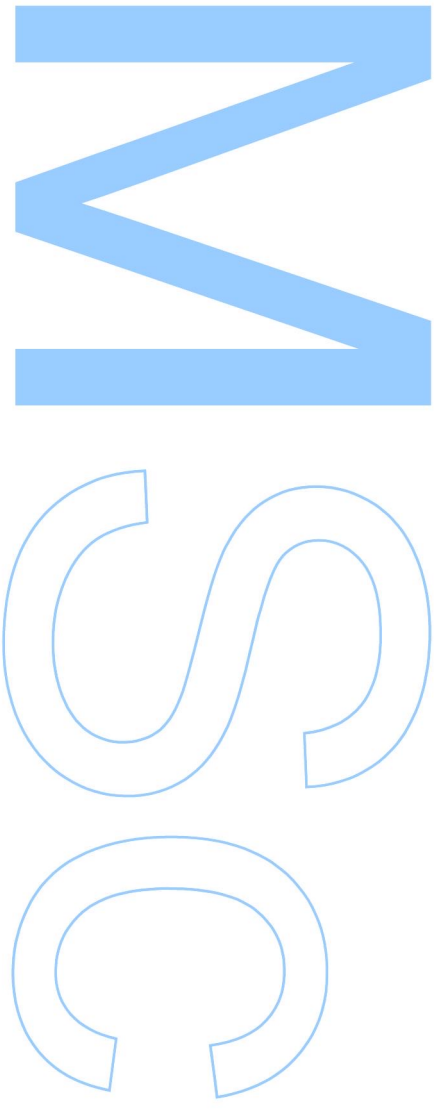
Data Leakage Detection with anti-causal learning

Margarida Antunes da Costa
Master's degree in Computer Science
Departamento de Ciência de Computadores
2022

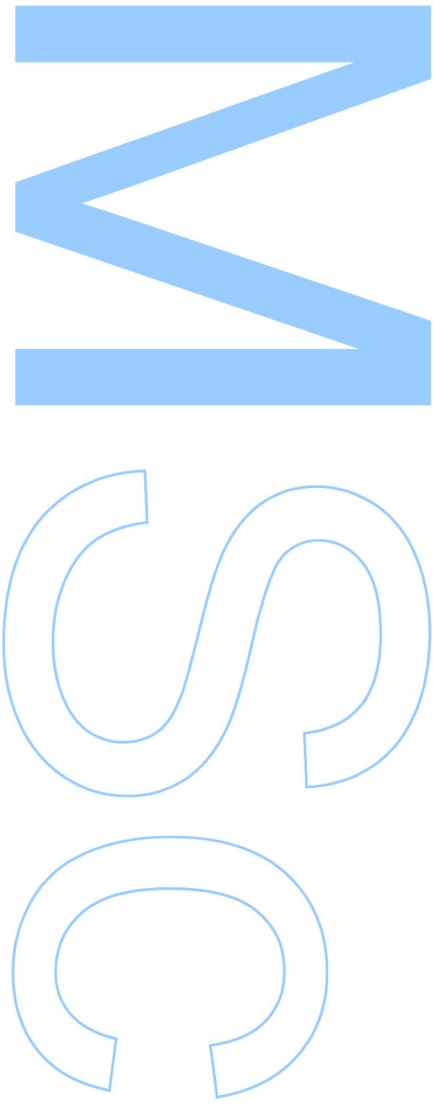
Orientador
João Nuno Vinagre Marques da Silva, Professor Auxiliar Convidado,
Faculdade de Ciências da Universidade do Porto



Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/_____/_____



Declaração de Honra

Eu, Margarida Antunes da Costa, inscrito(a) no Mestrado em Ciência de Computadores da Faculdade de Ciências da Universidade do Porto declaro, nos termos do disposto na alínea a) do artigo 14.º do Código Ético de Conduta Académica da U.Porto, que o conteúdo da presente dissertação de projeto reflete as perspetivas, o trabalho de investigação e as minhas interpretações no momento da sua entrega.

Ao entregar esta dissertação de projeto, declaro, ainda, que a mesma é resultado do meu próprio trabalho de investigação e contém contributos que não foram utilizados previamente noutros trabalhos apresentados a esta ou outra instituição.

Mais declaro que todas as referências a outros autores respeitam escrupulosamente as regras da atribuição, encontrando-se devidamente citadas no corpo do texto e identificadas na secção de referências bibliográficas. Não são divulgados na presente dissertação de projeto quaisquer conteúdos cuja reprodução esteja vedada por direitos de autor.

Tenho consciência de que a prática de plágio e auto-plágio constitui um ilícito académico.

