performance on the target data. Figure adapted from [79].

On the other hand, when we consider the label shift (i.e., $p(y)$ changes, but $p(\mathcal{X}|y)$ does not), if the model assumes that $p(\mathcal{X}|y)p(y)$ is valid from the source and the target datasets to predict the labels, as we would like to predict y it is not a surprise that this form of distribution shift will affect the model prediction. For example, the Naive

Bayes is an algorithm that makes this assumption and this algorithm uses the Bayes rule to infer $p(y|\mathcal{X})$ (i.e., $p(y|\mathcal{X}) = p(y)p(\mathcal{X}|y)/p(\mathcal{X})$). Thus, in this case the estimated model performance will be affected by changes in $p(y)$ between target and source datasets distributions [79]. Figure 2.2 exemplifies the label shift.

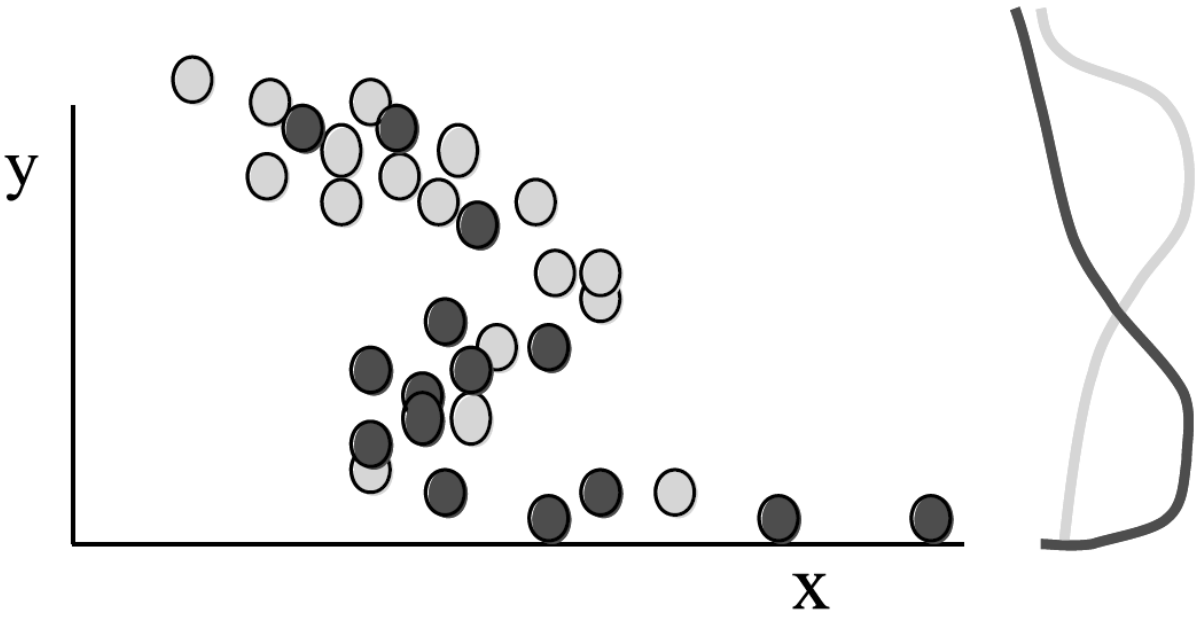


Figure 2.2: A label shift example. The y distribution changed in relation to the source and target datasets, which can impact the predictions and the model performance estimation. Figure extracted from [79].

Concept drift (also referred to as posterior shift) can be considered the most complex distribution shift to be treated when compared with the three presented distribution shift types. Once, concept drift occurs when there are changes in the conditional distribution of the output given an input (i.e., $p(y|x)$ in the source and the target datasets and a model normally learns $p(y|x)$. In simpler terms, in the concept drift, we have the same input, with different output [79].

To illustrate this, consider that in 2019 in Brazil, we had a model to predict car prices based on their features. Before the start of COVID-19 pandemic , in 2019, the average price of a new car was R$ 76,430.59. In 2020, it rose to R$ 86,385.63. In 2021, the average price exceeded R$ 100 thousand, reaching R$ 111,938.31 [1]. In this case, the distribution of the car features remained the same. However, the conditional distribution of car prices given the car features, suffered a significant shift [107].

In addition to the three presented types of distribution shift, [79] describe the Source Component Shift (aka Natural Distribution Shift [31]). In the Source Component Shift, the data come from multiple sources, and each source has its own characteristics. Hence, the data distribution of these sources may vary between the source and the target datasets in various ways. Source Component Shift can be the most frequent distribution shift type.

The Component Source Shift is present in different situations. For example, (1) vote expectations in an election vary depending on the type of the voters' job, and different locations in a country have different distributions of jobs; (2) a large furniture store wants to analyze advertising effectiveness across multiple simultaneous advertising media, but the effectiveness of each will likely vary; (3) the volume of network traffic on a computer system of a university can vary with the time of the year because different groups of students are present or absent at different times [79]. Next, we present the standard k-fold k-cross-validation (CV), a typical strategy to estimate the model performance. Additionally, we show the problems of applying the CV in distribution shift scenarios.

2.2 k-Fold Cross-Validation

The k-fold cross-validation (CV) is a strategy to evaluate a classification model. Essentially, the CV aims to estimate the model performance in a new dataset [28].

In the k-fold cross-validation (CV) we divide the dataset into k subsets. Then, we use k-1 subsets for training and one subset (remainder) to test the model. This process is repeated k times by alternating the test subset. After k iterations, we calculate the model performance as the average performance in the k-subsets [28].

Another strategy to estimate the model performance is the Leave-One-Out approach. The Leave-One-Out approach is a specific case of the CV, where the k data subset is equal to the number of examples in the dataset. An advantage of the Leave-One-Out approach is that it can provide information regarding the model variation in all available labeled data. However, the main disadvantage of this method is that it shows a high computational cost. Once in the Leave-One-Out, we need to train a number of models equal to the labeled examples [28].

The CV controls the computational cost by limiting the number of built models in k. This limitation is essential, especially when we evaluate the model in large datasets [28]. Thus, the Machine Learning community widely uses CV because of these properties. Generally, the 10-fold cross-validation is recommended for the model performance estimation, even when computational cost allows more folds, due to the 10-CV relatively low bias and variance [34, 65].

However, for the CV to effectively estimate the model performance, the source (training) and target (test) datasets must come from the same distribution, i.e., the source and target datasets must be independently identically distributed (*iid*). For example, we can achieve this situation when we have a collection of documents to be classified; next, we sample the source dataset from this document collection; and then, we build a classification model from this sample, as Figure 2.3 exemplifies. The requirement of the source and the target datasets to be *iid* can be statically demonstrated [106, 84, 61, 54] and we can observe this when we use a model in a real-world application [96, 21, 67, 25, 33].

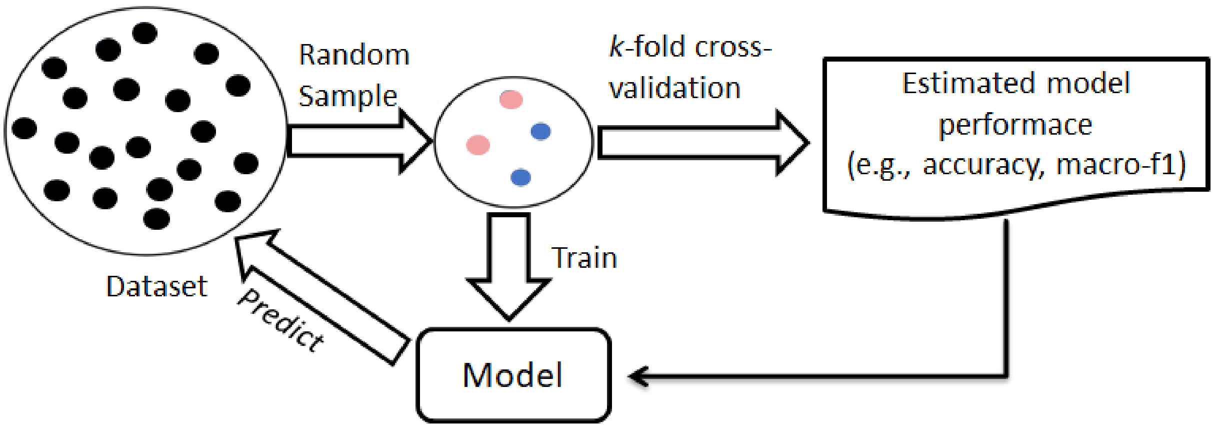

Random Sample
k-fold cross-validation
Estimated model performace (e.g., accuracy, macro-f1)
Train
Dataset
Predict
Model

Figure 2.3: Example of a situation that the k-fold cross-validation is effective to estimate the model performance.

In other words, in many real-world applications, the data distribution can change in different and uncontrolled ways (e.g., in autonomous cars [11], when a hospital creates a model and other hospitals use the same model [33], and spam filters [79]). Hence, the traditional CV estimation, including the leave-one-out, may not reflect the model performance in this kind of real-world application [96, 78]. So, new approaches should be proposed to estimate the model performance when the data distribution can change (aka distribution shift). Next, we present the concept of causality learning and the algorithms for causality graph inference used in this thesis.

2.3 Causality Learning

Causality learning, or causality inference, aims to identify cause-and-effect relationships between variables (i.e., features). Specifically, causality is a relationship between two variables where changes in one variable cause changes in the other variable. Causality inference is important for many empirical sciences [87].

Causality and correlation differ once the correlation between variables indicates that changes in one variable are associated with changes in another. In other words, correlation measures the strength of a relationship between two variables. Causality learning goes beyond correlation and tries to determine whether changes in one variable directly cause changes in another. In this sense, (1) just correlation does not imply causality, and (2) there is causality between two variables does not guarantee that there is a correlation between the same variables [93].

The final goal of causality inference is to discover the causal structure among the variables of the observational data. Normally, the inferred causal relationships are represented as a causal graph [93]. A causal graph of a dataset is a directed acyclic graph (DAG) where we represent the features as vertices, and edges are the causal relationship between features (i.e., when one feature is directly influenced by another feature) [103, 35]. For instance, Figure 2.4 shows a causal graph for the problem of predicting if the pavement is slippery given four features: the season of the year (X_1), if it is raining (X_2), if the sprinkler is on (X_3), and if the pavement is wet (X_4) and a prediction $\mathcal{M}(X)$ of if the pavement is slippery, where $X = \{X_1, .., X_n\}$ [35, 103].

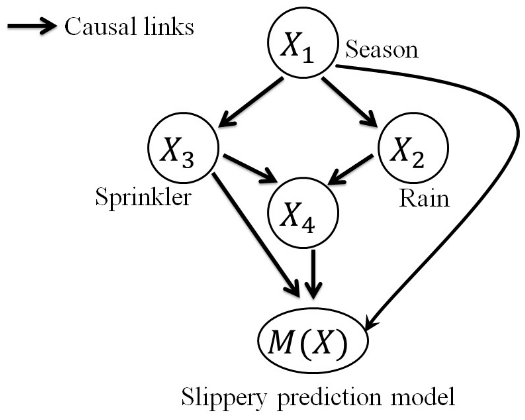


Figure 2.4: A causal graph example for the problem of predicting if the pavement is slippery given four features: the season of the year (X_1), whether it is raining (X_2), whether the sprinkler is on (X_3), and whether the pavement is wet (X_4) and a prediction $\mathcal{M}(X)$ of whether the pavement is slippery, where $X = \{X_1, .., X_n\}$. Figure adapted from [103].

In this thesis, we used the following algorithms for causality inference: DirectLinGAM [87], Notears [114], and PC [93]. Next we present the used causality inference algorithms in this thesis.

2.3.1 DirectLinGAM

The core of the DirectLinGAM algorithm [87] is an approach to estimate a causal ordering of variables (or features) with no prior knowledge of the structure. Causal order is a sequence that indicates whether a variable can have causal relationships (i.e., can cause) with the previous variables in the sequence. The method estimates the causal order of variables by successively removing the most independent variable from the input data. Once the causal order of variables is identified, the connection strengths among the variables can be estimated using a covariance-based methods such as least squares regression.

The DirectLinGAM is based on the Linear Non-Gaussian Acyclic Model (LiNGAM) [86]. The LiNGAM assumes that the observational data are generated from a process involving a causal graph. Let us represent the causal graph with a $m \times m$ adjacency matrix $B = b_{ij}$ where each $b_{ij} \in B$ represents the connection strength from a variable x_j to a variable x_i in the causal graph. Adicionally, let k_i denotes the causal order of variables x_i in the causal graph. Specifically, the causal order indicates that x_i has no path to any $x_{k_j} | j > i$ in the causal graph (a path from x_i to x_j is defined as a sequence of directed edges connecting x_i to x_j). Furthermore, the LiNGAM assumes linearity in the relationships among variables. This is expressed as:

$$x_i = \sum_{k_j < k_i} b_{ij} x_j + e_i,$$

where e_i represent an external influence. Specifically, e are a continuous random variables. [87] emphasize that x_i is equal to e_i whether any variable x_j $(j \neq i)$ has a causal relationship to x_i (i.e.,$b_{Ij} = 0 \forall j \neq i$). In this scenario, x_i is referred to as an exogenous variable. Otherwise, e_i is an error. For instance, considering the model:

$$\begin{aligned} x_2 &= e_2 \\ x_1 &= 1.5x_2 + e_1 \\ x_3 &= 0.8x_1 + 1.5x_2 + e_3, \end{aligned}$$

where x_2 is equal to e_2 since it is not determined by either x_1 or x_3. Consequently, x_2 is an exogenous variable, while e_1 and e_3 are errors. In the LiNGAM, there is at least one exogenous variable $x_i (= e_i)$ because of the acyclicity assumption and the causal graph characteristics. Algorithm 1 shows the DirectLiNGAM algorithm based on the LiNGAM.

Many independence measure can be used in the DirectLiNGAM algorithm. However, [87] proposed the use of following kernel-based independence measure:

$$T_{kernel}(x_j, U) = \sum_{i \in U, i \neq j} MI_{kernel}(x_j, r_i^{(j)}), \tag{2.1}$$

where MI is the common independence measure mutual information, computed by

$$MI_{kernel}(y_1, y_2) = -\frac{1}{2} \log \frac{det \mathcal{K}_k}{det \mathcal{D}_k}, \tag{2.2}$$

where

Algorithm 1: DirectLinGAM

```
   Input: Dataset $X_{original}$, Set of features $U$
   Result: $B$ (adjacency matrix of the inferred causal graph)

 1 $X \leftarrow copy X_{original}$
 2 $K \leftarrow \emptyset$
 3 while $size\ of\ K \leq p - 1$ do
       /*Conduct least squares regressions of $x_i$ on $x_j$ for all $i \in U \backslash K$
       (where $i \neq j$), and calculate the corresponding residual vectors
       $r^{(j)}$ as well as the residual data matrix $R^{(j)}$ from the data matrix
       $X$ for all $j \in U \backslash K$*/
 4     for $j \in U \backslash K$ do
 5         for $i \in U \backslash K (i \neq j)$ do
 6             $r_i^{(j)} = X_i - cov(X_i, X_j)/var(X_j) \times X_j$ and
 7             $R_i^{(j)} = r_i^{(j)}$
           end
       end
       /*Find the most independent feature*/
 8     $X_m = arg\ min_{j \in U \backslash K}$ independenceMeasure($X_j, U \backslash K$)
 9     Append $m$ to the end of $K$
10     $X \leftarrow R^{(m)}$
   end
11 Append the remaining feature to the end of K
   /*Build the adjacency matrix $B$ according to the order in $K$, and
   determine the connection strengths $b_{i,j}$ by employing conventional
   covariance-based regression on the original dataset $X_{original}$ */
   $B \leftarrow$ adjacency matrix with 0 in all positions
12 for $i \leftarrow 1\ until\ i \leq size\ of\ K$ do
       /*$k_{0:i}$ denotes the values in $k$ from the index 0 until the index $i$
       */
13     $predictors \leftarrow k_{0:i}$
14     $target \leftarrow k_i$
15     $regressor \leftarrow$ build a linear regressor from $< X_{predictors}, X_{target} >$
       $B_{target,predictors} \leftarrow$ coefficients estimated by the $regressor$
   end
```

$$\mathcal{K}_k = \begin{bmatrix} (K_1 + \frac{nk}{2}I)^2 & K_1K_2 \\ K_2K_1 & (K_2 + \frac{nk}{2}I)^2 \end{bmatrix}$$

and

$$\mathcal{D}_k = \begin{bmatrix} (K_1 + \frac{nk}{2}I)^2 & 0 \\ 0 & (K_2 + \frac{nk}{2}I)^2 \end{bmatrix},$$

where

$$K_1(y_1^{(i)}, y_1^{(j)}) = exp(-\frac{1}{2\sigma^2}||y_1^{(i)} - y_1^{(j)}||^2)$$

and

$$K_2(y_2^{(i)}, y_2^{(j)}) = exp(-\frac{1}{2\sigma^2}||y_2^{(i)} - y_2^{(j)}||^2)$$

.

. Where k and σ are a small positive constants, n is the number of observed values in a variable (e.g., size of X_i), and I is an $n \times n$ identity matrix. In this thesis, we used the described algorithm to infer a causal graph from a dataset with the DirecLiNGAM. Next, we present the PC-algorithm.

2.3.2 PC Algorithm

The PC (Peter and Clark) algorithm is a causal inference algorithm designed to identify the conditional independence relationships among variables in a dataset. The PC algorithm is widely used for causality inference in observational data and has been applied in various research areas [93, 42].

The PC algorithm first builds a complete undirected graph. Next, the PC algorithm systematically performs conditional independence tests (e.g., chi-square or Fisher's z-transform) between each pair of variables (i, j) in the dataset, given a set of conditioning variables. Then, based on these conditional independence tests, the PC removes edges of the complete undirected graph. Thus, the PC algorithm builds an undirected graph known as the skeleton.

Algorithm 2: PC

```
Input: Vertex Set $V$, Conditional Independence Test
Result: Causal graph $B$

1  Build a complete undirected graph $G$ on the vertex set V
2  $n = -1; B = G$
3  for each ordered pair of adjacent vertices $i, j : |adj(C, i)\backslash\{j\}| < n$ do
4      $n = n + 1$
5      do
6          Select a (new) ordered pair of vertices $i, j$ that are adjacent in $B$ such that
           $|adj(B, i)\backslash\{j\}| \geq n$
7          do
8              Choose (new) $k \subseteq adj(B, i)\backslash\{j\}with|k| = n$
9              if $i$ and $j$ are conditionally independent given $k$ then
10                 Remove the edge $i$, $j$ from $B$
11                 $Sepset(i, j) = k$ and $Sepset(j, i) = k$
               end
           while until edge $i, j$ is deleted or all $k \subseteq adj(B, i)\backslash\{j\}with|k| = n$ have
           been selected
       while until all ordered pairs of adjacent variables $i$ and $j$ such that
       $|adj(B, i)\backslash\{j\}| \geq nandk \subseteq adj(B, i)\backslash\{j\}with|k| = n$ have been tested for
       conditional independence
   end
12 for each triple of vertices i, j, z such that the pair i, j and the pair j, z are each
   adjacent in j but the pair i, z are not adjacent in j do
13     if $j \notin Sepset(i, z)$ then
14         orient $i - j - z$ as $i \rightarrow j \leftarrow z$
       end
   end
15 do
16     if $i \rightarrow j$, $j$ and $z$ are adjacent, $i$ and $z$ are not adjacent, and there is no
       arrowhead at $j$ then
17         orient $j - z$ as $j \rightarrow z$
       end
18     if there is a directed path from $i$ to $j$, and an edge between $i$ and $j$ then
19         orient $i - j$ as $i \rightarrow j$
       end
   while until no more edges can be oriented
```

Then, the PC algorithm applies rules to determine the directions of the edges in the skeleton (i.e., to identify the causal direction of edges in the graph). Algorithm 2 details the PC algorithm [93, 42].

A great challenge of algorithms such as PC and DirectLiNGAM is to estimate conditional independence from finite data. [42] focused on Gaussian scenarios (i.e., where random variables follow a multivariate normal distribution) and assumed that conditional independence relations correspond to d-separations. Here, d-separation is a criterion used to determine whether a set A of variables is independent of another set C, given a third set Z in the causal graph (where the term d-separated emphasizes the directional (d) characteristic of the causal graph). The idea is to associate dependence between the pair of variables i, j with the existence of a path connecting i and j, while independence is associated with unconnectedness or separation.

Thus, given these considerations, Kalisch et al. [42] suggest that conditional independence can be deduced from partial correlations, and the partial correlation of the sample can be calculated via regression. Then, Kalisch et al. [42] use Fisher's z-transformation to test the conditional independence from the partial correlation.

Figure 2.5 (extracted from Peter et al. [93]) illustrates the PC process to build the skeleton graph, considering that a conditional independence test can correctly detect the dependence cases between two variables. Note that when $n = 0$, there is no independence among the variables because there is a path between all pairs of variables in the true graph. However, when $n = 1$, considering the B (in other words, whether there is a path from A to other vertices excluding B), there is no dependence among A and the other vertices, so A is independent of the C, D, and E. Similarly, when $n = 2$, considering $\{C, D\}$ there is no dependence between B and E.

Following the example in Figure 2.5, Figure 2.6 shows how the undirected graph at the bottom of Figure 2.5 is partially oriented by the PC algorithm. These edge orientations occurred because the triples of variables with two adjacent vertices among them are:

$$A - B - C$$
$$C - B - D$$
$$B - D - E$$
$$A - B - D$$
$$B - C - E$$
$$C - E - D$$

In this instance, considering the triple $C - E - D$, $E \notin \mathit{Sepset}(C, D)$ (*Sepset* denotes *separation set*), $C - E$ and $E - D$ collide at the vertex E. The other triplets do not have this kind of collider. So, the resulting pattern generated by the algorithm is shown in Figure 2.6.

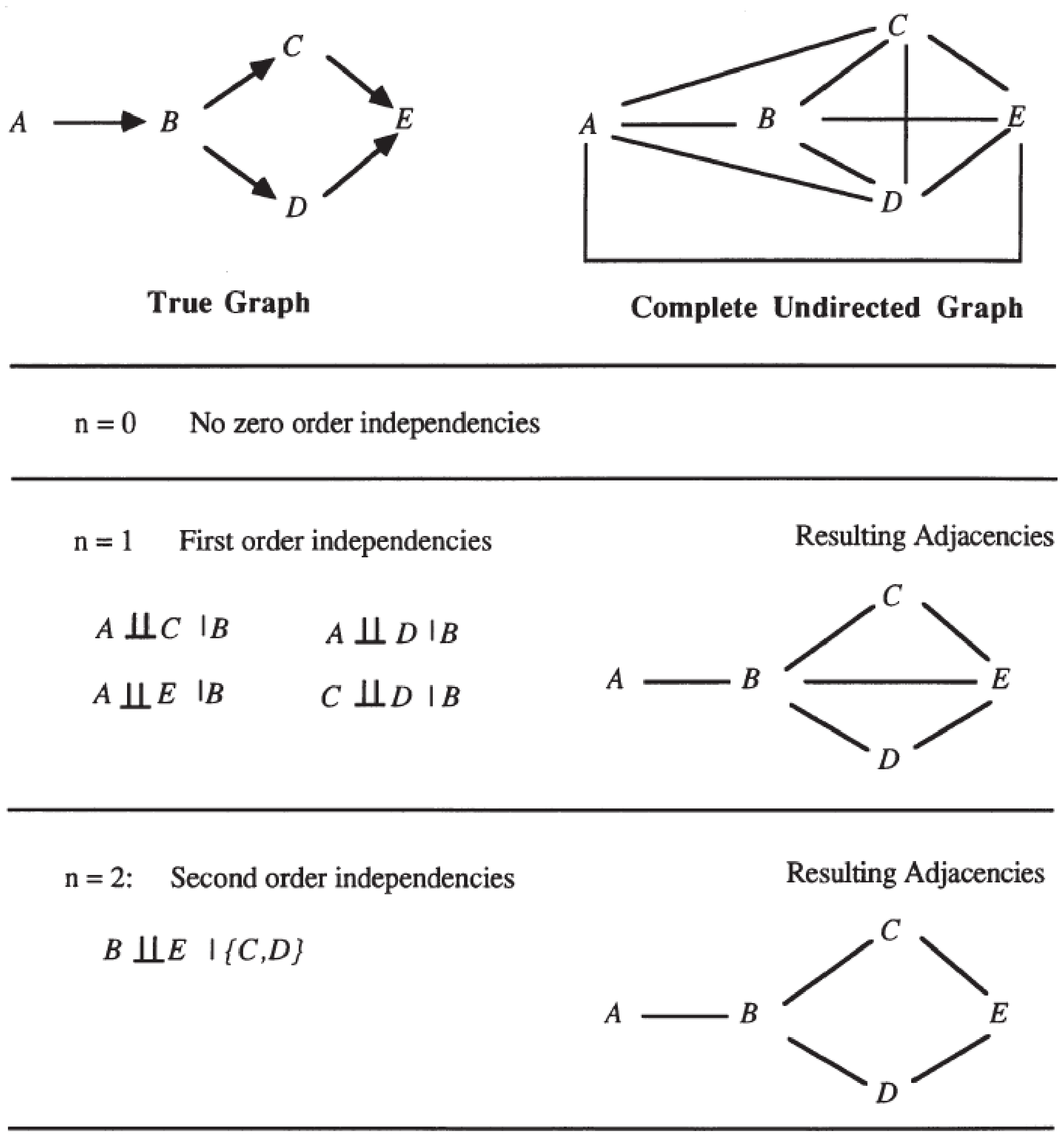

True Graph
Complete Undirected Graph
n = 0 No zero order independencies
n = 1 First order independencies
A ⫫ C | B
A ⫫ D | B
A ⫫ E | B
C ⫫ D | B
Resulting Adjacencies
n = 2: Second order independencies
B ⫫ E | {C,D}
Resulting Adjacencies

Figure 2.5: Illustration of the PC process to build the skeleton graph. Where n denotes the size of the set of conditioning variables. Figure extracted from [93].

Peter et al. [93] highlight that the pattern presented in Figure 2.6 shows a indistinguishability class. Once every orientation of the undirected edges in Figure 2.5 is valid, since it does not include a collision at B. Thus, the PC result is a partially directed acyclic graph (CPDAG), i.e., a graph that can include both directed and undirected edges.

Despite this, the PC algorithm is advantageous because it can handle a large number of variables and efficiently discover causal structures without assuming linearity or specific functional forms. Next, we present the Notears algorithm.

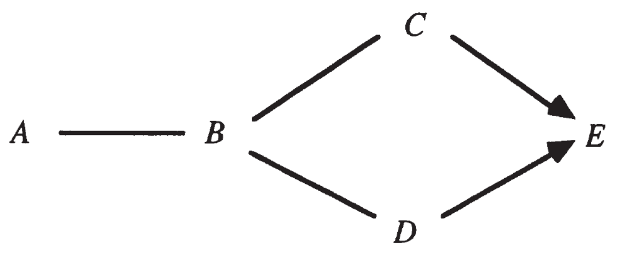


Figure 2.6: The final causal graph of the PC process example. Figure extracted from [93].

2.3.3 NOTEARS

Zheng et al. [114] proposed the method *Non-combinatorial Optimization via Trace Exponential and Augmented lagRangian for Structure learning* (NOTEARS) for inference of causal graphs from data based on a continuous optimization program.

Zheng et al. [114] formulated the causal graph inference as an optimization problem, in which Zheng et al. [114] replaced the constraint of inferring a graph that is a DAG by a function $h(W)$. Given that W denotes a weighted adjacency matrix of a graph $G(W)$, $h(W)$ measures the violations to the inferred graph $G(W)$ to become a DAG. Thus, the goal of the NOTEARS is to solve the following optimization problem:

$$\begin{aligned} &\max_{\alpha \in \mathbb{R}} D(\alpha), \\ \text{where } D(\alpha) &= \min_{W \in \mathbb{R}^{d \times d}} L^{\rho}(W, \alpha), \\ L^{\rho}(W, \alpha) &= F(W) + \frac{\rho}{2}|h(W)|^2 + \alpha h(w), \\ F(W) &= \ell(W, X) + \lambda ||W||_1 \\ &= \frac{1}{2n}||X - XW||^2 + \lambda ||W||_1. \end{aligned}$$

Where $\rho(> 0)$ is a penalty parameter, $X \in R^{n \times d}$ is a data matrix of observations (i.e., a dataset), $\ell(W, X)$ is the least-squares loss (i.e., $\frac{1}{2n}||X - XW||^2$). $F(W)$ adds the ℓ_1-regularization ($||W||_1 = ||vec(W)||_1$) to the graph inference process.

Algorithm 3 shows the main steps of the NOTEARS. We can see that in the optimization process α is updated as $\alpha \leftarrow \alpha + \rho h(W_{\alpha}^{*})$, where W_{α}^{*} is the local W which has been optimizing. Besides, the NOTEARS applies a threshold in the estimates of coefficients to reduce the number of false discovered edges. So, NOTEARS thresholds the edges weight as: after obtaining $\widetilde{W}$ (in the line 7), given a threshold $\omega > 0$, set the weights smaller than ω to zero. Besides, as h($\widetilde{W}$) measures the $\widetilde{W}$ violations to become a DAG, a small threshold ω is enough to remove edges included in cycles. Next, we introduce the similarity functions used in this thesis to compute the similarity between the causal graphs inferred from the source and target datasets.

Algorithm 3: NOTEARS

```
Input: Initial guess $(W_0, \alpha_0)$, progress rate $c \in (0,1)$, tolerance $\varepsilon > 0$, threshold $\omega$ ¿ 0
Result: Causal graph $\widehat{W}$

1 for $t \in \{0, 1, 2, ...\}$ do
2     $W_{t+1} \leftarrow arg\ min_W L^{\rho}(W_t, \alpha_t)$ with $\rho$ such that $h(W_{t+1}) < c\ h(W_t)$
3     $\alpha_{t+1} \leftarrow \alpha_t + \rho\ h(W_{t+1})$
4     if $h(W_{t+1}) < \varepsilon$ then
5         $\widetilde{W} \leftarrow W_{t+1}$
6         break
      end
  end
7 $\widehat{W} \leftarrow \widetilde{W} \circ (\widetilde{W} > \omega)$ // change to 0 all $\widetilde{W}$ positions < $\omega$
```

2.4 Similarity Functions

A similarity function quantifies the similarity between two arrays. Our proposed algorithm C-Eval requires a metric to measure the similarity between two causal graphs. We used the Cosine similarity, the Jaccard Index, and Macro F1 as similarity functions. Firstly, we used these similarity functions because:

1. We represented the causal graphs as adjacency matrices and we can represent a matrix as an array, which makes it possible to use the Cosine, the Jaccard Index, and Macro F1 similarity functions.

2. Furthermore, the used similarity metrics have a smaller computational cost than the typical graph similarity functions, which involve subgraph isomorphism [74]. Moreover, given the small size of the causal graphs, the typical graph similarity functions cannot be applicable.

3. Our proposed method requires that the similarity function return a value between zero and one.

Specifically, we used the Cosine similarity and the Jaccard Index because these are among the most used similarity functions [34]. Besides, considering that the causal graphs are represented as adjacency matrices: (1) these similarity functions are indicated when the arrays that the similarity will be measured are composed of values between zero and one, and (2) these similarity functions are commonly used to compute the similarity of space data [92, 74], which is the case of the graphs represented as adjacency matrices since in this case we have a small number of edges compared with the possible number of edges in the graph.

Cosine similarity measures the cosine of the angle between two vectors. Given two vectors a and b, Han et al. [34] define cosine similarity as:

$$\frac{a \cdot b}{||a|| \cdot ||b||} = \frac{\sum a_i \cdot b_i}{\sum \sqrt{a_i^2} \cdot \sum \sqrt{b_i^2}} \tag{2.3}$$

The Jaccard index, also known as the Jaccard similarity coefficient, is defined as the size of the intersection divided by the size of the union of the evaluated arrays [34]:

$$\frac{a \cap b}{a \cup b} \tag{2.4}$$

In turn, Macro F1 is a metric commonly used to measure the model performance in imbalanced data. Because the Macro F1 averages the model performance on each class (i.e., the Macro F1 attributes the same importance to each class) [22]. We generally get imbalanced data when representing causal graphs with adjacency matrices once the causal graphs have fewer edges than the potential number of edges. Thus, we hypothesized that the proposed method could effectively use the Macro F1 as a similarity function. Besides, once Macro F1 was the metric used to measure the model performance, the interval of values of the graphs similarity will be similar to the range of values of the model performance.

The Macro F1 is computed using the average of the F1 score for each class c. So, given two arrays a and b, and considering a as true labels and b as predictions, and the number of the True Positives (tp), False Positives (fp), and False Negative (fn) for each class c, we compute F1 score as [34]:

$$precision(c) = \frac{tp(c)}{tp(c) + fp(c)} \tag{2.5}$$

$$recall(c) = \frac{TP(c)}{tp(c) + FN(c)} \tag{2.6}$$

$$f1(c) = 2 \cdot \frac{precision(c) \cdot recall(c)}{precision(c) + recall(c)} = \frac{2tp(c)}{2tp(c) + fp(c) + fn(c)} \tag{2.7}$$

.

Finally, we compute the Macro F1 as:

$$macroF1 = \frac{1}{|classes|} \sum_{c \in classes} f1(c) \tag{2.8}$$

.

Next, we present the explainability types used in this thesis to provide information for the proposed X-Eval algorithm.

2.5 Explainability

Given the complexity of some popular classification models (often referred to as black-box models), it raised the necessity to propose explainability approaches for understanding the model predictions. Explainability refers to methods and metrics that help to understand and interpret the predictions made by a black-box model (i.e., a non-explainable model). In this thesis, to provide information for the proposed AutoEval algorithm X-Eval, we used the following metrics typically employed to explainability: model prediction confidence, agreement between model, and SHAP values. Next, we present these explainability metrics. We start by presenting how the model prediction confidence can be employed for explainability.

2.5.1 Explainability based on Prediction Confidence

The prediction confidence is a measure that indicates the possibility that the model prediction is correct. In other words, the prediction confidence quantifies the level of certainty that the model has in its prediction. Generally, an automatic classification model returns a prediction confidence value with a prediction. Thus, the prediction confidence can be used to improve the understanding and trusting in the model prediction [113, 102, 50, 101].

For instance, Le et al. [50] introduced approaches based on counterfactual explanations of prediction confidence to help the users to better understand and trust the model prediction. The counterfactual explanation is the smallest changes in a target instance values to change the model prediction to a desired output. Thus, Le et al. [50] proposed approaches to generate target examples that increase or decrease the model prediction confidence to a specific value.

Additionally, Zhang et al. [113] conducted a study of the impact on the model prediction trust when we present the users the prediction confidence for a particular prediction. Zhang et al. [113] demonstrated that showing the prediction confidence to users can improve the user's trust in the model prediction.

Thus, as Zhang et al. [113] and Le et al. [50] evidenced, approaches based on confidence can provide insights and explainability regarding the model. Thus, this evidence supports the use of prediction confidence to provide information for AutoEval in the proposed X-Eval algorithm.

2.5.2 Explainability based on Agreement between Models

There are strategies that use an explainable model (e.g., Decision Tree or Naive Bayes) to explain the black-box model predictions (e.g., Surrogates Tree). Similarly, some strategies use the agreement between the black-box model and the explainable model to explain the black-box model predictions [49]. Two models agree in a prediction whether both predict the same class to a target instance.

For example, Kuttichira et al. [49] proposed an explainability approach using a decision tree. They train the black-box model and the decision tree in the same training set. Subsequently, they identify the region in which the predictions presented significant differences between the black-box model and the decision tree. They sample a small set of examples in this region and train the decision tree using this data labeled by the black-box model. This process is repeated until the differences between the two models become small. After this process, decision tree rules can be used to explain the black-box model.

Some popular algorithms that build interpretable models are Decision Tree, Decision Rules, k-Nearest Neighbors (k-NN), and Naive Bayes. For instance, Naive Bayes is interpretable because of the independence among features assumption. Hence, the Naive Bayes model provides the possibility of checking the contribution of each feature to the prediction, since we can interpret the conditional probability [10].

Hence, we hypothesized that when the black-box model agrees with an interpretable model, the predictions of the black-box model can be easily explained. Thus, once the model explainability is known to improve the trust in the model predictions [81], we expect that the agreement with an interpretable model will increase the probability of the correct model predictions.

In this thesis, we used the Naive Bayes to compute the agreement between the evaluated model and the interpretable model. From this comparison, we generate the agreement information to be used in the X-Eval algorithm. We chose the Naive Bayes algorithm to bring diversity to the predictions because of the different internal work of this algorithm to the evaluated model. Once, in this thesis, we focused on evaluating tree ensemble algorithms (e.g., the XGBoost). Besides, making predictions with the Naive Bayes is relatively faster than the k-NN. Next, we present SHAP values, the metric used in the X-Eval to measure the importance of the features in the model predictions.

2.5.3 SHAP values

SHapley Additive exPlanations (SHAP) is a popular explainability approach [58, 109, 64, 53, 12]. SHAP values is a way to explain a model by setting the impact or importance of each feature in the model prediction. In the SHAP values, we represent the explanation of a model $\mathcal{M}$ prediction as an m-dimensional array $\mathcal{E}(\mathcal{M}) = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_m\}$ showing which features are driving the model prediction. Specifically, $\mathbf{e}_i$ takes a value that corresponds to the respective feature influence on the model decision. Since we do not assume feature independence while learning $\mathcal{M}$, then correlated features within model $\mathcal{M}$ should share impact or importance [2, 72].

Thus, given an instance x, SHAP provides the weighted contribution of each feature in the outcome (i.e., local feature importance) by learning how the model behaves in the x proximity. SHAP uses an *explanation model* g that is capable of approximating the output of the model $\mathcal{M}(x)$. The explanation model is described as:

$$g(z') = \phi_0 + \sum_{i=1}^{m} \phi_i z'_i \tag{2.9}$$

, where $z' \in \{0,1\}^m$, and $\phi_i \in \mathbb{R}$ is the feature contribution [20, 66]. In the binary classification (i.e., when the model output is 0 or 1), the SHAP values range from -1 to 1, where negative SHAP values indicate that the feature impact for the model predicts the negative label (represented in this thesis by the label 0), and the positive SHAP values indicate the feature impact for the model predicts the positive label (represented by the label 1).

Lundberg and Lee [58] describe SHAP as a permutation-based approach for feature importance attribution which defines a model as a cooperation of features, and it assigns a value for each feature in the cooperation based on its contribution to the model decisions. More precisely, SHAP is a solution to Equation:

$$\phi_i(\mathcal{M}, x) = \sum_{z' \subseteq x'} \frac{|z'|!(m - |z'| - 1)!}{m!} [\mathcal{M}(z') - \mathcal{M}(z' \backslash i)] \tag{2.10}$$

, where

- $z' \subseteq x'$ denotes all z' vectors where the non-zero entries are a subset of the non-zero entries in x';
- $|z'|$ is the number of entries different from zero in z';
- $\mathcal{M}_x(z') = E[\mathcal{M}(z)|z_S]$, S is the feature set of indexes different from zero in z', z_S denotes the original feature values but within missing values for features not in the set S;

- $z' \backslash i$ denotes setting $z'_i = 0$;
- x' is simplified input, which corresponds to the original input x by a mapping function $x = h_x(x')$, where $x'_i = 0 \rightarrow \phi_i = 0$

There are many other feature attribution methods (e.g., [81, 82]), but SHAP is the only method with the three desirable properties:

- Local accuracy: the explanation model $g(z')$ should match the output of the model $\mathcal{M}(x)$ when $x = h_x(x')$ (i.e., $g(z') = \mathcal{M}(h_x(z'))$ when $z' = x'$);
- Missingness: missing features have no attributed impact on the model decisions;
- Consistency: if a model changes and some feature impact increases or stays the same independent of the other features, the feature impact of that feature should not decrease. In other words, SHAP values only change if there is a change in the importance of a feature. This property ensures a consistent interpretation of the model behavior, even after alterations in the model architecture or parameters.

The exact computation of SHAP values is challenging [58]. However, approximated options to calculate the explanation model g include Linear SHAP, Kernel SHAP, Deep SHAP, and Tree SHAP [58, 57]. In this thesis, we used (1) Tree SHAP to compute the SHAP values of models built with ensemble tree classifiers (such as the XGBoost). and (2) Kernel SHAP for the TabNet classifier.

Tree SHAP only takes allowed paths within the feature set, which means that the Tree SHAP does not include non-realistic combinations of features as in other permutation-based methods. Instead, it takes the weighted average of all the model decisions reachable by a certain coalition of features [57]. On the other hand, the Kernel SHAP is a sample-based method that approximates SHAP values using a weighted linear regression [58].

SHAP values differ from the agreement and prediction confidence approaches regarding the granularity of information they offer. Once SHAP values provide information by feature, and the other approaches provide information considering the overall prediction. Consequently, the SHAP values can provide more detailed information regarding the model. In the following section, we present the Bayesian Search optimization strategy used in the X-Eval.

2.6 Bayesian Search

Bayesian Search is an optimization strategy based on the Bayesian methods to guide the search for the best solution in a parameter space. The Bayesian Search is regularly used when the objective function is noisy and expensive to evaluate. The Bayesian Search core idea is to estimate the objective function outcomes using probabilistic models, typically Gaussian Processes (GPs), to decide the next solution to be evaluated [9]. This strategy is interesting because estimating the outcome of the objective function using a probabilistic model has a lower cost than executing the actual objective function. For instance, in the model hyperparameters turning of a Neural Network, the objective function is to build the model and to compute the model performance in a test set [49].

According to Brochu et al. [9], in the Bayesian Search, first, we randomly generate a set of initial candidate solutions. Then, we evaluated this solution using an objective function.

Next, we built a probabilistic surrogate model (e.g., a Gaussian Process) from the observed data solutions. The goal of this surrogate model is to provide an estimate of the objective function outcome. Thus, we use the surrogate model to estimate the conditional probability of the objective function (obf), given the available data (D) (i.e., $p(obf|D)$).

Many techniques can be used to define the surrogate model, although the most popular is using a regression model. A Gaussian Process (GP) is a model that builds a conditional probability distribution over the data, assuming a multivariate Gaussian distribution. The GP can efficiently summarize a great number of functions and smooth the transition as more data are provided to the model.

We can build a GP regression model from a sample of solutions ($X = \{x_1, x_2, ..., x_n\}$) and their objective function outcomes ($y = obf(x_i) \forall x_i \in X$). That is, the candidate solutions and their objective function outcomes are generated to define a dataset $D = \{< x_i, obf(x_i) >, ..., < x_n, obf(x_n) >\}$ to train the GP regressor. We can consider that the GP regressor will estimate the outcome for the new provided solutions as $p(obf|D) = p(D|obf) * p(obf)$. Thus, GP regressor predictions will change as more candidate solutions are provided for the GP regressor.

So, an acquisition function is defined based on the surrogate model. We use the surrogate model to estimate the outcome of a new set of candidate solutions in the space parameters. For instance, we can define the candidate solutions randomly and estimate the objective function outcome using the surrogate model. The next step is to select the candidate solution based on the highest estimated outcome. Then, we evaluate the selected solution with the actual objective function to get its true outcome.

Finally, the new observation is incorporated into the dataset D, and the surrogate model is updated with the selected candidate solution. The process is repeated from the definition of the surrogate model until a stopping criterion (e.g., a specified number of iterations or achievement of a predetermined result). Algorithm 4 summarizes the Bayesian Search process.

Algorithm 4: Bayesian Search

```
Input: $seachspace, obf$
Result: $x_t$ // selected solution

1  $X$ = generate candidate solutions from the $seachspace$
2  $y = obf(x_i) \forall x_i \in X$
3  $D =< X, y >$
4  GP = build a regression model from $D$
5  for $t \in \{1, 2, ..., \textit{until a stopping criterion}\}$ do
6      C = generate candidate solutions // possibly randomly
8      // GP(x) estimates the $obf$ outcome
10     Find $x_t$ by the acquisition function over the GP: $x_t = argmax_{x \in C}$ GP$(x)$
12     $y_t = obf(x_t)$
14     $D = D \cup < x_t, y_t >$
16     Update the GP with $D$
   end
```

Bayesian Search is particularly effective in scenarios where the objective function is expensive to evaluate, and noisy. Bayesian Search has been used in hyperparameters tuning, and other optimization problems [49]. In this thesis, we used the Bayesin Search to looking for the best values of a specific hyperparameters of the proposed algorithm X-Eval, which is presented in the Secton 4.1.2. In the following chapter, we present our related work.

Chapter 3

Related Work

In the literature, there are different strategies to estimate the model performance in an unknown distribution (e.g., Regression-based AutoEval [25, 33, 99], Agreement Among Models [59, 77, 76, 14], and Threshold-based AutoEval [31]). This chapter presents an overview of works related to AutoEval.

3.1 Regression-based AutoEval

This section presents: (1) works that used regression models to predict model performance [25, 33, 99]; and (2) works that introduced metrics highly correlated with model performance in out-of-distribution target data, which can be used in regression to predict model performance [24, 100, 110].

Deng and Zheng [25] have recently introduced the term Automatic model Evaluation (or simply, AutoEval) to reference the problem expressed by the following question:

"*how can we estimate model performance on a test set without labels?*"

To investigate the AutoEval problem, they constructed a meta-dataset in which each instance is a transformed dataset from original images. Specifically, the authors used transformations such as rotation, background substitution, and foreground scaling. To predict the accuracy of the classification model on an unlabeled dataset, Deng and Zheng [25] trained regression models using features such as the Fréchet distance (FD) between the meta-dataset and original images. The authors compared three approaches: (i) based on threshold and predictions confidence, (ii) linear regression trained with FDs, and (iii) neural network regression also trained with FDs. In their experiments, the neural network regression approach achieved better results.

In a similarly way, Guillory et al.[33] also proposed AutoEval approaches based on regression methods. Guillory et al. [33] estimated the model performance on unlabeled data using regression models trained with: (i) the average confidence (AC) of the model predictions; and (ii) the Difference of average Confidences (DoC) on the training set and the test set. The authors evaluated their approaches over a variety of distribution shifts. In their results, Guillory et al. [33] observed that DoC outperformed the feature statistics-based methods (e.g., FD and Maximum Mean Discrepancy) and AC in the AutoEval task.

Sun et al. [99] used distribution shapes, clusters, and representative samples extracted from image features to propose a semi-structured training set representation for regression-based AutoEval. In the experiments, their AutoEval method achieved consistent and competitive results in comparison to existing methods such as FD Deng and Zheng [25], AC, and DoC Guillory et al. [33] for generation of training sets to AutoEval.

Deng et al. [24] applied dispersity to predict model performance in distribution shift situations. Dispersity measures the diversity and distribution of the model predictions among the classes. Deng et al. [24] use the nuclear norm to characterize confidence and dispersity and show that the nuclear norm strongly correlates with the model performance in out-of-distributions datasets.

Weijie et al. [100] proposed a bag-of-prototypes (BoP) dataset representation, which is a semantic bag at the dataset level. In BoP, they create K clusters from a reference dataset. Next, they create a K-dimensional histogram by counting the number of images in each cluster. The BoP representation aims to provide a characterization of the dataset semantic distribution. Then, Weijie et al. [100] use the Jensen-Shannon (JS) divergence to measure the similarity between two datasets. Their experiments demonstrated a strong correlation between the JS distance of source and target datasets represented as BoP and the model accuracy.

Similarly, XIE et al. [110] proposed an approach that uses feature dispersion as an indicator of the model performance in an out-of-distribution target dataset. They allocate the target instances into different clusters based on model predictions. Then, they compute a dispersion score by using the average distances between each cluster centroid and the center of all features, weighted by the sample size of each cluster. The proposed metric was highly correlated with the model performance in their experiments.

Huang et al. [37] aim to improve model performance estimation on out-of-distribution datasets by customizing a meta-dataset. In their proposal, each set of samples in this meta-dataset exhibits a reasonable distribution shift relative to the target dataset.

This section presented approaches to generate statistics regarding the model prediction to train regression models to predict the model performance. However, Garg et al. [31] showed that the correlation between these presented statistics and the true performance could significantly vary when we consider different natures of distribution shifts. Furthermore, in this kind of approach, the information used to predict the model performance is aggregated at the dataset level, thus, it is impossible to consider specific characteristics of an instance in the test set. Next, we present approaches that use Agreement Among Models (which provide instance-level information about the model prediction) to predict the model performance.