Margarida Costa

Porto, 30 de setembro de 2022

Abstract

In today's world, the amount of available data is growing faster than the ability to extract knowledge from it. As a result, it is now more challenging to keep the data reliable during the data collection, storage and processing, leading to problems such as noise, missing values and data leakage, among others. In this dissertation we focus on data leakage. This is a problem in supervised learning tasks that happens when information that can only be known after the true value of the target is revealed is present in predictor variables. This artificially boosts a model's performance offline. When the same model is taken to the real-world, it performs much worse. Surprisingly, there is very little research on leakage detection in the literature. For this reason, we decided to study and delve further. Our basis of the study was the research of Schölkopf et al. [1], where the authors show that `Semi-supervised Learning` can work in the anti-causal direction, but not in the causal direction. In other words, it does not work when the target variable is a causal effect of the predictive variables.

Building on this principle, we propose a method for detecting *leakage*. In essence, the method is based on the assumption that if semi-supervised works on datasets or streams, this means that there is some predictor(s) that have information that is an effect of the target. If semi-supervised learning works on a dataset that is known to be causal, the only explanation is that the predictors have leaked information from the target.

We use multiple datasets in classification and regression tasks, in batch and stream learning settings, and manipulate them by artificially introducing leakage. We then apply a method using the principle described above. Our results show that our method can be used to detect leakage in all settings. In addition to this, we develop a stream-based `Co-Training` algorithm that would allow us to split the set of attributes according to the assumptions of its creators [2].

Our contributions include code for reproducing results that can be useful in detecting *leakage* and creating an algorithm for splitting the set of attributes, using the key concept of conditional mutual information in the `Co-Training` technique.

Keywords: Machine Learning; Data Processing; Causality; Leakage; Semi-supervised Learning; Co-Training; Conditional Mutual Information;

Resumo

No mundo de hoje, a quantidade de dados disponíveis é maior do que a capacidade de extrair conhecimento deles. Como resultado, agora é mais desafiador manter os dados confiáveis durante a extração, armazenamento e processamento de dados, levando a problemas como *noise*, valores ausentes e *data leakage*, entre outros.

Este é um problema em tarefas de *machine learning* supervisionada que acontece quando informações que só podem ser conhecidas após a revelação do verdadeiro valor do alvo estão presentes em variáveis preditivas. Isso aumenta artificialmente o desempenho offline dum modelo. Quando o mesmo modelo é levado para o mundo real, ele tem um desempenho muito pior.

Surpreendentemente, há muito pouca pesquisa sobre detecção de *leakages* na literatura. Por esse motivo, decidimos estudar e aprofundar. A nossa base de estudo foi a pesquisa de Schölkopf et al. [1], onde os autores mostram que `Aprendizagem Semi-supervisionada` pode trabalhar na direção anti-causal, mas não na direção causal . Em outras palavras, não funciona quando a variável alvo é um efeito causal das variáveis preditivas.

Com base neste princípio, propomos um método para detectar *leakages*. Em essência, o método é baseado na suposição de que, se a aprendizagem semi-supervisionada funcionar em conjuntos de dados ou fluxos, isso significa que existem alguns preditores que possuem informações que são um efeito da variável-alvo. Caso funcione num conjunto de dados conhecido como causal, a única explicação é que os preditores contêm informações do alvo.

Nas nossas experiências usamos vários conjuntos de dados em tarefas de classificação e regressão, em diferentes configurações tanto em *batch* como em *stream*, e os manipulamos introduzindo *leakage* artificialmente. Em seguida, aplicamos um método usando o princípio descrito acima. Nossos resultados mostram que nosso método pode ser usado para detectar *leakages* em todas as configurações. Além disso, desenvolvemos um algoritmo `Co-Training` baseado em stream que nos permitiria dividir o conjunto de atributos de acordo com as premissas de seus criadores [2].

As nossas contribuições incluem código para reproduzir resultados que podem ser úteis na detecção de *leakage* e na criação dum algoritmo para dividir o conjunto de atributos, usando o conceito chave de informação mútua condicional na técnica `Co-Training`.

Palavras-chave: Machine Learning; Processamento dos Dados; Leakage; Causalidade;

Semi-supervised Learning; Co-Training; Informação Mútua Condicional;

Acknowledgments

This dissertation marks the end of my journey as a master's student at DCC-FCUP. This path was accomplished with some adversities and constant challenges, but that allowed me to always learn more and be curious in several areas of computing. For this, a thank you to the people that are part of it and helped in my path.

First of all, I would like to thank the person who helped, taught and guided me in this dissertation, Professor João Vinagre, who showed great patience, availability and always willingness to increase my knowledge. I would also like to thank Professor Ana Paula Tomás, who, even not being my supervisor, was available to transmit her knowledge and help me.

I am also grateful for the friends and boyfriend I met on this incredible journey, who never let my frustrations and uncertainties influence my academic path.

Last but not least, I have to thank my family. Without them, I would not have the opportunities that I have been provided throughout my life, which allowed me to be here today, completing another stage in my academic journey. In addition to the opportunities provided, they also never let me give up, so I hope they will be proud of my work.

This work is supported by European Structural and Investment Funds in the FEDER component, through the Operational Competitiveness and Internationalization Programme (COMPETE 2020) [Project nº 047264; Funding Reference: POCI-01-0247-FEDER-047264].

To my family, boyfriend, friends and supervisor, who always supported me during this long path...

Contents

List of Tables

List of Figures

Acronyms

AUC ROC Area Under the ROC Curve

CondMI Class-conditional Mutual Information

F1 F1 Score

RF Random Forest

ML Machine Learning

MAE Mean Absolute Error

MSE Mean Square Error

ICM Independent Causal Mechanism

InterCMI Inter-Conditional Mutual Information

IntraCMI Intra-Conditional Mutual Information

MIP Mixed-Integer Programming

Prec Precision

Rec Recall

RMSE Root Mean Square Error

R^2 R Square

SSL Semi-supervised Learning

SCM Structural Causal Model

SL Supervised Learning

Chapter 1

Introduction

Over the last few years, the collection of large amounts of data has increased. As a result, it has become more difficult to keep data free of negative artifacts, such as noise, lack of integrity, missing values and leakage, introduced in previous stages of data collection, processing, transfer and storage. At the same time, the rapid development of data collection and extraction techniques makes it possible to achieve higher levels of research and study.

Data leakage, in particular, is considered "one of the top ten data mining mistakes"[3]. It consists of the existence of information about the target variable of a problem that should not be normally available. For example, assuming we have a model that predicts whether or not a patient is infected with HIV and in the data, there is a variable indicating whether or not the patient has taken the HIV antibiotic. It would be possible for the machine to use that data to correctly predict a large part of the cases instead of focusing on the variables it will have access to in a real case.

In the context of this significant data era and the emergence of various problems such as data leakage, we examine datasets and discover the dependency relationships between different variables with the target variable using semi-supervised learning methods, exploiting the causal structure of the prediction problem. With this new approach, we intend to eliminate data leakage, make the models more reliable, and achieve better results.

1.1 Objectives

The main aim is to develop novel methods to detect and prevent data leakage in machine learning using semi-supervised learning.

To achieve this objective we shall:

- Review the state-of-the-art in data leakage detection and prevention;
- Collect multiple datasets for supervised learning (classification and regression) with a

known causal structure;

- Use this causal structure to distinguish between "leaked" and "clean" features in batch and streaming settings.

1.2 Contributions

Our contributions involve a review of the state-of-the-art in leakage detection and a methodology to systematically identify variables with leakages in datasets, both in batch and stream. We obtained a set of experiments with ten datasets in regression problems and ten datasets in classification problems.

We also contributed a code to reproduce results that could be the basis for a leakage detection tool.

Finally, our contribution also involves the creation of an algorithm to achieve the ideal split of the feature set into two subsets per the presumptions established by the co-training technique's creators.

1.3 Organization

This dissertation is organised as follows: in Chapter 2, we explain the core concepts of machine learning and causality. Then, we present an overview of machine learning workflow where we talk about data pre-processing, different methods of evaluation of models and learning techniques - supervised, unsupervised and reinforcement learning. Then, in a more specific way, we approach the semi-supervised learning technique and its several approaches: self-training, co-training and graph-label propagation. Also, in this chapter, we explore the literature on existing leakage detection and also semi-supervised learning. Looking at these researches allows us to validate the results obtained.

Then, in Chapter 3, we explore twenty datasets, with each ten representing different types of problems, regression and classification. Leakage imputation on the variables allows us to analyse their behaviour in the models. In this chapter, we also describe the experimental setting and prove that our approach has a relevant contribution. We also research an algorithm for the optimal division of the feature set into two subsets to advantage the semi-supervised learning technique, co-training better.

In Chapter 4, we expose our results and briefly discuss them.

Finally, in Chapter 5, we exhibit a summary of the work done and the limitations found, thus allowing possible future work.

Chapter 2

Background and State of the art

A machine learning algorithm is implemented to detect data leakage, exploring the concept of causality, which uses the semi-supervised learning strategy.

In this chapter, a systematic review of machine learning will be carried out to understand better how the process works. In addition, the concept of causality in the machine learning community will also be addressed.

2.1 Machine Learning

Machine Learning (ML) is an area of computing, more precisely artificial intelligence, that refers to a wide range of algorithms that perform inference and prediction on datasets, which can be static or streaming. Inference deals with drawing insights, i.e. inferring the causes of events and behaviours, and prediction contributes to the outcome of future events.