3.2 Agreement Among Models

There are initiatives that use agreement (or disagreement) on ensemble classifiers to predict the model performance (e.g., [59, 77, 76, 14]). This kind of approach uses the predictions of two (or more) independently-trained models. Then, the model performance is defined based on how often the models agree in their predictions of the unlabeled data (or some statistics about it) [59].

Madani et al. [59] named this approach as co-validation, and they used the disagreement for performing error estimation and model selection. Madani et al. [59] use as extreme example of distribution shift the Active Learning, where we choose the most informative training instances using a process that does not reflect the test distribution. To compute the agreement among models on the test set, Madani et al. [59] partitioned the training set into equal halves, and they trained a model on each half. Madani et al. [59] showed empirically that in Active Learning, agreement among models: (1) provides a more accurate estimation of the model performance, and (2) is more useful as an indicator for model selection.

Platanios et al. [77] introduced a Bayesian approach that uses the predictions of many classifiers to estimate the model performance on unlabeled data. The central idea is that each model in the ensemble takes a sampled label for each example and changes the label with a probability equal to the model error rate. In their experiments, the proposed methods outperformed state-of-the-art approaches based on agreement among models to estimate the model accuracy.

Platanios et al. [76] proposed a method based on logical constraints to estimate classifiers accuracy, using only unlabeled data. The proposed method is based on the intuition that when the models make a prediction that violates a pre-built constraint, at least one model should have made a mistake.

Chen et al. [14] combined an ensemble of classifiers and self-training to identify misclassification. First, the proposed method iteratively trains an ensemble of models. Then, the proposed method performs self-training to improve the ensemble with the identified misclassified examples. They evaluated the proposed method in 59 tasks related to image classification and Natural Language Processing. The proposed method achieved state-of-the-art results in the experiments.

Baek et al. [6] explored the correlation between the agreement between models in the source and the target datasets to estimate the model performance. They call this correlation of agreement-on-the-line phenomenon. They introduced the ALine-D method, which leverages the agreement-on-the-line phenomenon and uses linear regression to estimate the model performance under distribution shift.

Rosenfeld and Garg [83] introduced a method that trains a secondary model maximizing two objectives: (1) the agreement with the evaluated classifier on the source distribution and (2) the disagreement on the target distribution. Then, the method uses the trained model to calculate the estimation error.

Baek et al. [6] explored the correlation between the agreement between models in the source and the target datasets to estimate the model performance. They call this correlation of agreement-on-the-line phenomenon. They introduced the ALine-D method, which leverages the agreement-on-the-line phenomenon and uses linear regression to estimate the model performance under distribution shift.

This kind of approach that uses agreement among models is intuitive, and we could find good results reported with this strategy in the literature.

3.3 Generalization Bounds

Some studies have proposed generalization bounds using theoretical and empirical complexity measures [26, 5, 40, 19, 41]. This kind of research bounds the generalization gap from complexity measures calculated on a trained model.

Arora et al. [5] proposed generalization bounds that are based on reparametrization of the trained neural network from a compression-based framework. Jiang et al. [40] proposed a measure based on the distances of the training examples to the decision boundary to predict the model generalization gap. Corneanu et al. [19] proposed an algorithm to estimate the model performance without any testing dataset. They used network topology measures to identify and to compute the model error in the test set, even with no access to it. Jiang et al. [41] investigated more than 40 complexity measures and they found out that the Sharpness-based measures (e.g., PAC-Bayesian bounds [26]) perform best.

These bounds (1) are usually numerically loose regarding the true generalization error [31], and (2) they do not focus on predict the model performance in a specific unlabeled dataset. Thus, the works that focus on the prediction of a model performance on an unlabeled dataset with unknown distribution are more relevant to our work.

3.4 AutoEval based on the Model Confidence or Parameters

Garg et al. [31] proposed the Average Thresholded Confidence (ATC), a method to define a threshold to be applied to the confidence of the model prediction. Next, the ATC estimates the accuracy of a model with the portion of unlabeled examples that the model confidence is greater than the defined threshold. ATC identifies a threshold t on the training set such that the number of training examples, with a score less than the threshold, match the error of the evaluated model. Then ATC estimates the model error on the test set by the number of test examples with a score less than t. Garg et al. [31] evaluated as score function the maximum confidence, and the negative entropy. In the experiments, ATC outperforms the baselines, and the ATC estimated the performance on the test set from 2 to 4 times more accurately than the baselines.

Inspired by Garg et al. [31], Lu et al. [56] proposed the Confidence Optimal Transport with Thresholding (COTT) method. COTT leverages optimal transport theory to compute a threshold $t \in \{0, 1\}$. Then, the COTT uses t to estimate the model performance by comparing the threshold t with the transport cost matrix.

Yu et al. [112] proposed the Projection Norm to predict the model performance on an out-of-distribution target dataset. First, the Projection Norm technique uses the model predictions to assign labels to the target dataset. Next, the Projection Norm builds a new model based on these assigned labels. Then, the Projection Norm estimates the model performance from the Euclidean distance of the parameters of the two models (i.e., the models trained with the source and the target data).

Our proposed C-Eval approach is similar to the Yu et al. [112] approach. However, instead of using the similarity between the models parameters, we used the similarity between the causal graphs extracted from the source and the target datasets. We based our idea on literature that suggests a relation between changes in the causal graph and distribution shift [94].

3.5 Domain Adaptation using Causality

Many works use causal graphs to learn a feature subset invariant for distribution shifts. These works use these invariant features to build models to make predictions from a new target dataset. This problem is known as domain adaptation [94, 60, 95]. However, in this thesis, our focus is on a different issue than the domain adaptation problem. Specifically, we aim to address the challenge of estimating the model performance in a specific out-of-distribution target dataset.

Typically, the works about domain adaptation provide theoretical guarantees about minimax optimal performance (i.e., the best possible performance of a model in the worst-case scenario) [94]. However, these works do not estimate the model performance in a specific target dataset. This kind of estimation is important because the distribution can be stable among different datasets and there can be different distribution shifts among other datasets [94].

3.6 AutoEval based on Feature Importance

Mougan et al. [70] proposed the use of explanation shifts as a way to detect distribution shifts that can impact on the model predictions. The authors conducted a mathematical analysis to demonstrate that these measures of explanation shifts are more reliable indicators of the model performance than other measures of distribution shifts.

3.7 Weighting the examples based on the target data

Shimodaira [88] introduced an approach for weighting the source examples based on the impact on the model. This approach assigns higher weights to source examples in regions with high density in the target data. Sugiyama et al. [98] extended this concept, they proposed a method to estimate generalization error considering covariate shift.

Subsequently, Sugiyama et al. [96] and Sugiyama et al. [97] used the adaptability of importance weighting to improve the cross-validation (CV) performance estimation.

Our proposed C-Eval approach is similar to the Sugiyama et al. [96] and Sugiyama et al. [97] approaches. However, rather than using the density to weight the examples in the CV, we regularize the (e.g., CV) performance estimated by the similarity between the causal graph inferred from the source and target datasets. Once, according to Subbaswamy et al. [94], causality relations are strongly related to distribution shifts.

3.8 Our Work

Our work differs from the aforementioned ones because, to the best of our knowledge: (a) our work was the first to use explanations to Automatic Model Evaluation [91], and (b) our initiative is the first work that uses causality in the classification model evaluation to estimate the model performance on an out-of-distribution target dataset (i.e., dataset drift).

We emphasize that the goal of our approach is to estimate the model performance in a specific out-of-distribution dataset (i.e., AutoEval), rather than focusing on improving or guaranteeing model performance in scenarios involving distribution shifts (e.g., building *minimax* optimal models), as was the case in [104, 52, 29, 7, 105, 48, 94, 39]. While there are some similarities between these two tasks, certain characteristics justify separating the research in each field. For instance, suppose that there are multiple models with high performance on a source dataset, and we aim to select the most suitable one to apply to a specific target dataset. In such situations, our focus is not on creating a new model but choosing the best existing model. Another example is when we would like to monitor the model performance after deployment in the real world. Next, we present the algorithms proposed in this thesis.

Chapter 4

Proposed Approaches

This chapter presents our proposed approaches to AutoEval: eXplainability for Automatic Model Evaluation (X-Eval) and Causality for Automatic Model Evaluation (C-Eval). The proposed algorithms aim to estimate the model performance in an unlabeled dataset with an unknown distribution. That is, given a classification model $\mathcal{M}$ trained on a source (training) dataset $\mathcal{D}$=<$\mathcal{X}_{train}, y_{train}$> (where $\mathcal{X}_{train}$=$\{x_1, .., x_n\}$, x_i is an array with the values of m features, and y_{train} is an array with the label for each instance $x_i \in \mathcal{X}_{train}$), and a target (test) dataset $\mathcal{T}$=<$\mathcal{X}_{target}, ?$> (where $\mathcal{X}_{target}$=$\{x_1, .., x_o\}$, and x_i is an array with the values of m features), considering that $\mathcal{T}$ is unlabeled and its distribution is unknown, the proposed algorithm aims to estimate the performance of a $\mathcal{M}$ on $\mathcal{T}$. Next, we present the X-Eval.

4.1 X-Eval: eXplainability for Automatic Model Evaluation

This section presents the proposed algorithm to AutoEval with common metrics used to explainability. We name this algorithm as eXplainability for Automatic Model Evaluation (referred to as X-Eval). We split the X-Eval into two algorithms: (1) the X-EvalCore, an interactive algorithm to estimate model performance on a specific target dataset, and (2) the complete X-Eval, an extension of the X-EvalCore using Bayesian Search to improve the performance estimation and the approximation of the target data distribution. The next section presents the X-EvalCore.

4.1.1 X-EvalCore: The X-Eval Fundamental Concept

The main idea of the X-EvalCore is to approximate the performance of the model ($\mathcal{M}$) in the target data ($\mathcal{T}$) by the average of k estimations of the performance of the model $\mathcal{M}$ in the $\mathcal{T}$. To estimate the performance of the model $\mathcal{M}$ in the target data $\mathcal{T}$, we use a second model (denoted as $\mathcal{M}_{autoEval}$) to classify each prediction of $\mathcal{M}$ as correct or incorrect. We train the second model $\mathcal{M}_{autoEval}$ using data commonly used to explain the model. We denoted the true performance of $\mathcal{M}$ on $\mathcal{T}$ as $\ell(\mathcal{M}, \mathcal{T})$ and the estimation of the $\ell(\mathcal{M}, \mathcal{T})$ as $\ell'(\mathcal{M}, \mathcal{T}, p_{corIncor})$, where $p_{corIncor}$ is a array which $p^{i}_{corIncor}$ indicates whether the i-th prediction is correct or incorrect.

We used the following model explainability approaches to provide data to the X-Eval process: (1) confidence in the model prediction [33], (2) agreement among two models [14], and (3) SHAP values (i.e., feature importance in the prediction) [90, 70].

As Figure 4.1 illustrates, the core premise of the X-Eval algorithm is that there are similar patterns in the model explanation in the source and the target datasets, which are kept in different data distributions. For instance, even with differences in the distributions of the target and the source datasets: (1) a low prediction confidence indicates an incorrect prediction in the target and in the source datasets; (2) a disagreement among classifiers can indicate an incorrect prediction in both datasets; and (3) if the model predicted the positive label and the most features impacted the model for the negative label, this indicates a mistake in the target and source datasets.

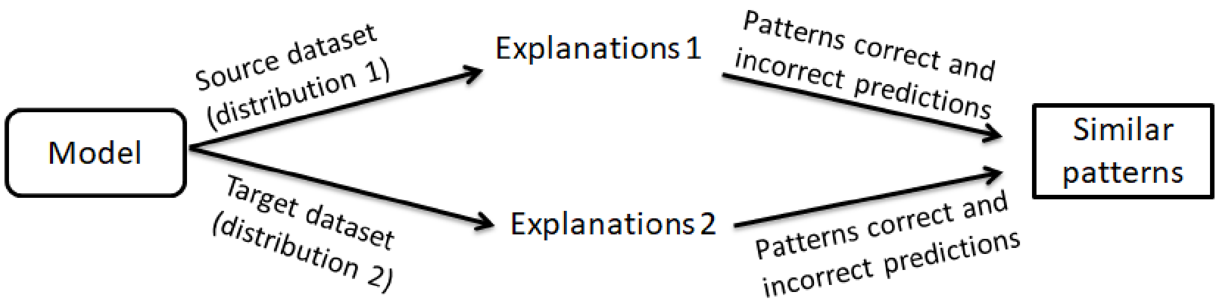

Model
Source dataset (distribution 1)
Target dataset (distribution 2)
Explanations 1
Explanations 2
Patterns correct and incorrect predictions
Patterns correct and incorrect predictions
Similar patterns

Figure 4.1: X-Eval central premise.

Figure 4.2 illustrates the main X-Eval idea and the X-EvalCore method implements this main idea. Algorithm 5 shows the main steps of the X-EvalCore method [90]. The goal of the X-EvalCore is to approximate the target dataset distribution with a classification model (referred to as AutoEval Model and denoted as $\mathcal{M}_{autoEval}$) built from the explanations of the model $\mathcal{M}$ predictions. For this, we use an explainer function $Explainer(\mathcal{M}, < \mathcal{X}, y >)$ that extracts explanations about the

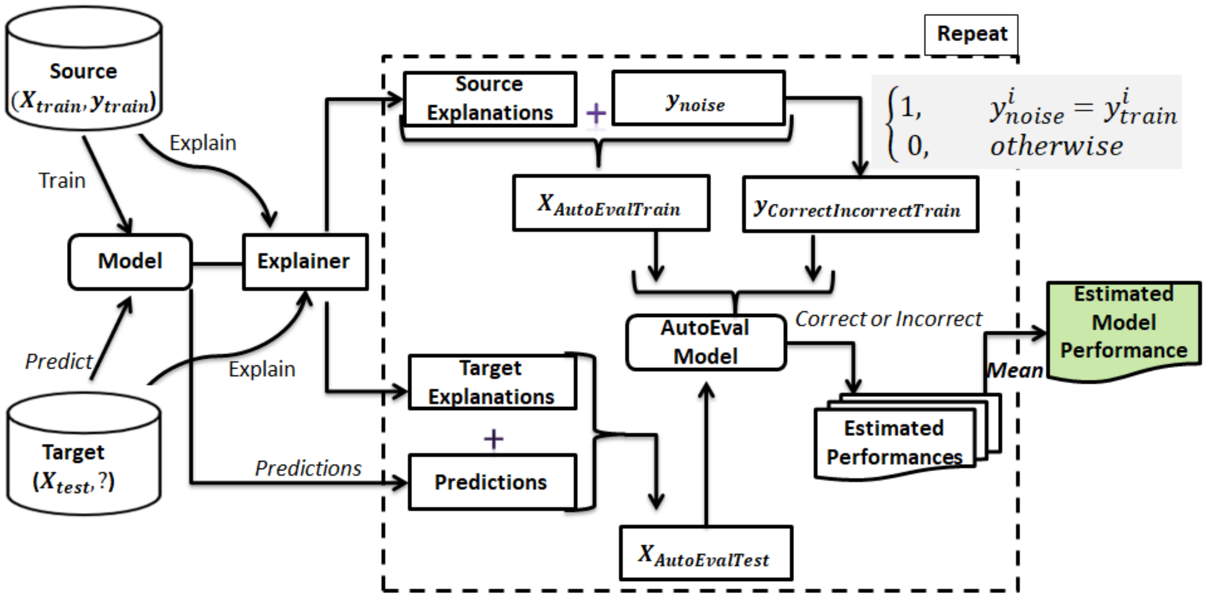


Figure 4.2: X-EvalCore fluxogram.

predictions of the model $\mathcal{M}$ on a dataset $\mathcal{X}$ (we describe the *Explainer* function in the Section 4.1.3, Section 4.1.4, Section 4.1.5, and Section 4.1.6).

Thus, from the explainer $Explainer(\mathcal{M}, \mathcal{D})$ we start the creation of a dataset $\mathcal{X}_{sci}$ (to train $\mathcal{M}_{autoEval}$), which can provide patterns of correct or incorrect predictions, i.e., $\mathcal{X}_{sci}$=$Explainer(\mathcal{M}, \mathcal{D})$. To complement the $\mathcal{M}_{autoEval}$ training dataset (¡$\mathcal{X}_{sci}$,, . ¿), we create an array y_{sci} of labels which indicates whether each $\mathcal{M}$ prediction is correct or incorrect. Thus, to create y_{sci}, we simulate predictions of a model building a noise array (y_{noise}) with possible labels to the y_{train}. We defined y_{noise} as a noised array to approximate different distributions that could occur on target data $\mathcal{T}$. One approach to construct the y_{noise} is by generating a random array (i.e., $y_{noise} = \{r\}^{|y_{train}|}$, where r is a random possible label for y_{train}). From the y_{noise} we create y_{sci} checking if y^i_{noise}=y^i_{train}, i.e.,

$$y^i_{sci} = \begin{cases} 1, & \text{if } y^i_{noise} = y^i_{train} \\ 0, & \text{otherwise} \end{cases} \forall i \in \{1, .., |y_n|\} \tag{4.1}$$

.

So, we concatenate $y^{\intercal}_{noise}$ to the columns of the $\mathcal{X}_{sci}$ (i.e., $\mathcal{X}_{sci} = y^{\intercal}_{noise} \cup \mathcal{X}_{sci}$). Then, we train $\mathcal{M}_{autoEval}$ on $<\mathcal{X}_{sci}$, $y_{sci}>$.

For the $\mathcal{M}_{autoEval}$ to classify $\mathcal{M}$ predictions as correct or incorrect and to estimate the model performance on the target dataset (i.e., to estimate the $\ell(\mathcal{M}, \mathcal{T})$), we also need to create a target dataset based on the model explanations on the $\mathcal{X}_{target}$. So we define $\mathcal{X}_{tci}$ by concatenating (1) the $\mathcal{M}$ predictions on the target data $\mathcal{T}$ and (2) the model explanations prediction in the target dataset (i.e., $Explainer(\mathcal{M}, \mathcal{T})$). That is, $\mathcal{X}_{tci}$=$\mathcal{M}(\mathcal{T})^{\intercal} \cup Explainer(\mathcal{M}, \mathcal{T})$. Then, we use the prediction of $\mathcal{M}_{autoEval}$ (i.e., $\mathcal{M}_{autoEval}(\mathcal{X}_{tci})$) to estimate the performance of the model $\mathcal{M}$ on the target dataset $\mathcal{T}$

Algorithm 5: X-EvalCore

```
Input: $\mathcal{M}, \mathcal{D}, \mathcal{T}, Y_{noise}$
Result: $estimatedPerformance$

1 $estimates \leftarrow \emptyset$
2 forall $y_{no} \in Y_{noise}$ do
3     $\mathcal{X}_{sci}^{y_{no}} = Explainer(\mathcal{M}, \mathcal{D}) \cup y_{no}^{\intercal}$
4     $y_{sci} = \begin{cases} 1, & \text{if } y_{no}^i = y_{train}^i \\ 0, & \text{otherwise} \end{cases} \forall i \in \{1, .., |y_n|\}$
5     $\mathcal{M}_{ci}^{y_{no}} \leftarrow$ model built from $< \mathcal{X}_{sci}, y_{sci} >$
6     $\mathcal{X}_{tci} \leftarrow Explainer(\mathcal{M}, \mathcal{T}) \cup \mathcal{M}(\mathcal{X}_{target})^{\intercal}$
7     $estimate \leftarrow \ell'(\mathcal{M}, \mathcal{T}, \mathcal{M}_{ci}^{y_{no}}(\mathcal{X}_{tci}))$
8     $estimates \leftarrow estimates \cup estimate$
  end
9 $estimatedPerformance \leftarrow$ average of $estimates$
```

(i.e.,$\ell'(\mathcal{M}, \mathcal{T}, \mathcal{M}_{autoEval}^i(\mathcal{X}_{tci}^i))$).

Given $Y_{noise}=\{y_{noise}^1, .., y_{noise}^k\}$, the X-EvalCore process is executed with each $y_{noise} \in Y_{noise}$ because: (1) X-EvalCore involves randomness to approximate different distributions which could occur on $\mathcal{T}$, and (2) this repetition is important to decrease the variance and the estimation error.

The final estimated performance is the average of all estimates made with each $y_{noise} \in Y_{noise}$. In other words, as the expected value ($E[X]$) of a random variable is the average of all possible values of the variable [106, 84], we approximated $E[\ell(\mathcal{M}, \mathcal{T})]$ using:

$$E[\ell(\mathcal{M}, \mathcal{T})] \approx \frac{1}{|Y_{noise}|} \sum_{y_{no} \in Y_{noise}} \ell'(\mathcal{M}, \mathcal{T}, \mathcal{M}_{autoEval}^{y_{no}}(\mathcal{X}_{tci})) \tag{4.2}$$

, where $\mathcal{M}_{autoEval}^{y_{no}}$ denotes the $\mathcal{M}_{autoEval}$ built from a $y_{no} \in Y_{noise}$.

This section presented the X-EvalCore, a novel method for estimating the model performance in the target dataset. Leveraging the X-EvalCore characteristics, we integrated the X-EvalCore into the X-Eval algorithm. Next, we present the X-Eval, which incorporates Bayesian Search to the X-EvalCore to improve the approximation of the target data distribution and to refine the performance estimation of the model performance in the target dataset.

4.1.2 X-Eval: Extending the X-EvalCore using Bayesian Search

To improve the X-EvalCore estimation, we used a Bayesian Search to search for Y_{noise} that better represents the target data distribution. According to Rossi [84], the X-EvalCore could be considered MSE-consistent if the mean squared error (MSE) of the estimator X-EvalCore and $\ell(\mathcal{M}, \mathcal{T})$ goes to zero as the number to the used y_{noise} (denoted as $|Y_{noise}|$) goes to infinity, i.e.,

$$\lim_{|Y_{noise}| \to \infty} \text{MSE}(\text{X-EvalCore}(\mathcal{M}, \mathcal{D}, \mathcal{T}, Y_{noise}), \ell(\mathcal{M}, \mathcal{T})) \to 0 \tag{4.3}$$

. Thus, the X-EvalCore can be as accurate as desired by setting a sufficiently large number of y_{noise} (i.e., $|Y_{noise}|$).

However, it is impossible to use an infinity number of y_{noise} (i.e., $|Y_{noise}| \to \infty$) in the X-EvalCore algorithm. In this sense, we hypothesized that the following components can improve the X-EvalCore estimation and approximate the X-EvalCore of the state of MSE-consistent:

1. The accuracy of the $\mathcal{M}_{autoEval}$, which depends on (a) the choice of a Y_{noise} set that decreases the estimation error, (b) the quality of the provided explanations (by the *Explainer*) used to judge each target instance as correct or incorrect, and (c) the effectiveness of the classifier used to build $\mathcal{M}_{autoEval}$;

2. The size $|Y_{noise}|$ must be large enough to make the estimation error goes to the desired value.

To deal with the components 1.a (the choice of a Y_{noise} set) and 2 (the size $|Y_{noise}|$ must be large enough), we proposed the use of Bayesian Search in the X-Eval algorithm. X-Eval uses Bayesian Optimization [68, 9] to search for the best values for each $y_{noise} \in Y_{noise}$ to be applied in the X-EvalCore.

To deal with component 1.b (the quality of the provided explanations), we showed the applicability and effectiveness of feature importance (SHAP values) to provide information to AutoEval.

To deal with component 1.c (the effectiveness of the classifier used to build $\mathcal{M}_{autoEval}$), in this thesis, we used the XGBoost classifier to create the AutoEval Model ($\mathcal{M}_{autoEval}$) because it is a state-of-art representative on automatic classification in the context of tabular data.

Thus, aiming to reduce the estimation error of the X-EvalCore by choosing better values for Y_{noise}, we proposed the use of the Bayesian Search to search for Y_{noise} that can approximate the estimated performance to the true performance. This algorithm is an interactive process to define the Y^k_{noise} based (1) on the previous $Y^1_{noise}, .., Y^{k-1}_{noise}$ and (2) in the objective value $(v^1, .., v^{k-1})$ associated to each $Y^1_{noise}, .., Y^{k-1}_{noise}$. The objective value v^i is a noisy measure of the X-EvalCore effectiveness using a $Y_{noise} \in \{Y^1_{noise}, .., Y^{k-1}_{noise}\}$ to estimate the true performance $\ell(\mathcal{M}, \mathcal{T})$. We chose the Bayesian Optimization to search for the best parameter Y_{noise} because this is a powerful strategy to seek the maximum and the minimum of a function when the objective function is complex, noisy, and expensive to evaluate [68, 9]. Thus, the Bayesian Optimization was a logical definition once our objective function is:

1. Complex and expensive to compute since it depends on a call of the X-EvalCore (as we will see next), and

2. Noisy, because we do not have enough information about the $\mathcal{T}$ distribution to make an accurate estimate of how far the X-EvalCore is from the true performance.

Algorithm 6 shows the main steps of the X-Eval algorithm. To define our objective function, we needed to modify the X-EvalCore to return a measure of how near the X-EvalCore estimation is to the true performance $\ell(\mathcal{M}, \mathcal{T})$ with a specific Y^k_{noise}. In this thesis, we used as objective function the average accuracy (acc) of $\mathcal{M}^{y_{no}}_{autoEval}$ $\forall y_{no} \in Y_{noise}$ on its training set $(¡\mathcal{X}^{y_{no}}_{sci}, y_{sci}¿)$ plus the average of a measure (con) of how consistent the Expected AutoEval Labels are to each $x^i_{target} \in \mathcal{X}_{target}$ by $\mathcal{M}(\mathcal{X}_{target})$ considering $\mathcal{M}_{autoEval}(\mathcal{X}_{tci})$.

Algorithm 6: X-Eval

```
   Input: $\mathcal{M}, \mathcal{D}, \mathcal{T}, l$
   Result: $estimatedPerformance$

   // The BayesianSearch chooses the $Y^0_{noise}$ randomly
 2 $Y^0_{noise} \leftarrow$ BayesianSearch($\emptyset, \emptyset$)
 4 $k \leftarrow 0$
 6 while $k < l$ do
 8     $estimate^k, v^k \leftarrow$ X-EvalCore($\mathcal{M}, \mathcal{D}, \mathcal{T}, Y^k_{noise}$)
10     $k \leftarrow k + 1$
12     $Y^k_{noise} \leftarrow$ BayesianSearch($\{Y^0_{noise}, .., Y^{k-1}_{noise}\}, \{v^0, .., v^{k-1}\}$)
13 end
15 $estimatedPerformance \leftarrow$ average of $\{estimate^0, .., estimate^{l-1}\}$
```

Definition 1 (Expected AutoEval Label)**.** The Expected AutoEval Label (eal) to a specific x^i_{target} (given a $y_{no} \in Y^k_{noise}$) is: **the own model $\mathcal{M}(x^i_{target})$ prediction** if its prediction was classified as correct by the AutoEval Model (i.e., $\mathcal{M}^{y_{no}}_{autoEval}(x^i_{tci}) = 1$), or **the opposite label predicted by the model $\mathcal{M}(x^i_{target})$** if its prediction was classified as incorrect by the AutoEval Model (i.e., $\mathcal{M}^{y_{no}}_{autoEval}(x^i_{tci}) = 0$), as Equation shows 4.4.

$$eal(x^i_{target}, y_{no}) = \begin{cases} \mathcal{M}(x^i_{target}), & \mathcal{M}^{y_{no}}_{autoEval}(x^i_{tci}) = 1 \\ 1 - \mathcal{M}(x^i_{target}), & \text{otherwise} \end{cases} \tag{4.4}$$

Specifically, our objective function (obf) is:

$$obf(X_{target}, Y^k_{noise}) = \frac{1}{|Y^k_{noise}|} \sum_{y_{no} \in Y^k_{noise}} acc(\mathcal{M}^{y_{no}}_{autoEval}(\mathcal{X}^{y_{no}}_{sci}), y_{sci}) + \frac{1}{|\mathcal{X}_{target}|} \sum_{x_{te} \in \mathcal{X}_{target}} con(x_{te}, Y^k_{noise}) \tag{4.5}$$

, where the average $\mathcal{M}^{y_{no}}_{autoEval}$ accuracy in the training set (acc) is

$$acc(\mathcal{M}^{yo}_{autoEval}(\mathcal{X}^{yo}_{sci}), y_{sci}) = \frac{1}{|\mathcal{X}^{yo}_{sci}|} \sum_{i}^{|\mathcal{X}^{yo}_{sci}|} \begin{cases} 1, & \mathcal{M}^{y_{no}}_{autoEval}(x^i_{sci}) = y^i_{sci} \\ 0, & \text{otherwise} \end{cases} \tag{4.6}$$

, and the consistency function con is

$$con(x^i_{target}, Y^k_{noise}) = \begin{cases} \overline{eal}(x^i_{target}, Y^k_{noise}), & \overline{eal}(x^i_{target}, Y^k_{noise}) > 0.5 \\ 1 - \overline{eal}(x^i_{target}, Y^k_{noise}), & \text{otherwise} \end{cases} \tag{4.7}$$

, where $\overline{eal}$ is the average expected AutoEval label, computed with

$$\overline{eal}(x^i_{target}, Y^k_{noise}) = \frac{1}{|Y^k_{noise}|} \sum_{y_{no} \in Y^k_{noise}} \begin{cases} \mathcal{M}(x^i_{target}), & \mathcal{M}^{y_{no}}_{autoEval}(x^i_{tci}) = 1 \\ 1 - \mathcal{M}(x^i_{target}), & \text{otherwise} \end{cases} \tag{4.8}$$

In the objective function obf, we used $\mathcal{M}_{autoEval}$ accuracy on the training dataset (acc in the Equation 4.6) because if $\mathcal{M}_{autoEval}$ cannot find patterns on the training set (i.e., $\mathcal{M}_{autoEval}$ has a low accuracy), $\mathcal{M}_{autoEval}$ hardly will find out patterns on the target dataset and accurate classify the model $\mathcal{M}$ predictions as correct or incorrect.

The consistency function (con in the Equation 4.7) is a measure of how frequently the Expected AutoEval Label does not change for different $y_{no} \in Y^k_{noise}$. Thus, the consistency of the $\mathcal{M}_{autoEval}$ predictions can indicate the confidence of the X-EvalCore in the $\mathcal{M}_{autoEval}$ classifications. Figure 4.3 shows the distribution of the con function over different values of average Expected AutoEval Label.

Besides the right choice of the parameter Y_{noise}, an essential component of the X-Eval effectiveness is the data provided to the X-Eval to classify each target instance as correct or incorrect since it is the data that should have patterns that are kept in different distributions. Thus, in this thesis, we focused on performing AutoEval with the X-Eval with the following measure typically used for the model explanation: (1) confidence in the model prediction [33], (2) agreement among models [14], and (3) we proposed the use of feature importance (i.e., SHAP values) [90, 70]. Next, we present how we used these explanations to provide information for AutoEval with the X-Eval.

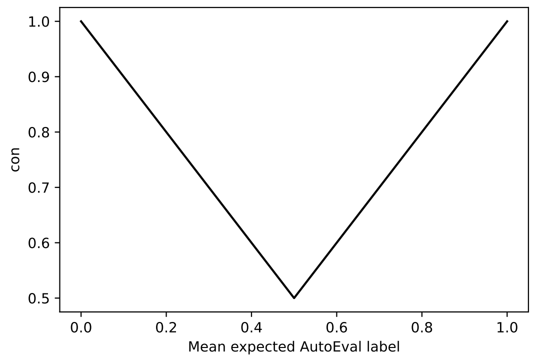


Figure 4.3: Distribution of the *con* function

4.1.3 Provided Confidence in the Model Prediction for the X-Eval

Given the predictions of the model $\mathcal{M}$ to a dataset $< \mathcal{X}, ? >$, $\mathcal{M}$ normally provides a confidence $(c_1, .., c_{|y|})$ of prediction for each possible label. The confidence in the model predictions is used in many approaches to model explainability.

In this thesis, we denote $\mathcal{C}onfidence(\mathcal{M}, \mathcal{X})$ as a function which returns $c_1, ..c_{|y|}$, where $c_i \in \mathbb{R}$, $0 \leq c_i \leq 1$, and c_i is the confidence of the predicted label is correct to the target instance x_i. We used the absolute difference of the confidence of model prediction for each label (i.e., $|confidenceLabel0 - confidenceLabel1|$) instead of the confidence for both labels. This way, we can reduce the data dimensionality, providing the same amount of information to the AutoEval Model $\mathcal{M}^{y_{no}}_{autoEval}$. In this thesis, we consider the scenario in which the summation of the model prediction confidence for all data classes is equals one (i.e., $\sum_{cl}^{classes} c_i^{cl} = 1$, where c_i^{cl} denotes the model prediction confidence to the class cl).

Thus, we can use $\mathcal{C}onfidence(\mathcal{M}, \mathcal{X})$ to provide data to the X-Eval to estimate the model performance. Next, we present how we used the agreement among models to provide information for the X-Eval.

4.1.4 Provided Agreement Among Models for the X-Eval

Different works use agreement among classifiers to AutoEval [59, 77, 14]. However, given the characteristics of the X-Eval algorithm, we needed to propose a novel approach to compute agreement between two classifiers. Thus, to compute agreement between models, we use a model second model $\mathcal{M}'$, different form the evaluated model $\mathcal{M}$ (i.e., $\mathcal{M}' \neq \mathcal{M}$), to compare the prediction of $\mathcal{M}'(\mathcal{X})$ and $\mathcal{M}(\mathcal{X})$. To train the AutoEval Model $\mathcal{M}_{autoEval}$ with explanations related to the model $\mathcal{M}$ in the source dataset, we perform a 10-fold cross-validation (Algorithm 7) using the source dataset and we get the predicted label to each $x_i \in X_{train}$ with $\mathcal{M}$ and $\mathcal{M}'$. That is:

- For the training set, we compute

  $$pred = CrossValPredict(\mathcal{M}, < \mathcal{X}_{train}, y_{train} >)$$

  and

  $$pred' = CrossValPredict(\mathcal{M}', < \mathcal{X}_{train}, y_{train} >)$$

  . Algorithm7 shows the main steps of the *CrossValPredict* function.

- For the test set, we compute $pred = \mathcal{M}(\mathcal{X}_{test})$ and $pred' = \mathcal{M}'(\mathcal{X}_{test})$.

  To compute the agreement between $\mathcal{M}$ and $\mathcal{M}'$, we perform $agreement = \begin{cases} 1, & \text{if } pred_i = pred'_i \\ 0, & \text{otherwise} \end{cases} \forall i \in \{1, .., |y_n|\}$.

  We referred to as $\mathcal{A}greement$ the presented algorithm to compute agreement between models. Algorithm 8 summarizes the $\mathcal{A}greement$ process. Next, we present how we used the SHAP values (i.e., feature importance) to provide information for the X-Eval.

Algorithm 7: CrossValPredict

```
    Input: $\mathcal{M}, < \mathcal{X}, y >$
    Result: $predictions$

1   $predictions \leftarrow \emptyset$
2   $Folds \leftarrow$ split $< \mathcal{X}, y >$ in 10-folds //without shuffling the data
3   forall ¡trainingSet,testSet¿ in Folds do
4       $\mathcal{M}_{copy} \leftarrow$ to copy $\mathcal{M}$
5       $\mathcal{M}_{copy} \leftarrow$ model built on the $trainingSet$
6       $predictions \cup \mathcal{M}_{copy}(testSet)$
    end
```

In this thesis, we used the Naive Bayes classifier in the agreement approach to create the model $\mathcal{M}'$. We chose the Naive Bayes because it is an explainable model. Besides, the Naive Bayes can bring diversity to the predictions since Naive Bayes applies a different classification strategy from the evaluated models in this thesis (i.e., decision tree ensembles, and the TabNet).

Algorithm 8: *Agreement*

```
Input: $\mathcal{M}, < \mathcal{X}, y >$
Result: $agreement$

1 $\mathcal{M}' \leftarrow model \neq \mathcal{M}$
2 if ¡$\mathcal{X}, y >$ is the training set then
3     $pred \leftarrow CrossValPredict(\mathcal{M}, < \mathcal{X}, y >)$
4     $pred' \leftarrow CrossValPredict(\mathcal{M}', < \mathcal{X}, y >)$
  else
5     To train $\mathcal{M}'$ on the source data $\mathcal{D}$
6     $pred \leftarrow \mathcal{M}(\mathcal{X})$
7     $pred' \leftarrow \mathcal{M}'(\mathcal{X})$
  end
8 $agreement = \begin{cases} 1, & \text{if } pred_i = pred'_i \\ 0, & \text{otherwise} \end{cases} \forall i \in \{1, .., |y_n|\}$
```

4.1.5 Provided SHAP values for the X-Eval

Section 2.5.3 provided the background regarding the SHAP values. As showed, the exact computation of SHAP values is challenging [58]. However, there are approximated options to calculate SHAP values, such as LinearSHAP, KernelSHAP, DeepSHAP, and TreeSHAP [58, 57].

In this thesis, we used TreeSHAP to compute the SHAP values of the decision tree ensemble classifiers (i.e., XGBoost, LightGBM, and CatBoost) and the KernelSHAP to compute the SHAP values to the TabNet classifier. We denote the function that generates SHAP values as $\mathcal{S}hap$. Next, we present how we combined these three approaches to explanation to provide information for the X-Eval.

4.1.6 Combining Different Types of Explanation for the X-Eval

An advantage of the X-Eval algorithm is that it allows the combination of different types of model explanations. In this work, we combined explanations: SHAP values, the confidence in the model predictions, and the agreement among models just concatenating the columns of the result of each explanation type. For instance, Algorithm 9 shows how we combined the information coming from SHAP values ($\mathcal{S}hap$), confidence in the prediction ($\mathcal{C}onficence$), and agreement among models ($\mathcal{A}greement$).

Algorithm 9: *Explaner*

```
Input: $\mathcal{M}, < \mathcal{X}, y >$
Result: $explanations$

$explanations \leftarrow$
  $\mathcal{S}hap(\mathcal{M}, \mathcal{X}) \cup$
  $\mathcal{C}onficence(\mathcal{M}, \mathcal{X})^\intercal \cup$
  $\mathcal{A}greement(\mathcal{M}, < \mathcal{X}, y >)^\intercal$
```

This kind of strategy is interesting because each type of explanation can be important to define whether a specific prediction is correct or not. Our experiments evaluated the X-Eval with the combination of feature importance (SHAP values), confidence in the model prediction, and agreement among models. Next, we present the time complexity analysis of the X-Eval.

4.1.7 Time Complexity Analysis of the X-Eval

In this section, we present the time complexity analysis of the X-Eval algorithm. We denote the source dataset number of examples as n, the target dataset number of instances as m, and the number of features in the training dataset as $|x|$. Considering the Algorithm 6, the X-Eval executes the BayesianSearch and the X-EvalCore method in l iterations. The BayesianSearch complexity is $O(l^3)$ and letting $O(XEvalCore)$ be the complexity of the X-EvalCore algorithm, the X-Eval complexity is $O(l^3) + O(l\ XEvalCore)$.

The X-EvalCore complexity is dependent on many experimental design decisions. Hence, next, we present the X-EvalCore complexity considering this thesis main experimental design decisions. Given the Algorithm 5, in the X-EvalCore the loop in line 2 is executed a number of interactions equals the size of the Y_{noise} (denoted as $|Y_{noise}|$). Inside this loop, we execute the *Explainer* in line 3. The *Explainer* is sum of the $\mathcal{S}hap$, the $\mathcal{C}onfidence$, and the $\mathcal{A}greement$ complexities. Given that:

- The $\mathcal{S}hap$ is $O(n\ t\ e\ d^2)$, where t be the number of trees, d the maximum depth of any tree, and e the number of leaves, considering that we used the TreeSHAP [57].
- The $\mathcal{A}greement$ is $(O(n\ |x|) + O(n\ |x|\ c)) + (O(t\ d\ |x|\ \log n) + O(n\ t\ d))$. Considering that a Gaussian Naive Bayes classifier takes (1) $O(n\ |x|\ c)$ to train, and (2) $O(n\ |x|\ c)$ to make the predictions of the dataset, where c the number of classes. Additionally, (1) the training of an XGBoost model is $O(t\ d\ |x|\ \log n)$, where t is the number

of trees, d is the height of the trees, and (2) predicting the labels of the dataset is $O(n\ t\ d)$.

- The $\mathcal{C}onfidence$ is $O(1)$, once we get the prediction confidence during the executions of the $\mathcal{A}greement$ function.

Thus, we have the following time complexities:
$O(n\ t\ e\ d^2) + O(n\ |x|) + O(n\ |x|\ c) + O(t\ d\ |x|\ \log n) + O(n\ t\ d)) + O(1)$

Once we used the default XGBoost hypeparameters[1] (i.e., $d = 6$ and $t = 100$), d and t can be considered constants. Then, we obtain:
$O(n\ e) + O(n\ |x|) + O(n\ |x|\ c) + O(|x|\ \log n) + O(n) + O(1) =$
$(n\ e) + (n\ |x|) + (n\ |x|\ c) + (\ |x|\ \log n) + n =$
$(n\ e) + (n\ |x|) + (n\ |x|\ c) + (\ |x|\ \log n) =$ (because n is dominated by $(n\ |x|\ c)$)
$(n\ e) + (n\ |x|\ c) + (\ |x|\ \log n) =$ (because $(n\ |x|)$ is dominated by $(n\ |x|\ c)$)
$O((n\ e) + (n\ |x|\ c))$ (because $(|x|\ \log n)$ is dominated by $(n\ |x|\ c)$)

In the Algorithm 5, the lines 4 and 7 are $O(m)$. In line 5, we train the XGBoost that is $O(|x| \log n)$ [15]. In line 6, we execute the *Explainer* in the target dataset, in which the complexity is $(m\ e) + (m\ |x|\ c)$. Additionally, in line 6, we make predictions with the XGBoost, in which the complexity is $O(m\ t\ d) = O(m)$, where d is the maximum depth of the tree and t is the total number of trees. Again, as we used the default XGBoost hyperparameters (i.e., $d = 6$ and $t = 100$), d and t can be considered constants [15].

However, in the real implementation, the *Explainer* is executed just once. So, the X-EvalCore is

$O((n\ e) + (n\ |x|\ c)) + O((m\ e) + (m\ |x|\ c)) + (l\ |Y_{noise}|(O(m) + O(|x|\ \log n) + O(m) + O(1))) =$

$(n\ e) + (n\ |x|\ c) + (m\ e) + (m\ |x|\ c) + (l\ |Y_{noise}|((|x|\ \log n) + 2m)) =$

$(n\ e) + (n\ |x|\ c) + (m\ e) + (m\ |x|\ c) + (l\ |Y_{noise}|((|x|\ \log n) + m)) =$ (removing the constant 2)

$n + (n\ |x|\ c) + m + (m\ |x|\ c) + (l\ |Y_{noise}|((|x|\ \log n) + m)) =$ (removing e because e can be considered a constant since it is limited by 2^d, ie, $e \leq 2^d$, and $d = 6$)

$(n\ |x|\ c) + (m\ |x|\ c) + (l\ |Y_{noise}|((|x|\ \log n) + m)) =$ (removing n and m, because they are dominated by $(n\ |x|\ c)$ and $(m\ |x|\ c)$)

$O((max(n, m)\ |x|\ c) + (l\ |Y_{noise}|((|x|\ \log n) + m)))$

In the X-Eval (Algorithm 6), the BayesianSearch complexity is $O(l^3)$, where l is maximum number of generated Y_{noise} [63]. So, the X-Eval is

$O(l^3 + ((max(n, m)\ |x|\ c) + (l\ |Y_{noise}|((|x|\ \log n) + m)))$

[1]https://xgboost.readthedocs.io/en/latest/parameter.html

Thus, the time complexity of our X-Eval implementation is dominated by (1) the Bayesian Search time complexity (i.e., l^3), (2) the number of evaluated y_{noise} (i.e., $l\ |Y_{noise}|$), (3) the time complexity of the *Explainer*, specifically to build and make predictions with the Naive Bayes classifier in the $\mathcal{A}greement$ function (i.e., $((n\ |x|\ c) + (m\ |x|\ c))$), and (4) the time complexity to build and make the predictions with the XGBoost in the AutoEval Model $\mathcal{M}_{autoEval}$ (i.e., $(l\ |Y_{noise}|((|x|\ \log n) + m))$).

We highlight that the presented time complexity analysis is dependent on this thesis main experimental design. Consequently, the time complexity can change according to the evaluated model, the chosen classifier to build the AutoEval Model ($\mathcal{M}_{autoEval}$), the classifier used in the $\mathcal{A}greement$ function, and the optimization algorithm used to search for Y_{noise} values. Moreover, we highlight that we used a small l and $|Y_{noise}|$ (i.e., $|Y_{noise}| \leq l \leq 300$, in the X-Eval hyperparameters analysis, and $|Y_{noise}| = l = 30$ in the other experiments). Next, we present the proposed algorithm based on causality to regularize the estimated model performance in distribution shift scenarios.

4.2 C-Eval: Causality for Automatic Model Evaluation

We proposed the approach Causality for Automatic Model Evaluation (C-Eval) to regularize the estimated performance of the classification model $\mathcal{M}$ on a specific target dataset, which can be out-of-distribution.

The hypothesis and the central idea of our approach is that the changes in the causal relations of the features in different datasets can indicate changes in the distribution, which will affect the estimated performance of the model. From this hypothesis, we use the similarity among the causal graphs inferred from different datasets to regulate the estimated performance reported by a performance estimator (e.g., the standard k-fold cross-validation). Figure 4.4 illustrates the main steps of the proposed algorithm C-Eval.

Specifically in our approach, first, we estimate the model performance in the target dataset with a performance estimator, denoted as $estimator(\mathcal{D}, \mathcal{T})$, (such as the X-Eval and the k-fold cross-validation). In our experiments, we used the 10-fold cross-validation (CV) because it is a commonly used and well-studied method that presents low bias and variance [46, 61]. In addition, we evaluated the C-Eval regularizing the proposed X-Eval estimate because the X-Eval showed a lower estimation error than the CV in our experiments.

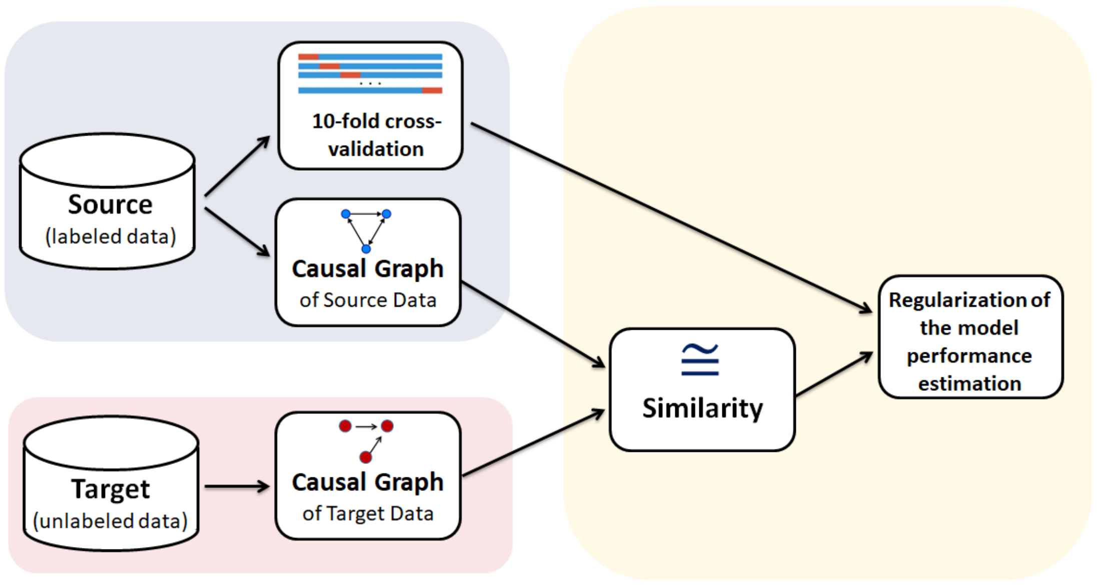


Figure 4.4: Illustration of the algorithm Causality for Automatic Model Evaluation (C-Eval).