This branch reaches a human level of semantic understanding, information extraction and pattern detection as it can learn rules from existing data.

A ML model (Fig.2.1) is a statistical representation of a process in an environment, which receives a set of data as input and returns a result with a certain probability. For example, if we request to predict the veracity of a written tweeter, we can create a model that receives a set of tweets.

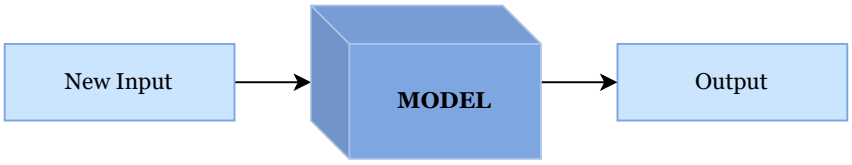


Figure 2.1: Representation of machine learning model

Classical machine learning is often classified by how the algorithm learns to be more accurate

in its predictions. It can be divided into four strategies: supervised, unsupervised, semi-supervised, and reinforcement learning, in Fig. 2.2 [4].

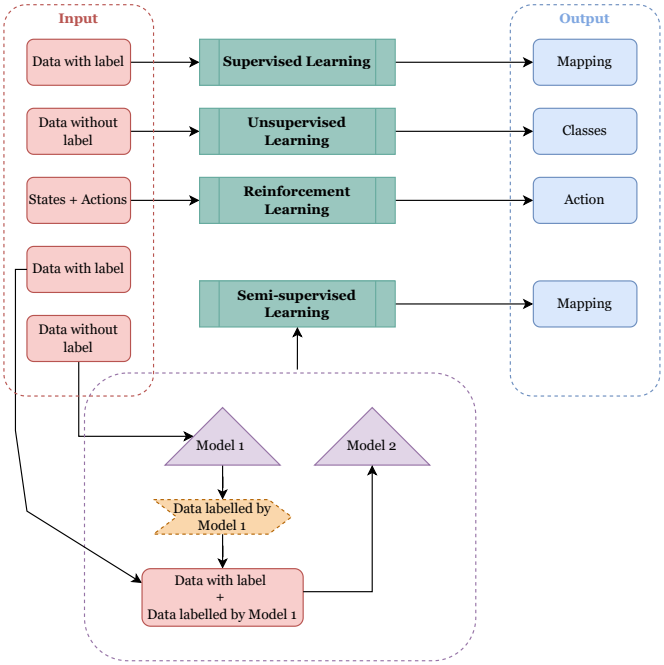


Figure 2.2: Supervised Learning vs Unsupervised Learning vs Reinforcement Learning vs Semi-Supervised Learning – Overview

2.1.1 Supervised Learning

In Supervised Learning (SL), the dataset used contains all its labelled examples [4][5], $(x_i, y_i)_{i=0}^{N-1}$, where x_i is a predictive variable, being part of a feature vector, x and y_i is the label which can be an element belonging to a finite set of classes. It can be an actual number or a more complex structure, e.g. a graph or a vector. Each feature vector, x, has a corresponding label, y. For example, each x represents the water potability verification dataset, which has several features, like x_0: ph, x_1: chloramines level, x_2: sulfates level, being the label, y: the water is potable, divided into two classes

$$\begin{cases} y_0 = \text{No} \\ y_1 = \text{Yes} \end{cases}$$

The goal of an SL algorithm is to use the dataset to produce a model that takes a feature vector, x, as input and output information that allows the label y to be deduced. Formally, the algorithm searches for a function f: $X \to Y$, where X is the feature vector input, Y is the class set to output and $f \in F$ (F is the hypothesis space).

This strategy can be divided into two major tasks, depending on the dataset used: Classification and Regression. The classification task is suitable for predictions of discrete values, while

the regression task establishes predictions of continuous variables. Classification is the technique of finding a function that helps categorize data into classes based on the parameters found in the feature vector. A subdivision can still be made, but according to the number of different classes: Binary Classifier and Multi-class Classifier. Regression is the method that looks for correlations between variables, thus verifying their dependence or independence.

2.1.2 Unsupervised Learning

In unsupervised learning, the dataset has no labelled examples [4][6], ${x_i}_{i=0}^{N-1}$, where x is the feature vector. This learning aims to create a model that takes the feature vector x as input and looks for interesting patterns in the sample data. There are different forms of unsupervised learning:

- Clustering: returns the cluster for each feature vector in the dataset;
- Dimensionality reduction: returns a feature vector, x' but with fewer features than the initial one, x;
- Outlier detection: returns a value that indicates how x takes different values from the other examples.

2.1.3 Reinforcement Learning

Reinforcement Learning [4] is when the model interacts with the environment throughout the runtime, represented by a state, the feature vector. In each state, different actions can be executed that have different rewards.

The goal is the learning of a function that captures the feature vector of a state as input and performs the optimal action to execute it. This function is called policy. An action is considered optimal when it allows the maximization of the given reward.

2.1.4 Semi-supervised Learning

Semi-supervised Learning (SSL) deals with labelled and unlabelled data in a dataset [4][7][8][9]. In traditional semi-supervised learning, called inductive semi-supervised learning, the algorithm takes in the training dataset labelled examples $(x_i, y_i)_{i=0}^{N-1}$ and unlabelled examples ${x_i}_{i=0}^{N-1}$ and learns from f: $X \to Y$, $f \in F$ where F is the hypothesis space. The goal is to learn a model that predicts better future test data than a model learned by fully labelled training data. Another form of semi-supervised learning is transductive learning, where they use the labelled training data to predict fully labelled test data by a rule of thumb. The goal is to find the labels without building an f-function, as their output is just a vector of labels through patterns and additional information present. However, this technique presents a constraint, according to A. Gammerman

et al. [10] if the interest is in predicting a particular example and not in a general rule for all test examples.

The essential prerequisite for SSL to be more meaningful than SL is that the knowledge about $P(X)$ gained through the unlabeled data must contain valuable information in inferring $P(Y|X)$. Otherwise, semi-supervised learning will not produce better rankings relative to supervised learning.

Some assumptions must be met for semi-supervised learning to work:

- Semi-supervised Smoothness Assumption: Data points closer to each other are more likely to have the same output label;
- Cluster Assumption: Data points in the same cluster are more likely to share an output level;
- Low-density Separation Assumption: Data points are located in sparse, low-density regions, thus forming the decision boundary between classes;
- Manifold Assumption: The data lie approximately on a manifold of a much lower dimension than the input space.

The first assumption is that the data should be capable of generalisation from a small number of labels. Data from two examples near, located in dense regions of the learning space, are very likely to have equal labels; otherwise, semi-supervised learning would be impeded due to the nature of the data.

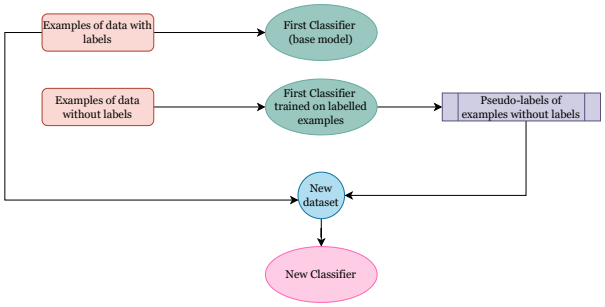
The others refer to conditions that can fill in the unlabelled data examples and under which a generalisation can be made from a small number of labelled examples. These assumptions differ where the data are located to label the examples: the data are grouped into homogeneous classes or dense spaces.

Several semi-supervised learning techniques differ in how the model predicts unlabeled instances: self-training, co-training and graph-label propagation.

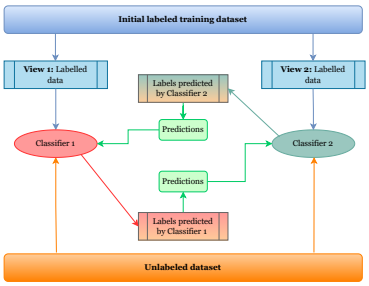
2.1.4.1 Self-training

In the semi-supervised learning technique, Fig. 2.3(a), self-training, a base learner is first trained using only the labelled data. The classifier is then used to label a randomly selected sample from the unlabeled pool using only the labelled data. The most accurately identified examples from these freshly labelled cases are added to the labelled set, and the classifier retrains on this expanded labelled set. More unlabelled samples are classified and then employed in retraining at each iterative stage in the process.

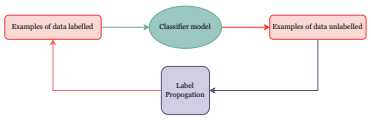
Traditional semi-supervised learning employs this methodology.



(a) Self-training



(b) Co-training



(c) Graph-based label propagation

Figure 2.3: SSL's approaches

2.1.4.2 Co-training

Blum and Michell [2], who created co-training, assumed that two naturally occurring learning perspectives are distinct. A classifier, in Fig. 2.3(b) is used in each view of the labelled and unlabelled data. The co-training algorithm chooses high-confidence samples scored by each classifier to create a self-labelled dataset in each iteration. Then, labelled data and an extra set of self-labelled data are added to the other classifier.