Next, we use a causality inference algorithm $\mathcal{C}$ to infer a causal graph from the source (training) dataset (denoted as $\mathcal{C}(\mathcal{D})$) and another causal graph from the target (test) dataset (denoted as $\mathcal{C}(\mathcal{T})$). A causal graph of a dataset is a directed acyclic graph (DAG) where the features and the label are represented as vertices and the edges denote causal relationship among these vertices (i.e., when one feature directly influences another feature) [103, 35]. Figure 2.4 shows a causal graph example. Once the labels of the target dataset ($\mathcal{T}$=<$\mathcal{X}_{target}, ?$>) are unknown (i.e., $y_{target} =?$), we fill the labels of the target dataset with the predictions of the evaluated model $\mathcal{M}$ (i.e., $y_{target} = \mathcal{M}(\mathcal{T})$), in the source dataset ($\mathcal{D}$=<$\mathcal{X}_{train}, y_{train}$>) we use the data label in the causal graph inference (i.e., we use the y_{train}).

Then, we compute the similarity from the inferred causal graphs (denoted as $s(\mathcal{C}(\mathcal{D}), \mathcal{C}(\mathcal{T}))$), where $0 \leq s(\mathcal{C}(\mathcal{D}), \mathcal{C}(\mathcal{T})) \leq 1$. Finally, we use this similarity to regularize the estimated performance $estimator(\mathcal{D}, \mathcal{T})$ reported by the Performance Estimator (e.g., the X-Eval or the k-fold cross-validation). Specifically, we estimate the regularized model performance with the equation:

$$\frac{\lambda\ estimator(\mathcal{D}, \mathcal{T}) + s(\mathcal{C}(\mathcal{D}), \mathcal{C}(\mathcal{T}))}{(1+\lambda)} \tag{4.9}$$

We use a factor $\lambda \geq 0$ to multiply the estimated performance $estimator(\mathcal{D}, \mathcal{T})$ to control the overfitting and underfitting, in a similar way that is made in ℓ_1 and ℓ_2 norm regularization approaches [47]. Another way to interpret the λ is the λ is the degree of confidence in the performance estimator ($estimator$), e.g., the more distribution shift is expected in the target dataset, the smaller should be the λ. We used $1 + \lambda$ to normalize the $\lambda\ estimator(\mathcal{D}, \mathcal{T})) + s(\mathcal{C}(\mathcal{D}), \mathcal{C}(\mathcal{T}))$ result between 0 and 1. Next, we present the time complexity analysis of the C-Eval algorithm.

4.2.1 Time Complexity Analysis of the C-Eval

In this section, we present the C-Eval time complexity analysis. We denote the source dataset number of examples as n, the target dataset number of instances as m, and the number of features in the training dataset as $|x|$. From Equation 4.9, we can see that the C-Eval complexity time is dominated by the time complexities of the used performance estimator, causality inference algorithm, and similarity function.

Considering the used causality inference algorithms in this thesis:

- The PC is $O(|x|^q)$, where q is the maximal degree of any vertex in the causal graph [93, 43].
- The DirectLinGAM is $O(n_d\ |x|^3\ M^2 +\ |x|^4\ M^3)$, where n_d is the number of examples, and M $(\ll n_d)$ is the maximal rank found by the low-rank decomposition used in the kernel-based independence measure [87].
- The Notears is $O(|x|^3)$ [114]

Additionally, considering that we evaluated the XGBoost performance (with the hyperparameters $t = 100$ and $d = 6$) and the used performance estimator in this thesis:

- The 10-fold cross-validation (10-CV) is $O((|x|\ \log n) +\ n)$. Once (1) the training of an XGBoost model is $O(\ |x|\ \log n)$, and (2) predicting the labels of the dataset is $O(n)$.
- The X-Eval complexity is
  $O(l^3 + ((max(n, m)\ |x|\ c) + (l\ |Y_{noise}|(((|x|\ \log n) + m)))$. Considering our experimental design decisions, as shown in Section 4.1.7.

Once we represent the causal graphs as adjacent matrices (which complexity is $O(|x|^2)$, the similarity (s) between the causal graphs can be computed with the following complexities, considering the used similarity functions in this thesis:

- The Cosine similarity is $O(|x|^2)$ [89];
- The Jaccard similarity is $O(|x|^2)$ [23].
- The Macro F1 similarity is $O(|x|^2)$.

Finally, we can define the C-Eval complexity according to the chosen causality inference algorithm, performance estimator, and similarity function. For instance:

- The C-Eval with the X-Eval performance estimator, DirectLinGAM, and Macro F1 similarity is $O(l^3 + ((max(n,m)\ |x|\ c) + (l\ |Y_{noise}|(((|x|\ \log n) + m)) + (n\ |x|^3\ M^2 + \ |x|^4\ M^3) + (m\ |x|^3\ M^2 + \ |x|^4\ M^3))$
- The C-Eval with the 10-CV performance estimator, DirectLinGAM, and Macro F1 similarity is $O((|x|\ \log n) + (n\ |x|^3\ M^2 + \ |x|^4\ M^3) + (m\ |x|^3\ M^2 + \ |x|^4\ M^3))$

We highlight that we used a small l and $|Y_{noise}|$ (i.e., $|Y_{noise}| \leq l \leq 300$, in the X-Eval hyperparameters analysis, and $|Y_{noise}| = l = 30$ in the other experiments). Next, we present the experimental evaluation of the proposed algorithms X-Eval and C-Eval.

Chapter 5

Experimental Design to Evaluate the Proposed Algorithms

In this chapter, we present the experimental design, i.e., the main decision to evaluate the X-Eval and C-Eval. Chapter 6 and Chapter 7 present these experimental evaluations. In our experiments, we used real-world and synthetic data.

In the experimental evaluation (Chapter 6 and Chapter 7), we analyzed the approximations of the X-Eval and the C-Eval to the true performance of the classifier XGBoost in a specific target out-of-distribution dataset. We used the XGBoost because it is representative of the state-of-the-art to deal with tabular data [15]. Additionally, in Section 6.2.2, we evaluated the estimation error of the models LightGBM, CatBoost, and TabNet since they are in the state-of-art of automatic classification for tabular data [8].

In the X-Eval evaluation, considering the substantial computational cost of executing the X-EvalCore, we opted for a Y_{noise} size of 30 and set the number of interactions in Bayesian Search (l) to 30. We used these settings because as demonstrated in the experimental analysis in Appendix D, increasing the values of these hyperparameters did not significantly decrease the estimation error.

Besides, in the C-Eval, we evaluated the algorithms PC (named after its authors, Peter and Clark) [93, 42], NOTEARS [114], and DirectLiNGAM [87] to infer the causal graphs. We chose these algorithms because they are state-of-the-art representative method for causal discovery [51]. In addition, we varied the hyperparameter λ factor from 0 to 10 with an interval of 0.01.

We evaluated the C-Eval with the similarity functions Cosine, Jaccard, and Macro F1 (i.e., we used these similarity functions to compute the causal graph similarities). We used these metrics because they are commonly used metrics when the feature values are between 0 and 1 [92]. Besides, in our proposed method C-Eval, we needed a similarity metric that the output is between 0 and 1. We evaluated the metric Macro F1 as similarity function because of its proprieties to compare imbalance data (Section 2.4 provides a longer explanation regarding the decision of the similarity functions). To calculate the similarity between the causal graphs, we represented a causal graph as an adjacency matrix and we transformed the adjacency matrix into an array as Figure 5.1 exemplifies.

$$\begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \longrightarrow [1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1]$$

Figure 5.1: Example of transformation of an adjacency matrix to an array.

We evaluated the proposed methods with real-world and synthetic data. The goal of the experiments with synthetic data was to characterize the performance of the proposed algorithm in different types and degrees of distribution shifts.

We used the standard 10-fold cross-validation (10-CV) as a baseline. According to Han et al. [34], the 10-CV is the recommended approach to evaluate a classification model, even when computational cost allows more folds, due to the 10-CV relatively low bias and variance. Besides, Marcot and Hanea [65] experiments support the common use of the 10-CV. However, some works suggest that defining the appropriated number of folds according to the data characteristics can improve the CV estimation [108, 73], but this process of defining the number of folds cannot remove the constraint of the target data must have the same distribution of the source dataset (i.e., must be *iid*). To demonstrate this empirically, Appendix C shows experiments indicating that only increasing the number of folds of the k-fold cross-validation is not enough to reduce the estimation error in distribution shift scenarios.

Our experiments aimed to investigate the following research questions (RQ):

RQ1: Can AutoEval approaches based on Causality Learning and Explainability be better than the standard 10-fold cross-validation to estimate the model performance in the situation which can occur distribution shift in the tabular data context?

RQ2: Can AutoEval approaches based on Causality Learning and Explainability be better than the standard 10-fold cross-validation to feature selection in the situation that can occur distribution shift in the tabular data context?

Chapter 6

Experimental Evaluation with Real-World Data

This chapter presents the experiments to evaluate the proposed methods in real-world situations in which we expect distribution shifts. In these experiments, we used real-world data related to the following three subjects: COVID-19, Alzheimer's disease, and School Dropout. For each covered subject, we selected two datasets that share the same features and classes, but there are some differences in the data generation process or population (i.e., different hospitals, different departments, and different periods). As showed in the Table 6.1, the datasets are imbalanced. Thus, we employed macro f1 [22] as the basic evaluation metric in our experiments. Besides, the Chapter 5 describes the main experimental decisions. Next, we detail the used datasets.

6.1 Datasets and Setups of Training and Test

This section presents the datasets used in our experiments and the defined training and test setups. Table 6.1 describes the main characteristics of our datasets[1]. We can observe that we used datasets of typical sizes found in the healthcare area [3] and large datasets, such as those related to COVID-19. We extracted the COVID-19 datasets from the Repository COVID-19 DataSharing/BR[2]. These datasets contain laboratory test results of patients with COVID-19, and the outcome indicates whether the patient has died or not. The data were provided by Hospital Beneficência Portuguesa de São Paulo (HBP)[3] and Hospital Sírio Líbanes (HSL)[4].

The datasets regarding the Alzheimer's disease (AD) subject comprise data from patients who may have AD (e.g., gender, age, education level, and laboratory test results).

[1]Datasets available on https://github.com/ismasantana/AutoEval.
[2]https://repositoriodatasharingfapesp.uspdigital.usp.br/
[3]https://repositoriodatasharingfapesp.uspdigital.usp.br/handle/item/101
[4]https://repositoriodatasharingfapesp.uspdigital.usp.br/handle/item/97

Table 6.1: Main characteristics of the used datasets

Subject	**Name**	**# intances**	**# features**	**positive class**
COVID-19	HBP	453	471	15%
	HSL	4,018		5%
Alzheimer	Geriatrics	106	18	72%
	Neurology	74		54%
School dropout	Students2008	143	15	13%
	Students2014	164		19%

The class indicates whether a patient has been diagnosed with AD or not. The datasets come from the Geriatrics and Neurology departments of the Universidade Federal de Minas Gerais[5], we denoted these datasets as: Geriatrics and Neurology. Each department receives patients with different characteristics.

In the School Dropout datasets, the data comprise diverse information about high school students. The outcome indicates whether a student will drop out school or not. The *Students2008* dataset contains data from students who started school in 2008, and the *Students2014* dataset contains data from students who started in 2014. We split the data in these periods because a student needs to complete the course within six years. Thus, to have a training dataset in the real world, we would wait for six years to determine if a student would drop out of school. Nevertheless, we believe that the characteristics of students can change after six years, causing a distribution shift.

We considered the following source (training) and target (test) datasets setups in the experiments. Given two datasets D_s=<X_s, y_s> and D_t=<X_t, y_t>, which share the same feature-set and possibly have differences in the distributions:

1. For COVID-19 and Alzheimer datasets, we performed the experiments in which (a) D_s was used for training and D_t for testing, and (b) D_t was used for training and D_s for testing.
2. For School Dropout datasets we used the *Students2008* dataset to train and the *Students2014* to test. Once we split the School Dropout Datasets in time, training the model with future data would not make sense.

In the experiments, we used all features of the Alzheimer and School dropout dataset, once domain experts selected and analyzed these features. However, due to the great number of features of the COVID-19 dataset, in each setup, we selected the most

[5]https://ufmg.br

informative features using the training set, and the Borutashap feature selection method [45]. Thus, BorutaShap selected 20 features for the HBP dataset, and 30 features for the HSL dataset. Next, we characterize the distribution shift in the datasets used in this thesis.

6.1.1 Distribution Shift Characterization in the Datasets

Nagarajan et al. [71] identified that spurious features are related to fundamental factors that cause models fail in the presence of the distribution shifts. In summary, spurious features are features correlated with a label in the training set but not in the test set.

To visualize the distribution shift and identify spurious features on the datasets of the same subject, we plotted approximations of the probability density function (PDF) to each feature in each label. To categorical features, we plotted histograms and to continue features we plotted the Kernel Density Estimation (KDE) [16]. However, to facilitate the visualization of the difference in the distributions of each dataset: (1) we subtracted the PDFs of the data with the label 1 and the data with the label 0 in each dataset, and (2) we plotted the result of the difference in each dataset in the same graph. That is, to each D_s and D_t about the same subject, we plotted in the same graph: (1) $PDF(D_s^{label=1}) - PDF(D_s^{label=0})$ and (2) $PDF(D_t^{label=1}) - PDF(D_t^{label=0})$, where $label = 0$ and $label = 1$ denote the examples in the label 0 and 1, respectively. Figure 6.1 shows an illustration of this process to visualize the spurious features and distribution shift. This way, the regions below 0 in the y-axis show a greater correlation with the label 0, and the regions above 0 show a greater correlation with the label 1. Consequently, the regions where the correlation is opposite in the datasets indicate spurious correlations.

We plotted some examples of features that could contribute to this analysis. In the COVID-19 datasets, due to the high number of features, we selected the most important features of each dataset individually, using the BorutaSHAP [45] feature selection method, and we considered in the analysis just the selected features.

Figure 6.2, Figure 6.3, and Figure 6.4 show the result of these plots. Considering that a model would be trained on a dataset and the same model would be tested in another, visually, we can identify some spurious features in the datasets, such as (1) *Peak-Low TF-aTFPI*, *Peak-High TF+APC*, and *APO E* in the datasets about Alzheimer, and (2) *Potassio_mEq/Lm*, *Fosforo_mg/dL*, *Volume plasquetario medio_%*, *pO2 venoso_mmHg*, and *HCO3 venoso_mmol/L* in the COVID-19 datasets. In this characterization, we cannot identify evident spurious features in the datasets about school dropout, we only could see

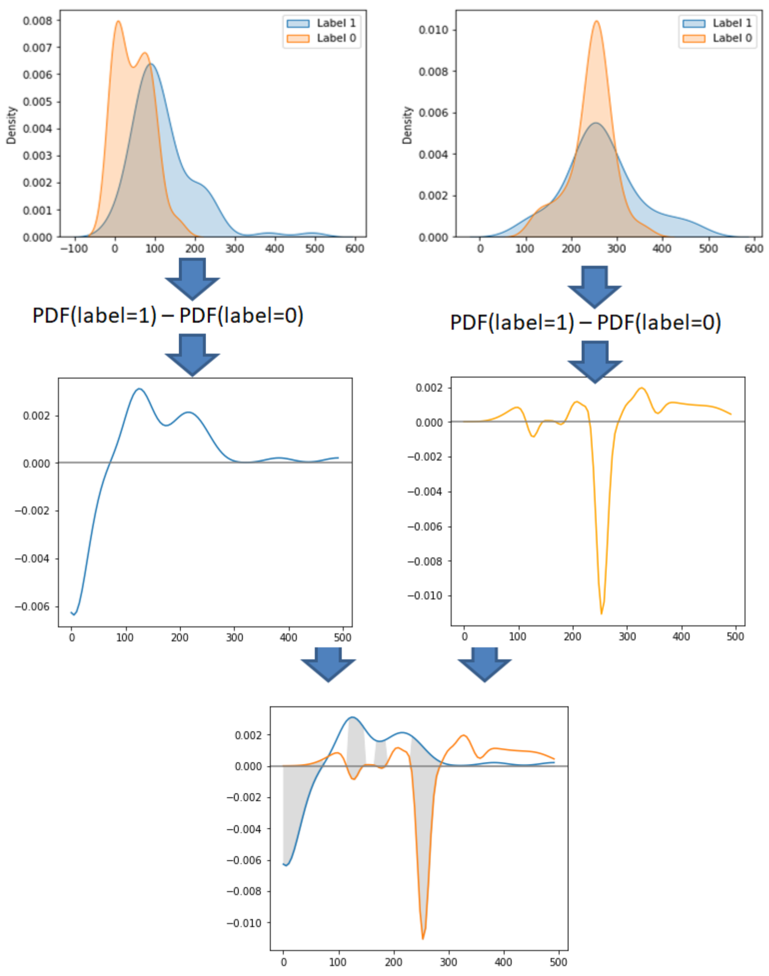


Figure 6.1: The process to visualize the spurious features and distribution shift.

small differences in the distributions (e.g., in the feature *1ANOFALTA*).

These characterizations indicate that the Alzheimer and COVID-19 datasets have significant natural distribution shifts (or source component shift [79]). On the other hand, the distributions of the datasets related to School Dropout are similar. These characteristics are expected because the data about School Dropout were collected from the same place, and the datasets about Alzheimer and COVID-19 were collected from different places. Next, we characterize the causality in the datasets used in the thesis.

6.1.2 Characterization of the Causality in the Datasets

This section characterizes the causal relations in the datasets used in this thesis. We employed the algorithms PC, DirectLinGAM, and Notears to infer a causal graph for each dataset. In this characterization, we used the true label of each dataset in the

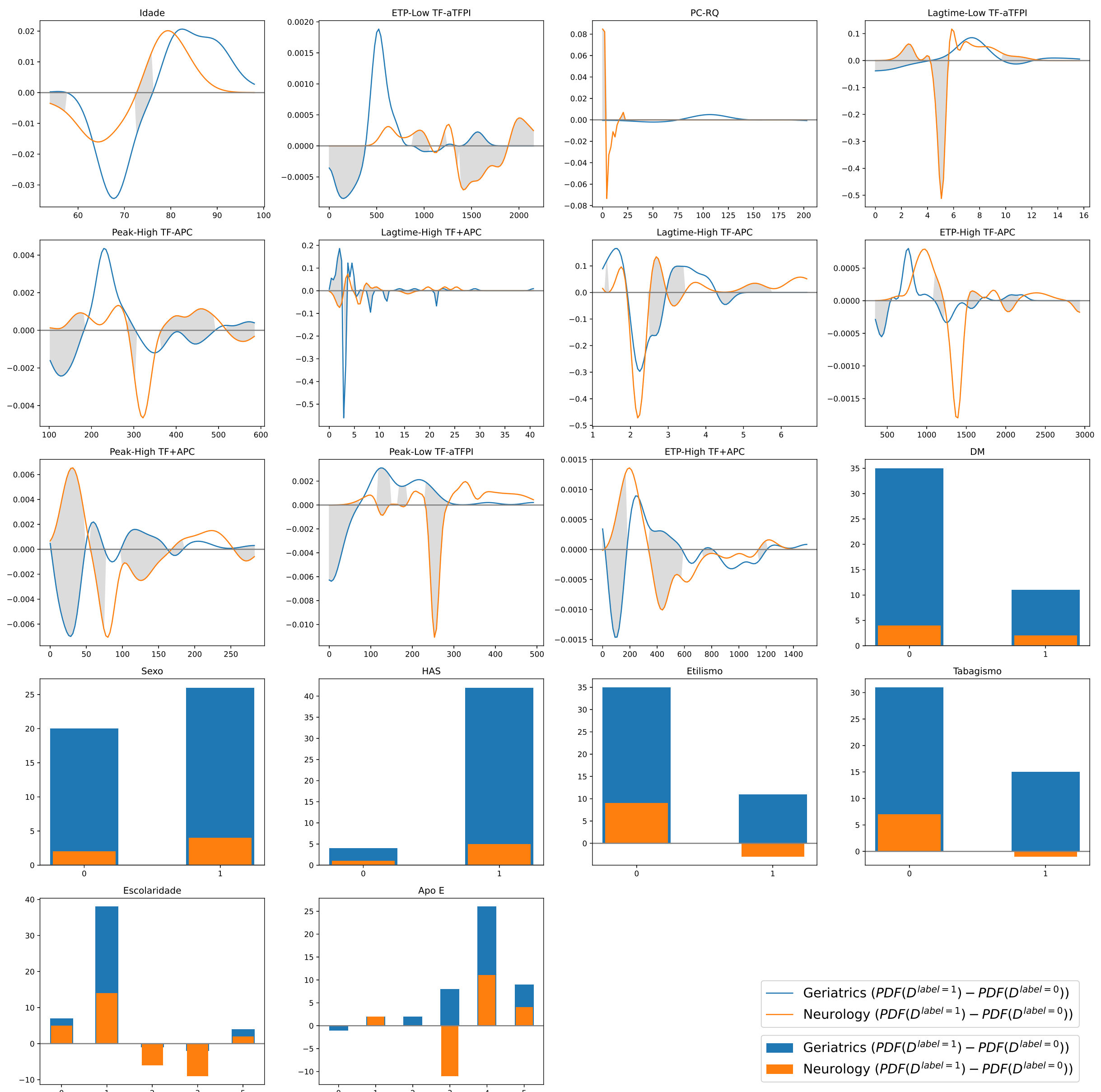


Figure 6.2: Differences between the Probability Density Function (PDF) of examples with label 1 and examples with label 0 in the datasets about Alzheimer. We filled out with the gray color the regions correlated with different labels in the evaluated datasets.

causality inference process. In addition, we highlighted the common edges between the graphs regarding the same subject.

Figure 6.5, Figure 6.6, and Figure 6.7 show the causal graph inferred by the algorithms PC, DirectLinGAM, and Notears for each dataset. We can observe that there are common edges between the graphs regarding the same subject in most cases. For instance, in Figure 6.5, we can see common edges among the graphs inferred from the datasets regarding the School Dropout when we used the algorithm Notears and DirectLiNGAM. However, the PC algorithm did not find causal relationships in the *Students2014* dataset and this implies in zero similarity between the causal graphs inferred to the datasets *Students2008* and *Students2014*. This lack of causal relations

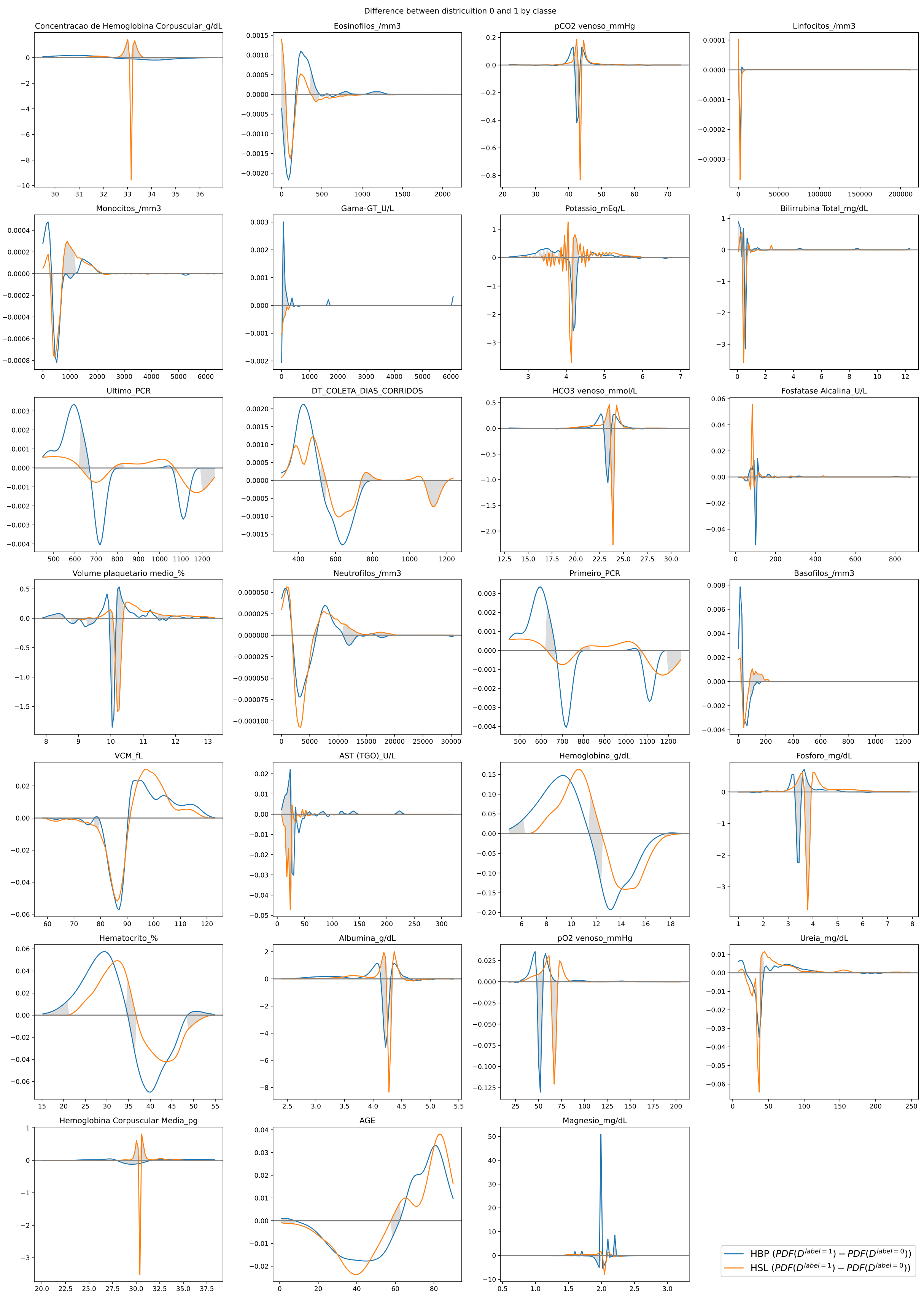


Figure 6.3: Differences between the Probability Density Function (PDF) of examples with label 1 and examples with label 0 in the datasets about COVID-19. We filled out with the gray color the regions correlated with different labels in the evaluated datasets.

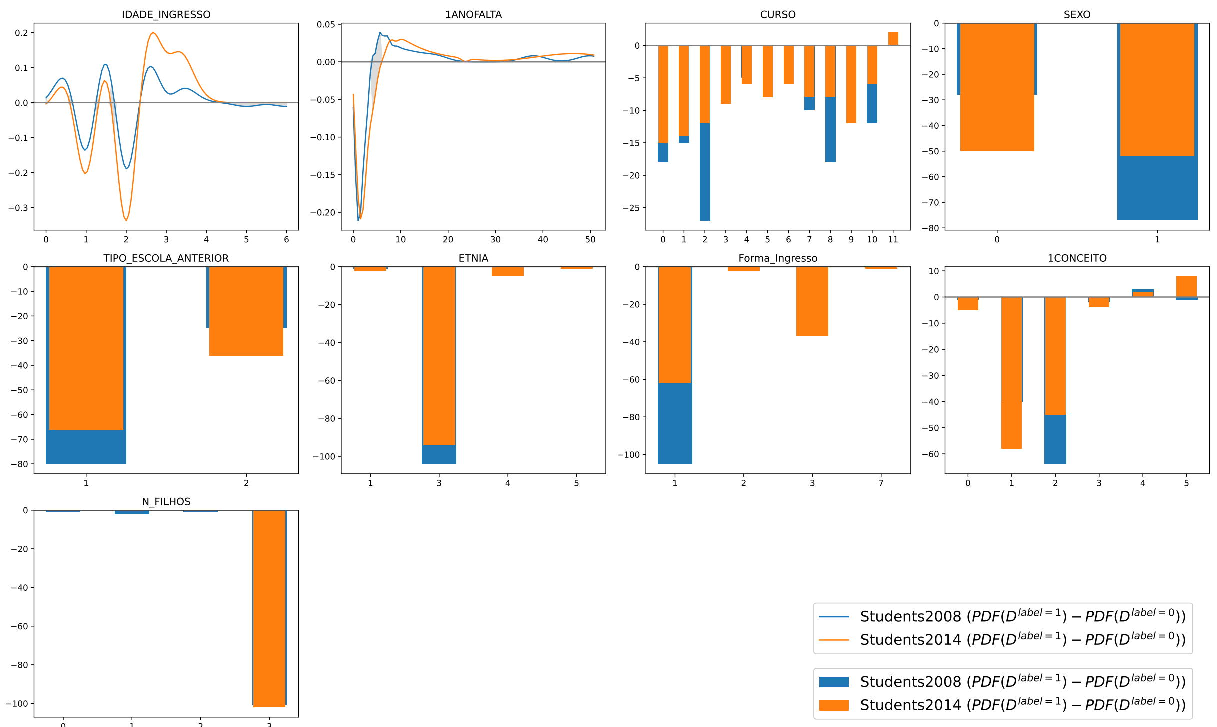
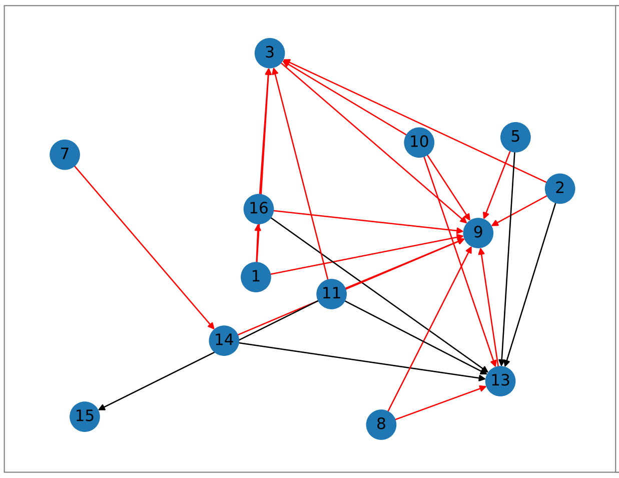


Figure 6.4: Differences between the Probability Density Function (PDF) of examples with label 1 and examples with label 0 in the datasets about School Dropout. We filled out with the gray color the regions correlated with different labels in the evaluated datasets.

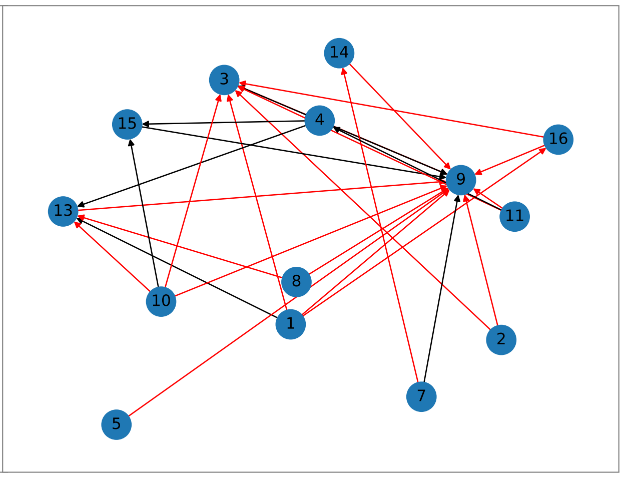
indicates that the PC algorithm may not be suitable for use in the proposed algorithm C-Eval in this dataset. In other words, whether the algorithm to infer causality relation does not find causal relations in the data, the C-Eval with this causal algorithm will not be effective for performance estimation. Thus, we must choose another algorithm to infer causality in this dataset.

Figure 6.6 shows the graphs inferred from the datasets regarding COVID-19. We can notice common edges between the graphs inferred for the same algorithm to the datasets of different population, independent of the used algorithm for causality discovery. However, in the graphs inferred by the Notears (Figure 6.6a and Figure 6.6b), there is concentration of features with causal relations with the feature *Platelets_/mm3* (i.e., the node 16). This concentration is interesting because some studies indicate the impacts of COVID-19 in the platelets, fundamental cells for the coagulation process [36, 27]. In addition, according to Fleury [27], platelets participate in the immune response, and changes in the platelets number are related to many diseases.

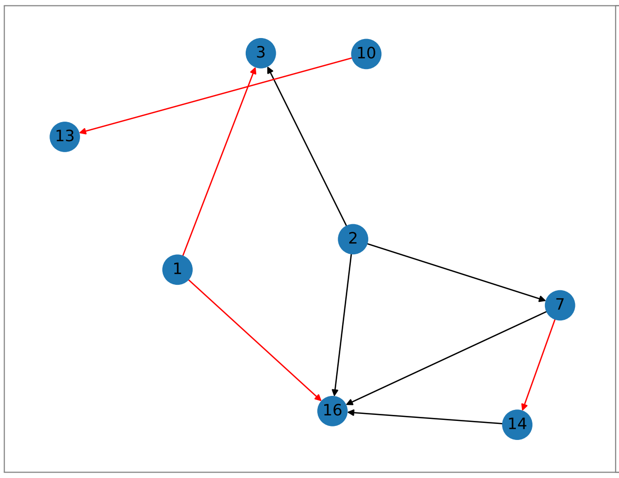
Figure 6.7 makes evident that the Notears algorithm tends to infer more causal relations. Besides, we can observe a few cases of edges to the class (node 1) in all graphs. However, this does not preclude using this kind of information in the C-Eval because changes in feature distributions also cause distribution shifts (e.g., covariate shifts). Next, we present the experimental evaluation of our proposed algorithms using real-world datasets.



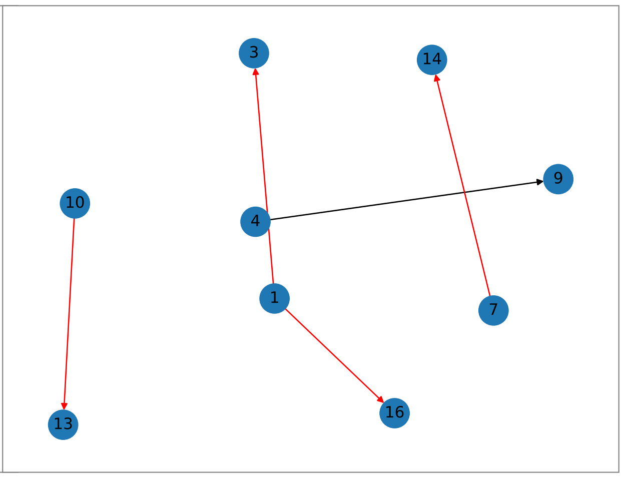
(a) *Students2008* - Notears



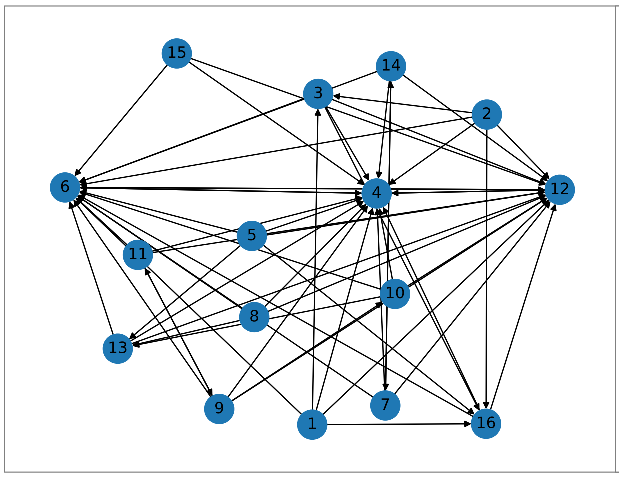
(b) *Students2014* - Notears



(c) *Students2008* - DirectLiNGAM



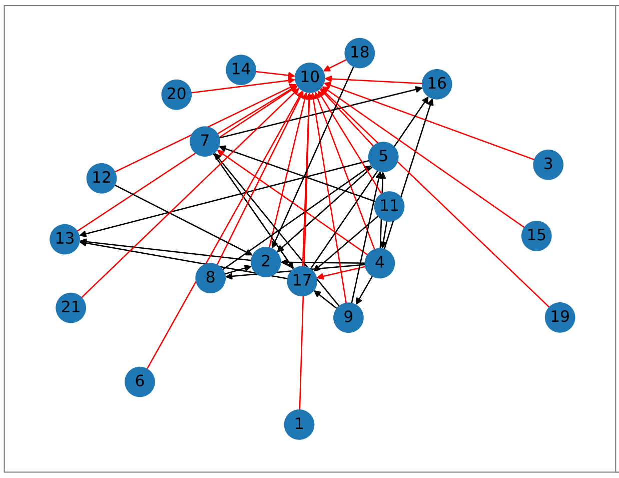
(d) *Students2014* - DirectLiNGAM

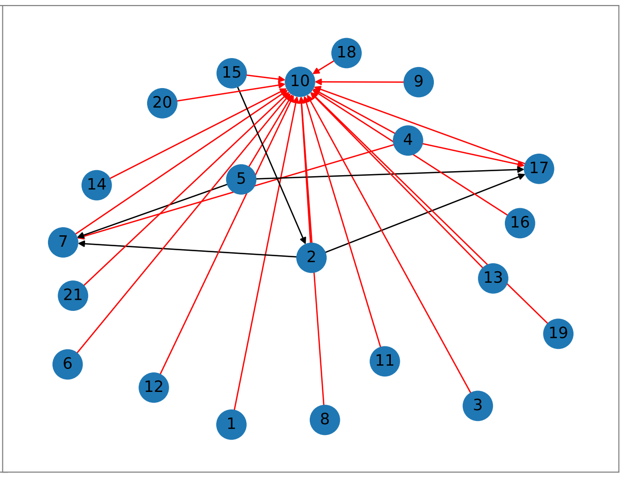


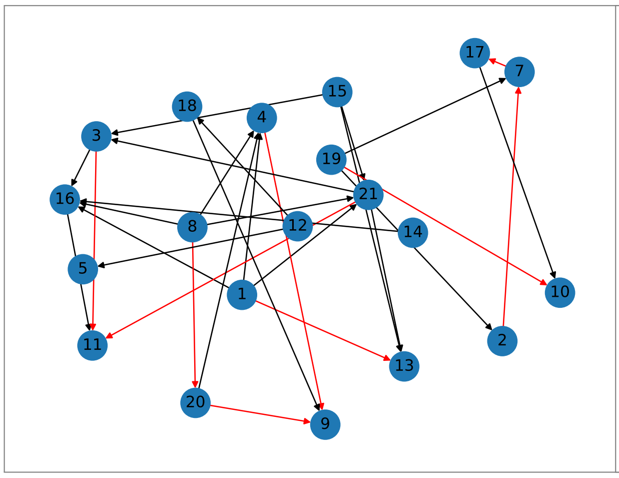
(e) *Students2008* - PC

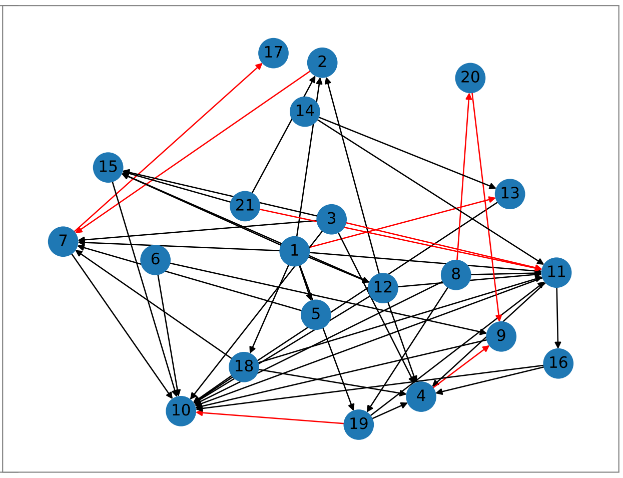
(f) *Students2014* - PC

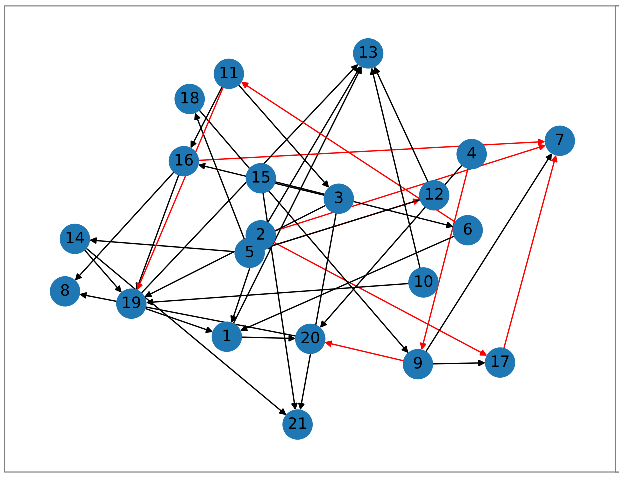
Figure 6.5: Causal graphs inferred from the datasets about **School Dropout** (i.e., *Students2008* and *Students2014*)) by each causal discovery algorithm used in this thesis. We showed the name of the causal discovery algorithm used to infer the graph in the legend of each subfigure. We highlighted the common edges among the two graphs inferred by the same algorithm with the red color. Appendix A shows the map among the node labels and the feature names.


(a) HBP - Notears


(b) HSL - Notears


(c) HBP - DirectLiNGAM


(d) HSL -DirectLiNGAM


(e) HBP - PC

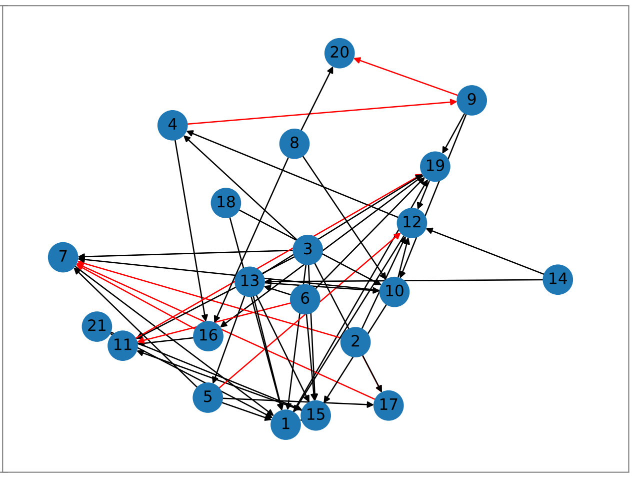
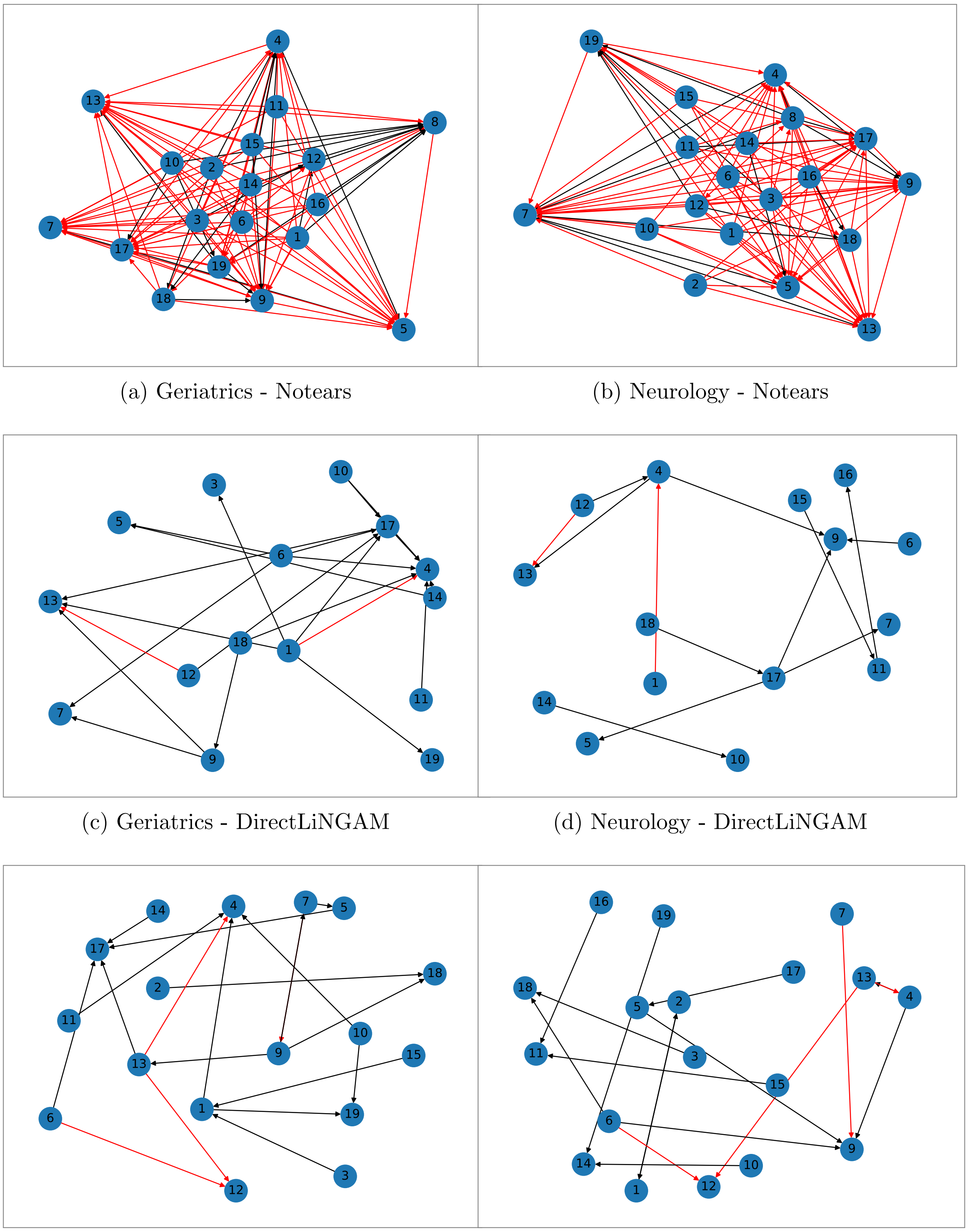

(f) HSL - PC

Figure 6.6: Causal graphs inferred from the datasets about **COVID-19** (i.e., HBP and HSL) by each causal discovery algorithm used in this thesis. We showed the name of the causal discovery algorithm used to infer the graph in the legend of each subfigure. We highlighted the common edges among the two graphs inferred by the same algorithm with the red color. Appendix A shows the map among the node labels and the feature names.



(a) Geriatrics - Notears

(b) Neurology - Notears

(c) Geriatrics - DirectLiNGAM

(d) Neurology - DirectLiNGAM

(e) Geriatrics - PC

(f) Neurology - PC

Figure 6.7: Causal graphs inferred from the datasets about **Alzheimer** (i.e., *Geriatrics* and *Neurology*) by each causal discovery algorithm used in this thesis. We showed the name of the causal discovery algorithm used to infer the graph in the legend of each subfigure. We highlighted the common edges among the two graphs inferred by the same algorithm with the red color. Appendix A shows the map among the node labels and the feature names.

6.2 Analysis of Generalization Gaps on Natural Distribution Shifts

In the first type of experiment, we evaluated the generalization gap (i.e., estimation error, which we computed as the absolute difference between the estimated performance and the true performance in a target dataset [40]). First, we estimated the performance of the XGBoost classifier with a 10-fold cross validation (CV) and with the proposed AutoEval approaches (X-Eval and C-Eval). Next, we computed the absolute error between each estimated performance and true performance when the model was applied to the target dataset (i.e., $|estimatedPerformance - truePerformance|$).

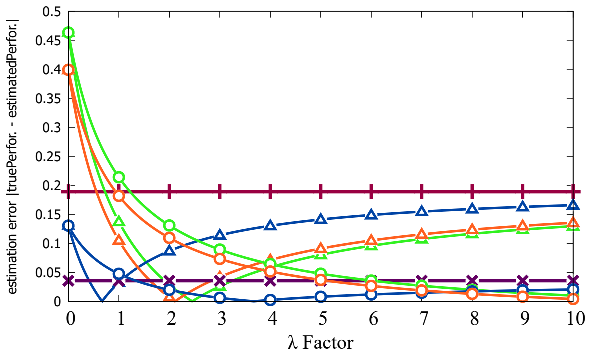
Figure 6.8, Figure 6.9, and Figure 6.10 show the absolute error between the estimated and true performance on each analyzed setup of source and target sets. Each figure shows the results of one evaluated similarity function (i.e., Cosine, Jaccard, and Macro F1). We investigated the effectiveness of C-Eval when applied to regularize the performance estimation of both CV and X-Eval. We show the results for different values of the λ factor, ranging the value of λ from 0 to 10 with an interval of 0.01, to evaluate the λ factor influence on C-Eval results. Besides, we used the algorithms PC [93, 42, 51], Notears [114], and DirectLiNGAM [87] to infer the causal graphs from a datatset in the C-Eval algorithm.

In Figure 6.8, Figure 6.9, and Figure 6.10, we can observe that the X-Eval estimation error remained lower or equal to the CV estimation error in all cases. The inherent characteristics of each approach explain this tendency. This is because, we designed the X-Eval to perform performance estimation in scenarios in which we expect distribution shifts and the CV was not prepared for such situations (i.e., the CV was designed for conditions where the target and source datasets are independent and identically distributed (iid)). In other words, the X-Eval considers k estimates of the model performance in the target data. In contrast, CV considers k estimates based only on the source data. Since the distributions of the source and the target datasets are different, we expect the CV to achieve a greater estimation error than the X-Eval.

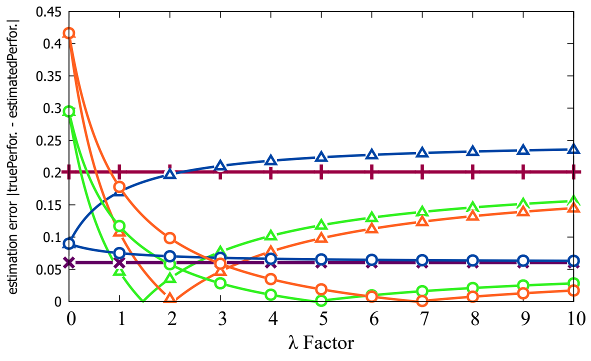
Specifically, we can attribute the superior X-Eval results to the following characteristics:

1. **Individual Instance Judgment**: The X-Eval employs a strategy to classify each target instance as correct or incorrect. This strategy is essential to treat covariate shifts.

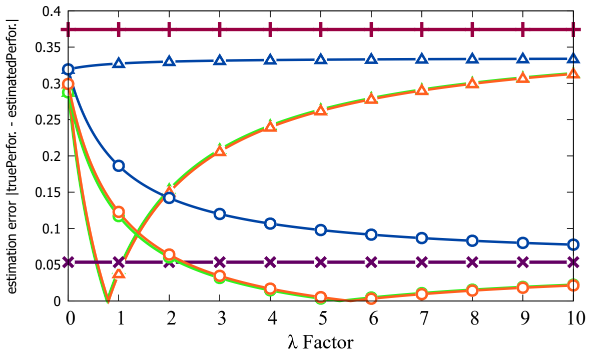
2. **Label Shift Simulation**: Another key X-Eval characteristic is the simulation of changes in the label distribution (i.e., $p(y)$) in the target dataset. This strategy



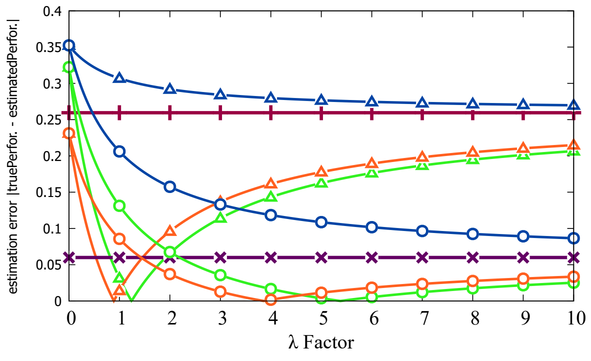
(a) Source: HBP, Target: HSL



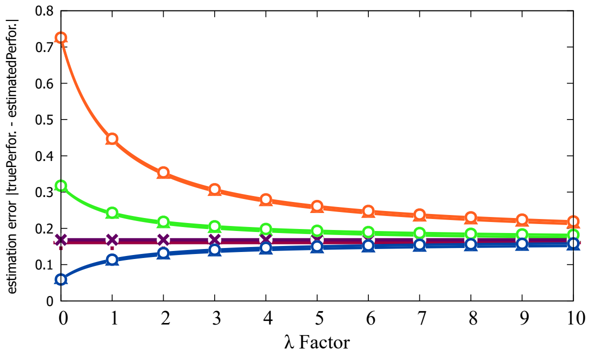
(b) Source: HSL, Target: HBP

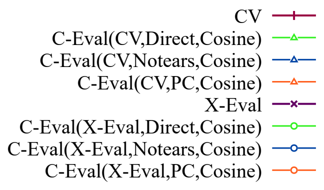


(c) Source: Geriatrics, Target: Neurology



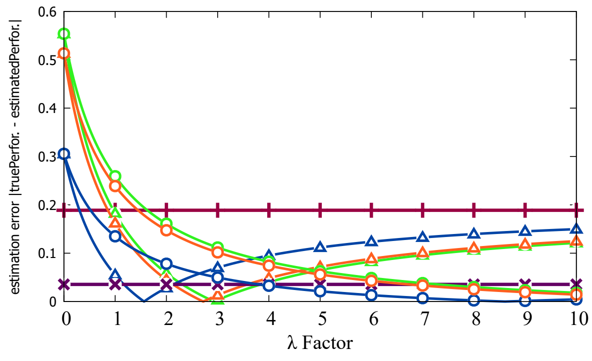
(d) Source: Neurology, Target: Geriatrics



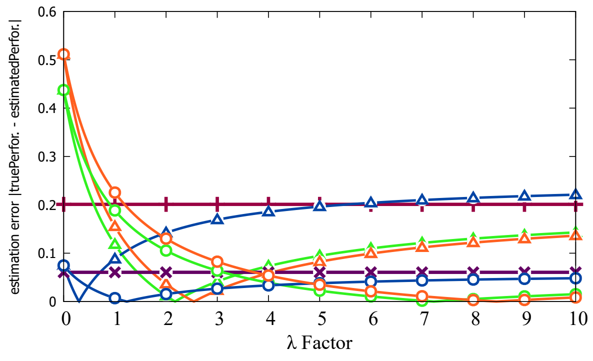


(e) Source: *Students2008*, Target: *Students2014*

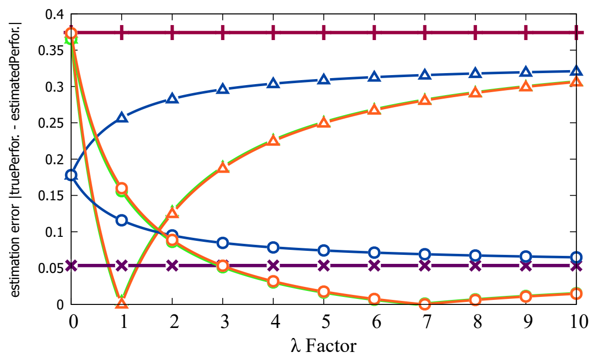
Figure 6.8: Generalization gap analysis of the X-Eval and the C-Eval with the **Cosine similarity function**. For each evaluated approach, we can see the absolute error between the true performance and the estimated model performance on the target dataset over different values to the λ factor. C-Eval($\mathcal{C}$, s) denotes the C-Eval with the causal discovery algorithm $\mathcal{C}$ and the similarity function s.



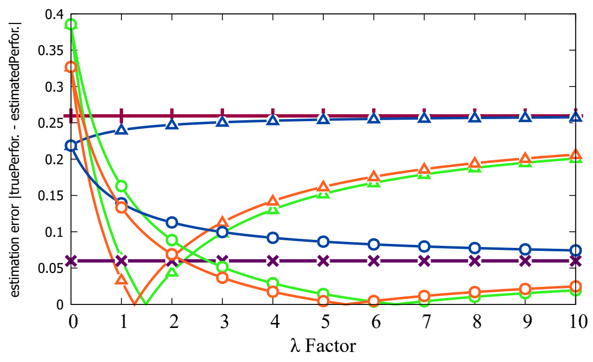
(a) Source: HBP, Target: HSL



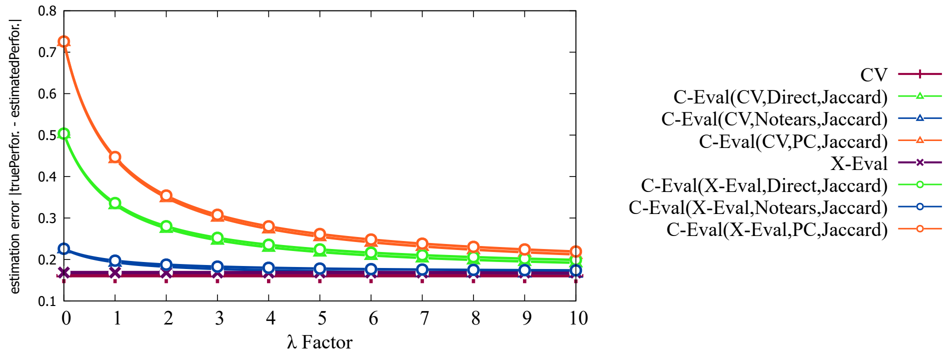
(b) Source: HSL, Target: HBP



(c) Source: Geriatrics, Target: Neurology

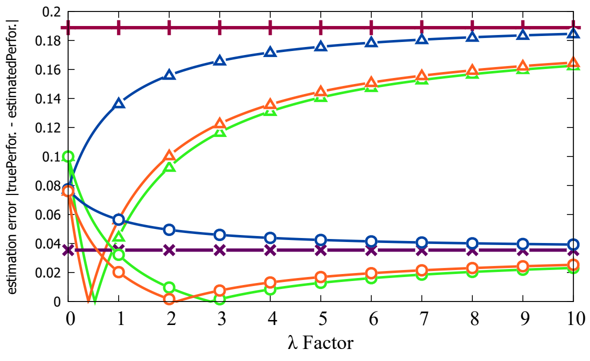


(d) Source: Neurology, Target: Geriatrics

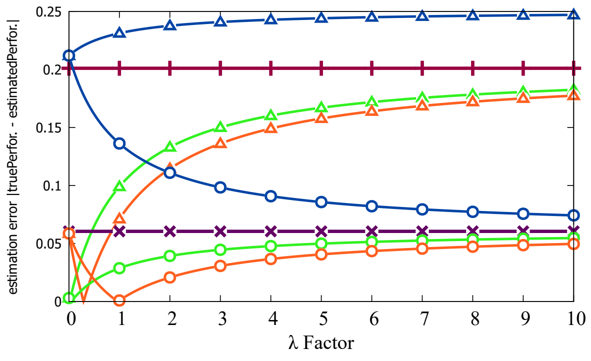


(e) Source: *Students2008*, Target: *Students2014*

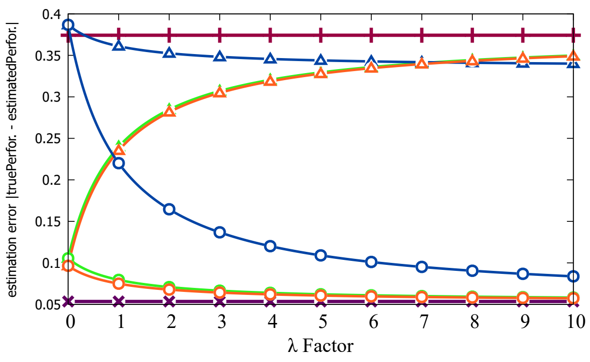
Figure 6.9: Generalization gap analysis of the X-Eval and the C-Eval with the **Jaccard similarity function**. For each evaluated approach, we can see the absolute error between the true performance and the estimated model performance on the target dataset over different values to the λ factor. C-Eval($\mathcal{C}$, s) denotes the C-Eval with the causal discovery algorithm $\mathcal{C}$ and the similarity function s.



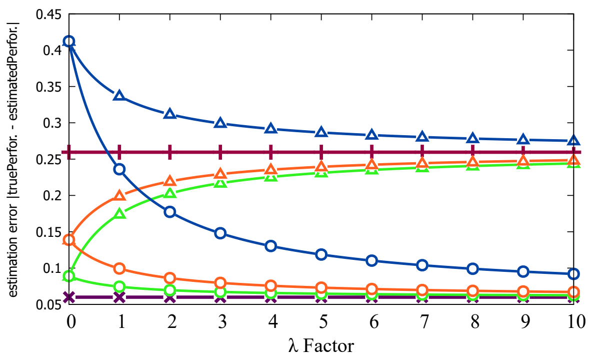
(a) Source: HBP, Target: HSL



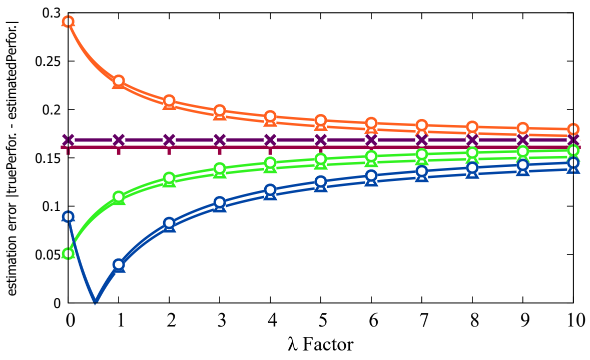
(b) Source: HSL, Target: HBP

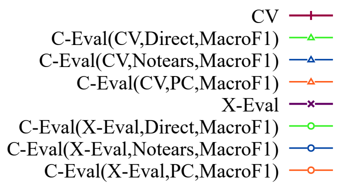


(c) Source: Geriatrics, Target: Neurology



(d) Source: Neurology, Target: Geriatrics





(e) Source: *Students2008*, Target: *Students2014*

Figure 6.10: Generalization gap analysis of the X-Eval and the C-Eval with the **Macro F1 similarity function**. For each evaluated approach, we can see the absolute error between the true performance and the estimated model performance on the target dataset over different values to the λ factor. C-Eval($\mathcal{C}$, s) denotes the C-Eval with the causal discovery algorithm $\mathcal{C}$ and the similarity function s.

addresses label shift by preparing the AutoEval model for different changes in the data distribution.

3. **Agreement as Complementary Feature**: X-Eval leverages the agreement as a complementary feature. When we combine the agreement with other features that the X-Eval uses, the X-Eval may deal with some cases of concept drift. For instance, when concept drift is caused by underspecification (i.e., when different solutions solve equivalent problems) [30, 21]).

Thus, as in Section 6.1.1 we identified distribution shifts among the datasets regarding COVID-19 and Alzheimer. In conjunction with the X-Eval characteristics, these distribution shifts explain the lower X-Eval estimation error in these datasets. However, in the case of the School Dropout datasets, as showed in Section 6.1.1, we did not observe significant distribution shifts among the datasets. In this situation, even without notable distribution shifts, the X-Eval estimation error remained equal to the CV estimation error.

Analyzing the C-Eval regularizing the CV estimate, this combination outperformed the standard CV in most cases. For instance, the C-Eval with CV and Macro F1 as a similarity function (Figure 6.10) outperformed the standard CV in all evaluated cases with the DirectLinGAM algorithm. The best results were achieved with the λ factor between 0 and 1. We attribute this result to the close approximation of the Macro F1 similarity to the true performance, as detailed in Table 6.2.

The C-Eval regularizing the CV estimate compared to the X-Eval (alone) approximated more to the true performance with specific hyperparameters. For instance, (1) in COVID-19 datasets with Cosine similarity, PC, and a λ factor equals 2 (Figure 6.8a and Figure 6.8b); (2) in Alzheimer datasets with Cosine similarity, PC, and a λ factor equals 1 (Figure 6.8c and Figure 6.8d); and (3) in dataset related to School Dropout with Macro F1 similarity, DirectLinGAM, and a λ factor equals 1 (Figure 6.10e).

Analyzing the λ factor impact in the C-Eval results, in Figure 6.8, Figure 6.9, and Figure 6.10, we can see that the C-Eval achieved the best results in the majority of cases according to the used λ factor. For instance, in the COVID-19 and Alzheimer datasets, the C-Eval with DirectLinGAM and PC, in most cases, achieved near zero estimation error when regularized the: (1) CV estimate, with λ factor between 1 and 3, and (2) the X-Eval estimate, with λ factor between 4 and 7. These ranges of λ factor are due to the fact that, as X-Eval tends to approximate more to the true performance compared to CV, the X-Eval estimate must have a greater weight in the C-Eval process than the CV estimate.

Overall, the results of the C-Eval with the causality inference algorithms DirectLinGAM and PC and the similarity function Cosine and Jaccard were similar. We can better understand these results by analyzing the Table 6.2. Once, (1) on the COVID-19 and Alzheimer datasets, the CV estimated a higher performance than the true performance, and (2) the similarities between the causal graphs inferred by the DirectLinGAM and PC algorithms were lower than the true performance, then the C-Eval decreased the estimated performance. This adjustment closely approximated the C-Eval estimate to the true performance. Besides, given the close similarities of the causality graphs inferred by the DirectLinGAM and PC algorithms, the C-Eval results with these algorithms were close.

Table 6.2: This table presents detailed metrics used in generalization gap analysis. It includes Cosine, Jaccard, and Macro F1 similarities among the causal graphs inferred by the DirectLiNGAM (Direct), PC, and Notears algorithms. Additionally, the last two lines of the table show the True Performance (macro f1) and the performance estimated with 10-fold Cross-Validation (CV).

Causality Inference Algorithm	**Metric**	**Source: Geriatrics Target: Neurology**	**Source: Neurology Target: Geriatrics**	**Source: HBP Target: HSL**	**Source: HSL Target: HBP**	**Source: Students2008 Target: Students2014**
Direct	**Cosine**	0.17	0.13	0.20	0.33	0.41
	Jaccard	0.09	0.06	0.11	0.19	0.22
	Macro-F1	0.56	0.54	0.56	0.62	0.67
PC	**Cosine**	0.16	0.22	0.26	0.21	0.00
	Jaccard	0.09	0.12	0.15	0.11	0.00
	Macro-F1	0.56	0.59	0.59	0.56	0.43
Notears	**Cosine**	0.78	0.80	0.53	0.71	0.67
	Jaccard	0.64	0.67	0.36	0.55	0.50
	Macro-F1	0.85	0.86	0.74	0.84	0.81
	True performance	0.46	0.45	0.66	0.62	0.73
	CV estimation	0.83	0.71	0.85	0.82	0.56
	X-Eval estimation	0.51	0.51	0.70	0.68	0.56

In the case of the School Dropout datasets, when considering C-Eval with the causal inference algorithms DirectLinGAM and PC and the similarity functions Cosine and Jaccard, the C-Eval results were worse or equal to the CV. We attribute these results to the following facts:

1. In the case of PC, the similarity was zero (Table 6.2). A simple adaptation to the C-Eval algorithm for this scenario would be using a second algorithm for causality inference if the causality inference algorithm does not identify causal relations in a dataset.

2. In the DirectLinGAM case, the similarity among the inferred causal graphs and the CV estimate was lower than the true performance (Table 6.2). Thus, C-Eval provided an even lower estimate, which increased the estimation error.

In general, the error in the C-Eval estimation with Notears was either greater or close to the error in the CV estimate, as illustrated in Figure 6.8b. This result is attributed to Notears tendency to infer a larger number of causal relationships, as evidenced in the graphs presented in Figure 6.5, Figure 6.6, and Figure 6.7. Consequently, the higher similarity between graphs inferred by Notears implies that the C-Eval estimates performance higher than the CV, as exemplified in Table 6.2 (e.g., considering Neurology as the source dataset, Notears, and Macro F1 similarity). In other words, with both the CV estimate and the Notears graphs similarity higher than the true performance, the error in C-Eval estimates tended to increase.

Given the experiments presented in this section, in general, the best results were achieved with the C-Eval regularizing the X-Eval estimate, using the following hyperparameters: Macro F1 as the similarity function, the DirectLinGAM algorithm for causality inference, and a λ factor between 2 and 3. The reasons for this are the following:

1. The estimation of the X-Eval approximated more to the true performance, outperforming the CV in the studied scenarios involving distribution shifts. Even in cases with minimal distribution changes, such as in the datasets regarding School Dropout, the error in the X-Eval estimation was nearly identical to the CV error. These results indicate that, even with no distribution shifts, the X-Eval estimation error is close to the CV estimation error.

2. The Macro F1 as the similarity function showed the best results because it was the same metric used to measure the model performance. Consequently, the similarity measured by this function tends to be closer to the true performance than the similarities obtained from Cosine and Jaccard functions. Another hypothesis is because of the characteristics of Macro F1 to measure similarity in imbalanced data. This characteristic becomes especially relevant considering the adjacency matrices of causal graphs, where the analyzed data is sparse, i.e., the causality graphs have few edges compared to the number of possible edges.

3. The DirectLinGAM outperformed the other evaluated causality inference algorithms because of its effectiveness and simplicity in causality discovery. Comparing the DirectLinGAM with the Notears, the Notears tends to infer a greater number of causal relationships, resulting in high similarity between causal graphs (which can be observed in Table 6.2, Figure 6.7, Figure 6.6, and Figure 6.5). With this high similarity, the estimated performance was not penalized in cases where it should

have been penalized. This situation occurred because the similarity between the causality graph was greater than the true performance, while the CV estimate was also greater than the true performance. Concerning the PC, despite the proximity of causality graphs inferred by the PC to those graphs inferred by the DirectLinGAM, the PC algorithm failed to identify causal relationships in one dataset about School Dropout. Consequently, the error of the C-Eval estimate with PC was higher than the error of the CV estimate in this dataset. This may have happened mainly because of the conditional independence test used in each algorithm. Once in the DirectLiNGAM we used mutual information [87] and in the PC we used Fisher's z-transformation [43].

4. The best results were achieved with the λ factor between 2 and 3 because of the close approximation of the X-Eval estimate and the Macro F1 similarity to the true performance. Consequently, the weight of these two measures in the analysis must be similar.

In the context of Alzheimer datasets, the performance of C-Eval with X-Eval and Macro F1 similarity is not superior to that of X-Eval (alone) (i.e., the errors in the estimation are practically identical) (Figure 6.10c and Figure 6.10d). This result is because both Macro F1 similarity and X-Eval estimate were close to the true model performance. As we can see in Table 6.2, the difference between the true performance and the Macro F1 similarity was 0.10, and between the true performance and the X-Eval estimate was 0.05. However, C-Eval with X-Eval demonstrates lower estimation error when compared to X-Eval (alone) when we use the Jaccard and Cosine similarity functions.

These results indicate that the proposed C-Eval approach can achieve better effectiveness if we select appropriate hyperparameters for each distribution shift scenario. Therefore, a challenge in the C-Eval approach may be the definition of appropriate hyperparameters for the data in analysis based on its characteristics and the expected type/degree of distribution shift. For instance, (1) in cases where a strong distribution shift is likely, we can use the Cosine similarity, as it tends to be smaller than the estimated performance; and (2) in cases where we expect a small distribution shift, we can use the Macro F1 similarity, as this metric returns a range of values closer to the true performance. However, the experiments showed that the combination of the C-Eval with the X-Eval consistently achieved the best results. Next, we analyze the impact of the explanation feature used in the AutoEval model on the X-Eval estimation.

6.2.1 Analysis of the Impact of the Features Used in the X-Eval

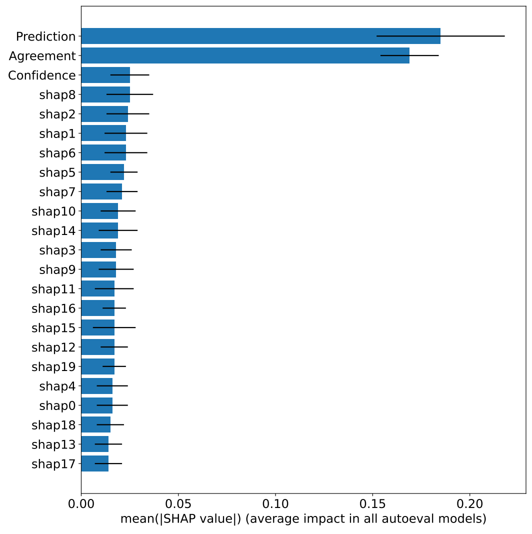
To classify each target example as correct or incorrect, the X-Eval builds an AutoEval model using features commonly used for model explanation. As these features come from different explainability approaches, it is interesting to understand the impact of each type of explanation feature in the AutoEval model.

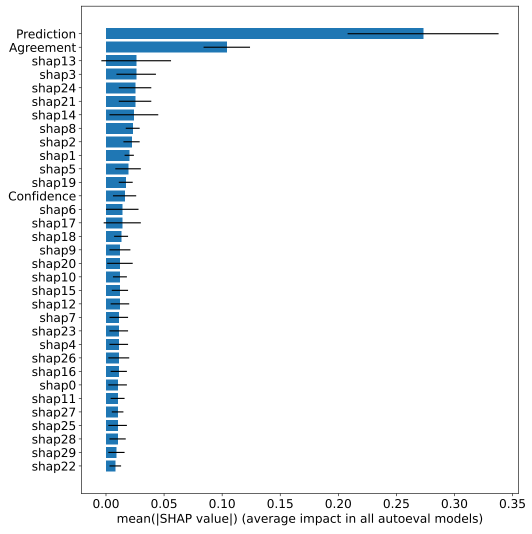
Initially, we computed the SHAP values (i.e., the impact of each feature on the model prediction) for every model built in the X-Eval process. Next, we calculated the average of the SHAP values for each feature across all models. This aggregation aims to facilitate the analysis. By analyzing these average SHAP values, we can have insights into the model behavior and identify the most important features by considering all AutoEval models in the X-Eval process.

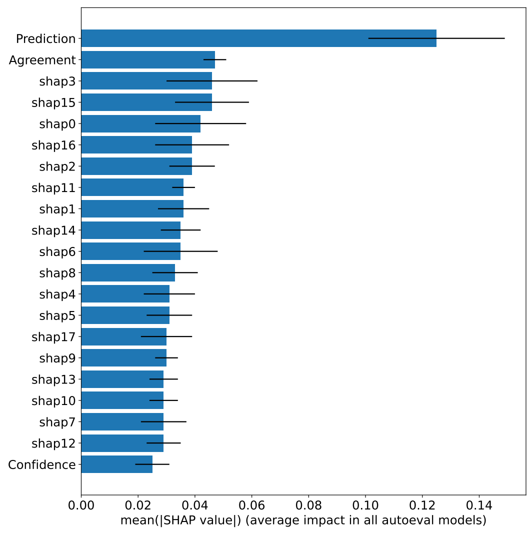
Figure 6.11 shows the average impact of each feature used in the X-Eval AutoEval model. We can see that the prediction of the evaluated model has the greatest impact in the majority of cases (4/5). This result is significant because it is the model prediction that the AutoEval model must judge if it is correct or not. The second feature that has the most impact is the agreement among the classifiers, this shows the importance of this feature to judge the prediction as correct or incorrect. However, we can see that we have a greater number of features created from the SHAP values, and if we consider the impact of all SHAP values features, we can consider this source of information as the most important.

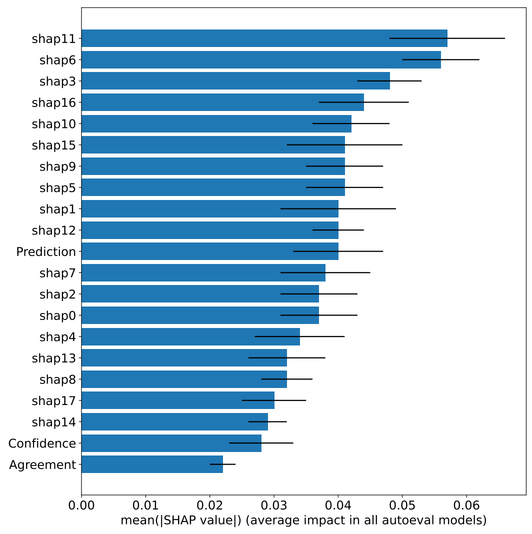
Figure 6.12 shows the absolute estimation error (i.e., $|TruePerformance - EstimatedPerformance|$) of X-Eval across various combinations of feature types. The combination of Agreement, Confidence, and SHAP values provided the most stable results. This combination consistently achieved the smallest error in three of the five setups of source and target datasets (3/5). Even in the Alzheimer datasets, where it achieved the second smallest error, the difference to the first approach was relatively small (i.e., ¡ 0.05). These evidences demonstrate the importance of combining the three feature types.

In this section, we evaluated the applicability of the proposed X-Eval and C-Eval approaches to estimate the model performance, focusing on the XGBoost classifier. The next section analyzes the proposed AutoEval approaches considering different classifiers.


(a) Source: HBP, Target: HSL


(b) Source: HSL, Target: HBP


(c) Source: Geriatrics, Target: Neurology


(d) Source: Neurology, Target: Geriatrics

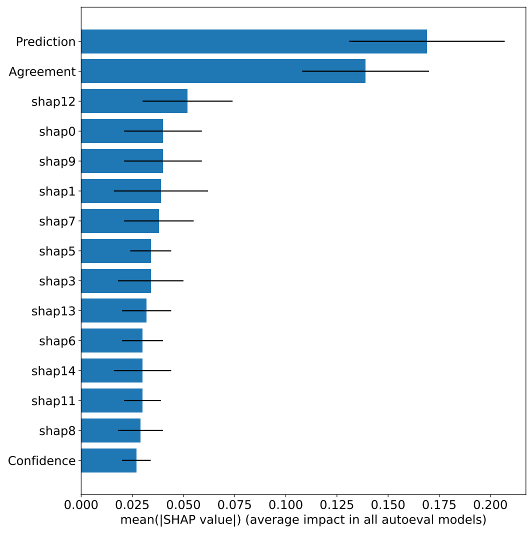

(e) Source: *Students2008*, Target: *Students2014*

Figure 6.11: Impact of the features used in the AutoEval model. The names of the features created using SHAP values start with *shap*.

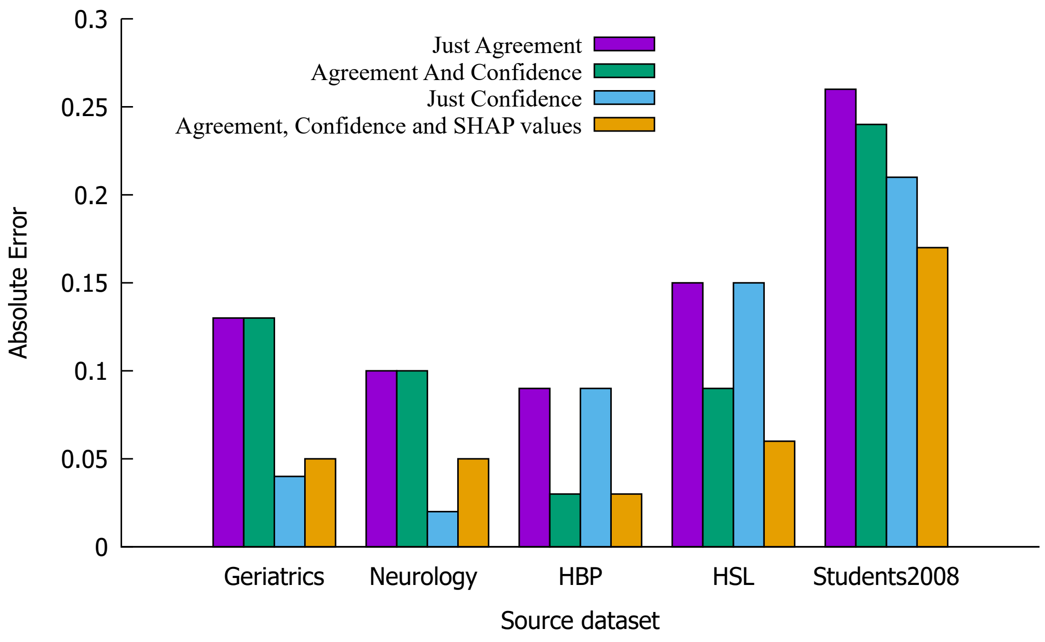


Figure 6.12: X-Eval absolute estimation error with different combinations of feature types.

6.2.2 Generation Gap Analysis with Different Classifiers

This section analyzes the estimation error of the proposed approaches for different classifiers. To facilitate the analysis, in this section we present the experiments with the C-Eval algorithm only with the DirectLinGAM causality inference algorithm, the Macro F1 similarity function, and factors 1, 3, and 6. We used these C-Eval hyperparameters because they achieved the best results in the previous analysis. The results for the other causality inference algorithms and similarity functions are in Appendix E.

In this experiment, we used the classifiers XGBoost, LightGBM, and CatBoost because they are in the state-of-art to automatic classification for tabular data [8]. Besides, we used the deep learning model TabNet because in the [4] experiments, the TabNet outperformed many state-of-art classifiers (such as ensemble decision trees methods, e.g., XGBoost) on the classification task on tabular datasets. To generate the SHAP values, we used (1) the TreeSHAP for XGBoost, LightGBM, and CatBoost classifiers and (2) the KernelSHAP for TabNet.

Table 6.3 shows the estimation error of the proposed autoEval approaches (X-Eval and C-Eval) and the CV. We can see that the proposed approaches achieved the smallest estimation error in most cases. The 10-fold cross-validation (CV) just achieved the smallest estimation in one analyzed case (i.e., 1/20), specifically in the experiment with the dataset regarding School Dropout and the TabNet classifier. However, we highlight that when the CV was the best, the TabNet achieved a very small performance in the target dataset (i.e., 0.17 performance in Macro F1) and in the CV estimation (i.e., 0.18

performance in Macro F1). So, in this case, in the real world, the TabNet would hardly ever be selected to build a classification model to the School Dropout data.

Table 6.3: Estimation error analysis for the classifiers XGBoost, LightGBM, CatBoost, and TabNet. We compared the proposed autoEval approaches (X-Eval and C-Eval) with the 10-fold cross-validation (CV). The C-Eval($Estimatior$, $Direct$, $MacroF1$, λ) denotes the C-Eval approach with one $Estimatior$ (i.e, CV or X-Eval), the $Direct$LiNGAM inference causality algorithm and one λ factor (i.e., 1, 3, or 6). We highlighted the smallest estimation error for each dataset considering each classifier in **bold**, and we also underline the second smallest estimation error.

	Performance Estimator	**Source: Geriatrics Target: Neurology**	**Source: Neurology Target: Geriatrics**	**Source: HBP Target: HSL**	**Source: HSL Target: HBP**	**Source: Students2008 Target: Students2014**
XGBoost	CV	0.37	0.26	0.19	0.20	0.16
	X-Eval	**0.05**	**0.06**	0.04	0.06	0.17
	C-Eval(CV, Direct, MacroF1, 1)	0.24	0.17	0.04	0.10	**0.11**
	C-Eval(X-Eval, Direct, MacroF1, 1)	0.08	0.07	0.03	**0.03**	**0.11**
	C-Eval(CV, Direct, MacroF1, 3)	0.31	0.22	0.12	0.15	0.13
	C-Eval(X-Eval, Direct, MacroF1, 3)	0.07	0.07	**0.00**	0.04	0.14
	C-Eval(CV, Direct, MacroF1, 6)	0.34	0.24	0.15	0.17	0.15
	C-Eval(X-Eval, Direct, MacroF1, 6)	0.06	**0.06**	0.02	0.05	0.15
LightGBM	CV	0.31	0.11	0.22	0.32	0.09
	X-Eval	**0.00**	0.09	0.07	0.16	0.12
	C-Eval(CV, Direct, MacroF1, 1)	0.17	**0.04**	0.07	0.17	**0.07**
	C-Eval(X-Eval, Direct, MacroF1, 1)	0.01	0.06	**0.00**	**0.09**	0.08
	C-Eval(CV, Direct, MacroF1, 3)	0.24	0.07	0.15	0.25	0.08
	C-Eval(X-Eval, Direct, MacroF1, 3)	0.01	0.07	0.04	0.13	0.10
	C-Eval(CV, Direct, MacroF1, 6)	0.27	0.09	0.18	0.28	0.09
	C-Eval(X-Eval, Direct, MacroF1, 6)	**0.00**	0.08	0.05	0.14	0.11
CatBoost	CV	0.33	0.09	0.25	0.30	0.15
	X-Eval	**0.01**	0.09	0.08	0.07	0.21
	C-Eval(CV, Direct, MacroF1, 1)	0.20	**0.00**	0.09	0.17	**0.12**
	C-Eval(X-Eval, Direct, MacroF1, 1)	0.03	0.08	**0.01**	**0.06**	0.14
	C-Eval(CV, Direct, MacroF1, 3)	0.26	0.04	0.17	0.24	0.14
	C-Eval(X-Eval, Direct, MacroF1, 3)	0.02	0.09	0.04	0.07	0.18
	C-Eval(CV, Direct, MacroF1, 6)	0.29	0.06	0.20	0.26	0.14
	C-Eval(X-Eval, Direct, MacroF1, 6)	0.01	0.09	0.06	0.07	0.19
TabNet	CV	0.10	0.12	0.07	0.03	**0.01**
	X-Eval	**0.04**	0.10	**0.00**	0.05	0.06
	C-Eval(CV, Direct, MacroF1, 1)	0.10	0.08	0.31	0.06	0.26
	C-Eval(X-Eval, Direct, MacroF1, 1)	0.07	**0.07**	0.27	0.02	0.29
	C-Eval(CV, Direct, MacroF1, 3)	0.10	0.10	0.19	0.05	0.13
	C-Eval(X-Eval, Direct, MacroF1, 3)	0.05	0.08	0.13	**0.01**	0.18
	C-Eval(CV, Direct, MacroF1, 6)	0.10	0.11	0.14	0.04	0.08
	C-Eval(X-Eval, Direct, MacroF1, 6)	0.05	0.09	0.07	0.03	0.13

Besides, it is interesting to observe that the X-Eval achieved a relatively small estimation error in the datasets regarding Alzheimer and COVID-19, once the estimation error was between 0.0 and 0.1. Additionally, even in all datasets, the X-Eval estimation error was relatively small since the greatest estimation error of the X-Eval was 0.21, and the greatest CV estimation error was 0.37. This result demonstrates the stability of the

X-Eval estimation. Furthermore, the C-Eval decreased the X-Eval estimation error in many cases. For instance, considering the approach C-Eval(X-Eval, $Direct, MacroF1, 1$), the C-Eval improved the X-Eval estimation error in 13/20 cases.

The results presented in this section demonstrated the applicability of the proposed autoEval approaches considering different classifiers. The next section deepens our experimental evaluation and demonstrates the applicability of the X-Eval and C-Eval to feature selection.

6.3 An In-depth Analysis of the Proposed Algorithms Across Diverse Feature Subsets

Once it is a challenge to get datasets with differences in the data generation process that share the same features and classes, this section presents an analysis of the proposed AutoEval approaches in the performance estimation of models built from different feature sets. Thus, this analysis aims to conduct a more robust evaluation.

Additionally, this methodology evaluates the proposed approaches in the feature selection task considering unlabeled data with an unknown distribution (i.e., a specific out-of-distribution target dataset). Feature selection is an important task to improve the effectiveness and interpretability of classification models. We can define feature selection as a process to select a subset of the most important features of a dataset [62].

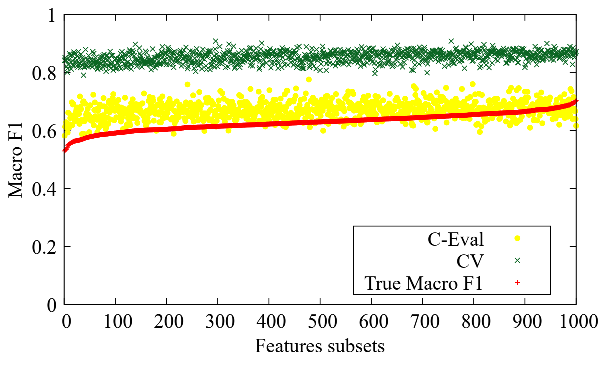
Thus, a relevant application to the proposed algorithms is to use them as indicators for model selection, also considering unlabeled data from an unknown distribution. This approach is interesting because most existing strategies consider only labeled data to select the best model to use [46]. So, we proposed to use AutoEval approaches to deal with the problem of selecting a suitable model for an unlabeled dataset with an unknown distribution. Specifically, we proposed a random search process to evaluate the proposed AutoEval method for feature selection for a specific and possible out-of-distribution dataset.

In this experiment, we used random feature subsets because, as we do not know the distribution of the target set, it would not make sense to use the source dataset distribution information to perform tasks such as feature selection. Besides, through this experiment, we can measure the effectiveness of AutoEval in choosing the best classification model for different distribution shifts. Thus, the proposed random search process consists of the following steps:

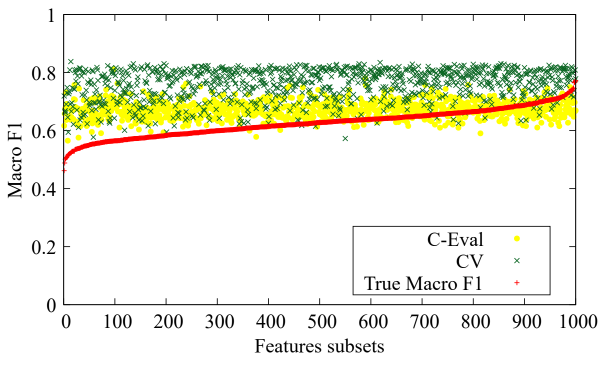
- First, we generated 1,000 different feature subsets, one subset with all features of the source dataset, and the remaining 999 were random feature subsets (in terms of size and items);

- Second, we built a model from the source dataset with each feature subset and we estimated each model performance using the proposed approaches and the 10-fold cross-validation (CV);
- Next, to perform a feature selection, we generated ranks of the models by estimations;
- Finally, we selected the best subset on the feature subset rankings.

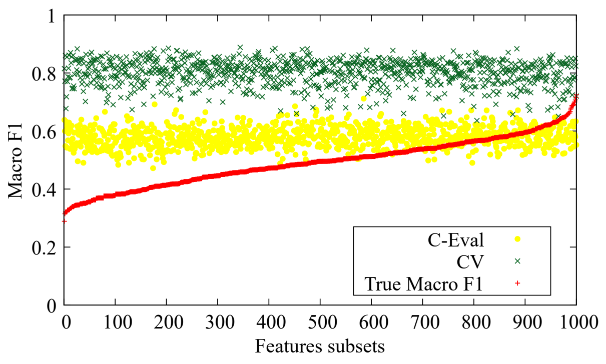
Figure 6.13 shows detailed examples of the proposed evaluation process with AutoEval using the XGBoost classifier. In Figure 6.13, we can compare the true performance (macro f1), the estimates of the C-Eval, and the CV for all generated feature subsets in four datasets. In all cases, the shape of the C-Eval points is closer to the true performance points than the CV points.



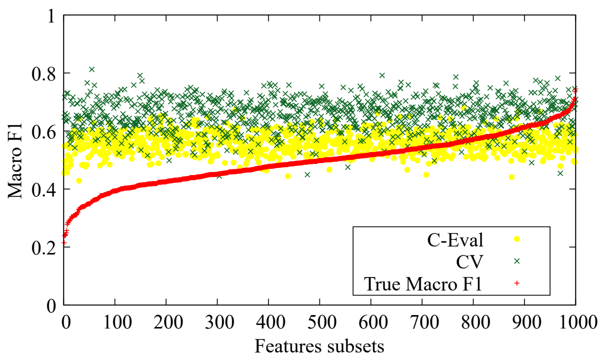
(a) Source: HBP, Target: HSL



(b) Source: HSL, Target: HBP



(c) Source: Geriatrics, Target: Neurology



(d) Source: Neurology, Target: Geriatrics

Figure 6.13: Detailed examples of executions of the proposed feature selection process. C-Eval and CV estimates, and true performance of the XGBoost classifier on 1,000 different features subsets. We sorted the feature subsets by the true performance. In this example, we executed the C-Eval regularizing the X-Eval estimate, the Macro F1 similarity function, the DirectLiNGAM causality inference algorithm, and λ factor 1.

To quantify this observation, first, we analyzed the number of models that the proposed approaches approximated more to the true performance than the CV (Figure 6.14, Figure 6.15, and Figure 6.16). We evaluated the C-Eval with the following settings: (1) with different values of the λ factor (i.e., we varied the value of λ from 0 to 10 with an interval of 0.01), (2) regularizing the estimates of the CV and the X-Eval, and (3) with the DirectLinGAM, PC, and Notears causality inference algorithms.

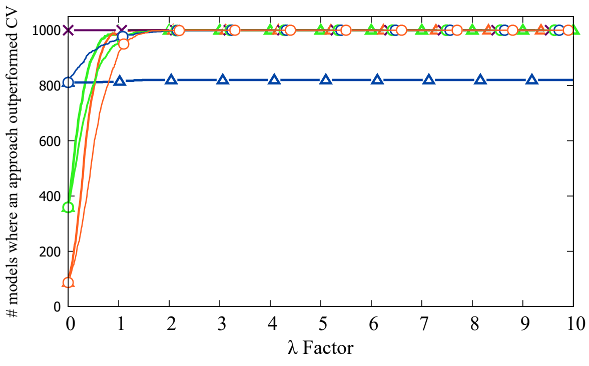
In Figure 6.14, Figure 6.15, and Figure 6.16, we can see that the X-Eval achieved a smaller estimation error than CV in at least 70% of the analyzed models in all datasets. This finding highlights the X-Eval potential to provide more precise estimates of model performance under distribution shift scenarios.

Furthermore, in most cases, the C-Eval regularizing the CV estimate improved the estimation error compared to the standard CV. For instance, we observed this in more than 60% of models in all datasets when the C-Eval used the DirectLinGAM and the Macro F1 similarity function (Figure 6.16).

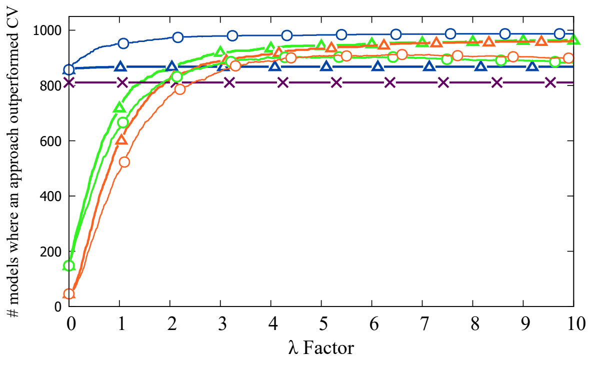
Overall, in this first analysis, the C-Eval results with Cosine and Jaccard similarities, and the causality inference algorithms DirectLinGAM and PC were similar (Figure 6.14 and Figure 6.15). The C-Eval with these similarity functions achieved an estimation error smaller than the CV estimation error in at least 60% of the analyzed models in four datasets, specifically in the datasets regarding COVID-19 and Alzheimer (Figure 6.14a, Figure 6.14b, Figure 6.14c, and Figure 6.14d) with the causality inference algorithm DirectLinGAM and PC and a λ factor ≥ 2.

In this first analysis, in the dataset about School Dropout (Figure 6.14e, Figure 6.15e), the C-Eval with Cosine and Jaccard similarities, and the causality inference algorithms DirectLinGAM and PC showed inferior results compared to other datasets. This result is explained by the fact that the distributions of the School Dropout datasets are similar (as we saw in Section 6.1.1), so the estimated performance by the CV or X-Eval is close to the true performance. However, the estimated performance decreases when we regularize the CV or X-Eval estimates using the Cosine or Jaccard similarities. Then, when we consider the datasets regarding COVID-19 and Alzheimer (Figure 6.20, and Figure 6.21), this decrease approximates the estimated performance to the true performance (i.e., this is the correct behavior). However, in the datasets regarding School Dropout, the estimated performance is lower than the true performance, so a decrease in the estimated performance increases the estimation error. Section 6.3.1 better demonstrates these phenomena.