As a result, the co-training approaches choose unlabeled samples with high confidence that the trained models would correctly predict them.

Co-training developed [2] has to follow two assumptions:

1. Sufficiency assumption: each view is sufficient to predict the class perfectly;
2. Independence assumption: the views are conditionally independent; that is, they are independent given a class.

Criteria of Feature Splits: The views and the splitting of features are determining factors in co-training performance. In a study by Nigam et al. [11] they concluded that co-training is better when views are truly independent rather than randomly split. However, Feger et al. [12] stated that the accuracy of co-training classifiers does not improve with views truly independent.

The measure used for feature splitting is the class-conditional mutual information [13], based on the research of Feger et al. Class-conditional Mutual Information (CondMI) is the reduction in uncertainty about a variable X that we learn from another variable Y, given the value of another variable. That is the expected value of mutual information of two variables given the value of another. Mutual information is a way to calculate how much one variable tells us about

the other in units of bits.

$$\text{CondMI}(X,Y) = H(X|Z) - H(X|Y,Z) = \sum_{x\in X}\sum_{y\in Y} p(x,y|Z=c) log_2 \frac{p(x,y|Z=c)}{p(x|Z=c)p(y|Z=c)} \tag{2.1}$$

In the Equation 2.1, Z=c means that the class of z is equal to c. Therefore, if the result is close to 0, we can conclude that the variables X and Y are independent.

Based on CondMI(X, Y), it is possible to evaluate the dependency between the two views and the intra-view. The dependency between views calculates summing the CondMI between the different attributes of the view V_1 and the attributes of the other view V_2, Equation 2.2. The smaller the value of Inter-Conditional Mutual Information (InterCMI) is, the dependency between views is also more minor. The intra-view dependency V_i is the sum of the CondMI between the different attributes present in the view, Equation 2.3. The larger the Intra-Conditional Mutual Information (IntraCMI) is, the more dependency of the view is prominent.

$$\text{InterCMI}(V_1,V_2) = \sum_{A_1\in V_1}\sum_{A_2\in V_2} CondMI(A_1,A_2) \tag{2.2}$$

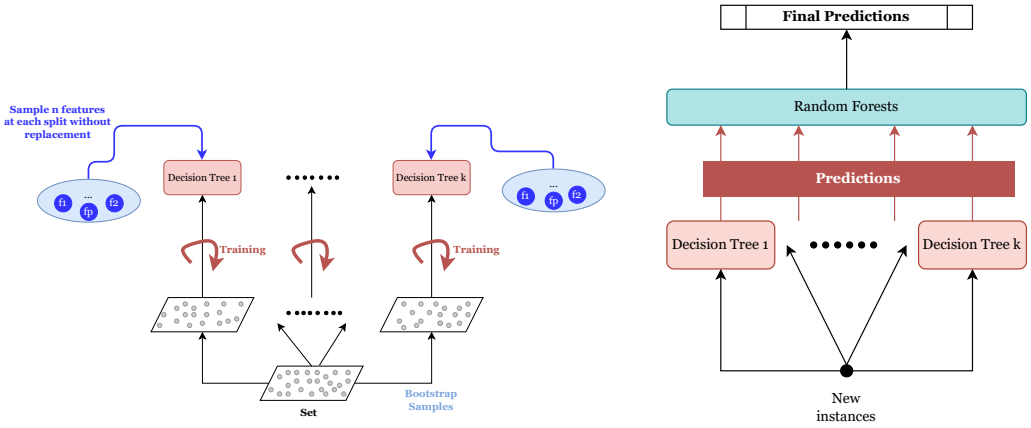
where A_1 and A_2 are the attributes present in the views V_1 and V_2, respectively.

$$\text{IntraCMI}(V_i) = \sum_{X,Y\in V_i} CondMI(X,Y) \tag{2.3}$$

Jun et al. [14] and Feter et al. [12] demonstrate five different methods for splitting features, shown in Table 2.1.

Splitting Methods	Start	How its works
Random Split [14]	Single View	Split into two views randomly for 20 times for each dataset. Choose the best combination.
Entropy Split [14]	Single View	Calculate the entropy of each feature in the single view based on the whole dataset. Sort the features in descending order of entropy value. Create a heuristic - The attributes of the odd positions go to the first view and the attributes of the even positions go to the second view.
Entropy-Start Hill Climbing [14]	Split features into two views based on Entropy Split	Obtain a group of new generated split, by exchanging a feature of one view with another. once time. Repeat process, until the split is not altered from the last iteration.
Random-Restart Hill Climbing [14]	Random Splits instead of only one deterministic Entropy Split	Same Entropy-Start Hill Climbing. Obtain the best split comparing all 20 hill climbing searches.
MaxInd Split [12]	Undirected graph, with features as vertices and the CondMI between two features as weight on the edge	Cut the graph into two disjoint sets of the same size. This split minimize the dependence between the two parts of the graph making the sum of the cut edges minimal, by using a graph partition heuristic [15]. The result of this graph cut is then used as feature split.