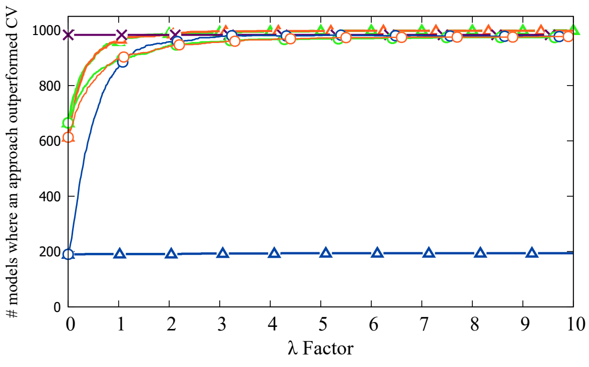
In contrast, the Notears causality inference algorithm was more effective in situations with a small distribution shifts (i.e., in the School Dropout dataset). In the School Dropout dataset, the C-Eval with Notears and Cosine similarity achieved a lower estimation error than the CV in more than 70% of the analyzed models (Figure 6.14e). Besides, C-Eval with Notears, Jaccard similarity, X-Eval, and factor $\lambda \geq 4$ achieved a



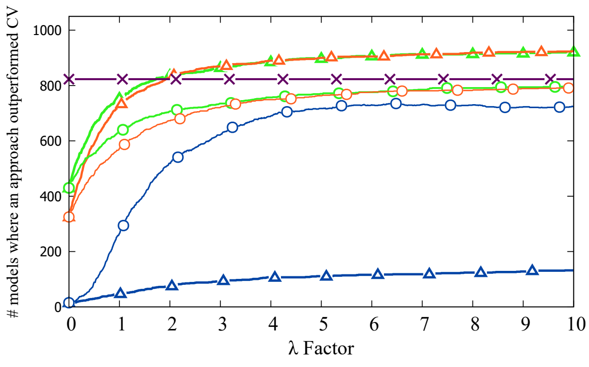
(a) Source: HBP, Target: HSL



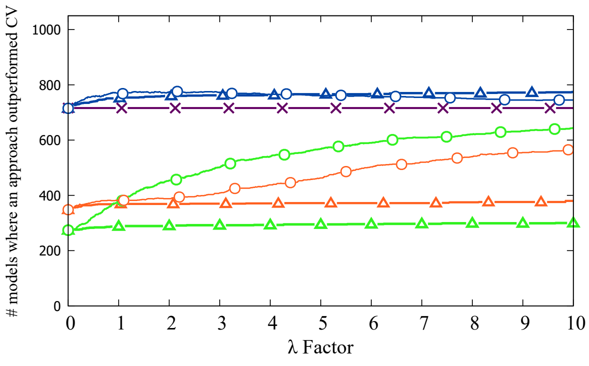
(b) Source: HSL, Target: HBP

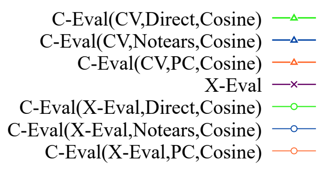


(c) Source: Geriatrics, Target: Neurology



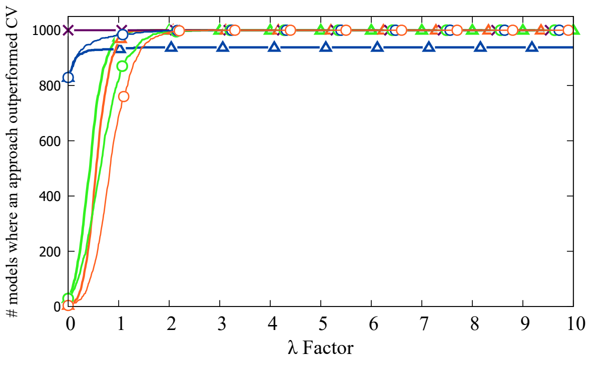
(d) Source: Neurology, Target: Geriatrics



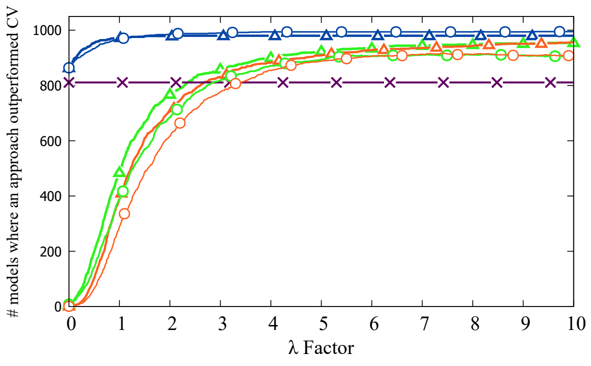


(e) Source: *Students2008*, Target: *Students2014*

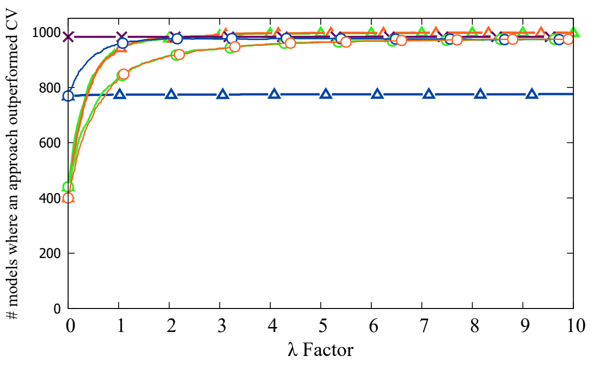
Figure 6.14: The number of models (created by sub-feature sets) that the X-Eval and the C-Eval with the **Cosine similarity function** approximated more to the true performance than the CV estimate. The graphs show the results for different values of the λ factor. C-Eval(C, s) denotes the C-Eval with the causal discovery algorithm C and the similarity function s.



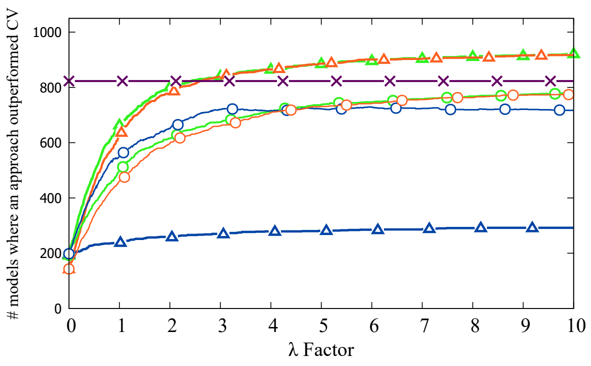
(a) Source: HBP, Target: HSL



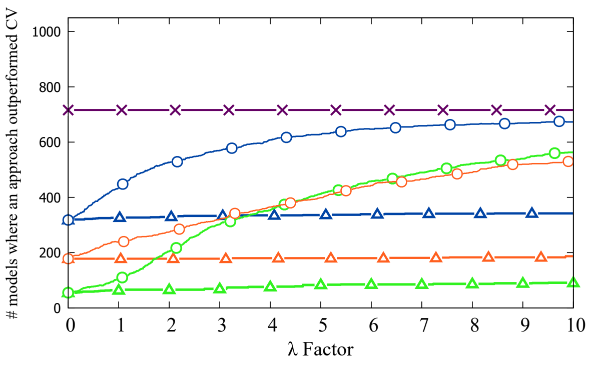
(b) Source: HSL, Target: HBP

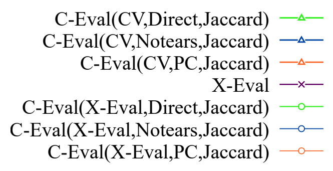


(c) Source: Geriatrics, Target: Neurology



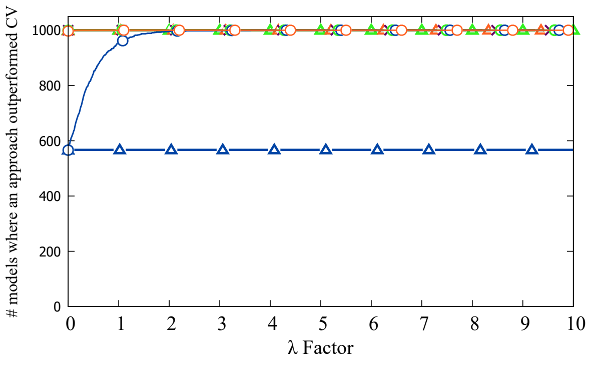
(d) Source: Neurology, Target: Geriatrics



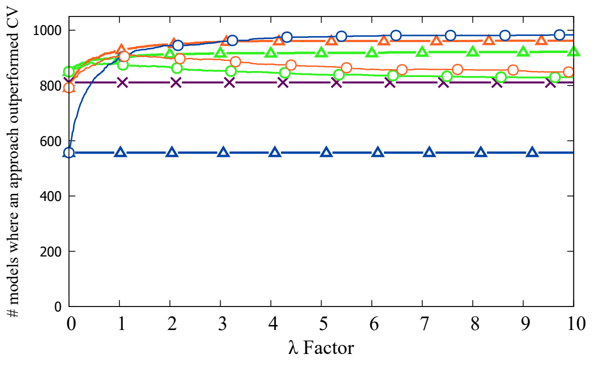

C-Eval(CV,Direct,Jaccard)
C-Eval(CV,Notears,Jaccard)
C-Eval(CV,PC,Jaccard)
X-Eval
C-Eval(X-Eval,Direct,Jaccard)
C-Eval(X-Eval,Notears,Jaccard)
C-Eval(X-Eval,PC,Jaccard)

(e) Source: *Students2008*, Target: *Students2014*

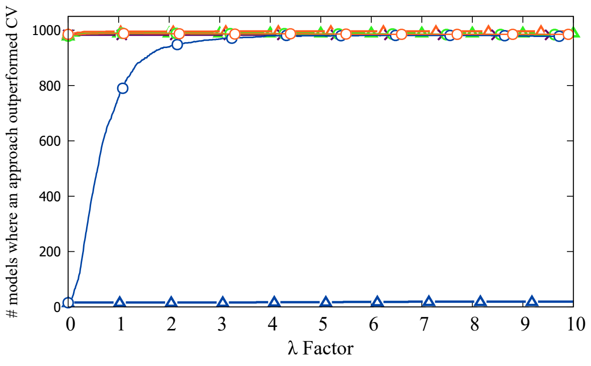
Figure 6.15: The number of models (created by sub-feature sets) that the X-Eval and the C-Eval with the **Jaccard similarity function** approximated more to the true performance than the CV estimate. The graphs show the results for different values of the λ factor. C-Eval(C, s) denotes the C-Eval with the causal discovery algorithm C and the similarity function s.



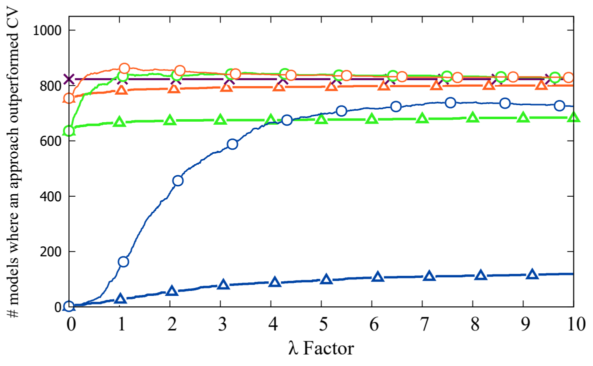
(a) Source: HBP, Target: HSL



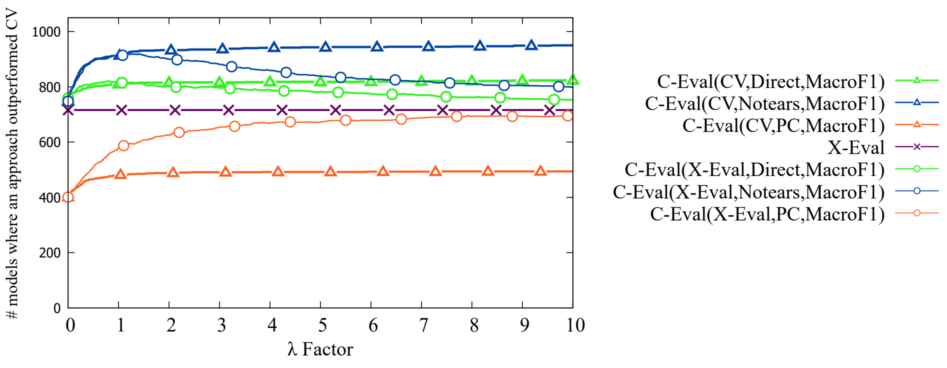
(b) Source: HSL, Target: HBP



(c) Source: Geriatrics, Target: Neurology



(d) Source: Neurology, Target: Geriatrics



(e) Source: *Students2008*, Target: *Students2014*

Figure 6.16: The number of models (created by sub-feature sets) that the X-Eval and the C-Eval with the **Macro F1 similarity function** approximated more to the true performance than the CV estimate. The graphs show the results for different values of the λ factor. C-Eval(C, s) denotes the C-Eval with the causal discovery algorithm C and the similarity function s.

lower estimation error than the CV in more than 60% of the analyzed models (Figure 6.15e).

The best results, in terms of consistency, were achieved with C-Eval regularizing the X-Eval estimate, using DirectLinGAM, Macro F1 similarity, and λ factor ≥ 1 (Figure 6.16). This combination achieved an estimation error smaller than the CV in at least 80% of the analyzed models. We attribute this result to the following reasons:

- X-Eval estimates are closer to the true performance than CV estimates. This is because the X-Eval considers the possible difference in the source and target dataset distributions, and the CV does not.

- DirectLinGAM is a simple and effective causal inference algorithm able to estimate causal relationships, and these causal relationships contribute to regularizing the performance estimation. Besides: (1) the Notears tends to infer many causal relationships, and (2) there are cases where the similarity between the PC graphs is zero. However, both extreme similarities (i.e., near zero or one) do not contribute to the regulation process.

- Macro F1 similarity is the same metric used to measure model performance. Hence, the range of values returned by this metric is closer to the true performance, which was adequate for regularization. Furthermore, Macro F1 is a recommended metric to measure similarity in imbalanced data, such as the causal graphs represented with adjacent matrices, which have few edges compared to the number of possible edges.

In a second analysis, we evaluated the proposed method as an indicator for feature selection. Figure 6.17, Figure 6.18, and Figure 6.19 show the gain of the proposed approaches in relation to CV using the described evaluation process as a feature selection process. We computed the gain as:

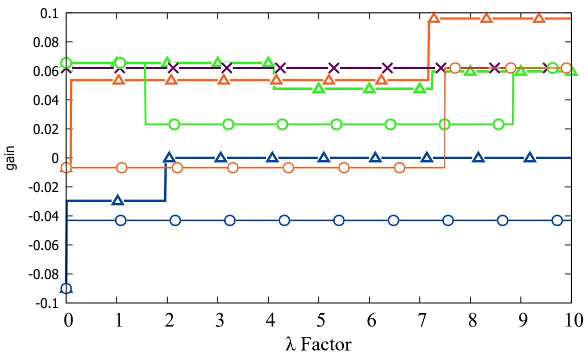
$$\frac{\ell(autoEval, \mathcal{D}_{target}) - \ell(CV, \mathcal{D}_{target})}{\ell(CV, \mathcal{D}_{target})} \tag{6.1}$$

. Where $\ell(autoEval, \mathcal{D}_{target})$ and $\ell(CV, \mathcal{D}_{target})$ denote the achieved performance (i.e., macro f1) in the target dataset with the proposed feature selection process using one proposed AutoEval approach and CV respectively.

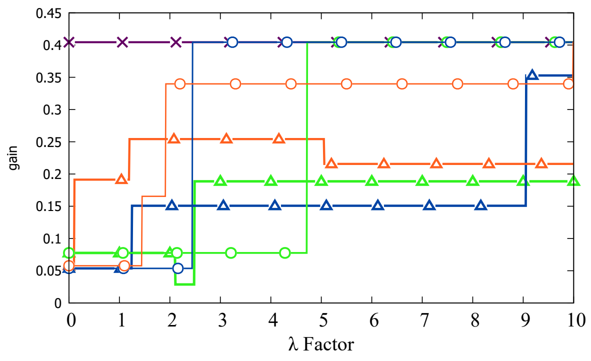
In this second analysis, the proposed approaches outperformed the standard CV in all analyzed cases with gains up to 77%, as evidenced by Figure 6.17d, Figure 6.18d, and Figure 6.19d. Our proposed approaches showed a lesser gain percentage in the School Dropout dataset, this is due to the small distribution shift between the School Dropout datasets (as Section 6.1.1 demonstrated), i.e., with a small distribution shift, the CV estimate tends to have a close approximation to the true performance.

Analyzing the X-Eval, from Figure 6.17, Figure 6.18, and Figure 6.19, we can see that the X-Eval showed gains when compared to the standard CV in all analyzed cases. These gains ranged from 2% to 40% . The gains up to 40% (e.g., in Figure 6.19d) highlight the X-Eval potential for improving the achieved performance in the challenge scenario of model selection in the presence of distribution shift.

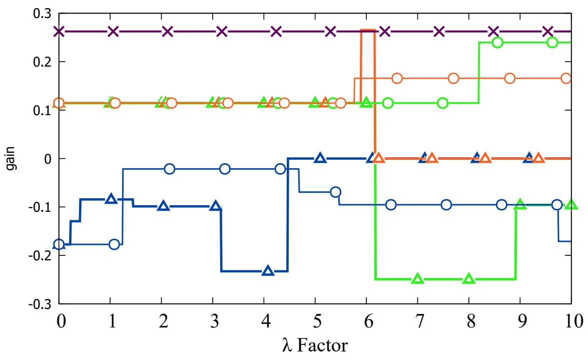
Regarding the C-Eval, in Figure 6.17, Figure 6.18, and Figure 6.19 we can



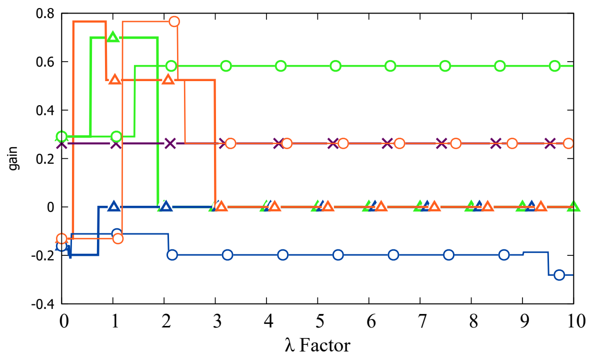
(a) Source: HBP, Target: HSL



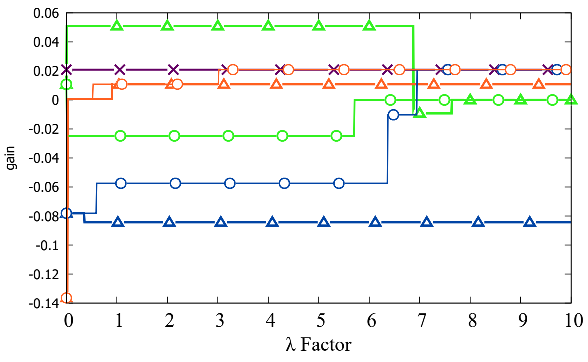
(b) Source: HSL, Target: HBP

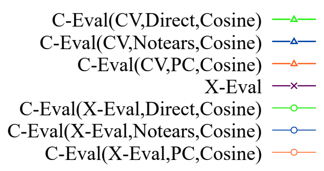


(c) Source: Geriatrics, Target: Neurology



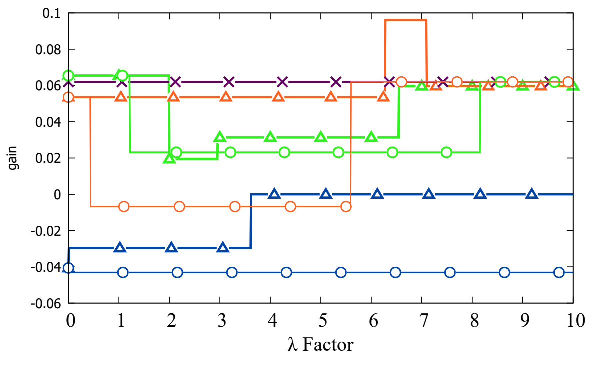
(d) Source: Neurology, Target: Geriatrics



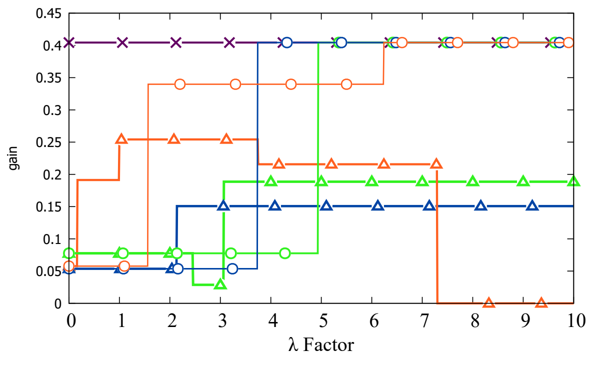


(e) Source: *Students2008*, Target: *Students2014*

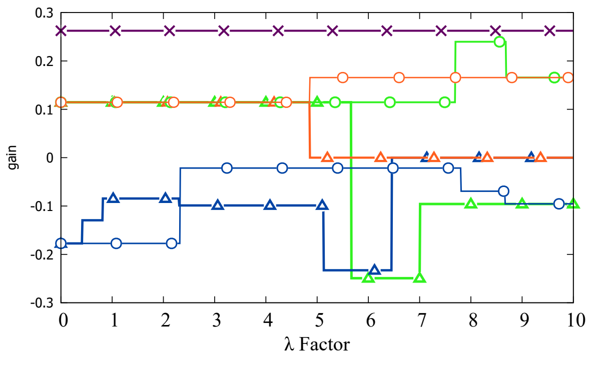
Figure 6.17: Gain of the X-Eval and the C-Eval, with the **Cosine similarity function**, compared to the CV in terms of achieved performance in the feature selection experiment. The graphs show the results for different values of the λ factor. C-Eval($\mathcal{C}$, s) denotes the C-Eval with the causal discovery algorithm $\mathcal{C}$ and the similarity function s.



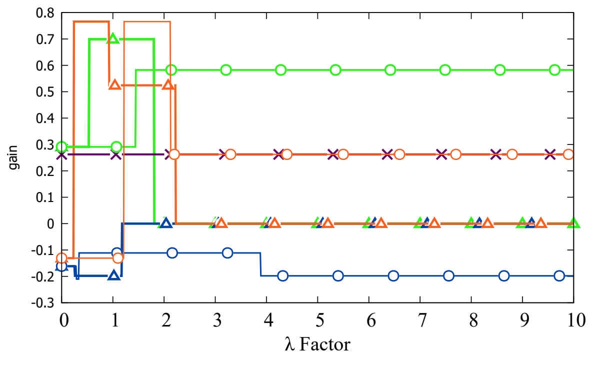
(a) Source: HBP, Target: HSL



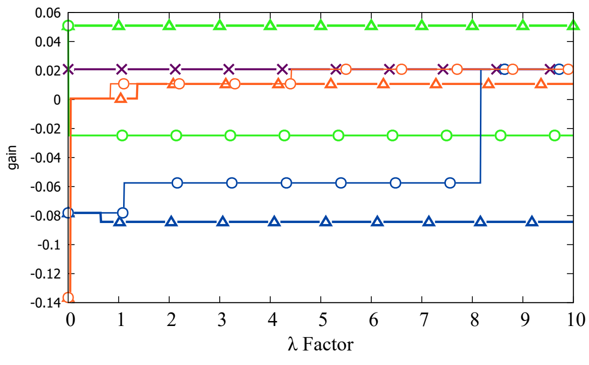
(b) Source: HSL, Target: HBP

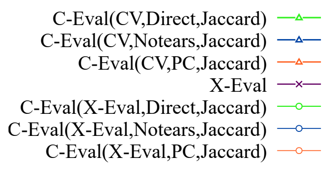


(c) Source: Geriatrics, Target: Neurology



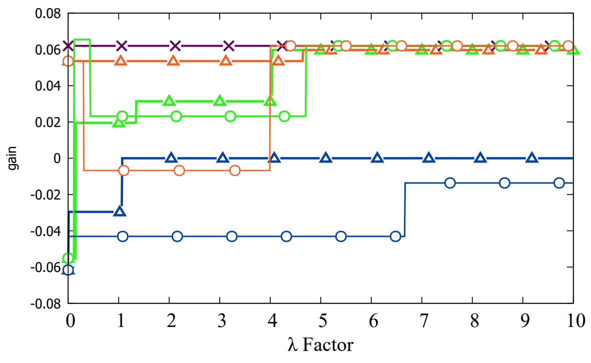
(d) Source: Neurology, Target: Geriatrics



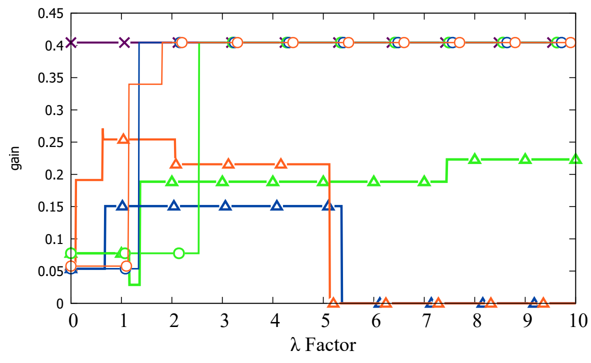


(e) Source: *Students2008*, Target: *Students2014*

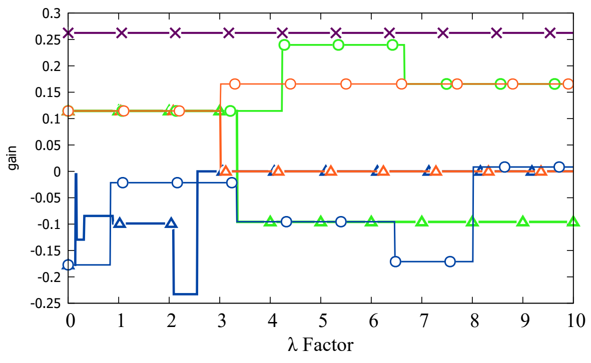
Figure 6.18: Gain of the X-Eval and the C-Eval, with the **Jaccard similarity function**, compared to the CV in terms of achieved performance in the feature selection experiment. The graphs show the results for different values of the λ factor. C-Eval($\mathcal{C}$, s) denotes the C-Eval with the causal discovery algorithm $\mathcal{C}$ and the similarity function s.



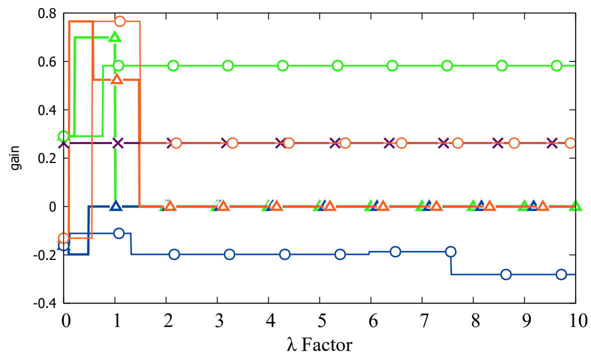
(a) Source: HBP, Target: HSL



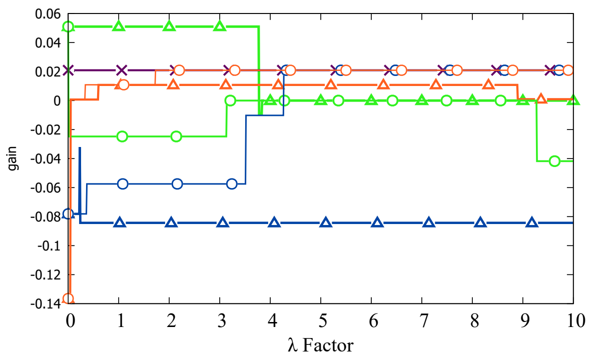
(b) Source: HSL, Target: HBP

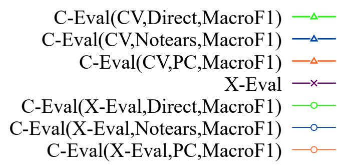


(c) Source: Geriatrics, Target: Neurology



(d) Source: Neurology, Target: Geriatrics





(e) Source: *Students2008*, Target: *Students2014*

Figure 6.19: Gain of the X-Eval and the C-Eval, with the **Macro F1 similarity function**, compared to the CV in terms of achieved performance in the feature selection experiment. The graphs show the results for different values of the λ factor. C-Eval($\mathcal{C}$, s) denotes the C-Eval with the causal discovery algorithm $\mathcal{C}$ and the similarity function s.

observe that C-Eval using DirectiLinGAM and PC causality inference algorithms is better than or equal to CV in most values of the λ factor ≥ 1, independent of the used similarity function, and methodology to estimate the model performance (i.e., CV or X-Eval). For instance, C-Eval with Cosine similarity, DirectLinGAM and λ factor ≤ 6 showed gains on the COVID-19 and Alzheimer datasets in all analyzed cases, with improvements up to 70% (Figure 6.17d). For the School Dropout dataset (Figure 6.17e): (1) C-Eval (with DirectLinGAM and Cosine) regularizing the CV estimate always achieved gains compared to CV with λ factors ≤ 7; (2) C-Eval (with DirectLinGAM and Cosine) regularizing the X-Eval estimate achieved the same performance as CV for λ factors ≥ 6 (Figure 6.17e). Besides, the C-Eval with PC causality inference algorithm showed gains in all analyzed cases with a λ factor greater than 0, reaching up to 77% improvement (Figure 6.17d). However, C-Eval with the Notears causality inference algorithm failed to demonstrate consistent gains compared to CV across most cases. This is attributed to the Notears tendency to infer a large number of causal relationships (as Section 6.1.2 demonstrated). Consequently, the C-Eval with the Notears did not apply adequate penalties to the estimated performance during regularization. From Figure 6.17 we can highlight that the C-Eval with the Cosine similarity consistently showed gain over CV (i.e., the C-Eval was always better or equal to the standard CV) with the settings:

- λ factor between 1 and 2, Cosine, CV and the DirectLinGAM algorithm;
- λ factor between 1 and 2, Cosine, CV and the PC algorithm;
- λ factor ≥ 6, Cosine, X-Eval, and the DirectLinGAM algorithm.

The C-Eval with Jaccard and Macro F1 similarities demonstrated similar results to Cosine similarity, with gains up to 77% (Figures 6.18d and Figure 6.19d). This is because the ranking of the models remains similar with both similarity functions.

Overall, we achieved the best results, in terms of gain in achieved performance with the proposed feature selection process, with the C-Eval regularizing the X-Eval with λ factor between 1 and 3, Macro F1 similarity, and the PC algorithm (Figure 6.19). This is because the C-Eval with these settings was always better than the standard CV and achieved a gain of up to 77% (e.g., Figure 6.19d).

These results related to the gain in the feature selection were expected because, as shown in Figures 6.14, 6.15, and 6.16, the proposed approaches approximated more to the true performance than the CV. Thus, these results indicate that the X-Eval and the C-Eval approaches can be more effective than the CV for feature selection on unlabeled data with an unknown distribution. In other words, the section findings demonstrate that the feature and model selection can be applications of the proposed AutoEval approaches.

Next, we characterize the results of the evaluation presented in this section analyzing the distribution of the metrics used to estimate the model performance.

6.3.1 Analysis of the Distributions of Used Metrics in Model Performance Estimation

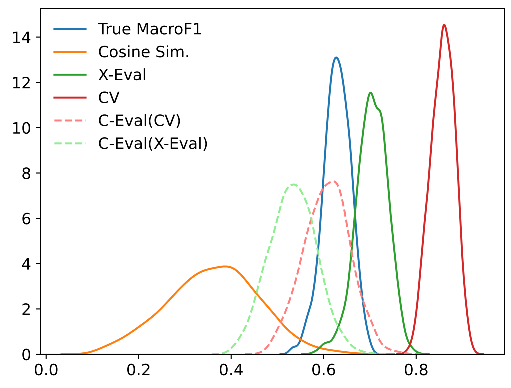
To better understand our results, this section characterizes the distributions of the true performance, similarity between the causality graphs, CV estimate, X-Eval estimate, and C-Eval estimate, on the proposed feature selection process. Figures 6.20, 6.21, 6.22 6.23, 6.24, 6.25, 6.26, 6.27, and 6.28 show the distribution of the analyzed metrics considering the 1,000 generated feature subsets in each analyzed similarity function and causality inference algorithm.

First, we highlight that when we compare the X-Eval estimate and the CV estimate distributions, the X-Eval is closer to the true performance distribution (Figure 6.20). This is because the X-Eval estimate is more precise than the CV estimate, as evidenced in the previous section (i.e., Section 6.3).

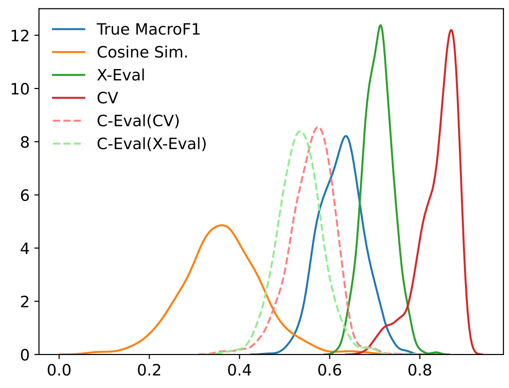
Furthermore, the Macro F1 similarity is the evaluated similarity function that the distribution more approximated to the true performance distribution (Figure 6.22, Figure 6.25, and Figure 6.28). This explains why the C-Eval achieved the best results with the Macro F1 similarity (Figure 6.16).

We can observe that the distributions presented a similar behavior in the COVID-19 and Alzheimer datasets (e.g., Figure 6.20a, Figure 6.20b, Figure 6.20c, and Figure 6.20d). For instance, in Figure 6.20a the distribution of the CV estimate is concentrated in a range of values above the true performance distribution. In contrast, the distribution of the cosine similarity between the causal graphs is concentrated in a range of values below the true performance distribution. Consequently, C-Eval regularization decreases the CV estimate, bringing the CV estimate distribution closer to the true performance distribution. This is an expected behavior in many distribution shift scenarios, where the model performance in the target data tends to drop in relation to the estimated performance in the source dataset [31]. The same occurred with the X-Eval estimate distribution (i.e., the C-Eval approximated more the X-Eval estimate distribution to the true performance distribution). This result reinforces the applicability of the C-Eval to estimate the model performance in a target dataset with distribution shift.

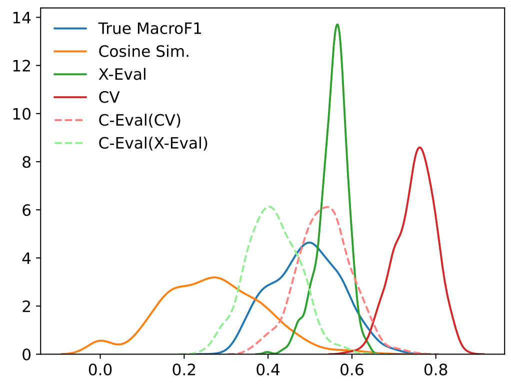
However, in the School Dropout dataset, we observed a different behavior. For the School Dropout dataset, the distributions of the estimated performance (with CV or



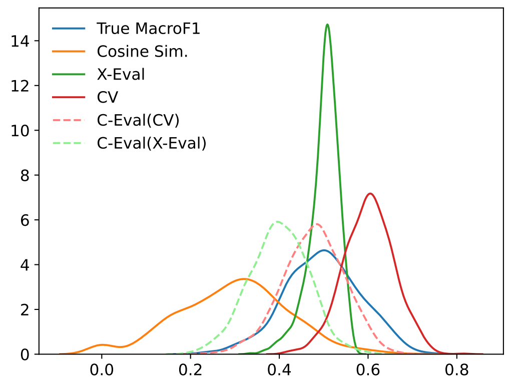
(a) Source: HBP, Target: HSL



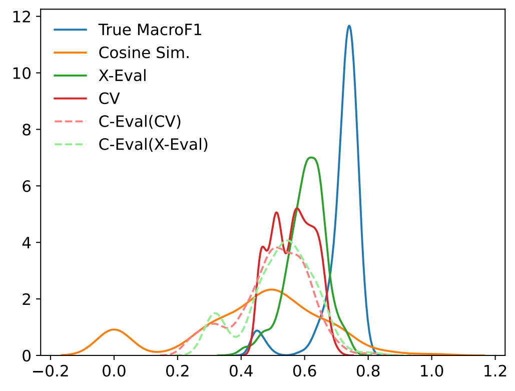
(b) Source: HSL, Target: HBP



(c) Source: Geriatrics, Target: Neurology

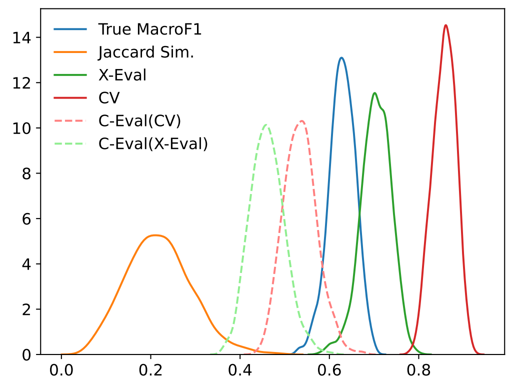


(d) Source: Neurology, Target: Geriatrics

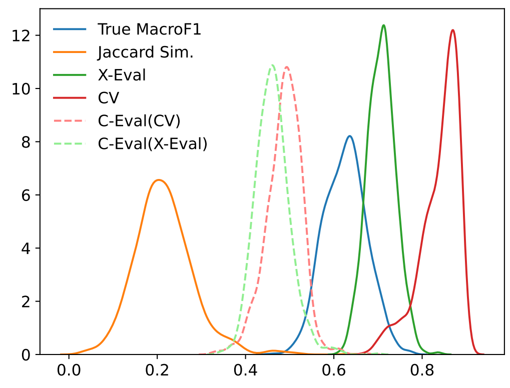


(e) Source: *Students2008*, Target: *Students2014*

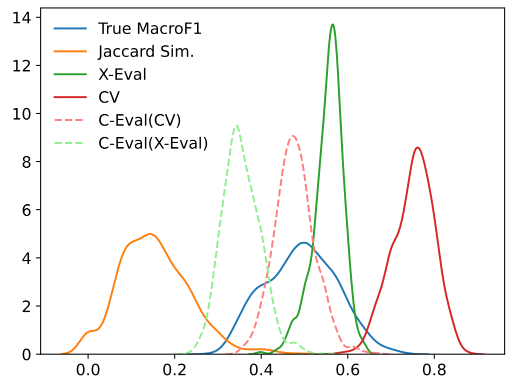
Figure 6.20: Distributions of metrics used in the experiments. C-Eval($\mathcal{V}$) denotes the C-Eval estimation with the **Cosine similarity function**, the algorithm **DirectLinGAM** and the λ factor 1, and the performance estimator $\mathcal{V}$.



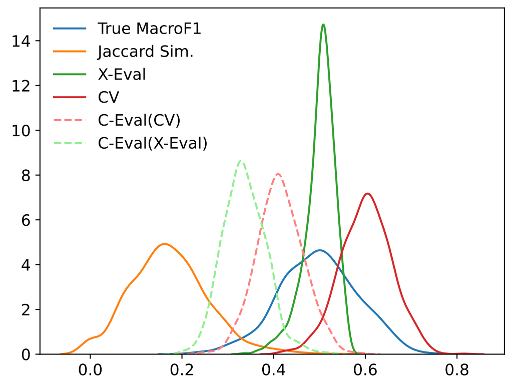
(a) Source: HBP, Target: HSL



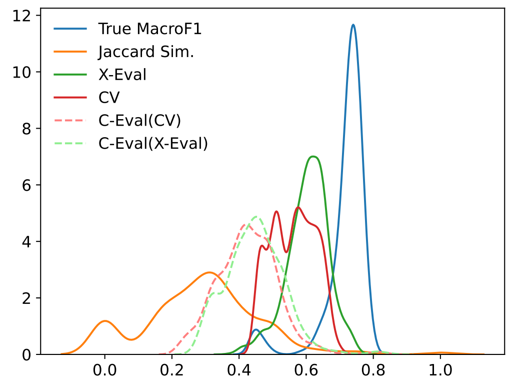
(b) Source: HSL, Target: HBP



(c) Source: Geriatrics, Target: Neurology

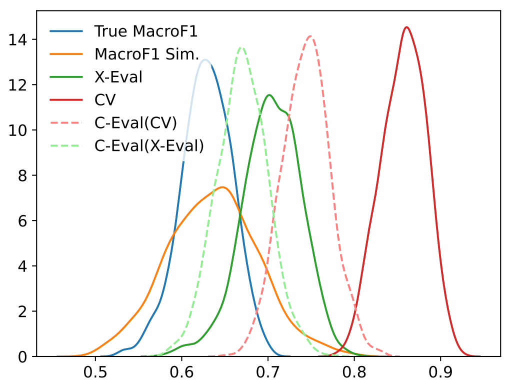


(d) Source: Neurology, Target: Geriatrics

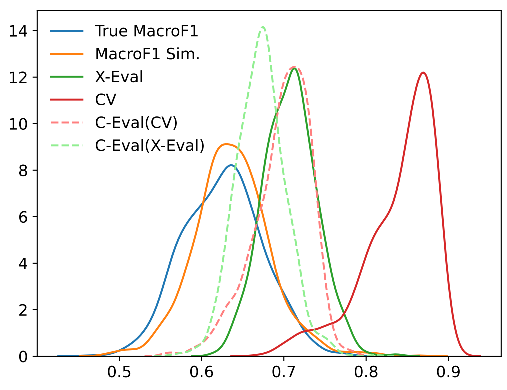


(e) Source: *Students2008*, Target: *Students2014*

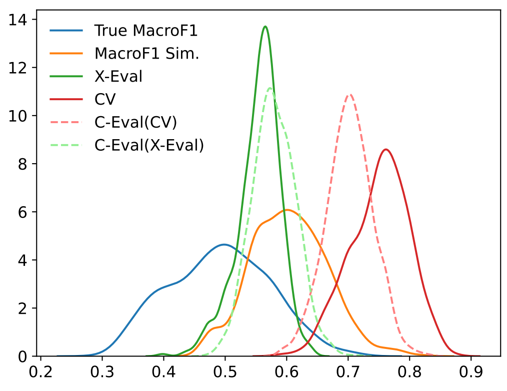
Figure 6.21: Distributions of metrics used in the experiments. C-Eval($\mathcal{V}$) denotes the C-Eval estimation with the **Jaccard similarity function**, the algorithm **DirectLinGAM** and the λ factor 1, and the performance estimator $\mathcal{V}$.



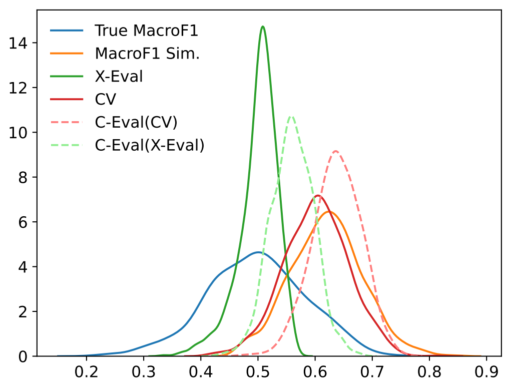
(a) Source: HBP, Target: HSL



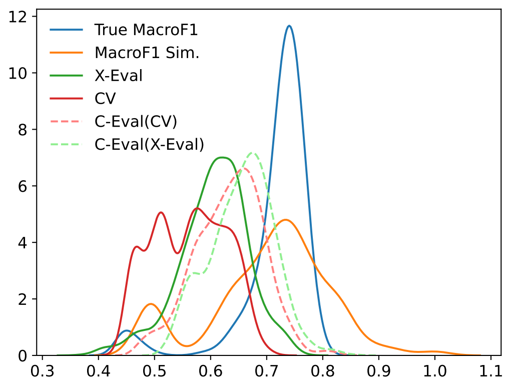
(b) Source: HSL, Target: HBP



(c) Source: Geriatrics, Target: Neurology

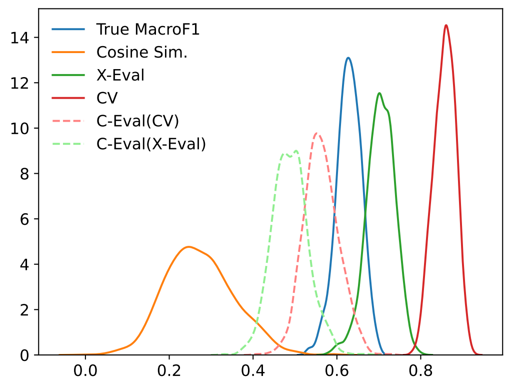


(d) Source: Neurology, Target: Geriatrics

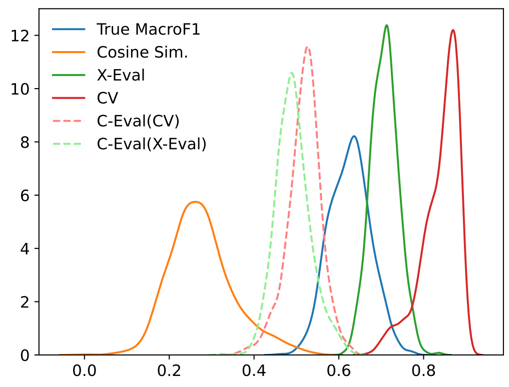


(e) Source: *Students2008*, Target: *Students2014*

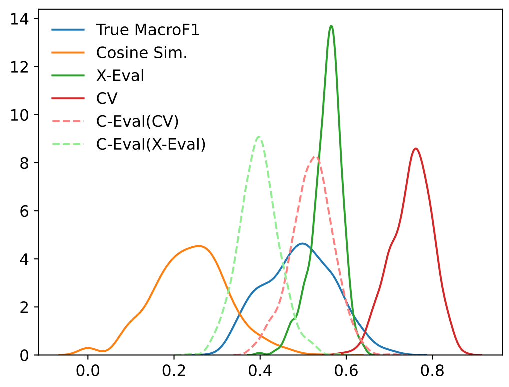
Figure 6.22: Distributions of metrics used in the experiments. C-Eval($\mathcal{V}$) denotes the C-Eval estimation with the **Macro F1 similarity function**, the algorithm **DirectLinGAM** and the λ factor 1, and the performance estimator $\mathcal{V}$.



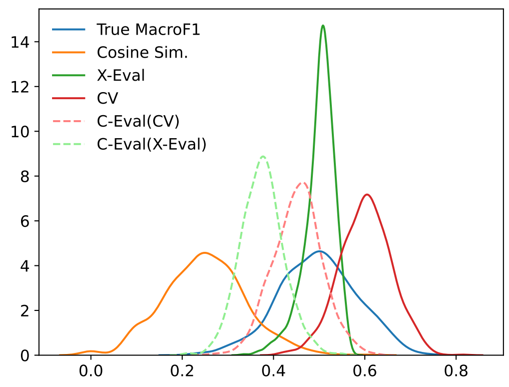
(a) Source: HBP, Target: HSL



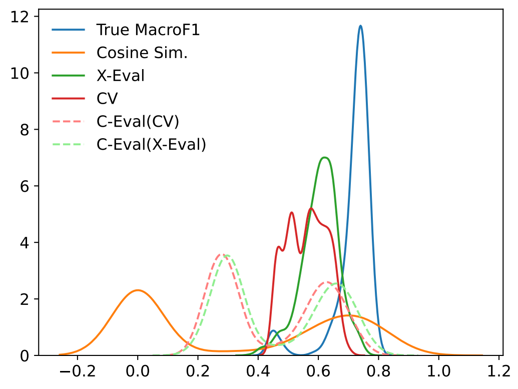
(b) Source: HSL, Target: HBP



(c) Source: Geriatrics, Target: Neurology

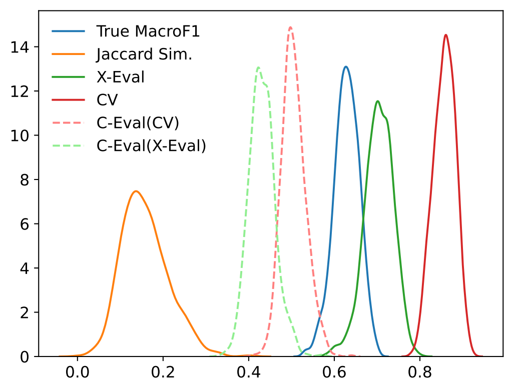


(d) Source: Neurology, Target: Geriatrics

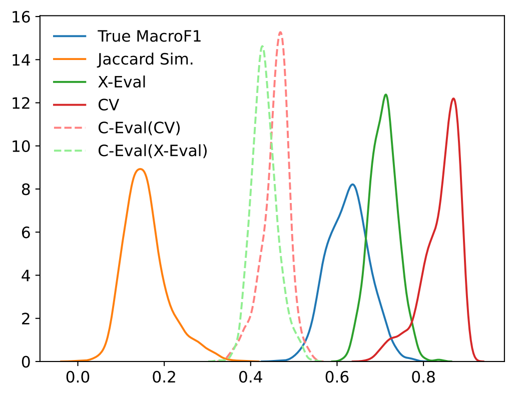


(e) Source: *Students2008*, Target: *Students2014*

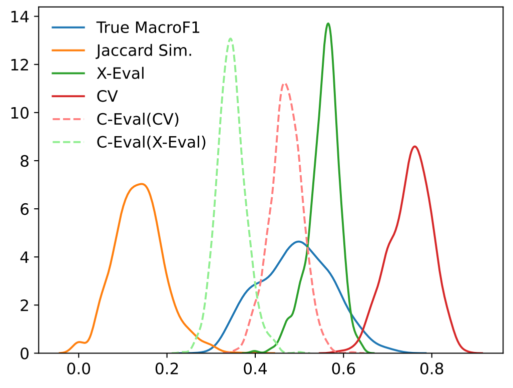
Figure 6.23: Distributions of metrics used in the experiments. C-Eval($\mathcal{V}$) denotes the C-Eval estimation with the **Cosine similarity function**, the algorithm **PC** and the λ factor 1, and the performance estimator $\mathcal{V}$.



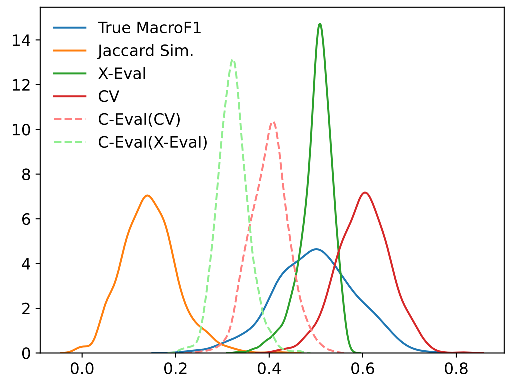
(a) Source: HBP, Target: HSL



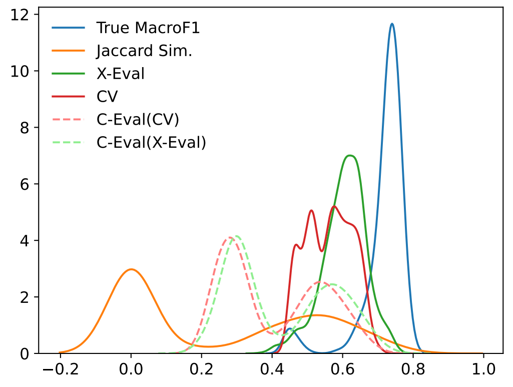
(b) Source: HSL, Target: HBP



(c) Source: Geriatrics, Target: Neurology

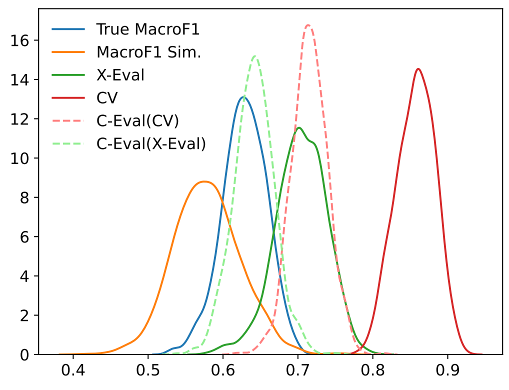


(d) Source: Neurology, Target: Geriatrics

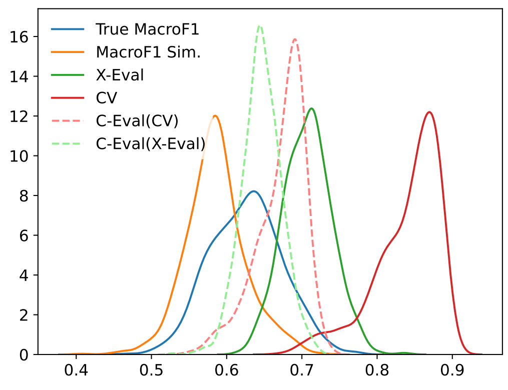


(e) Source: *Students2008*, Target: *Students2014*

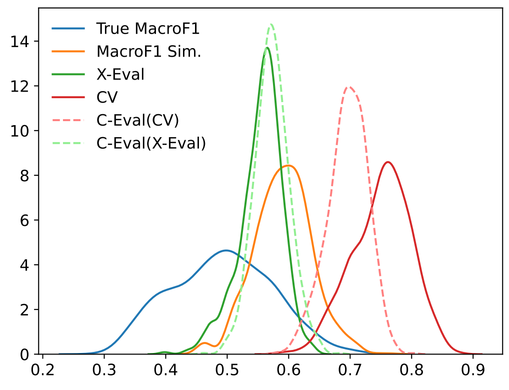
Figure 6.24: Distributions of metrics used in the experiments. C-Eval($\mathcal{V}$) denotes the C-Eval estimation with the **Jaccard similarity function**, the algorithm **PC** and the λ factor 1, and the performance estimator $\mathcal{V}$.



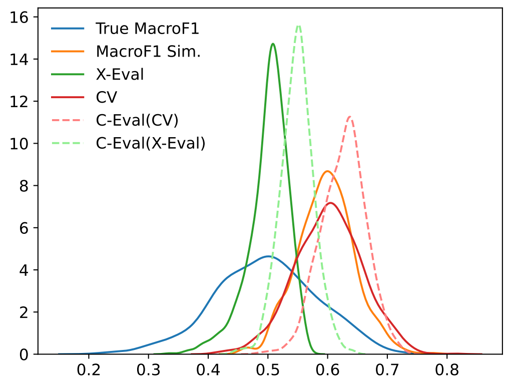
(a) Source: HBP, Target: HSL



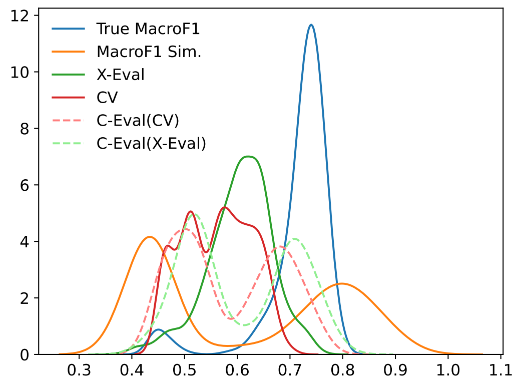
(b) Source: HSL, Target: HBP



(c) Source: Geriatrics, Target: Neurology

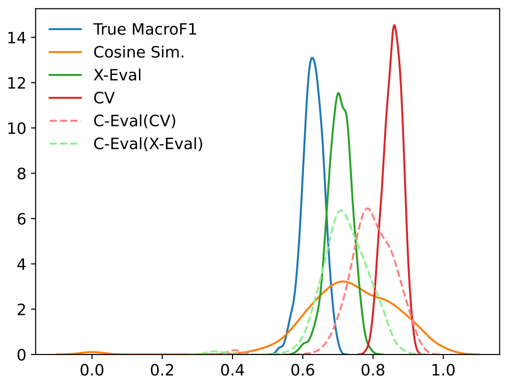


(d) Source: Neurology, Target: Geriatrics

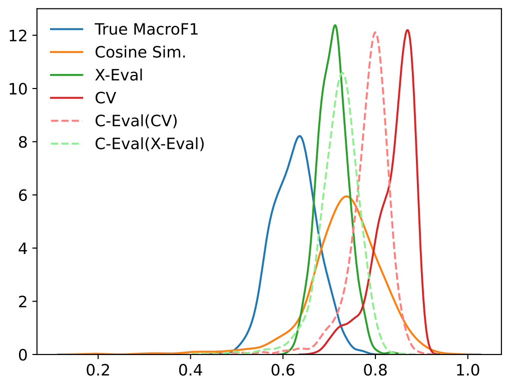


(e) Source: *Students2008*, Target: *Students2014*

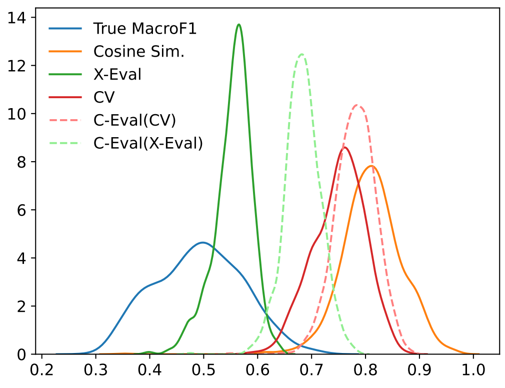
Figure 6.25: Distributions of metrics used in the experiments. C-Eval($\mathcal{V}$) denotes the C-Eval estimation with the **Macro F1 similarity function**, the algorithm **PC** and the λ factor 1, and the performance estimator $\mathcal{V}$.



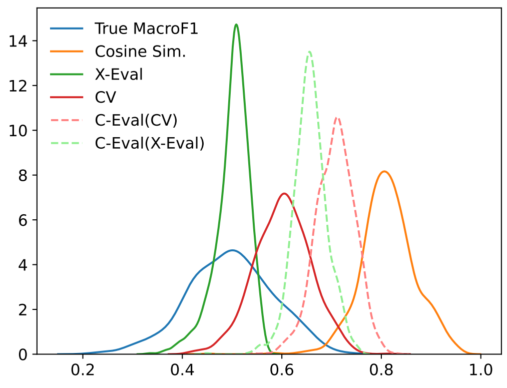
(a) Source: HBP, Target: HSL



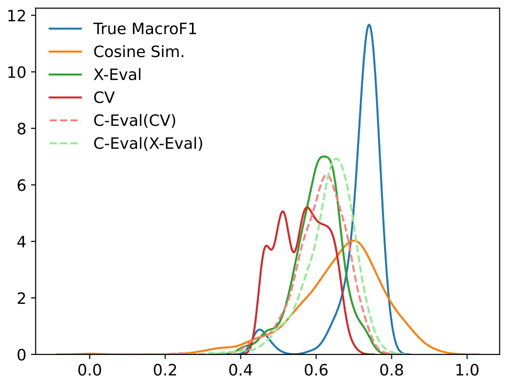
(b) Source: HSL, Target: HBP



(c) Source: Geriatrics, Target: Neurology

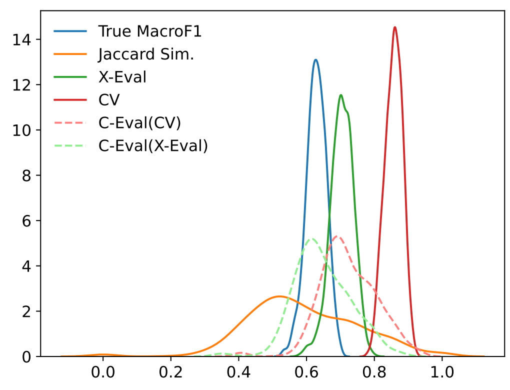


(d) Source: Neurology, Target: Geriatrics

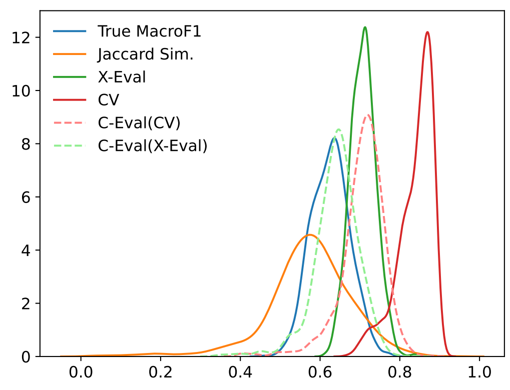


(e) Source: *Students2008*, Target: *Students2014*

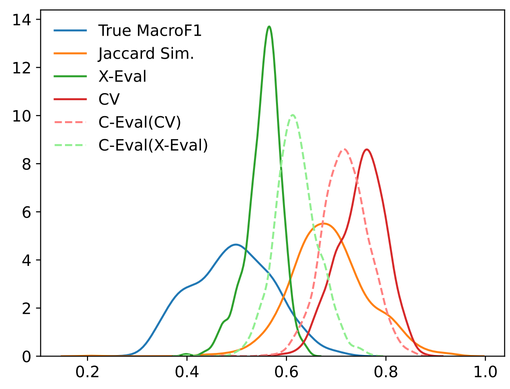
Figure 6.26: Distributions of metrics used in the experiments. C-Eval($\mathcal{V}$) denotes the C-Eval estimation with the **Cosine similarity function**, the algorithm **Notears** and the λ factor 1, and the performance estimator $\mathcal{V}$.



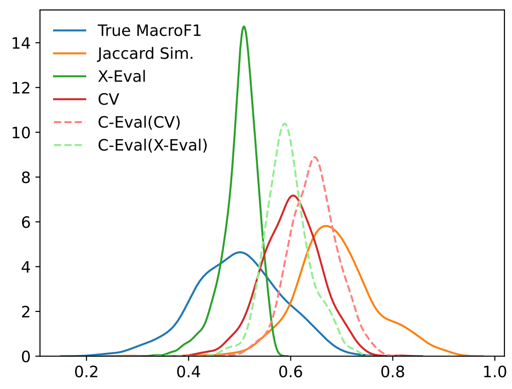
(a) Source: HBP, Target: HSL



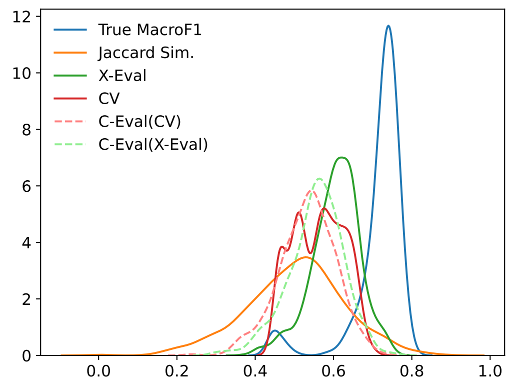
(b) Source: HSL, Target: HBP



(c) Source: Geriatrics, Target: Neurology

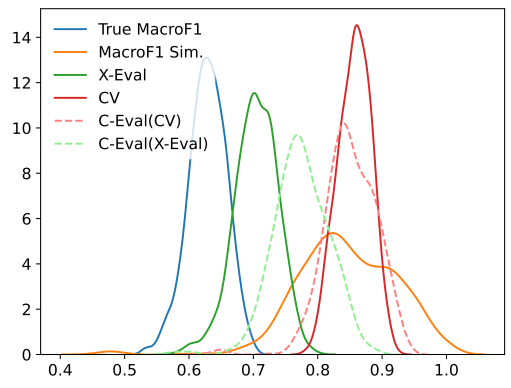


(d) Source: Neurology, Target: Geriatrics

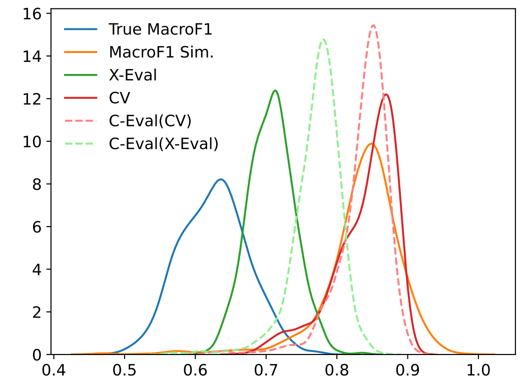


(e) Source: *Students2008*, Target: *Students2014*

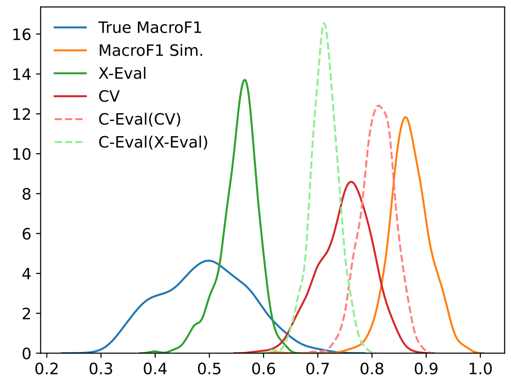
Figure 6.27: Distributions of metrics used in the experiments. C-Eval($\mathcal{V}$) denotes the C-Eval estimation with the **Jaccard similarity function**, the algorithm **Notears** and the λ factor 1, and the performance estimator $\mathcal{V}$.



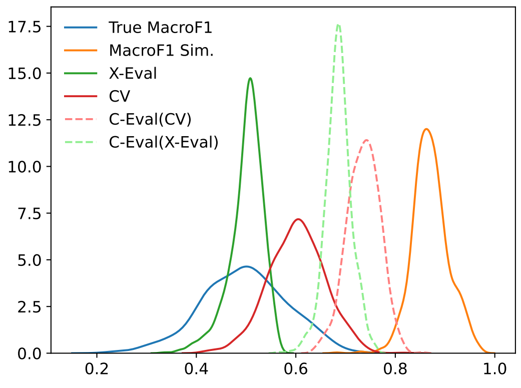
(a) Source: HBP, Target: HSL



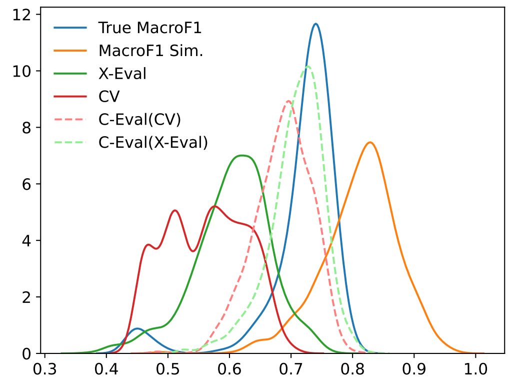
(b) Source: HSL, Target: HBP



(c) Source: Geriatrics, Target: Neurology



(d) Source: Neurology, Target: Geriatrics



(e) Source: *Students2008*, Target: *Students2014*

Figure 6.28: Distributions of metrics used in the experiments. C-Eval($\mathcal{V}$) denotes the C-Eval estimation with the **Macro F1 similarity function**, the algorithm **Notears** and the λ factor 1, and the performance estimator $\mathcal{V}$.

X-Eval) and the Cosine and Jaccard similarities between the causal graphs were concentrated below the true performance. So, the C-Eval with the DirectLinGAM and PC algorithms and similarity functions Cosine and Jaccard decreased the estimated performance (Figure 6.20e, Figure 6.21e, Figure 6.23e, and Figure 6.24e), which distanced the C-Eval distribution to the true performance distribution. Consequently, this also increased the estimation error observed in Figure 6.14e and Figure 6.15e. On the other hand, the Macro F1 similarity distribution closely approximated to the true performance distribution. This made the C-Eval regularization approximate more the distributions of the CV estimate and the X-Eval estimate to the true performance distribution (Figure 6.22e, Figure 6.25e, and Figure 6.28e). This also decreased the estimation error, as Figure 6.16e shows. In Figure 6.16e, the combination of C-Eval with Notears and Macro F1 similarity achieved the best results. We attribute this result to the Macro F1 similarity closer proximity to the true performance (Figure 6.28e) and the Notears tendency to predict a greater number of causal relationships (as Section6.1.2 demonstrated). This way, the penalization of the C-Eval with the Notears and Macro F1 similarity is smaller than the other C-Eval configurations. Thus, given the small distribution shift between the datasets about School Dropout, this configuration that causes a smaller penalization (i.e., C-Eval, with Notears and Macro F1) is more appropriate.

Additionally, still considering the School Dropout dataset, the distributions of the similarity functions Cosine (Figure 6.20e and Figure 6.23e) and Jaccard (Figure 6.21e and Figure 6.24e) showed peaks at zero when the causality inference algorithm was the PC and the DirectLiNGAM. However, the C-Eval improved the results in the School Dropout dataset with the Notears algorithm (Figure 6.16). In this case, the similarity distribution does not have peaks at zero. These peaks in the distributions also explain the superior performance of C-Eval using the Notears algorithm (that there is no peak at zero in the similarity distribution) in the School Dropout dataset (Figure 6.28e). Thus, as mentioned previously in Section 6.2, a future improvement in the C-Eval is to choose another causality inference algorithm when no causal relationships is inferred. This was observed with the PC algorithm, which inferred zero causal relationships for the School Dropout dataset (Figure 6.5).

The analyses realized in this section reinforce the observations made in the previous section, which indicated the X-Eval and C-Eval effectiveness in improving the precision in the model performance estimation. Next, we present the experimental evaluation of the proposed methods using synthetic data. This experimental evaluation aims to understand the behavior of the proposed methods across different types and intensities of distribution shifts.

Chapter 7

Experimental Evaluation with Synthetic Data

In this section, we evaluated the proposed methods using synthetic data. This experiment aims to analyze the behavior of the proposed method across different types and intensities of distribution shifts. We employed three approaches to generate synthetic data: (1) we simulated covariate shift by splitting a real-world dataset in source and target dataset based on the age feature present in the data; (2) we generated semi-synthetic data by shuffling subsets of feature values of a real-world dataset; (3) we created a full synthetic dataset and we shuffled specific portions of the feature values correlated to the labels. For a detailed description of the main experimental decisions, refer to Chapter 5. Next, we will describe how we generated and conducted the experiments with each type of synthetic data.

7.1 Experiments with Semi-Synthetic Data

In our experiments with semi-synthetic data, we chose a real-world dataset regarding the Polycystic Ovary Syndrome (PCOS) disease, which was introduced by Silva et al. [91]. We selected this dataset because we needed an easily separable dataset (i.e., a dataset in which the model has high performance) to analyze the effects of distribution shifts in the model performance (i.e., the degradation of the model performance over the distribution shift). Besides, the PCOS dataset is interesting because it deals with one of the most frequent endocrinopathy among women of reproductive age [91].

The PCOS dataset has 145 examples, 14 features, and two classes. Besides, the classes are balanced (i.e., with 49,66% of examples of the positive class and 50,34% in the negative class). The class indicates if the woman has PCOS or not.

7.1.1 Experiments with Covariate Shift Simulation

In this experiment, we split the PCOS data into source and target datasets based on the age feature. So, this way, we intend to simulate covariate shift (i.e., when there is a change in $p(\mathcal{X})$) in a similar way that Ginsberg [32] presented. For example, a model was built from data provided by a hospital that received people of a range of ages. However, the same model was applied to data from another hospital that received patients of a different range of age from the hospital in which the model was built. Specifically, for each unique age t in the PCOS dataset, we created: (1) a source dataset $\mathcal{D}^{(t)}$ with all examples with age $< t$; and (2) a target dataset $\mathcal{T}^{(t)}$ with the examples with the age $\geq t$.

Figure 7.1 shows the PCOS dataset distribution by the age feature. We used the source and target datasets created with ages (t) from 27 to 38 because the PCOS dataset has a few examples for causal graphs inference with ages less than 27 and greater than 38. Thus, we created 12 different source and target datasets with different data distributions (i.e., distribution shifts).

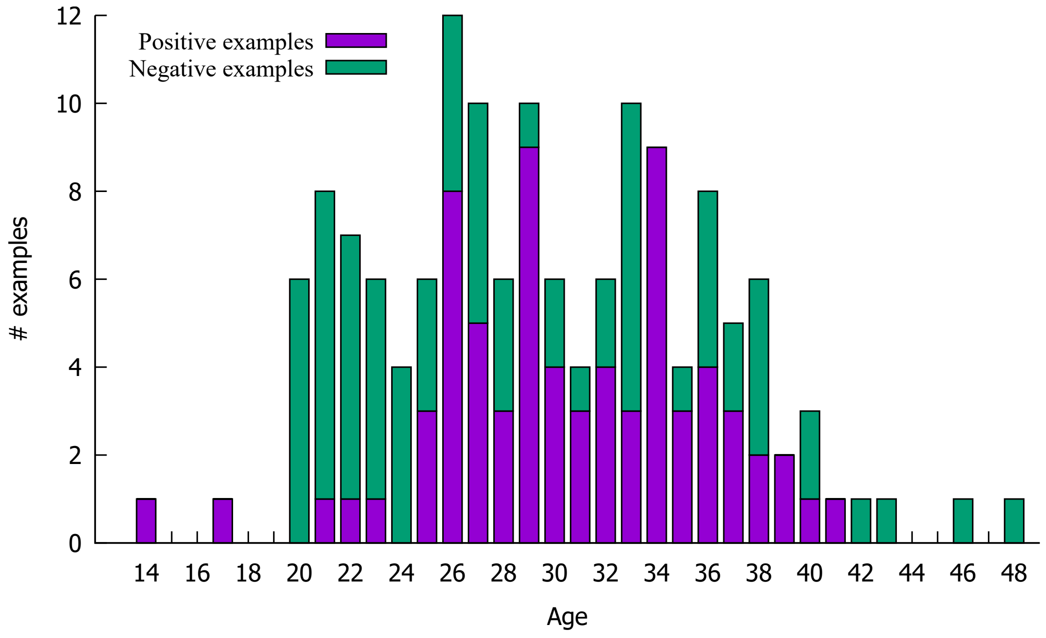
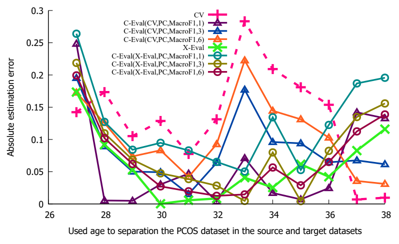


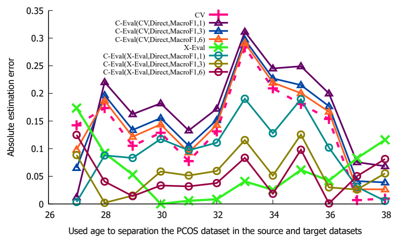
Figure 7.1: Data distribution by age in the PCOS dataset. Additionally, this figure presents the label distribution for each age.

In Figure 7.2, we can observe that the C-Eval showed more stable results with the Macro F1 similarity. These results reinforce our analysis and hypothesis made in Section 6 regarding the applicability of the Macro F1 similarity in the proposed C-Eval to AutoEval. Besides, in this experiment, the C-Eval achieved a smaller estimation error than the CV in most cases with the configurations:

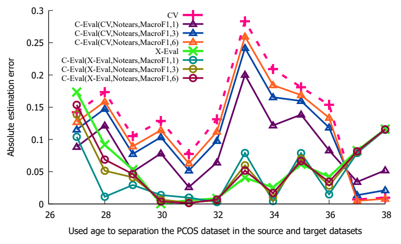
- C-Eval regularizing the X-Eval estimate with the Macro F1 similarity, independent of the other hyperparameters.
- C-Eval with the PC causality inference algorithm and Macro F1 similarity, independent of the other hyperparameters.
- C-Eval regularizing the CV estimate with the Macro F1 similarity, and the PC or Notears causality inference algorithms, independent of the used λ factor.



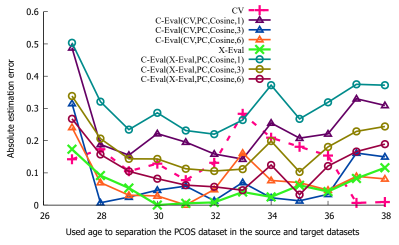
(a) PC/MacroF1



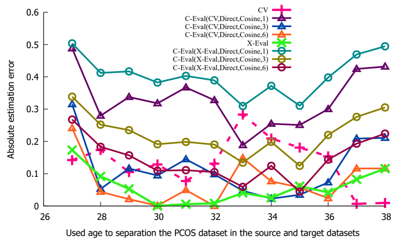
(b) DirectLiNGAM/MacroF1



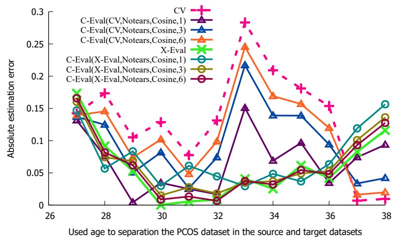
(c) Notears/MacroF1



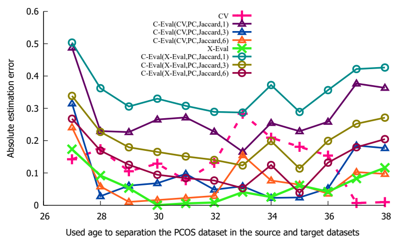
(d) PC/Cosine



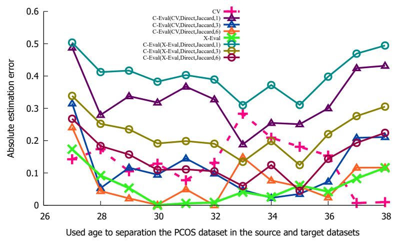
(e) DirectLiNGAM/Cosine



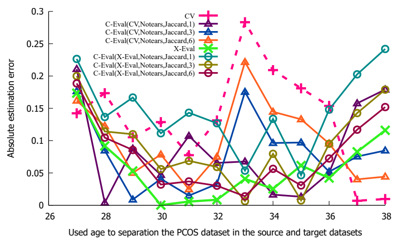
(f) Notears/Cosine



(g) PC/Jaccard



(h) DirectLiNGAM/Jaccard



(i) Notears/Jaccard

Figure 7.2: Absolute estimation error on covariate shift simulation by splitting the PCOS dataset by age. Each sub-figure presents the name of the algorithm used for causal graph inference in the legend.

Figure 7.2 shows that the X-Eval achieved a smaller estimation error than the CV in most cases. We attribute these results to the variation of the used metrics in the X-Eval as Figure 7.3 and Figure 7.4 show. Figure 7.3 shows the variation of the average impact of each feature in the model. Additionally, Figure 7.4 shows the variation of the average prediction confidence and agreement percentage over the created training and test dataset split by age. Thus, the X-Eval used this variation to detect patterns of correct and incorrect prediction and estimate the model performance in a closer approximated way than the CV.

These results indicate the applicability of the proposed X-Eval and C-Eval approaches for approximating the model generalization in covariate shift scenarios. Next, we present the experiments simulating distribution shifts caused by noisy data.

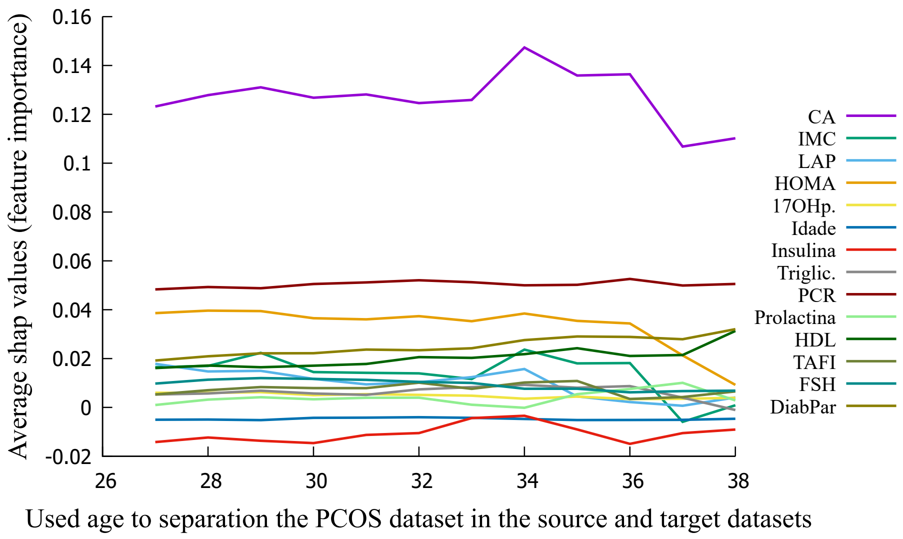


Figure 7.3: This figure shows the average SHAP values (feature importance) over covariate shift simulation by splitting the data by age.

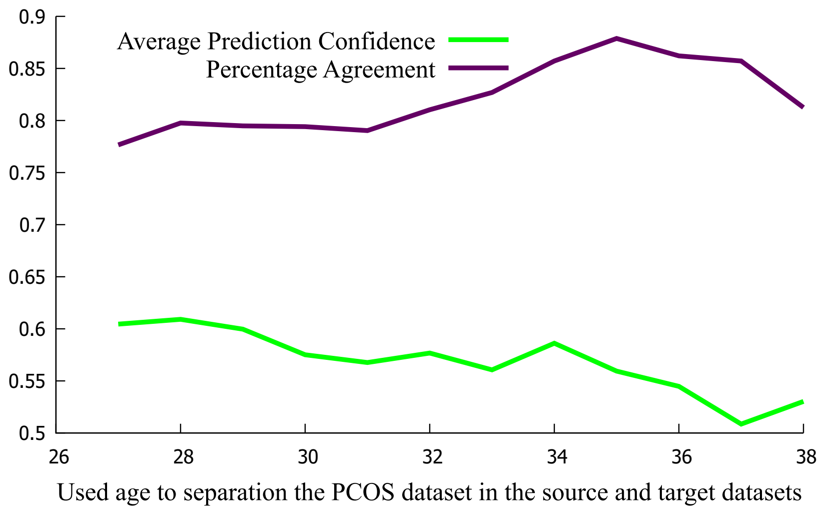


Figure 7.4: Average prediction confidence (standard deviation between 0.34 and 0.36) and percentage of agreement over covariate shift simulation by splitting the data by ages.

7.1.2 Experiments Simulating Distribution Shift Caused by Noisy Data in Semi-Synthetic Data

In this experiment, we used the PCOS dataset. Figure 7.5 illustrates the process of creating the test sets for this experiment. First, we split the dataset into a training (70% of the data) and a test set (30% of the data). Next, we ranked the features by importance using the SHAP values [58, 57]. Then, we started shuffling 10% of the feature values in the test set and we created a new test set with this percent of values shuffled. We repeated this process increasing the percentage of the shuffled values in 10% until we achieved 100% of this feature values shuffled. Then, we continued the process with the next feature in the rank and we repeated this process of shuffling of the feature values. At the end of the process, we created 140 test sets, each test set with a percentage of shuffled feature values and the last test set has all the feature values shuffled. Introducing noise into the data is a common way to simulate distribution shift in the classification task [25, 31, 111].

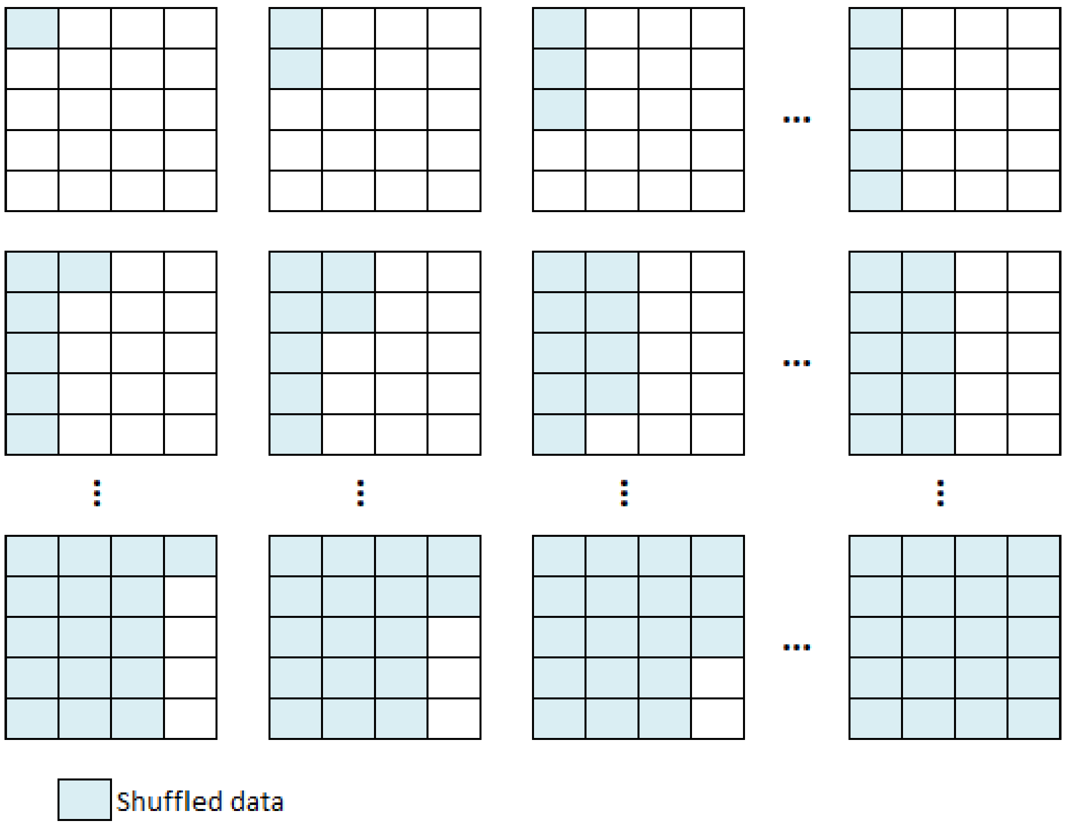


Figure 7.5: Illustrations of the process to create the synthetic data shuffling the feature values. The grids represent the datasets, each column represents a feature, and a cell represents a set of feature values.

Specifically, to generate the semi-synthetic data, given a dataset $\mathcal{D} =< X, y >$, where $X = \{x_1, .., x_n\}$ and x_i is an array of feature $F = \{f_1, .., f_m\}$ and y is an array with the labels associated with each x_i. We split $\mathcal{D}$ into two subsets, training $\mathcal{D}_{train}$ and test $\mathcal{D}_{test}$. Considering a function $GenSynt(\mathcal{D}, z, f_i)$ to generate a synthetic dataset by shuffling $z\%$ of the feature f_i, where $z \in \{0.1, 0.2, .., 0.9, 1.0\}$ and $f_i \in F$. So, we created $\mathcal{D}_{test,i} = GenSynt(\mathcal{D}_{test}, z_i, f_i) \forall z_i \in Z \; \forall f_i \in F$.

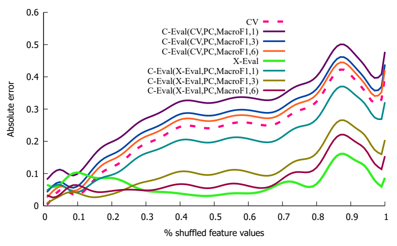
Next, we built a model $\mathcal{M}$ from $\mathcal{D}_{train}$ and we estimated the model performance of the $\mathcal{M}$ in each $\mathcal{D}_{test,i}$ using the 10-fold cross-validation (CV), and the proposed AutoEval approaches (X-Eval and C-Eval). We did experiments with the C-Eval with: (1) the causality inference algorithms discovery PC, DirectLiNGAM, and Notears; (2) the λ factors 1, 3, and 6; and (3) the similarity functions Cosine, Jaccard, and Macro F1.

We adopted the strategy of shuffling data because this way the feature values are possible values. For example, if we increase the feature values by a constant, we could create an impossible example (e.g., a person aged 150). Besides, this strategy is used in algorithms such as SHAP values [58, 57]. We employed SHAP values to rank the features because some works have shown that this method can achieve better results than other commonly used algorithms for feature selection [109, 64, 53].

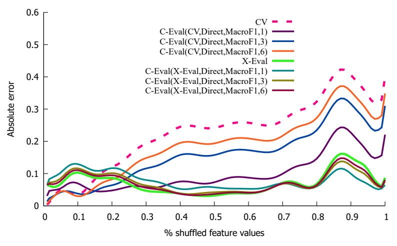
Figure 7.6 shows the results of the experiments with the generated synthetic data. This figure shows the absolute error between the estimated and the true performance as the percentage of shuffled feature values increases. We compared the absolute estimation error (i.e., *—truePerformance-estimatedPerformance—*) of the proposed approaches with the 10-fold cross-validation (CV) absolute estimation error. We can observe that as the rate of the shuffled feature values increases, the estimation error of the proposed approaches tends to be smaller than the CV estimation error in all cases.

Besides, one can see the C-Eval results with the Macro F1 similarity were more stable than the C-Eval results with the other similarity functions (Figure 7.6a, Figure 7.6b, and Figure 7.6c). As mentioned previously, this is because the Macro F1 similarity approximated more to the true performance, and we attribute this result for: (1) the Macro F1 was the same metric used to measure the model performance, and (2) the Macro F1 recommendation to measure similarity in imbalanced data. Furthermore, Figure 7.7 shows that the Macro F1 similarity between the causal graphs is closer approximated to true performance than the other similarity functions, supporting our hypothesis.

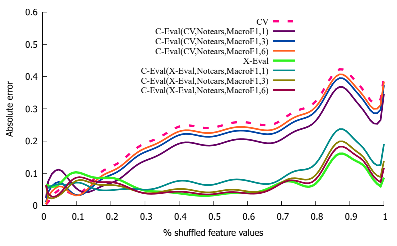
Considering the X-Eval and the C-Eval with the Macro F1 similarity (Figure 7.6a, Figure 7.6b, and Figure 7.6c), specifically when the percentage of the shuffled data was small (i.e., < 0.2), the estimation error of the proposed approaches was close to the CV estimation error, because with a small distribution shift the CV is precise to estimate the model performance. However, we consider this a positive result because this close estimation error indicates that the proposed approaches do not



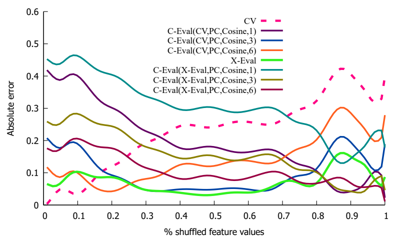
(a) PC/MacroF1



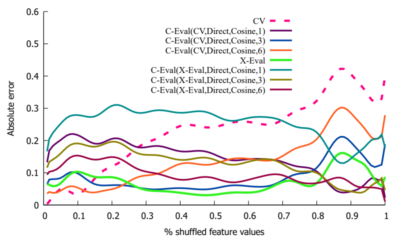
(b) DirectLiNGAM/MacroF1



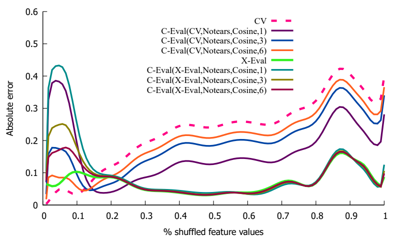
(c) Notears/MacroF1



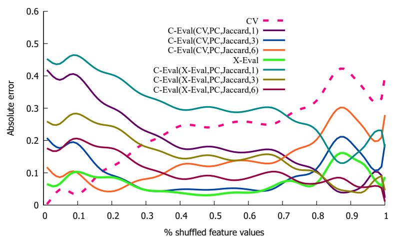
(d) PC/Cosine



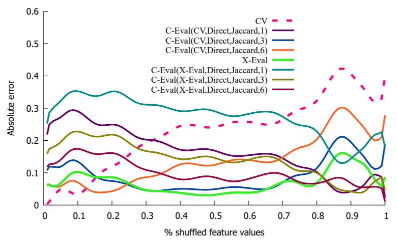
(e) DirectLiNGAM/Cosine



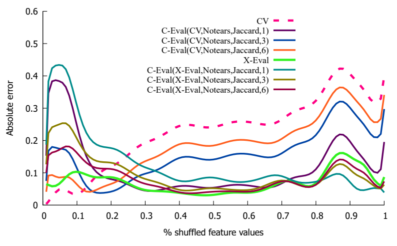
(f) Notears/Cosine



(g) PC/Jaccard



(h) DirectLiNGAM/Jaccard



(i) Notears/Jaccard

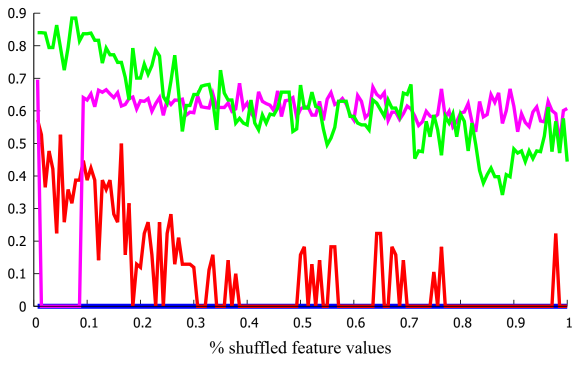
Figure 7.6: Absolute estimation error on incrementally shuffled feature values. In each subfigure, the legend shows the name of the algorithm for causal graph inference and the similarity function used in the C-Eval.

increase the estimation error achieved with the CV in scenarios with no distribution shift or with a small distribution shift. Besides, these results are another indication of the Macro F1 superiority in the evaluated distribution shift scenarios.

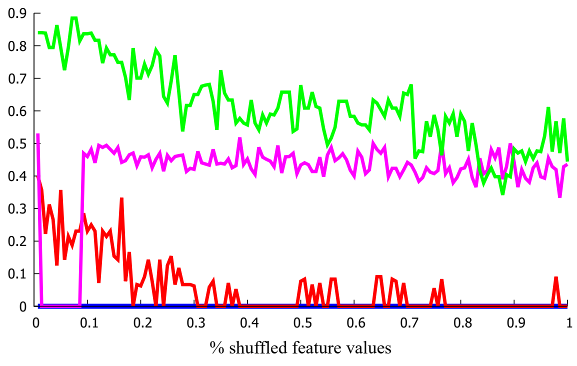
Analyzing the X-Eval results, in Figure 7.6a, we may observe that the X-Eval achieved the smallest estimation error and outperformed the CV in most cases. Besides, the X-Eval showed stable results in all series (i.e., when we increased the percentage of shuffled data). For instance, the X-Eval maximum estimation error was 0.2, whereas the CV maximum estimation error was 0.4.

An explanation for this result is the variation in the features used by the X-Eval in the estimation of the model performance (these variations can be observed in Figure 7.8 and Figure 7.9). Figure 7.8 and Figure 7.9 show that when the percentage of the shuffled data increases: (1) the average feature importance changes in different ways; and (2) the prediction confidence and agreement between the model decreases. Thus, the X-Eval uses these variations (in the feature importance, prediction confidence, and agreement) to discover patterns to classify each model prediction as correct or incorrect.

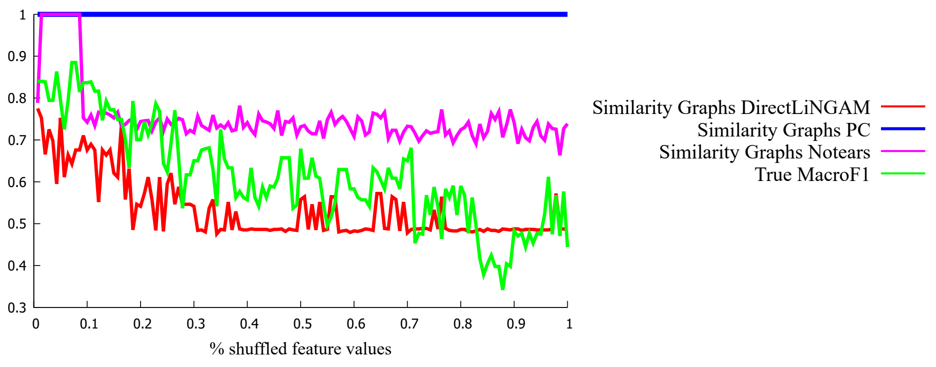
Besides, still regarding the X-Eval results, focus on the beginning of the series in Figure 7.6a, when the percentage of the shuffled data was less than 0.2, the CV was



(a) Cosine Similarity



(b) Jaccard Similarity



(c) Macro F1 Similarity

Figure 7.7: This figure shows (1) the graphs similarity by causality inference algorithm, and (2) the True Performance (Macro F1). Each sub-figure shows the name of the used similarity function in the legend.

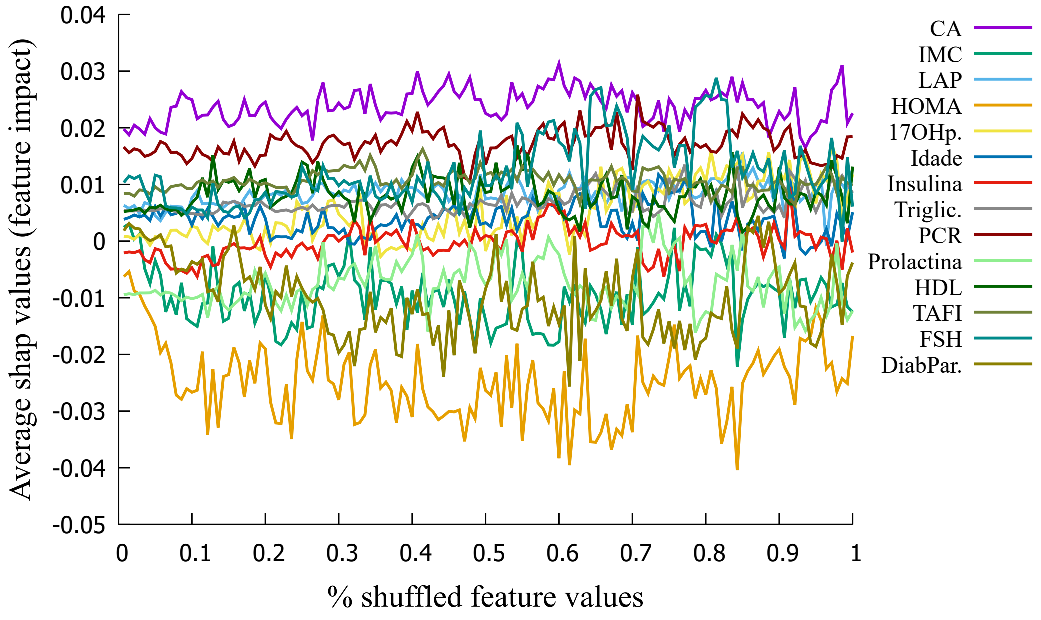


Figure 7.8: Impact of the shuffling in the feature values on the features importance. This figure shows the average SHAP values (feature importance) over different percentages of shuffled feature values in the PCOS dataset.