Table 2.1: Description of splitting methods proposed



(a) Random Forests: Training (b) Random Forests: Prediction

Figure 2.4: Random Forests' Steps

2.1.4.3 Graph-based Label Propagation

The idea behind graph-based, Fig. 2.3(c) is to construct a graph connecting similar data points where some nodes are labelled and many are unlabelled, [16]. Next, each hidden or observable label is considered a random variable at the nodes of this graph that is interpreted as an undirected graph model or Markov Random Field. So if an edge contains a high weight, then it means that its two nodes will have similar labels.

Iteratively, for any labelled node, we can propagate its label to the neighbouring unlabeled nodes according to its weight. The final prediction for each unlabelled node is when the labels on the nodes with no labels reach an equilibrium.

This work will only focus on semi-supervised learning, using self-training for batch data and co-training for data streams.

2.1.5 Algorithm used - Random Forests

The Random Forest (RF) algorithm is an ensemble method that uses a decision tree as the base estimator. Each estimator is trained on a different bootstrap sample of the same size as the dataset, allowing the introduction of randomization. This randomization searches for the best feature among a random subset of features instead of searching for only one feature.

This algorithm is mighty because it limits overfitting without substantially increasing the error due to bias due to using numerous trees.

The Random Forest algorithm has two major steps:

- Training, Fig. 2.4(a): Each tree forming the set is trained on a different bootstrap sample. Furthermore, when a tree is trained, at each node, only n features are sampled from all

features without replacement. To maximize the information gain, split the node.

- Prediction, Fig. 2.4(b): Once trained, predictions are made on new instances, where each is fed to the different base estimators, producing a prediction. These predictions are collected by the random forests metaclassifier. Then a final prediction is made depending on the nature of the problem: if it is classification, the final prediction is the majority vote; if it is regression, the final prediction is the average of all labels predicted by the base estimators.

Random Forest was the algorithm chosen for its ease of implementation and performance for the development of this dissertation.

2.1.5.1 Adaptive Random Forest

Traditional RandomForests require numerous transactions on the input data in order to create bootstraps for each tree. In learning data streams, it is not possible to perform these transactions. Therefore, according to Gomes, Heitor Murilo et al. [17] an adaptation of random forest depend:

1. On an online bootstrap aggregation process;
2. A limitation of each leaf splitting decision to a subset of features.

The first need is to change the type of distribution that bootstrap sampling follows. On static data, each tree is trained on a bootstraps sample. When there is a small amount of data, the sample follows a binomial distribution [18]; if the amount of data is large, it switches to a poison distribution with a mean equal to 1, Poison(λ =1). On streaming data, the mean becomes equal to λ =6, having the practical effect of increasing the probability of assigning higher weights to instances while the model is trained. The second restricts the set of features considered for the divisions of the future random subsets by modifying the base tree algorithm.

This algorithm trains background trees, which start training if a warning is detected during training. If the warning intensifies, they replace the active tree.

2.1.6 Encoding of Categorical Variables

Several machine learning algorithms, like Random Forest, work only with numeric attributes, so we must encode categorical attributes.

There are several different approaches to this encoding. However, not all have been explored.

One approach is Label Encoding [19], which involves converting each categorical value in a column into a numerical value.

The other approach is Target Encoding [20] which encodes the categories, changing them to a measurement of their effect on the target variable. Independently encodes each feature value for each target value.

The other existing approaches are one-hot encoding [19], dummy encoding [21] and feature hashing encoding [22].

2.1.7 Feature Scaling

Feature Scaling is a technique for normalizing independent features in a fixed range. It is performed at the pre-processing data stage, dealing with high magnitudes between variables. If this is not performed, the learning algorithm looks at the higher values as more relevant than the lower values for model training, regardless of the unit. Scaling can significantly differentiate between a weak and a better model.

There are several ways of scaling columns [23], like MaxAbs Scaler, Robust Scaler, Quantile Transformer Scaler, Power Transform Scaler and Unit Vector Scaler. Besides these, there is one that was used in the development of the leakage detection method: MinMax Scaler and Standard Scaler.

The MinMax Scaler transforms the columns, scaling each in a given interval.

$$X_{new} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{2.4}$$

This estimator scales each feature so that it is in the same range, reducing the data within the range from -1 to 1 if there are negative values. This is good on data where the standard deviation is low and does not have a Gaussian/normal distribution.