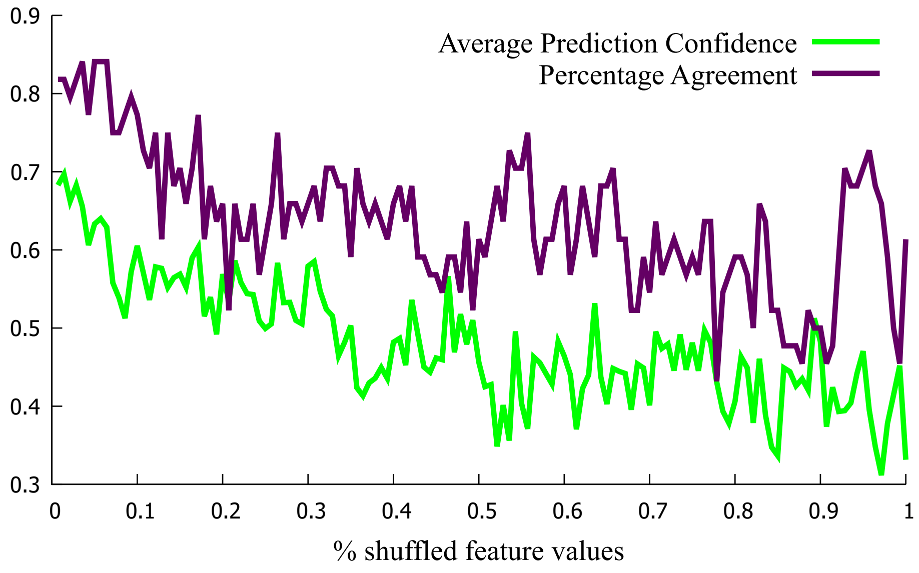


Figure 7.9: Average prediction confidence (standard deviation between 0.24 and 0.44) and percentage of agreement over different percentages of shuffled feature values on the PCOS dataset.

better than the X-Eval. This is because when there is a small distribution shift between the source and the target datasets, the CV is precise to estimate the model performance. However, we can observe that the X-Eval showed an estimation error ≤ 0.1, which can be considered relatively low compared to the highest CV estimation error in the series, which was 0.4.

On the other hand, from 10% up to 30% of the feature values shuffled the proposed approaches presented an evident lower estimation error than the CV, as Figure 7.6a, Figure 7.6b, and Figure 7.6c show. This is an interesting result because in the real-world we can consider up to 30% a reasonable percentage of noise in the data. Besides, with 30% of feature values shuffled, the error of the CV was 0.19 and the proposed approach C-Eval achieved an estimation error of up to 0.04 (Figure 7.6b). Thus, we can see a reduction of 78% of the estimation error.

In addition, the C-Eval, with Macro F1 similarity, stabilized the X-Eval estimation error where the percent of feature values shuffled was small (i.e., < 0.1), making the estimation error always smaller or close to the CV estimation error in the complete series, with the following setting:

- The C-Eval regularized the X-Eval estimate using the Macro F1 similarity function, the PC causality inference algorithm, and λ factors 3 and 6 (Figure 7.6a);
- The C-Eval regularized the X-Eval estimate using the Macro F1 similarity function, the causality inference algorithm DirectLinGAM, and λ factor 6 (Figure 7.6b).

Similarly, the C-Eval regularizing the CV estimate achieved an estimation error always smaller or close to the CV error when:

- The C-Eval regularized the CV estimate using the Macro F1 similarity function, the causality inference algorithm DirectLinGAM, and λ factor 3 (Figure 7.6b);
- The C-Eval regularized the CV estimate using the Macro F1 similarity function, the Notears causality inference algorithm, and λ factor 6.

These results indicate the applicability of the proposed methods to estimate the performance of the classification model when noise in the data causes distribution shifts. Once, the proposed methods outperformed the standard CV in the reasonable situation (i.e., with a small percentage of noised data) and hard situation (i.e., with noise in the greatest part of the data). Next, we present our experiments simulating distribution shift in full synthetic data.

7.2 Experiments Simulating Distribution Shift Caused by Noisy Data in Full Synthetic Data

In this experiment with synthetic data, we generated a completely synthetic dataset following the approach presented by [85]. Basically, in this approach, given the number of informative features k (i.e., feature correlated to the class) and the number of non-informative features j, the algorithm creates a dataset with k informative features correlated to the classes (using a cluster approach) and with j non-informative features applying noise [85]. Thus, we created a synthetic dataset with 1000 examples and 20 features, using [85] approach. However, only two features were informative (i.e., features correlated to the class) and the other features were randomly generated. Then, we split the dataset into a training (70% of the data) and a test (30% of the data) dataset. So, we simulated distribution shifts as illustrated in Figure 7.10. Thus, we started shuffling 10% of the values of the two informative features in the test set and we created a new test set considering the shuffled and non-randomized values. We repeated this process increasing the percentage of the shuffled values by 10% until we shuffled all the values of the original test set.

Specifically, we used [85] to create a dataset $\mathcal{D}^s = < \mathcal{X}, y >$, where $\mathcal{X} = \{x_1, .., x_n\}$ and x_i is an array of feature and y is an array with the labels associated with each x_i. We split $\mathcal{D}^s$ into two subsets, training $\mathcal{D}^s_{train}$ and test $\mathcal{D}^s_{test}$. Considering a function $GenSynt(\mathcal{D}, z)$ to simulate distribution shift by shuffling z% of the informative features individually (i.e., the values of a feature f_i were shuffled with the values of the

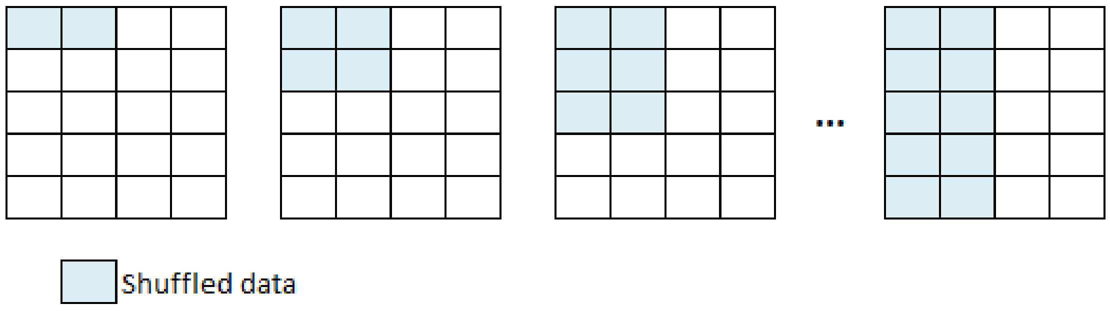


Figure 7.10: Illustrations of the process to create the synthetic data shuffling the feature values. The grids represent the datasets, each column represents a feature, and a cell represents a set of feature values. Considering that we had just two informative features in the synthetic dataset, we shuffled the values of both features simultaneously.

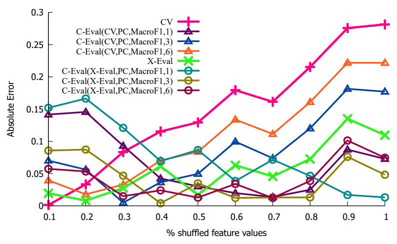
same feature f_i), where $\mathcal{Z} = \{0.1, 0.2, .., 0.9, 1\}$ and $z \in \mathcal{Z}$. So, we created $D_{test,i} = GenSynt(D, z_i) \forall z_i \in \mathcal{Z}$.

Next, we built a model $\mathcal{M}$ from $\mathcal{D}_{train}$ and we estimated the model performance of the $\mathcal{M}$ in each $\mathcal{D}_{test,i}$ using the 10-fold cross-validation (CV), the X-Eval, and the proposed C-Eval approach. When we used the C-Eval, we performed the experiments with: (1) the causal graph algorithm discovery PC, DirectLiNGAM and Notears; (2) the factors 1, 2, and 3; and (3) the similarity functions MacroF1, Cosine, and Jaccard.

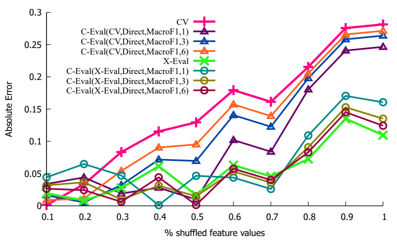
Figure 7.11 shows the results of this experiment with the generated synthetic data. This figure displays the estimation error across various percentages of shuffled data (i.e., simulations of distribution shifts). As expected, the error of the CV estimation increases when the percentage of shuffled data increases because we performed the CV estimation considering just the source dataset, which does not represent the distributions of the noisy target datasets.

In Figure 7.11, we can see that the X-Eval estimation error was consistently smaller than the CV estimation error. Besides, the X-Eval was stable with the error always less than 0.15 (e.g., Figure 7.11a). We attribute these results to the variation in the explanation metrics used in the X-Eval (specifically, the SHAP values, agreement between classifiers as Figure 7.12 and Figure 7.13 evidence). Figure 7.12 and Figure 7.13 show that when the percentage of the shuffled data increased, the average SHAP values (i.e., feature importance), the average prediction confidence and the percentage of agreement decreased.

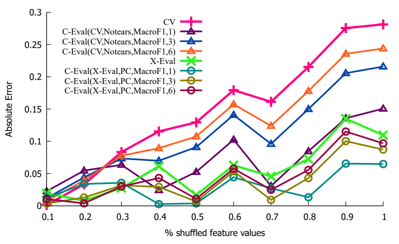
One can see, if we evaluate the C-Eval results regarding the similarity function, the C-Eval estimation error was more stable using the Macro F1 as similarity function because the C-Eval estimation error was smaller than CV estimation error in most cases, independent of the used causality learning algorithm (Figure 7.11a, Figure 7.11b, and Figure 7.11c). Figure 7.14 shows the variation of the similarity among the causal graphs over the percentage of the shuffled feature values; these variations explain these C-Eval



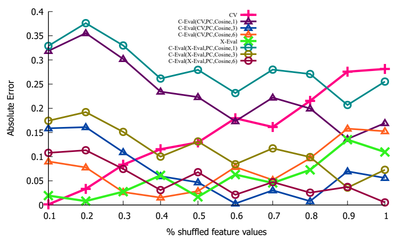
(a) PC/MacroF1



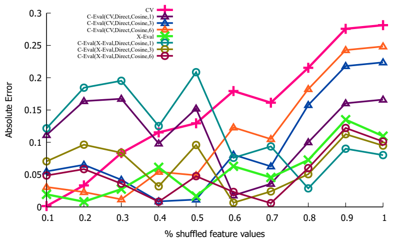
(b) DirectLiNGAM/MacroF1



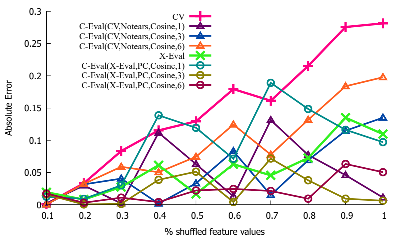
(c) Notears/MacroF1



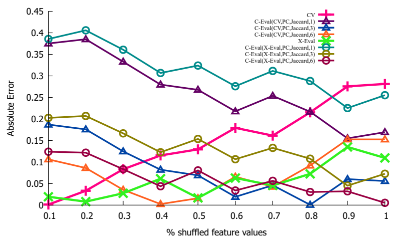
(d) PC/Cosine



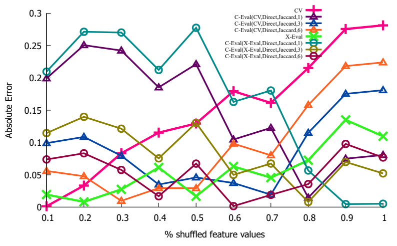
(e) DirectLiNGAM/Cosine



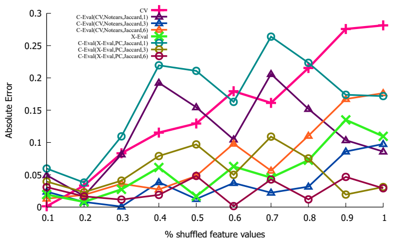
(f) Notears/Cosine



(g) PC/Jaccard



(h) DirectLiNGAM/Jaccard



(i) Notears/Jaccard

Figure 7.11: Absolute estimation error over different percentages of shuffled feature values in the full-synthetic data with just two informative features. We shuffled the two informative features values simultaneously. Each subfigure shows the name of the causality inference algorithm and the similarity function used in the C-Eval algorithm.

results. Figure 7.14 shows that the Macro F1 Similarity is closer approximated to the true performance than the other similarity functions. These results are similar to the C-Eval results presented in the previous experimental evaluation sections (i.e., Section 6, Section 7.1.2, and Section 7.1.1), which reinforces our finds.

Besides, the C-Eval decreased the X-Eval estimation error in most cases with:

- The causality inference algorithm PC with the percentage of shuffled data $> 30\%$ (Figure 7.11a).
- The DirecLinGAM up to 70% of data shuffled, which we consider reasonable that in the real world the target data present less than 70% of noise (Figure 7.11b).
- The Notears causality inference algorithm in all series (Figure 7.11c).

Specifically, the C-Eval achieved the best result with the Notears causality inference algorithm and the Cosine and Jaccard similarity functions. These approaches were stable and showed estimation errors of less than 0.1 during the experiment.

These results indicate the applicability of the proposed algorithms X-Eval and C-Eval to estimate the model performance in this kind of distribution shift. Once, our

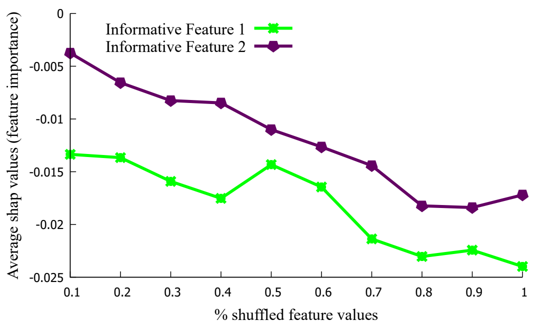


Figure 7.12: Impact of the shuffled in the feature values in the features importance. This figure shows the average SHAP values (standard deviation between 0.018 and 0.28) over different percentages of shuffled feature values in the full-synthetic data with just two informative features.

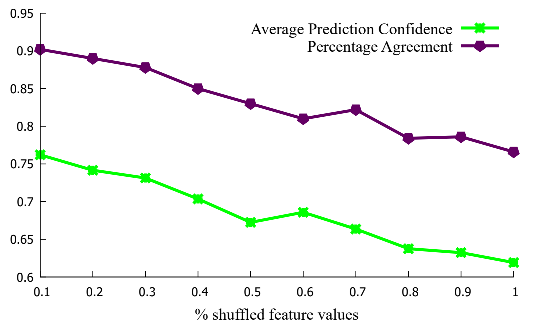
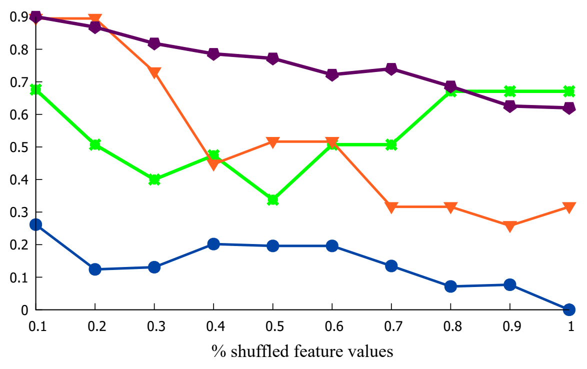
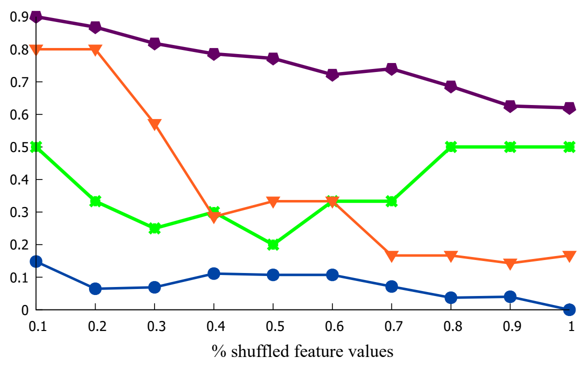


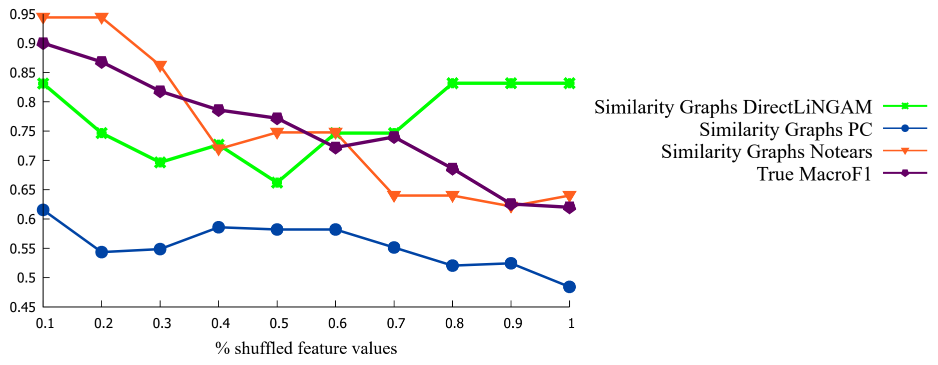
Figure 7.13: Average prediction confidence (standard deviation between 0.35 and 0.38) and percentage of agreement over different percentages of shuffled feature values in the full-synthetic data with just two informative features.



(a) Cosine Similarity



(b) Jaccard Similarity



(c) Macro F1 Similarity

Figure 7.14: This figure shows (1) the graphs similarity by causality inference algorithm, and (2) the True Performance (Macro F1) in the full-synthetic data with just two informative features. Each sub-figure shows the name of the used similarity function in the legend.

algorithms presented an estimation error consistently smaller than the standard CV estimation error. Next, we present a discussion of the results.

Chapter 8

Discussion of the Results

In Chapter 6 and Chapter 7 , we presented the evaluation of the proposed algorithms C-Eval and X-Eval with more than 5,150 models, including models built from synthetic and real-world datasets. Our experiments showed that the proposed X-Eval and C-Eval algorithms outperformed the standard 10-fold cross-validation (CV) in different distribution shift situations.

Specifically, we observed that the X-Eval consistently showed a smaller estimation error than the CV. We attribute these results to the following X-Eval characteristics:

1. **Individual Instance Judgment**: The X-Eval classifies each target instance as correct or incorrect. This strategy is essential to treat covariate shift.

2. **Label Shift Simulation**: Another key X-Eval characteristic is the simulation of changes in the label distribution (i.e., $p(y)$) in the target dataset. This strategy addresses label shift by preparing the autoEval model for different label distribution changes.

3. **Agreement as Complementary Feature**: X-Eval leverages the agreement as a complementary feature. When we combine the agreement with other features that the X-Eval uses, the X-Eval may deal with some cases of concept drift.

The C-Eval, regularizing both the CV and X-Eval estimates, outperformed the standard CV. Besides, the C-Eval improved the X-Eval estimate in most cases by closer approximating the X-Eval estimate to the true performance. The C-Eval showed better results using the Macro F1 similarity because it is the same metric used to measure the model performance since the similarities with the Macro F1 approximated more to the true performance. Besides, C-Eval with the causality inference algorithms DirectLinGAM and PC presented the best results. However, the DirectLinGAM presented an advantage over the PC because the PC predicted zero causality relationships in some datasets.

In addition, we observed that the C-Eval regularizing the CV estimate achieved better results using a λ factor between 1 and 3, which indicates that the weight of similarity between the causal graphs and the CV estimate should be near. On the other hand, C-Eval regularizing the X-Eval estimate achieved better results using the λ factor between

5 and 7 because as the X-Eval estimate closer approximated to the true performance, the X-Eval estimate must have a greater weight in the C-Eval process. In other words, the similarity between the causal graphs can influence the CV estimate more than the X-Eval estimate.

These results demonstrated the applicability of the proposed algorithms to approximate the model performance in distribution shift scenarios. Besides, the results showed the superiority of the proposed algorithms over the CV in approximating the true performance in distribution shift scenarios. Next, we discuss some limitations of our research.

8.1 Research Limitations

In this section, we discuss the limitations of our research. One limitation is the C-Eval dependence of hyperparameters. However, experimentally, we could define a set of hyperparameters indicated for the C-Eval.

In addition, the proposed algorithms C-Eval and X-Eval have a higher computational cost than the CV, as showed in the Section 4.1.7, Section 4.2.1, and evidenced by the process time analysis presented in the Appendix B. However, considering the challenging task of estimating the model performance in distribution shift scenarios, this higher computational cost can be considered less relevant in the first moment because (1) a precise model performance estimation in situations that can occur distribution shifts can bring benefits to many areas (e.g., Transfer Learning, Continual Learning, and Semi-Supervised Learning), and (2) the AutoEval approaches allow a continuous evaluation of the model in a real-world application (i.e., a deployed model). Besides, in the future, we can propose strategies to decrease the computational cost (e.g., to sample a target dataset to estimate the model performance). However, even with the processing time presented in Appendix B, applying the proposed algorithms in real-world environments is feasible because the greatest processing time was less than 20 minutes.

Finally, the small size of some datasets used to evaluate our approaches may limit the results of our experiments in terms of process time. However, in the real world, it is possible to control the size of the used target data to estimate model performance, or the size of the target dataset can be reduced by sampling the data. Besides, we used datasets of sizes commonly encountered in the healthcare area [3] and large datasets, such as those related to COVID-19. Next, we present our conclusions and future work.

Chapter 9

Conclusions and Future Work

This thesis focused on the important and underexplored problem of Automatic Model Evaluation (AutoEval) [25, 33, 111] on the tabular data context. We proposed novel AutoEval methods (i.e., the X-Eval and the C-Eval) to approximate the true performance of a classification model on unlabeled data of unknown distribution. The proposed methods benefit from the possibility of extracting explanations and causal graphs from labeled and unlabeled data.

Specifically, the proposed X-Eval method uses explanations because they represent the model from different perspectives. Thus, we investigated the applicability of these perspectives to provide information to model evaluation (i.e., to approximate the estimated performance to the true performance). Thus, we investigated the hypothesis that model representation with the explanation can diverge when there is a distribution shift between two datasets. Our experiments demonstrated these changes in the model explanations in distribution shift scenarios. Besides, this becomes more evident in the experiments with synthetic data.

Regarding the C-Eval, from the concept of causality relationships are stronger relations than the correlation, and that causal relations should not change from one dataset to another, we investigated the hypotheses that: (1) changes in the causal graphs of the source and target datasets indicate distribution shifts and (2) this difference can be used to regularize the estimated model performance with the 10-fold cross-validation (CV) or X-Eval.

We evaluated the proposed method in terms of gap between estimation and observed true performance. We also showed the application of the AutoEval approaches to model selection from a case study on feature selection. In this case study we proposed a simple process to feature selection with the estimations of proposed approach on unlabeled data of unknown distribution.

The proposed algorithms outperformed the standard 10-fold cross-validation (CV) in all experiments. Regarding the gap between estimated and true performance, the proposed methods showed a reduction of up to 100% in the error estimation when compared with the standard CV approach. Regarding feature selection, our proposed method achieved gains up to 76% in comparison to the CV. These results indicate that the proposed method can be used to improve the model performance estimation and model selection (e.g., hyperparameter tuning, and feature selection) in scenarios where distribution shift can occur.

In the future, the proposed algorithms can be studied in application to constant monitoring of the model performance in environments in which the distribution of future data can change [61]. In this sense, our results can benefit research areas (e.g., Semi-Supervised Learning, Ensemble Learning, Misclassification detection, Active Learning, and Continual Learning) and industrial processes that use machine learning models for automatic data classification in distribution shift environments (e.g., automatic classification in data stream scenarios). Besides, the proposed methods can be easily applied or adapted to Computer Vision and Natural Language Processing problems. Additionally, the proposed approach to compute the similarity between datasets can be used in another application to estimate distribution difference [25] and distance between dataset (e.g., for datasets clustering).

References

[1] Exame invest. https://exame.com/invest/minhas-financas/preco-do-carro-novo-no-brasil-aumentou-90-em-cinco-anos/, 2023. Accessed: 2023-11-07.

[2] Tiago Alves, Alberto Laender, Adriano Veloso, and Nivio Ziviani. Dynamic prediction of ICU mortality risk using domain adaptation. In *Proceedings of the 2018 IEEE International Conference on Big Data*, pages 1328–1336. IEEE, 2018.

[3] Mohammad R Arbabshirani, Sergey Plis, Jing Sui, and Vince D Calhoun. Single subject prediction of brain disorders in neuroimaging: Promises and pitfalls. *Neuroimage*, 145:137–165, 2017.

[4] Sercan RÖ. Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8):6679–6687, May 2021.

[5] Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. In *International Conference on Machine Learning*, pages 254–263. PMLR, 2018.

[6] Christina Baek, Yiding Jiang, Aditi Raghunathan, and J. Zico Kolter. Agreement-on-the-line: Predicting the performance of neural networks under distribution shift. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 19274–19289. Curran Associates, Inc., 2022.

[7] Parikshit Bansal, Yashoteja Prabhu, Emre Kiciman, and Amit Sharma. Using interventions to improve out-of-distribution generalization of text-matching systems. In *NeurIPS 2022 Workshop on Causality for Real-world Impact*, 2022.

[8] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[9] Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

[10] Nadia Burkart and Marco F Huber. A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research*, 70:245–317, 2021.

[11] Feiyang Cai and Xenofon Koutsoukos. Real-time out-of-distribution detection in cyber-physical systems with learning-enabled components. *IET Cyber-Physical Systems: Theory & Applications*, pages 212–234, 2022.

[12] Ahmet Furkan Çelik, Bedirhan Sağlam, and Sedef Demirci. Developing explainable intrusion detection systems for internet of things. In *2023 16th International Conference on Information Security and Cryptology (ISCTürkiye)*, pages 1–6. IEEE, 2023.

[13] Horng-Jinh Chang and Ming-Chen Lee. Applying computer simulation to analyze the normal approximation of binomial distribution. *Journal of Computers*, 28(5):116–131, 2017.

[14] Jiefeng Chen, Frederick Liu, Besim Avci, Xi Wu, Yingyu Liang, and Somesh Jha. Detecting errors and estimating accuracy on unlabeled data with self-training ensembles. *Advances in Neural Information Processing Systems*, 34:1–13, 2021.

[15] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[16] Yen-Chi Chen. A tutorial on kernel density estimation and recent advances. *Biostatistics & Epidemiology*, 1(1):161–187, 2017.

[17] Lu Cheng, Ruocheng Guo, Raha Moraffah, Paras Sheth, K Selçuk Candan, and Huan Liu. Evaluation methods and measures for causal learning algorithms. *IEEE Transactions on Artificial Intelligence*, 3(6):924–943, 2022.

[18] Michael Chui, James Manyika, Mehdi Miremadi, Nicolaus Henke, Rita Chung, Pieter Nel, and Sankalp Malhotra. Notes from the ai frontier: Insights from hundreds of use cases. *McKinsey Global Institute*, page 28, 2018.

[19] Ciprian A Corneanu, Sergio Escalera, and Aleix M Martinez. Computing the testing error without a testing set. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2677–2685, 2020.

[20] Gessé Dafé, Adriano Veloso, Mohammed Zaki, and Wagner Meira Jr. Learning sequential classifiers from long and noisy discrete-event sequences efficiently. *Data Min. Knowl. Discov.*, 29(6):1685–1708, 2015.

[21] Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D Hoffman, et al. Underspecification presents challenges for credibility in modern machine learning. *The Journal of Machine Learning Research*, 23(1):10237–10297, 2022.

[22] Kevin De Angeli, Shang Gao, Ioana Danciu, Eric B Durbin, Xiao-Cheng Wu, Antoinette Stroup, Jennifer Doherty, Stephen Schwartz, Charles Wiggins, Mark Damesyn, et al. Class imbalance in out-of-distribution datasets: Improving the robustness of the textcnn for the classification of rare cancer types. *Journal of biomedical informatics*, pages 1–11, 2021.

[23] Fan Deng, Stefan Siersdorfer, and Sergej Zerr. Efficient jaccard-based diversity analysis of large document collections. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1402–1411, 2012.

[24] Weijian Deng, Yumin Suh, Stephen Gould, and Liang Zheng. Confidence and dispersity speak: Characterizing prediction matrix for unsupervised accuracy estimation. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 7658–7674. PMLR, 23–29 Jul 2023.

[25] Weijian Deng and Liang Zheng. Are labels always necessary for classifier accuracy evaluation? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15069–15078, 2021.

[26] Gintare Karolina Dziugaite and Daniel M Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017.

[27] Marcos Kneip Fleury. A covid-19 e o laboratório de hematologia: uma revisão da literatura recente. *RBAC*, 52(2):131–7, 2020.

[28] Tadayoshi Fushiki. Estimation of prediction error by using k-fold cross-validation. *Statistics and Computing*, 21:137–146, 2011.

[29] Jean-Christophe Gagnon-Audet, Kartik Ahuja, Mohammad Javad Darvishi Bayazi, Pooneh Mousavi, Guillaume Dumas, and Irina Rish. WOODS: Benchmarks for out-of-distribution generalization in time series. *Transactions on Machine Learning Research*, 2023. Featured Certification.

[30] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37, 2014.

[31] Saurabh Garg, Sivaraman Balakrishnan, Zachary Lipton, Behnam Neyshabur, and Hanie Sedghi. Leveraging unlabeled data to predict out-of-distribution performance. In *International Conference on Learning Representations (ICLR)*, pages 1–28, 2022.

[32] Tom Ginsberg. Identifying and characterizing high dimensional covariate shift with learning models. 2023.

[33] Devin Guillory, Vaishaal Shankar, Sayna Ebrahimi, Trevor Darrell, and Ludwig Schmidt. Predicting with confidence on unseen distributions. In *Proc. of the IEEE/CVF International Conference on Computer Vision*, pages 1134–1144, 2021.

[34] Jiawei Han, Jian Pei, and Hanghang Tong. *Data mining: concepts and techniques*. Morgan kaufmann, 2022.

[35] Tom Heskes, Evi Sijben, Ioan Gabriel Bucur, and Tom Claassen. Causal shapley values: Exploiting causal knowledge to explain individual predictions of complex models. *Advances in neural information processing systems*, 33:4778–4789, 2020.

[36] Eugenio D Hottz, Isaclaudia G Azevedo-Quintanilha, Lohanna Palhinha, Lívia Teixeira, Ester A Barreto, Camila RR Pão, Cassia Righy, Sérgio Franco, Thiago ML Souza, Pedro Kurtz, et al. Platelet activation and platelet-monocyte aggregate formation trigger tissue factor expression in patients with severe covid-19. *Blood, The Journal of the American Society of Hematology*, 136(11):1330–1341, 2020.

[37] Yan Huang, Zhang Zhang, Yan Huang, Qiang Wu, Han Huang, Yi Zhong, and Liang Wang. Customized meta-dataset for automatic classifier accuracy evaluation. *Pattern Recognition*, 146:110026, 2024.

[38] iStock. Alley de neve pela manhã - imagem em alta resolução. https://www.istockphoto.com/br/foto/alley-de-neve-pela-manh%C3%A3-gm533292615-56247296?irgwc=1&cid=IS&utm_medium=affiliate&utm_source=Du%C5%A1an+Bi%C4%8Danski+Pr+Digitalnio&clickid=1tQUzj3mjxyPUwjX9tT9ITDaUkFWGaw3-y2vXc0&utm_content=258824&irpid=1404368, 2023. Accessed: 2023-10-06.

[39] Dominik Janzing. Causal regularization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[40] Yiding Jiang, Dilip Krishnan, Hossein Mobahi, and Samy Bengio. Predicting the generalization gap in deep networks with margin distributions. In *International Conference on Learning Representations*, pages 1–19, 2019.

[41] Yiding Jiang*, Behnam Neyshabur*, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. In *International Conference on Learning Representations*, pages 1–26, 2020.

[42] Markus Kalisch and Peter Bühlman. Estimating high-dimensional directed acyclic graphs with the pc-algorithm. *Journal of Machine Learning Research*, 8(3):613–636, 2007.

[43] Markus Kalisch and Peter Bühlman. Estimating high-dimensional directed acyclic graphs with the pc-algorithm. *Journal of Machine Learning Research*, 8(3), 2007.

[44] Zhiqi Kang, Enrico Fini, Moin Nabi, Elisa Ricci, and Karteek Alahari. A soft nearest-neighbor framework for continual semi-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11868–11877, 2023.

[45] E Keany. BorutaShap Project. https://github.com/Ekeany/Boruta-Shap, 2020. [Online; accessed 17-Aug-2022].

[46] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI*, volume 14, pages 1137–1145. Montreal, Canada, 1995.

[47] Johnson Kolluri, Vinay Kumar Kotte, MSB Phridviraj, and Shaik Razia. Reducing overfitting problem in machine learning using novel l1/4 regularization method. In *2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184)*, pages 934–938. IEEE, 2020.

[48] Abhinav Kumar, Amit Deshpande, and Amit Sharma. Causal effect regularization: Automated detection and removal of spurious correlations. In A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 20942–20984. Curran Associates, Inc., 2023.

[49] Deepthi Praveenlal Kuttichira, Sunil Gupta, Cheng Li, Santu Rana, and Svetha Venkatesh. Explaining black-box models using interpretable surrogates. In *PRICAI 2019: Trends in Artificial Intelligence: 16th Pacific Rim International Conference on Artificial Intelligence, Cuvu, Yanuca Island, Fiji, August 26–30, 2019, Proceedings, Part I 16*, pages 3–15. Springer, 2019.

[50] Thao Le, Tim Miller, Ronal Singh, and Liz Sonenberg. Explaining model confidence using counterfactuals. *arXiv preprint arXiv:2303.05729*, 2023.

[51] 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.

[52] Jiashuo Liu, Zheyan Shen, Yue He, Xingxuan Zhang, Renzhe Xu, Han Yu, and Peng Cui. Towards out-of-distribution generalization: A survey. *arXiv preprint arXiv:2108.13624*, 2021.

[53] Yuchun Liu, Zhihui Liu, Xue Luo, and Hongjingtian Zhao. Diagnosis of parkinson's disease based on shap value feature selection. *Biocybernetics and Biomedical Engineering*, 42(3):856–869, 2022.

[54] Victoria López, Alberto Fernández, and Francisco Herrera. On the importance of the validation technique for classification with imbalanced datasets: Addressing covariate shift when data is skewed. *Information Sciences*, 257:1–13, 2014.

[55] Paul Louangrath. Minimum sample size method based on survey scales. *Int. J. Res. Methodol. Soc. Sci*, 3(3):44–52, 2017.

[56] Yuzhe Lu, Yilong Qin, Runtian Zhai, Andrew Shen, Ketong Chen, Zhenlin Wang, Soheil Kolouri, Simon Stepputtis, Joseph Campbell, and Katia P. Sycara. Characterizing out-of-distribution error via optimal transport. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[57] Scott M. Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M. Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable ai for trees. *Nature Machine Intelligence*, 2(1):2522–5839, 2020.

[58] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.

[59] Omid Madani, David Pennock, and Gary Flake. Co-validation: Using model disagreement on unlabeled data to validate classification algorithms. *Advances in neural information processing systems*, 17:873–880, 2004.

[60] Sara Magliacane, Thijs Van Ommen, Tom Claassen, Stephan Bongers, Philip Versteeg, and Joris M Mooij. Domain adaptation by using causal inference to predict

invariant conditional distributions. *Advances in neural information processing systems*, 31:1–11, 2018.

[61] Sebastián Maldonado, Julio López, and Andrés Iturriaga. Out-of-time cross-validation strategies for classification in the presence of dataset shift. *Applied Intelligence*, pages 1–14, 2021.

[62] Sebastián Maldonado, Richard Weber, and Fazel Famili. Feature selection for high-dimensional class-imbalanced data sets using support vector machines. *Information sciences*, 286:228–246, 2014.

[63] Mohit Malu, Gautam Dasarathy, and Andreas Spanias. Bayesian optimization in high-dimensional spaces: A brief survey. In *2021 12th International Conference on Information, Intelligence, Systems & Applications (IISA)*, pages 1–8. IEEE, 2021.

[64] Wilson E Marcílio and Danilo M Eler. From explanations to feature selection: assessing shap values as feature selection mechanism. In *2020 33rd SIBGRAPI conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 340–347. Ieee, 2020.

[65] Bruce G Marcot and Anca M Hanea. What is an optimal value of k in k-fold cross-validation in discrete bayesian network analysis? *Computational Statistics*, 36(3):2009–2031, 2021.

[66] Christian Masdeval and Adriano Veloso. Mining citizen emotions to estimate the urgency of urban issues. *Inf. Syst.*, 54:147–155, 2015.

[67] John Miller, Karl Krauth, Benjamin Recht, and Ludwig Schmidt. The effect of natural distribution shift on question answering models. In *International Conference on Machine Learning*, pages 6905–6916. PMLR, 2020.

[68] Jonas Mockus. Application of bayesian approach to numerical methods of global and stochastic optimization. *Journal of Global Optimization*, 4(4):347–365, 1994.

[69] Jose G Moreno-Torres, Troy Raeder, Rocío Alaiz-Rodríguez, Nitesh V Chawla, and Francisco Herrera. A unifying view on dataset shift in classification. *Pattern recognition*, 45(1):521–530, 2012.

[70] Carlos Mougan, Klaus Broelemann, Gjergji Kasneci, Thanassis Tiropanis, and Steffen Staab. Explanation shift: Detecting distribution shifts on tabular data via the explanation space. In *NeurIPS 2022 Workshop on Distribution Shifts: Connecting Methods and Applications*, pages 1–9, 2022.

[71] Vaishnavh Nagarajan, Anders Andreassen, and Behnam Neyshabur. Understanding the failure modes of out-of-distribution generalization. *arXiv preprint arXiv:2010.15775*, 2020.

[72] Eduardo Nigri, Nivio Ziviani, Fábio Cappabianco, Augusto Antunes, and Adriano Veloso. Explainable deep cnns for mri-based diagnosis of alzheimer's disease. In *Proceedings of the 2020 International Joint Conference on Neural Networks*, pages 1–8, 2020.

[73] Isaac Kofi Nti, Owusu Nyarko-Boateng, and Justice Aning. Performance of machine learning algorithms with different k values in k-fold cross-validation. *J. Inf. Technol. Comput. Sci*, 6:61–71, 2021.

[74] Santiago Ontañón. An overview of distance and similarity functions for structured data. *Artificial Intelligence Review*, 53(7):5309–5351, 2020.

[75] Pexels. https://www.pexels.com/pt-br/foto/estrada-de-concreto-entre-arvores-1563356/, 2023. Accessed: 2023-10-06.

[76] Emmanouil Platanios, Hoifung Poon, Tom M Mitchell, and Eric J Horvitz. Estimating accuracy from unlabeled data: A probabilistic logic approach. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 1–10. Curran Associates, Inc., 2017.

[77] Emmanouil Antonios Platanios, Avinava Dubey, and Tom Mitchell. Estimating accuracy from unlabeled data: A bayesian approach. In *International Conference on Machine Learning*, pages 1416–1425. PMLR, 2016.

[78] Drew Prinster, Suchi Saria, and Anqi Liu. Jaws-x: addressing efficiency bottlenecks of conformal prediction under standard and feedback covariate shift. In *International Conference on Machine Learning*, pages 28167–28190. PMLR, 2023.

[79] Joaquin Quinonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. *Dataset shift in machine learning*. Mit Press, 2008.

[80] Haider Raza, Girijesh Prasad, and Yuhua Li. Ewma model based shift-detection methods for detecting covariate shifts in non-stationary environments. *Pattern Recognition*, 48(3):659–669, 2015.

[81] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "“Why should I trust you?”: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM*

SIGKDD international conference on knowledge discovery and data mining, pages 1135–1144, 2016.

[82] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, pages 1527–1535, 2018.

[83] Elan Rosenfeld and Saurabh Garg. (almost) provable error bounds under distribution shift via disagreement discrepancy. In A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 28761–28784. Curran Associates, Inc., 2023.

[84] Richard J Rossi. *Mathematical statistics: an introduction to likelihood based inference*. John Wiley & Sons, 2018.

[85] scikit learn. sklearn.datasets.make_classification. https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_classification.html, 2007. Accessed: August 29, 2023.

[86] Shohei Shimizu, Patrik O Hoyer, Aapo Hyvärinen, Antti Kerminen, and Michael Jordan. A linear non-gaussian acyclic model for causal discovery. *Journal of Machine Learning Research*, 7(10), 2006.

[87] Shohei Shimizu, Takanori Inazumi, Yasuhiro Sogawa, Aapo Hyvärinen, Yoshinobu Kawahara, Takashi Washio, Patrik O Hoyer, and Kenneth Bollen. Directlingam: A direct method for learning a linear non-gaussian structural equation model. *The Journal of Machine Learning Research*, 12:1225–1248, 2011.

[88] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

[89] Grigori Sidorov, Alexander Gelbukh, Helena Gómez-Adorno, and David Pinto. Soft similarity and soft cosine measure: Similarity of features in vector space model. *Computación y Sistemas*, 18(3):491–504, 2014.

[90] I. S. Silva and A. Veloso. Automatic model evaluation using feature importance patterns on unlabeled data. In *IJCNN International Joint Conference on Neural Networks*, pages 1–8, 2022.

[91] IS Silva, CN Ferreira, LBX Costa, MO Sóter, LML Carvalho, J de C. Albuquerque, MF Sales, AL Candido, FM Reis, AA Veloso, et al. Polycystic ovary syndrome:

Clinical and laboratory variables related to new phenotypes using machine-learning models. *Journal of Endocrinological Investigation*, pages 1–9, 2022.

[92] Pradeep Kumar Singh, Pijush Kanti Dutta Pramanik, and Prasenjit Choudhury. A comparative study of different similarity metrics in highly sparse rating dataset. In Valentina Emilia Balas, Neha Sharma, and Amlan Chakrabarti, editors, *Data Management, Analytics and Innovation*, pages 45–60, Singapore, 2019. Springer Singapore.

[93] Peter Spirtes, Clark N Glymour, and Richard Scheines. *Causation, prediction, and search*. MIT press, 2000.

[94] Adarsh Subbaswamy, Bryant Chen, and Suchi Saria. A unifying causal framework for analyzing dataset shift-stable learning algorithms. *Journal of Causal Inference*, 10(1):64–89, 2022.

[95] Adarsh Subbaswamy and Suchi Saria. Counterfactual normalization: Proactively addressing dataset shift using causal mechanisms. In *UAI*, pages 947–957, 2018.

[96] Masashi Sugiyama, Benjamin Blankertz, Matthias Krauledat, Guido Dornhege, and Klaus-Robert Müller. Importance-weighted cross-validation for covariate shift. In *Joint Pattern Recognition Symposium*, pages 354–363. Springer, 2006.

[97] Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert Müller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8(5), 2007.

[98] Masashi Sugiyama and Klaus-Robert Müller. Input-dependent estimation of generalization error under covariate shift. *Statistics & Decisions*, 23(4):249—-279, 2005.

[99] Xiaoxiao Sun, Yunzhong Hou, Hongdong Li, and Liang Zheng. Label-free model evaluation with semi-structured dataset representations. *arXiv preprint arXiv:2112.00694*, 2021.

[100] Weijie Tu, Weijian Deng, Tom Gedeon, and Liang Zheng. A bag-of-prototypes representation for dataset-level applications. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2881–2892, June 2023.

[101] Jasper van der Waa, Tjeerd Schoonderwoerd, Jurriaan van Diggelen, and Mark Neerincx. Interpretable confidence measures for decision support systems. *International Journal of Human-Computer Studies*, 144:102493, 2020.

[102] Danding Wang, Wencan Zhang, and Brian Y Lim. Show or suppress? managing input uncertainty in machine learning model explanations. *Artificial Intelligence*, 294:103456, 2021.

[103] Jiaxuan Wang, Jenna Wiens, and Scott Lundberg. Shapley flow: A graph-based approach to interpreting model predictions. In *International Conference on Artificial Intelligence and Statistics*, pages 721–729. PMLR, 2021.

[104] Ruoyu Wang, Mingyang Yi, Zhitang Chen, and Shengyu Zhu. Out-of-distribution generalization with causal invariant transformations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 375–385, 2022.

[105] Zhao Wang, Kai Shu, and Aron Culotta. Enhancing model robustness and fairness with causality: A regularization approach. In Amir Feder, Katherine Keith, Emaad Manzoor, Reid Pryzant, Dhanya Sridhar, Zach Wood-Doughty, Jacob Eisenstein, Justin Grimmer, Roi Reichart, Molly Roberts, Uri Shalit, Brandon Stewart, Victor Veitch, and Diyi Yang, editors, *Proceedings of the First Workshop on Causal Inference and NLP*, pages 33–43, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[106] Larry Wasserman. *Expectation*, pages 48–61. Springer New York, New York, NY, 2004.

[107] Geoffrey I Webb, Roy Hyde, Hong Cao, Hai Long Nguyen, and Francois Petitjean. Characterizing concept drift. *Data Mining and Knowledge Discovery*, 30(4):964–994, 2016.

[108] Tzu-Tsung Wong. Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation. *Pattern recognition*, 48(9):2839–2846, 2015.

[109] Xia Xiaomao, Zhang Xudong, and Wang Yuanfang. A comparison of feature selection methodology for solving classification problems in finance. In *Journal of Physics: Conference Series*, volume 1284, page 012026. IOP Publishing, 2019.

[110] RENCHUNZI XIE, Hongxin Wei, Lei Feng, Yuzhou Cao, and Bo An. On the importance of feature separability in predicting out-of-distribution error. In A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 27783–27800. Curran Associates, Inc., 2023.

[111] Han Yu, Jiashuo Liu, Xingxuan Zhang, Jiayun Wu, and Peng Cui. A survey on evaluation of out-of-distribution generalization. *arXiv preprint arXiv:2403.01874*, 2024.

[112] Yaodong Yu, Zitong Yang, Alexander Wei, Yi Ma, and Jacob Steinhardt. Predicting out-of-distribution error with the projection norm. In *International Conference on Machine Learning*, volume 162, pages 25721–25746. PMLR, 2022.

[113] Yunfeng Zhang, Q Vera Liao, and Rachel KE Bellamy. Effect of confidence and explanation on accuracy and trust calibration in ai-assisted decision making. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pages 295–305, 2020.

[114] Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. Dags with no tears: Continuous optimization for structure learning. *Advances in neural information processing systems*, 31:1–12, 2018.

Appendix A

Map among the Node Labels in the Causal Graph and the Features

Mapping among the node labels of the causal graph that was presented in the Figure 6.7 and the features of the Alzheimers datasets:

- 1:Alzheimer (class),
- 2:Escolaridade,
- 3:Apo E,
- 4:Peak-Low TF-aTFPI,
- 5:ETP-High TF+APC,
- 6:Lagtime-High TF-APC,
- 7:ETP-High TF-APC,
- 8:PC-RQ,
- 9:Peak-High TF-APC,
- 10:HAS,
- 11:Etilismo,
- 12:Lagtime-Low TF-aTFPI,
- 13:ETP-Low TF-aTFPI,
- 14:DM,
- 15:Tabagismo,
- 16:Sexo,
- 17:Peak-High TF+APC,

- 18:Lagtime-High TF+APC, and
- 19:Idade

Mapping among the node labels of the causal graph that was presented in the Figure 6.6 and the features of the COVID-19 datasets:

- 1:Veio_a_obito (class),
- 2:DT_COLETA_DIAS_CORRIDOS,
- 3:Concentracao de Hemoglobina Corpuscular_g/dL,
- 4:ALT (TGP)_U/L,
- 5:Ureia_mg/dL,
- 6:Eritrocitos_milhoes/mm3,
- 7:Primeiro_PCR,
- 8:Bilirrubina Indireta_mg/dL,
- 9:AST (TGO)_U/L,
- 10:Platelets_/mm3,
- 11:Hematocrito_
- 12:Creatinina_mg/dL,
- 13:AGE,
- 14:Potassio_mEq/L,
- 15:VCM_fL,
- 16:Hemoglobina_g/dL,
- 17:Ultimo_PCR,
- 18:Proteina C-Reativa_mg/dL,
- 19:RDW_
- 20:Bilirrubina Total_mg/dL, and
- 21:Hemoglobina Corpuscular Media_pg.

Mapping among the node labels of the causal graph that was presented in the Figure 6.5 and the features of the Dropout School datasets:

- 1:Evadiu,
- 2:SEXO,
- 3:1ANOFALTA,
- 4:Forma_Ingresso,
- 5:N_FILHOS,
- 6:ESTADO_CIVIL,
- 7:IDADE,
- 8:ETNIA,
- 9:ESCOLA_FUNDAMENTAL,
- 10:CIDADE_ORIGEM,
- 11:TIPO_ESCOLA_ANTERIOR,
- 12:TIPO_CONCOMITANCIA,
- 13:Cidade,
- 14:IDADE_INGRESSO,
- 15:CURSO, and
- 16:1CONCEITO.

Appendix B

Processing Time Analysis

This appendix presents the analysis of the processing time of the evaluated algorithm in this thesis. We executed the experiments in the hardware with 2-core Intel(R) Xeon(R) CPU @ 2.20GHz, 13GB RAM, and 33GB HDD. The machine there was no graphics card. Regarding the software specifications, we used the Ubuntu 22.04.3 LTS operating system with Python 3.10 installed.

Table B.1 shows the average process time of 30 executions of the algorithms 10-fold cross-validation, the X-Eval, and the C-Eval. The table shows the C-Eval results (1) by used causality inference algorithm, (2) regularizing the CV and the X-Eval estimates, and (3) with the Cosine similarity function. We present just the C-Eval results with the Cosine similarity because the used similarity function did not impact the processing time.

Table B.1: Average process time in 30 executions of the algorithms. We show the standard deviation in parentheses.

	Source: Geriatrics Target: Neurology	**Source: Neurology Target: Geriatrics**	**Source: HBP Target: HSL**	**Source: HSL Target: HBP**	**Source: Students2008 Target: Students2014**
CV	00:00:02 (00:00:01)	00:00:02 (00:00:01)	00:00:04 (00:00:01)	00:00:03 (00:00:02)	00:00:02 (00:00:01)
X-Eval	00:03:07 (00:01:14)	00:02:21 (00:00:08)	00:04:16 (00:00:42)	00:08:01 (00:00:19)	00:03:06 (00:00:38)
C-Eval(Direct,CV)	00:00:06 (00:00:02)	00:00:13 (00:00:06)	00:00:11 (00:00:04)	00:00:43 (00:00:13)	00:00:04 (00:00:02)
C-Eval(Notears,CV)	00:05:16 (00:00:31)	00:10:47 (00:00:53)	00:02:40 (00:00:23)	00:09:07 (00:00:37)	00:00:34 (00:00:07)
C-Eval(PC,CV)	00:00:02 (00:00:01)	00:00:03 (00:00:01)	00:00:10 (00:00:01)	00:00:54 (00:00:03)	00:07:38 (00:00:17)
C-Eval(Direct,X-Eval)	00:03:12 (00:01:16)	00:02:32 (00:00:13)	00:04:24 (00:00:46)	00:08:43 (00:00:34)	00:03:09 (00:00:39)
C-Eval(Notears,X-Eval)	00:08:22 (00:01:28)	00:13:07 (00:00:56)	00:06:59 (00:01:10)	00:17:20 (00:00:59)	00:03:40 (00:00:43)
C-Eval(PC,X-Eval)	00:03:07 (00:01:14)	00:02:23 (00:00:08)	00:04:23 (00:00:42)	00:08:50 (00:00:19)	00:10:43 (00:00:56)

We computed the processing time of the algorithms with the real-world datasets described in Section 6.1. In these experiments, we used all features of the Alzheimer and School dropout datasets. However, due to the great number of features of the COVID-19 dataset, in each setup, we selected the most informative features using the training set and the Borutashap feature selection method [45]. Thus, BorutaShap selected 20 features for the HBP dataset, and 30 features for the HSL dataset.

In Table B.1, we can see that the processing time of the proposed algorithms X-Eval and C-Eval are much greater than the CV process time. However, these greater processing times are less important in the first moment, given the challenging task of estimating the model performance in scenarios where we expect distribution shifts.

Appendix C

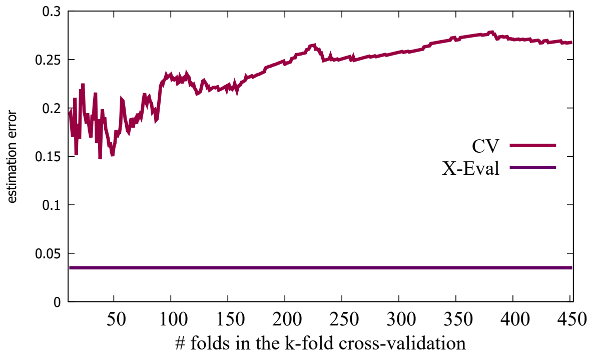
Analysis of the Number of Folds Impact in the k-fold Cross-Validation in Distribution Shift Scenarios

In this appendix, we analyze how the variation in the number of folds in the k-fold cross-validation (CV) changes the CV estimation error in distribution shift scenarios. Figure C.1 shows the CV and the X-Eval estimation errors in each analyzed setup of source and target datasets. In this experiment, we used the features described in Section 6.1. We estimated the model performance with the CV varying the number of folds (k) ranging from 11 to the number of examples in the source dataset, with an interval of one. We compared the CV only with the X-Eval to facilitate analysis because, in this experiment, we focused on the analysis of the effect of increasing the number of folds (i.e., it is not our intention to compare all proposed approaches).

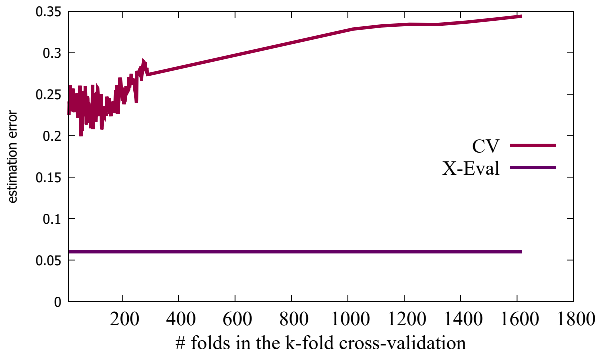
In Figure C.1, we can see that in most cases, when we increase the number of folds, the estimation error is not consistently reduced. Besides, even with the variation in the number of folds, the X-Eval outperformed the CV persistently in four setups (i.e., Figure C.1a, Figure C.1b, Figure C.1c, and Figure C.1d). These results are attributed to the fact that in the cases of COVID-19 and Alzheimer, there are evident distribution shifts between the source and the target datasets (as shown in Section 6.1.1), and the CV requires that the source and the target datasets share the same distribution [61, 54, 33].

Additionally, we emphasize that only in the datasets related to School Dropout (Figure C.1e) did the estimation error decrease significantly until approximately forty folds. However, after forty folds, the estimation error increased again. This result is attributed to the fact that we identified in Section 6.1.1 that there is no strong distribution shift between the datasets regarding School Dropout. Hence, the CV estimation tends to be precise. Despite the observed significant reduction in the estimation error with around 30 folds, we cannot assume this parameter as default for all datasets. Once defining the optimal number of folds for each dataset is a challenging task [108, 73].

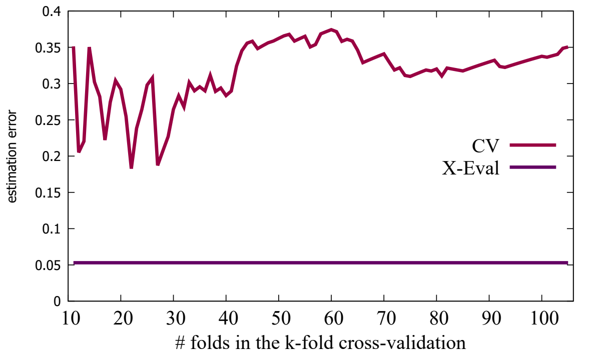
These findings indicate that only increasing the folds in the k-fold cross-validation does not improve the performance estimation when there is a distribution shift between the source and the target datasets. Consequently, proposing new approaches considering distribution shifts between the datasets is important.



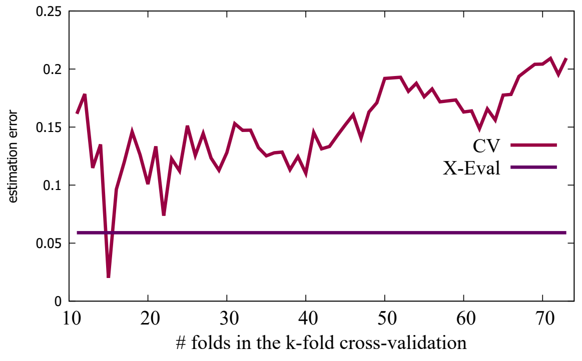
(a) Source: HBP, Target: HSL



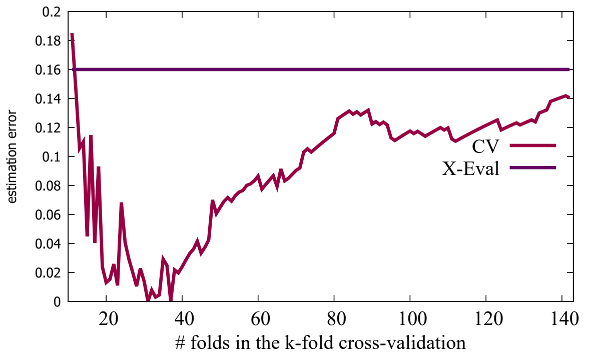
(b) Source: HSL, Target: HBP



(c) Source: Geriatrics, Target: Neurology



(d) Source: Neurology, Target: Geriatrics



(e) Source: *Students2008*, Target: *Students2014*

Figure C.1: Analysis of the CV estimation error ($|truePerformance - estimatedPerformance|$) when we increase the number of folds (k).

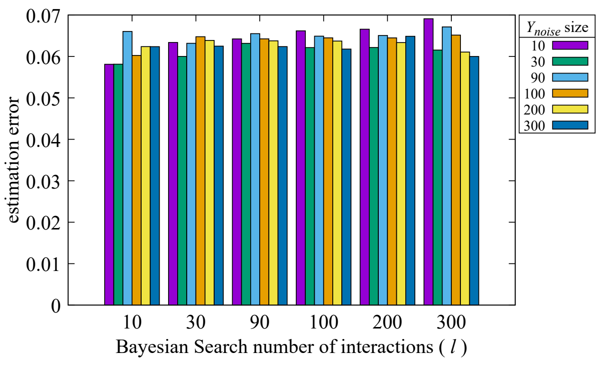
Appendix D

X-Eval Hyperparameter Analysis

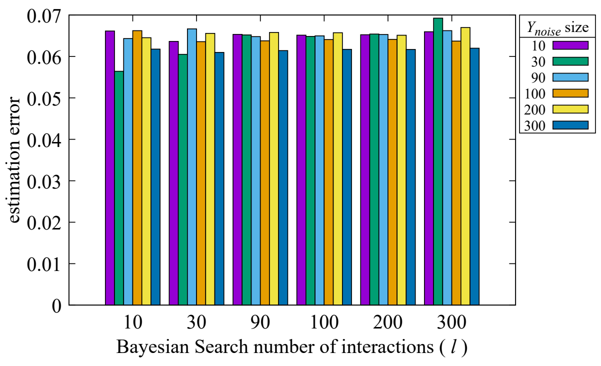
In this appendix, we show the conducted experimental analysis to define the X-Eval hyperparameters Y_{noise} size and the number of interactions (l) in Bayesian Search. We evaluated the X-Eval using Y_{noise} sizes and l values of 10, 30, 90, 100, 200, and 300. In this experiment, we used the features described in Section 6.1.

Figure D.1 shows the X-Eval estimation errors in each analyzed setup of source and target datasets. In Figure D.1, we see that there was no consistent improvement in the estimation error with the variations in the hyperparameters.

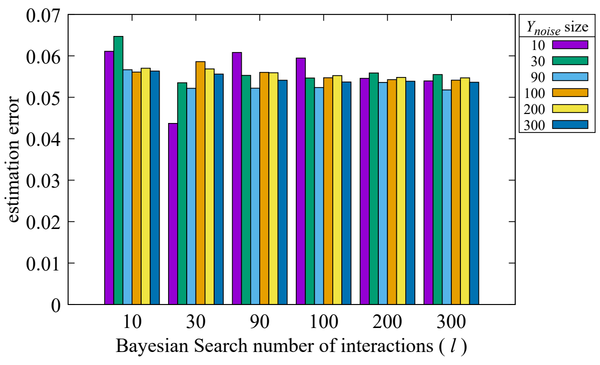
Hence, in the experiments involving the X-Eval in the rest of this thesis, we set the size of Y_{noise} to 30 and the Bayesian Search number of interactions (l) to 30. This decision was because (1) 30 is considered the minimum sample size to make statistical conclusions regarding a population [13, 55], and (2) during the experiments in this appendix, the X-Eval estimation error did not present a significant reduction with the variations in these hyperparameters. Thus, given the considerable computational cost to execute the X-Eval (as demonstrated in Appendix B and Section 4.1.7), minimizing the repetitions number is desirable.



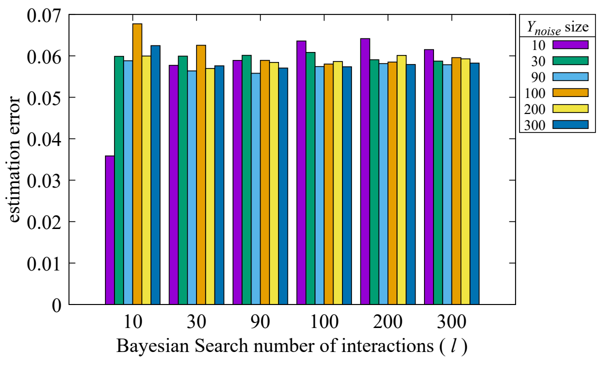
(a) Source: HBP, Target: HSL



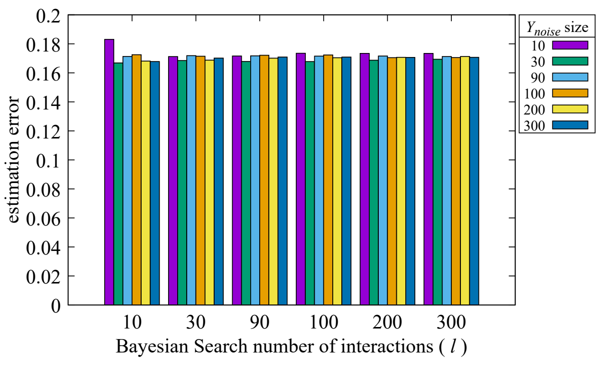
(b) Source: HSL, Target: HBP



(c) Source: Geriatrics, Target: Neurology



(d) Source: Neurology, Target: Geriatrics



(e) Source: *Students2008*, Target: *Students2014*

Figure D.1: X-Eval hyperparameters analysis.

Appendix E

Complementary Tables of the Estimation Error Analysis for Diverse Classifiers

Section 6.2.2 presented an analysis of the estimation error of the proposed approaches for different classifiers. However, Section 6.2.2 showed only the C-Eval results with the DirectLinGAM causality inference algorithm and the Macro F1 similarity function. Hence, this appendix includes tables presenting the C-Eval estimation error with the other causal inference algorithms and similarity functions used in this thesis, which were not shown in Section 6.2.2.

Table E.1: Estimation error analysis for the classifiers XGBoost, LightGBM, CatBoost, and TabNet. We compared the proposed autoEval approaches (X-Eval and C-Eval) with the 10-fold cross-validation (CV). The C-Eval(*Estimatior*, *Direct*, *Cosine*, λ) denotes the C-Eval approach with one *Estimatior* (i.e., CV or X-Eval), the **DirectLiNGAM** (*Direct*) causality inference algorithm, the **Cosine** similarity function, and one λ factor (i.e., 1, 3, or 6). We highlighted the smallest estimation error for each dataset considering each classifier in **bold**, and we also underline the second smallest estimation error.

	Performance Estimator	**Source: Geriatrics Target: Neurology**	**Source: Neurology Target: Geriatrics**	**Source: HBP Target: HSL**	**Source: HSL Target: HBP**	**Source: Students2008 Target: Students2014**
XGBoost	CV	0.37	0.26	0.19	0.20	**0.16**
	X-Eval	0.05	0.06	0.04	0.06	0.17
	C-Eval(CV,*Direct*,Cosine,1)	0.04	0.03	0.14	0.05	0.24
	C-Eval(X-Eval,*Direct*,Cosine,1)	0.12	0.13	0.21	0.12	0.24
	C-Eval(CV,*Direct*,Cosine,3)	0.21	0.11	**0.03**	0.08	0.20
	C-Eval(X-Eval,*Direct*,Cosine,3)	0.03	0.04	0.09	0.03	0.21
	C-Eval(CV,*Direct*,Cosine,6)	0.28	0.18	0.10	0.13	0.18
	C-Eval(X-Eval,*Direct*,Cosine,6)	**0.00**	**0.01**	0.04	**0.01**	0.19
LightGBM	CV	0.31	0.11	0.22	0.32	**0.09**
	X-Eval	**0.00**	0.09	0.07	0.16	0.12
	C-Eval(CV,*Direct*,Cosine,1)	0.04	0.14	0.10	**0.03**	0.21
	C-Eval(X-Eval,*Direct*,Cosine,1)	0.19	0.24	0.17	0.05	0.23
	C-Eval(CV,*Direct*,Cosine,3)	0.14	**0.02**	0.06	0.17	0.15
	C-Eval(X-Eval,*Direct*,Cosine,3)	0.09	0.16	0.05	0.06	0.17
	C-Eval(CV,*Direct*,Cosine,6)	0.21	0.04	0.13	0.24	0.13
	C-Eval(X-Eval,*Direct*,Cosine,6)	0.05	0.13	**0.00**	0.10	0.15
CatBoost	CV	0.33	0.09	0.25	0.30	**0.15**
	X-Eval	**0.01**	0.09	0.08	0.07	0.21
	C-Eval(CV,*Direct*,Cosine,1)	0.02	0.22	0.08	0.04	0.26
	C-Eval(X-Eval,*Direct*,Cosine,1)	0.14	0.31	0.17	0.07	0.29
	C-Eval(CV,*Direct*,Cosine,3)	0.18	0.07	0.08	0.17	0.21
	C-Eval(X-Eval,*Direct*,Cosine,3)	0.07	0.20	0.05	**0.00**	0.25
	C-Eval(CV,*Direct*,Cosine,6)	0.24	**0.00**	0.15	0.23	0.18
	C-Eval(X-Eval,*Direct*,Cosine,6)	0.04	0.15	**0.01**	0.03	0.23
TabNet	CV	0.10	0.12	0.07	0.03	**0.01**
	X-Eval	0.04	**0.10**	**0.00**	0.05	0.06
	C-Eval(CV,*Direct*,Cosine,1)	0.09	0.27	0.16	0.21	0.12
	C-Eval(X-Eval,*Direct*,Cosine,1)	0.12	0.26	0.12	0.17	0.15
	C-Eval(CV,*Direct*,Cosine Sim.,3)	**0.00**	0.20	0.11	0.12	0.06
	C-Eval(X-Eval,*Direct*,Cosine Sim.,3)	0.04	0.18	0.06	0.06	0.11
	C-Eval(CV,*Direct*,Cosine Sim.,6)	0.04	0.16	0.10	0.08	0.04
	C-Eval(X-Eval,*Direct*,Cosine,6	0.01	0.14	0.03	**0.02**	0.09

Table E.2: Estimation error analysis for the classifiers XGBoost, LightGBM, CatBoost, and TabNet. We compared the proposed autoEval approaches (X-Eval and C-Eval) with the 10-fold cross-validation (CV). The C-Eval(*Estimatior*, *Notears*, *Cosine*, λ) denotes the C-Eval approach with one *Estimatior* (i.e, CV or X-Eval), the **Notears** causality inference algorithm, the **Cosine** similarity function, and one λ factor (i.e., 1, 3, or 6). We highlighted the smallest estimation error for each dataset considering each classifier in **bold**, and we also $\underline{\text{underline}}$ the second smallest estimation error.

	Performance Estimator	**Source: Geriatrics Target: Neurology**	**Source: Neurology Target: Geriatrics**	**Source: HBP Target: HSL**	**Source: HSL Target: HBP**	**Source: Students2008 Target: Students2014**
XGBoost	CV	0.37	0.26	0.19	0.20	0.16
	X-Eval	**0.05**	**0.06**	0.04	**0.06**	0.17
	C-Eval(CV,Notears,Cosine,1)	0.33	0.31	$\underline{0.03}$	0.17	**0.11**
	C-Eval(X-Eval,Notears,Cosine,1)	0.19	0.21	0.05	0.08	**0.11**
	C-Eval(CV,Notears,Cosine,3)	0.33	0.28	0.11	0.21	$\underline{0.14}$
	C-Eval(X-Eval,Notears,Cosine,3)	0.12	0.13	**0.01**	$\underline{0.07}$	$\underline{0.14}$
	C-Eval(CV,Notears,Cosine,6)	0.33	0.27	0.15	0.23	0.15
	C-Eval(X-Eval,Notears,Cosine,6)	$\underline{0.09}$	$\underline{0.10}$	**0.01**	**0.06**	0.15
LightGBM	CV	0.31	0.11	0.22	0.32	$\underline{0.09}$
	X-Eval	**0.00**	0.09	0.07	0.16	0.12
	C-Eval(CV,Notears,Cosine,1)	0.28	0.14	0.12	0.21	**0.08**
	C-Eval(X-Eval,Notears,Cosine,1)	0.13	$\underline{0.04}$	**0.05**	**0.13**	0.1
	C-Eval(CV,Notears,Cosine,3)	0.29	0.12	0.17	0.27	$\underline{0.09}$
	C-Eval(X-Eval,Notears,Cosine,3)	0.06	**0.02**	$\underline{0.06}$	$\underline{0.15}$	0.11
	C-Eval(CV,Notears,Cosine,6)	0.30	0.12	0.19	0.29	$\underline{0.09}$
	C-Eval(X-Eval,Notears,Cosine,6)	$\underline{0.04}$	0.05	0.07	$\underline{0.15}$	0.11
CatBoost	CV	0.33	0.09	0.25	0.30	0.15
	X-Eval	**0.01**	0.09	0.08	0.07	0.21
	C-Eval(CV,Notears,Cosine,1)	0.28	0.13	0.09	0.14	**0.13**
	C-Eval(X-Eval,Notears,Cosine,1)	0.11	$\underline{0.04}$	**0.01**	**0.02**	0.16
	C-Eval(CV,Notears,Cosine,3)	0.30	0.11	0.17	0.22	$\underline{0.14}$
	C-Eval(X-Eval,Notears,Cosine,3)	0.06	**0.02**	$\underline{0.04}$	$\underline{0.05}$	0.18
	C-Eval(CV,Notears,Cosine,6)	0.31	0.10	0.20	0.25	0.15
	C-Eval(X-Eval,Notears,Cosine,6)	$\underline{0.04}$	0.05	0.06	0.06	0.19
TabNet	CV	0.10	0.12	$\underline{0.07}$	0.03	**0.01**
	X-Eval	**0.04**	0.10	**0.00**	0.05	$\underline{0.06}$
	C-Eval(CV,Notears,Cosine,1)	0.19	$\underline{0.04}$	0.35	0.07	0.22
	C-Eval(X-Eval,Notears,Cosine,1)	0.16	0.05	0.31	0.03	0.25
	C-Eval(CV,Notears,Cosine,3)	0.14	$\underline{0.04}$	0.21	0.05	0.11
	C-Eval(X-Eval,Notears,Cosine,3)	0.10	**0.02**	0.15	**0.01**	0.15
	C-Eval(CV,Notears,Cosine,6)	0.12	0.08	0.15	0.04	0.07
	C-Eval(X-Eval,Notears,Cosine,6)	$\underline{0.07}$	0.05	0.09	$\underline{0.02}$	0.12

Table E.3: Estimation error analysis for the classifiers XGBoost, LightGBM, CatBoost, and TabNet. We compared the proposed autoEval approaches (X-Eval and C-Eval) with the 10-fold cross-validation (CV). The C-Eval(*Estimatior*, *Direct*, *Jaccard*, λ) denotes the C-Eval approach with one *Estimatior* (i.e., CV or X-Eval), the **DirectLiNGAM** (*Direct*) causality inference algorithm, the **Jaccard** similarity function, and one λ factor (i.e., 1, 3, or 6). We highlighted the smallest estimation error for each dataset considering each classifier in **bold**, and we also underline the second smallest estimation error.

	Performance Estimator	**Source: Geriatrics Target: Neurology**	**Source: Neurology Target: Geriatrics**	**Source: HBP Target: HSL**	**Source: HSL Target: HBP**	**Source: Students2008 Target: Students2014**
XGBoost	CV	0.37	0.26	0.19	0.2	**0.16**
	X-Eval	0.05	0.06	$\underline{0.04}$	0.06	$\underline{0.17}$
	C-Eval(CV,*Direct*,Jaccard,1)	**0.00**	0.06	0.18	0.12	0.33
	C-Eval(X-Eval,*Direct*,Jaccard,1)	0.16	0.16	0.26	0.19	0.34
	C-Eval(CV,*Direct*,Jaccard,3)	0.19	0.1	**0.00**	$\underline{0.04}$	0.25
	C-Eval(X-Eval,*Direct*,Jaccard,3)	0.05	$\underline{0.05}$	0.11	0.06	0.25
	C-Eval(CV,*Direct*,Jaccard,6)	0.27	0.17	0.08	0.11	0.21
	C-Eval(X-Eval,*Direct*,Jaccard,6)	$\underline{0.01}$	**0.00**	0.05	**0.01**	0.22
LightGBM	CV	0.31	0.11	0.22	0.32	**0.09**
	X-Eval	**0.00**	0.09	0.07	0.16	$\underline{0.12}$
	C-Eval(CV,*Direct*,Jaccard,1)	0.07	0.19	0.15	$\underline{0.04}$	0.29
	C-Eval(X-Eval,*Direct*,Jaccard,1)	0.22	0.29	0.22	0.12	0.31
	C-Eval(CV,*Direct*,Jaccard,3)	0.12	$\underline{0.04}$	$\underline{0.03}$	0.14	0.19
	C-Eval(X-Eval,*Direct*,Jaccard,3)	0.11	0.19	0.08	**0.02**	0.21
	C-Eval(CV,*Direct*,Jaccard,6)	0.2	**0.02**	0.11	0.22	0.15
	C-Eval(X-Eval,*Direct*,Jaccard,6)	$\underline{0.06}$	0.14	**0.01**	0.08	0.17
CatBoost	CV	0.33	0.09	0.25	0.3	**0.15**
	X-Eval	**0.01**	0.09	0.08	0.07	$\underline{0.21}$
	C-Eval(CV,*Direct*,Jaccard,1)	$\underline{0.03}$	0.24	0.13	$\underline{0.03}$	0.34
	C-Eval(X-Eval,*Direct*,Jaccard,1)	0.2	0.32	0.22	0.15	0.37
	C-Eval(CV,*Direct*,Jaccard,3)	0.15	$\underline{0.08}$	$\underline{0.06}$	0.13	0.25
	C-Eval(X-Eval,*Direct*,Jaccard,3)	0.1	0.21	0.07	0.04	0.29
	C-Eval(CV,*Direct*,Jaccard,6)	0.23	**0.01**	0.14	0.2	$\underline{0.21}$
	C-Eval(X-Eval,*Direct*,Jaccard,6)	0.05	0.15	**0.01**	**0.01**	0.25
TabNet	CV	0.1	$\underline{0.12}$	0.07	**0.03**	**0.01**
	X-Eval	0.04	**0.1**	**0.00**	0.05	0.06
	C-Eval(CV,*Direct*,Jaccard,1)	0.13	0.31	0.09	0.28	0.04
	C-Eval(X-Eval,*Direct*,Jaccard,1)	0.16	0.3	0.05	0.24	0.07
	C-Eval(CV,*Direct*,Jaccard,3)	**0.02**	0.22	0.08	0.16	$\underline{0.02}$
	C-Eval(X-Eval,*Direct*,Jaccard,3)	0.06	0.2	0.02	0.1	0.07
	C-Eval(CV,*Direct*,Jaccard,6)	$\underline{0.03}$	0.18	0.08	0.1	$\underline{0.02}$
	C-Eval(X-Eval,*Direct*,Jaccard,6)	**0.02**	0.15	$\underline{0.01}$	$\underline{0.04}$	0.07

Table E.4: Estimation error analysis for the classifiers XGBoost, LightGBM, CatBoost, and TabNet. We compared the proposed autoEval approaches (X-Eval and C-Eval) with the 10-fold cross-validation (CV). The C-Eval($Estimatior$, $Notears$, $Jaccard$, λ) denotes the C-Eval approach with one *Estimatior* (i.e, CV or X-Eval), the **Notears** causality inference algorithm, the **Jaccard** similarity function, and one λ factor (i.e., 1, 3, or 6). We highlighted the smallest estimation error for each dataset considering each classifier in **bold**, and we also underline the second smallest estimation error.

	Performance Estimator	**Source: Geriatrics Target: Neurology**	**Source: Neurology Target: Geriatrics**	**Source: HBP Target: HSL**	**Source: HSL Target: HBP**	**Source: Students2008 Target: Students2014**
XGBoost	CV	0.37	0.26	0.19	0.20	**0.16**
	X-Eval	**0.05**	**0.06**	0.04	0.06	0.17
	C-Eval(CV,Notears,Jaccard,1)	0.26	0.24	0.06	0.09	0.19
	C-Eval(X-Eval,Notears,Jaccard,1)	0.12	0.14	0.13	**0.01**	0.20
	C-Eval(CV,Notears,Jaccard,3)	0.30	0.25	0.07	0.17	0.18
	C-Eval(X-Eval,Notears,Jaccard,3)	0.08	0.10	0.05	0.03	0.18
	C-Eval(CV,Notears,Jaccard,6)	0.31	0.26	0.12	0.20	0.17
	C-Eval(X-Eval,Notears,Jaccard,6)	0.07	0.08	**0.01**	0.04	0.18
LightGBM	CV	0.31	0.11	0.22	0.32	**0.09**
	X-Eval	**0.00**	0.09	0.07	0.16	0.12
	C-Eval(CV,Notears,Jaccard,1)	0.21	0.07	0.03	0.13	0.17
	C-Eval(X-Eval,Notears,Jaccard,1)	0.06	**0.03**	0.04	**0.05**	0.18
	C-Eval(CV,Notears,Jaccard,3)	0.26	0.09	0.13	0.23	0.13
	C-Eval(X-Eval,Notears,Jaccard,3)	0.03	0.06	**0.01**	0.10	0.15
	C-Eval(CV,Notears,Jaccard,6)	0.28	0.10	0.17	0.27	0.11
	C-Eval(X-Eval,Notears,Jaccard,6)	0.02	0.07	0.04	0.13	0.14
CatBoost	CV	0.33	0.09	0.25	0.30	**0.15**
	X-Eval	**0.01**	0.09	0.08	0.07	0.21
	C-Eval(CV,Notears,Jaccard,1)	0.20	0.06	**0.00**	0.04	0.21
	C-Eval(X-Eval,Notears,Jaccard,1)	0.04	**0.03**	0.08	0.08	0.24
	C-Eval(CV,Notears,Jaccard,3)	0.27	0.07	0.12	0.17	0.18
	C-Eval(X-Eval,Notears,Jaccard,3)	0.02	0.06	**0.00**	**0.00**	0.23
	C-Eval(CV,Notears,Jaccard,6)	0.29	0.08	0.18	0.22	0.17
	C-Eval(X-Eval,Notears,Jaccard,6)	0.02	0.07	0.03	0.03	0.22
TabNet	CV	0.10	0.12	0.07	0.03	**0.01**
	X-Eval	**0.04**	0.10	**0.00**	0.05	0.06
	C-Eval(CV,Notears,Jaccard,1)	0.12	0.03	0.25	0.17	0.13
	C-Eval(X-Eval,Notears,Jaccard,1)	0.09	**0.02**	0.21	0.13	0.16
	C-Eval(CV,Notears,Jaccard,3)	0.11	0.08	0.16	0.10	0.07
	C-Eval(X-Eval,Notears,Jaccard,3)	0.06	0.06	0.10	0.04	0.11
	C-Eval(CV,Notears,Jaccard,6)	0.10	0.10	0.12	0.07	0.04
	C-Eval(X-Eval,Notears,Jaccard,6)	0.05	0.07	0.06	**0.00**	0.09

Table E.5: Estimation error analysis for the classifiers XGBoost, LightGBM, CatBoost, and TabNet. We compared the proposed autoEval approaches (X-Eval and C-Eval) with the 10-fold cross-validation (CV). The C-Eval($Estimatior$, $Notears$, $MacroF1$, λ) denotes the C-Eval approach with one *Estimatior* (i.e, CV or X-Eval), the **Notears** causality inference algorithm, the **Macro F1** similarity function, and one λ factor (i.e., 1, 3, or 6). We highlighted the smallest estimation error for each dataset considering each classifier in **bold**, and we also underline the second smallest estimation error.

	Performance Estimator	**Source: Geriatrics Target: Neurology**	**Source: Neurology Target: Geriatrics**	**Source: HBP Target: HSL**	**Source: HSL Target: HBP**	**Source: Students2008 Target: Students2014**
XGBoost	CV	0.37	0.26	0.19	0.20	0.16
	X-Eval	**0.05**	**0.06**	**0.04**	**0.06**	0.17
	C-Eval(CV,Notears,MacroF1,1)	0.36	0.34	0.14	0.23	**0.04**
	C-Eval(X-Eval,Notears,MacroF1,1)	0.22	0.24	0.06	0.14	**0.04**
	C-Eval(CV,Notears,MacroF1,3)	0.35	0.30	0.17	0.24	0.10
	C-Eval(X-Eval,Notears,MacroF1,3)	0.14	0.15	0.05	0.10	0.10
	C-Eval(CV,Notears,MacroF1,6)	0.34	0.28	0.18	0.25	0.13
	C-Eval(X-Eval,Notears,MacroF1,6)	0.10	0.11	**0.04**	0.08	0.13
LightGBM	CV	0.31	0.11	0.22	0.32	0.09
	X-Eval	**0.00**	0.09	**0.07**	**0.16**	0.12
	C-Eval(CV,Notears,MacroF1,1)	0.32	0.17	0.19	0.28	**0.01**
	C-Eval(X-Eval,Notears,MacroF1,1)	0.16	0.08	0.12	0.19	0.02
	C-Eval(CV,Notears,MacroF1,3)	0.31	0.14	0.21	0.30	0.05
	C-Eval(X-Eval,Notears,MacroF1,3)	0.08	**0.01**	0.10	0.18	0.07
	C-Eval(CV,Notears,MacroF1,6)	0.31	0.13	0.21	0.31	0.07
	C-Eval(X-Eval,Notears,MacroF1,6)	0.05	0.04	0.09	0.17	0.09
CatBoost	CV	0.33	0.09	0.25	0.30	0.15
	X-Eval	**0.01**	0.09	**0.08**	**0.07**	0.21
	C-Eval(CV,Notears,MacroF1,1)	0.31	0.16	0.18	0.22	**0.06**
	C-Eval(X-Eval,Notears,MacroF1,1)	0.15	0.08	0.10	0.11	0.08
	C-Eval(CV,Notears,MacroF1,3)	0.32	0.12	0.22	0.26	0.11
	C-Eval(X-Eval,Notears,MacroF1,3)	0.08	**0.00**	0.09	0.09	0.15
	C-Eval(CV,Notears,MacroF1,6)	0.33	0.11	0.23	0.28	0.13
	C-Eval(X-Eval,Notears,MacroF1,6)	0.05	0.04	**0.08**	0.08	0.17
TabNet	CV	0.10	0.12	0.07	0.03	**0.01**
	X-Eval	**0.04**	0.10	**0.00**	0.05	0.06
	C-Eval(CV,Notears,MacroF1,1)	0.23	0.07	0.42	**0.01**	0.30
	C-Eval(X-Eval,Notears,MacroF1,1)	0.20	0.08	0.38	0.05	0.33
	C-Eval(CV,Notears,MacroF1,3)	0.16	0.03	0.24	**0.01**	0.16
	C-Eval(X-Eval,Notears,MacroF1,3)	0.12	**0.01**	0.19	0.05	0.20
	C-Eval(CV,Notears,MacroF1,6)	0.13	0.07	0.17	0.02	0.09
	C-Eval(X-Eval,Notears,MacroF1,6)	0.08	0.05	0.10	0.05	0.14

Table E.6: Estimation error analysis for the classifiers XGBoost, LightGBM, CatBoost, and TabNet. We compared the proposed autoEval approaches (X-Eval and C-Eval) with the 10-fold cross-validation (CV). The C-Eval(*Estimatior*, *PC*, *Cosine*, λ) denotes the C-Eval approach with one *Estimatior* (i.e, CV or X-Eval), the **PC** causality inference algorithm, the **Cosine** similarity function, and one λ factor (i.e., 1, 3, or 6). We highlighted the smallest estimation error for each dataset considering each classifier in **bold**, and we also underline the second smallest estimation error.

	Performance Estimator	**Source: Geriatrics Target: Neurology**	**Source: Neurology Target: Geriatrics**	**Source: HBP Target: HSL**	**Source: HSL Target: HBP**	**Source: Students2008 Target: Students2014**
XGBoost	CV	0.37	0.26	0.19	0.20	**0.16**
	X-Eval	0.05	0.06	0.04	0.06	0.17
	C-Eval(CV,PC,Cosine,1)	0.04	**0.01**	0.11	0.11	0.44
	C-Eval(X-Eval,PC,Cosine,1)	0.12	0.09	0.18	0.18	0.45
	C-Eval(CV,PC,Cosine,3)	0.21	0.14	0.04	0.05	0.30
	C-Eval(X-Eval,PC,Cosine,3)	0.03	**0.01**	0.07	0.06	0.31
	C-Eval(CV,PC,Cosine,6)	0.28	0.19	0.10	0.11	0.24
	C-Eval(X-Eval,PC,Cosine,6)	**0.00**	0.02	**0.03**	**0.01**	0.25
LightGBM	CV	0.31	0.11	0.22	0.32	**0.09**
	X-Eval	**0.00**	0.09	0.07	0.16	0.12
	C-Eval(CV,PC,Cosine,1)	0.03	0.12	0.21	0.14	0.42
	C-Eval(X-Eval,PC,Cosine,1)	0.18	0.22	0.29	0.22	0.43
	C-Eval(CV,PC,Cosine,3)	0.14	**0.01**	**0.00**	0.09	0.26
	C-Eval(X-Eval,PC,Cosine,3)	0.09	0.15	0.11	**0.03**	0.28
	C-Eval(CV,PC,Cosine,6)	0.21	0.04	0.10	0.19	0.19
	C-Eval(X-Eval,PC,Cosine,6)	0.05	0.12	0.03	0.05	0.21
CatBoost	CV	0.33	0.09	0.25	0.30	**0.15**
	X-Eval	**0.01**	0.09	0.08	0.07	0.21
	C-Eval(CV,PC,Cosine,1)	0.02	0.15	0.04	**0.01**	0.46
	C-Eval(X-Eval,PC,Cosine,1)	0.14	0.23	0.12	0.11	0.49
	C-Eval(CV,PC,Cosine,3)	0.18	0.03	0.10	0.15	0.31
	C-Eval(X-Eval,PC,Cosine,3)	0.07	0.16	**0.02**	0.02	0.35
	C-Eval(CV,PC,Cosine,6)	0.24	**0.02**	0.17	0.22	0.24
	C-Eval(X-Eval,PC,Cosine,6)	0.04	0.13	**0.02**	0.02	0.29
TabNet	CV	0.10	0.12	0.07	0.03	0.01
	X-Eval	0.04	**0.10**	**0.00**	0.05	0.06
	C-Eval(CV,PC,Cosine,1)	0.09	0.27	0.14	0.20	0.08
	C-Eval(X-Eval,PC,Cosine,1)	0.12	0.26	0.10	0.16	0.05
	C-Eval(CV,PC,Cosine,3)	**0.01**	0.20	0.10	0.12	0.04
	C-Eval(X-Eval,PC,Cosine,3)	0.04	0.18	0.05	0.06	**0.00**
	C-Eval(CV,PC,Cosine,6)	0.04	0.16	0.09	0.08	0.02
	C-Eval(X-Eval,PC,Cosine,6)	**0.01**	0.14	0.02	**0.01**	0.03

Table E.7: Estimation error analysis for the classifiers XGBoost, LightGBM, CatBoost, and TabNet. We compared the proposed autoEval approaches (X-Eval and C-Eval) with the 10-fold cross-validation (CV). The C-Eval(*Estimatior*, *PC*, *Jaccard*, λ) denotes the C-Eval approach with one *Estimatior* (i.e, CV or X-Eval), the **PC** causality inference algorithm, the **Jaccard** similarity function, and one λ factor (i.e., 1, 3, or 6). We highlighted the smallest estimation error for each dataset considering each classifier in **bold**, and we also underline the second smallest estimation error.

	Performance Estimator	**Source: Geriatrics Target: Neurology**	**Source: Neurology Target: Geriatrics**	**Source: HBP Target: HSL**	**Source: HSL Target: HBP**	**Source: Students2008 Target: Students2014**
XGBoost	CV	0.37	0.26	0.19	0.20	**0.16**
	X-Eval	0.05	0.06	0.04	0.06	0.17
	C-Eval(CV,PC,Jaccard,1)	**0.00**	0.03	0.16	0.16	0.44
	C-Eval(X-Eval,PC,Jaccard,1)	0.16	0.13	0.24	0.23	0.45
	C-Eval(CV,PC,Jaccard,3)	0.19	0.11	**0.01**	**0.02**	0.30
	C-Eval(X-Eval,PC,Jaccard,3)	0.05	0.04	0.10	0.08	0.31
	C-Eval(CV,PC,Jaccard,6)	0.27	0.18	0.09	0.10	0.24
	C-Eval(X-Eval,PC,Jaccard,6)	0.01	**0.00**	0.04	**0.02**	0.25
LightGBM	CV	0.31	0.11	0.22	0.32	**0.09**
	X-Eval	**0.00**	0.09	0.07	0.16	0.12
	C-Eval(CV,PC,Jaccard,1)	0.06	0.18	0.21	0.14	0.42
	C-Eval(X-Eval,PC,Jaccard,1)	0.21	0.28	0.29	0.22	0.43
	C-Eval(CV,PC,Jaccard,3)	0.12	**0.03**	**0.00**	0.09	0.26
	C-Eval(X-Eval,PC,Jaccard,3)	0.11	0.18	0.11	**0.03**	0.28
	C-Eval(CV,PC,Jaccard,6)	0.20	**0.03**	0.10	0.19	0.19
	C-Eval(X-Eval,PC,Jaccard,6)	0.06	0.14	0.03	0.05	0.21
CatBoost	CV	0.33	0.09	0.25	0.30	**0.15**
	X-Eval	**0.01**	0.09	0.08	0.07	0.21
	C-Eval(CV,PC,Jaccard,1)	0.03	0.20	0.10	0.06	0.46
	C-Eval(X-Eval,PC,Jaccard,1)	0.20	0.28	0.19	0.17	0.49
	C-Eval(CV,PC,Jaccard,3)	0.15	0.06	0.07	0.12	0.31
	C-Eval(X-Eval,PC,Jaccard,3)	0.10	0.18	0.06	0.05	0.35
	C-Eval(CV,PC,Jaccard,6)	0.23	**0.01**	0.15	0.20	0.24
	C-Eval(X-Eval,PC,Jaccard,6)	0.05	0.14	**0.00**	**0.00**	0.29
TabNet	CV	0.10	0.12	0.07	**0.03**	0.01
	X-Eval	0.04	**0.10**	**0.00**	0.05	0.06
	C-Eval(CV,PC,Jaccard,1)	0.13	0.31	0.08	0.27	0.08
	C-Eval(X-Eval,PC,Jaccard,1)	0.16	0.30	0.04	0.23	0.05
	C-Eval(CV,PC,Jaccard,3)	**0.02**	0.22	0.08	0.15	0.04
	C-Eval(X-Eval,PC,Jaccard,3)	0.06	0.20	0.02	0.09	**0.00**
	C-Eval(CV,PC,Jaccard,6)	0.03	0.18	0.08	0.10	0.02
	C-Eval(X-Eval,PC,Jaccard,6)	**0.02**	0.15	0.01	**0.03**	0.03

Table E.8: Estimation error analysis for the classifiers XGBoost, LightGBM, CatBoost, and TabNet. We compared the proposed autoEval approaches (X-Eval and C-Eval) with the 10-fold cross-validation (CV). The C-Eval($Estimatior$, PC, $MacroF1$, λ) denotes the C-Eval approach with one $Estimatior$ (i.e, CV or X-Eval), the **PC** causality inference algorithm, the **Macro F1** similarity function, and one λ factor (i.e., 1, 3, or 6). We highlighted the smallest estimation error for each dataset considering each classifier in **bold**, and we also underline the second smallest estimation error.

	Performance Estimator	**Source: Geriatrics Target: Neurology**	**Source: Neurology Target: Geriatrics**	**Source: HBP Target: HSL**	**Source: HSL Target: HBP**	**Source: Students2008 Target: Students2014**
XGBoost	CV	0.37	0.26	0.19	0.20	**0.16**
	X-Eval	**0.05**	**0.06**	0.04	0.06	0.17
	C-Eval(CV,PC,MacroF1,1)	0.24	0.20	0.06	0.07	0.23
	C-Eval(X-Eval,PC,MacroF1,1)	0.07	0.10	0.02	**0.00**	0.23
	C-Eval(CV,PC,MacroF1,3)	0.30	0.23	0.12	0.14	0.19
	C-Eval(X-Eval,PC,MacroF1,3)	0.06	0.08	**0.01**	0.03	0.20
	C-Eval(CV,PC,MacroF1,6)	0.33	0.24	0.15	0.16	0.18
	C-Eval(X-Eval,PC,MacroF1,6)	0.06	0.07	0.02	0.04	0.19
LightGBM	CV	0.31	0.11	0.22	0.32	**0.09**
	X-Eval	**0.00**	0.09	0.07	0.16	0.12
	C-Eval(CV,PC,MacroF1,1)	0.17	**0.05**	0.03	0.09	0.20
	C-Eval(X-Eval,PC,MacroF1,1)	0.02	**0.05**	0.05	**0.01**	0.21
	C-Eval(CV,PC,MacroF1,3)	0.24	0.08	0.12	0.21	0.15
	C-Eval(X-Eval,PC,MacroF1,3)	0.01	0.07	**0.01**	0.09	0.17
	C-Eval(CV,PC,MacroF1,6)	0.27	0.09	0.16	0.25	0.12
	C-Eval(X-Eval,PC,MacroF1,6)	0.01	0.07	0.04	0.12	0.15
CatBoost	CV	0.33	0.09	0.25	0.30	**0.15**
	X-Eval	**0.01**	0.09	0.08	0.07	0.21
	C-Eval(CV,PC,MacroF1,1)	0.20	**0.04**	0.11	0.15	0.25
	C-Eval(X-Eval,PC,MacroF1,1)	0.03	0.05	**0.03**	**0.04**	0.27
	C-Eval(CV,PC,MacroF1,3)	0.26	0.06	0.18	0.23	0.20
	C-Eval(X-Eval,PC,MacroF1,3)	0.02	0.07	0.05	0.06	0.24
	C-Eval(CV,PC,MacroF1,6)	0.29	0.07	0.21	0.26	0.18
	C-Eval(X-Eval,PC,MacroF1,6)	**0.01**	0.08	0.06	0.06	0.23
TabNet	CV	0.10	0.12	0.07	0.03	**0.01**
	X-Eval	**0.04**	0.10	**0.00**	0.05	0.06
	C-Eval(CV,PC,MacroF1,1)	0.10	0.08	0.30	0.05	0.13
	C-Eval(X-Eval,PC,MacroF1,1)	0.07	**0.06**	0.26	0.02	0.16
	C-Eval(CV,PC,MacroF1,3)	0.10	0.10	0.19	0.04	0.07
	C-Eval(X-Eval,PC,MacroF1,3)	0.05	0.08	0.13	**0.01**	0.11
	C-Eval(CV,PC,MacroF1,6)	0.10	0.11	0.14	0.04	0.04
	C-Eval(X-Eval,PC,MacroF1,6)	0.05	0.09	0.07	0.03	0.09