Standard Scaler follows a distribution with a mean equal to 0 and a standard deviation equal to 1. It assumes that the data are normally distributed.

$$X_{new} = \frac{x - \mu}{\sigma} \tag{2.5}$$

Centring and scaling occur independently on each feature by calculating the relevant statistics over the samples in the set. This scaler was used on the streams' data since, in the long run, not knowing these means and previous standard deviations is not dire for the learning algorithm's performance.

2.1.8 Model Evaluation

2.1.8.1 Evaluation Metrics

In creating machine learning models, an essential step is the evaluation of the implemented models to visualise their differences in their prediction. There are several conventional metrics for this evaluation, or we can create our own, depending on the problem at hand. The choice of metric can significantly influence the system, differing in the type of task: classification or regression.

In the classification context, the metrics used are Accuracy, Precision (Prec), Recall (Rec), F_1 Score, Log Loss and Area Under the ROC Curve (AUC ROC),[24]. Accuracy is the ratio of the number of predictions hit to the number of predictions made, Equation 2.6. This should be used on balanced datasets with no class imbalances, so it is not a good choice when target values are sparse. Precision is the ratio of true positives to positive values, Equation 2.7, and is useful when we want to be sure of our prediction. Another measure is Recall which answers the question of how many positives, Equation 2.8 have been correctly classified, allowing us to capture a larger number of positives. The fact that these two are very useful led to the creation of another one which is the F_1 score score, calculated by Equation 2.9. It is a harmonic mean of precision and recall, thus maintaining a balance between precision and recall: when precision increases, the $F_1 score$ also increases and when recall increases, the F1 score also increases. Rahul [24] mentions a trivial example that illustrates a good situation about its use: "If you are a police inspector and you want to catch criminals, you want to be sure that the person you catch is a criminal (Precision) and you also want to capture as many criminals (Recall) as possible. The F1 score manages this tradeoff". However, some problems prefer more importance to recall or precision, so the F1 score had to be adapted by adding weight,β, F1 Score beta, Equation 2.10.

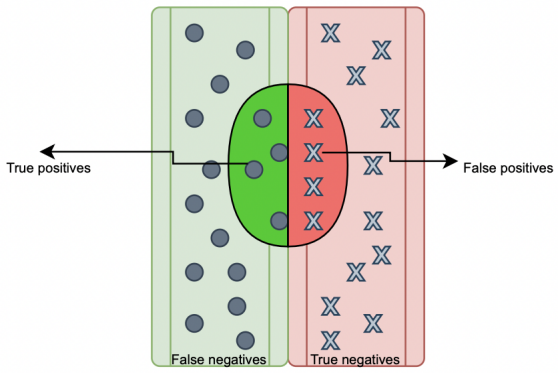


Figure 2.5: Diagram about True Positives, True Negatives, False Positives and False Negatives

$$Accuracy = \frac{(TP + TN)}{(TP + FP + FN + TN)} \tag{2.6}$$

$$\text{Prec} = \frac{TP}{(TP + FP)} \tag{2.7}$$

$$\text{Rec} = \frac{TP}{(TP + FN)} \tag{2.8}$$

$$F_1 = 2 * \frac{\text{Prec} * \text{Rec}}{\text{Prec} + \text{Rec}} \tag{2.9}$$

$$F_\beta = (1 + \beta^2) * \frac{\text{Prec} * \text{Rec}}{\beta^2 * \text{Prec} + \text{Rec}} \tag{2.10}$$

Another widely used metric is Log Loss used when the output is forecast probabilities, taking into account the uncertainty of the same based on how much it varies from the true label. Its used in machine learning models that use the LogisticRegression or Neural Networks algorithms. The following equation does the calculation:

$$LogLoss = \frac{-1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} y_{ij} * log(p_{ij}) \tag{2.11}$$

where

$$\text{N} = \text{Number of observations;}$$

$$\text{M} = \text{Number of classes;}$$

$$y_{ij} = \begin{cases} 1 & \text{if the observation i is in class j} \\ 0 & \text{otherwise} \end{cases}$$

$$p_{ij} = \text{Probability of classifier predicting class j for an observation i.}$$

LogLoss is not well advised for when datasets are unbalanced.

The last metric to be referenced is AUC ROC which computes how well the classes present in the target variable are divided. As its name mentions, it is equal to the area under a ROC (Receiver Operating Characteristic) curve that shows the trade-off between sensitivity, called True Positive Rate (TPR), also called Recall, Equation 2.8 and specificity which is equal to 1 - False Positive Rate (FRP), calculated by $\frac{TP}{(TP+FN)}$, as shown in Figure . This curve shows the performance of our model for all classification thresholds.

The main regression metrics [25] are R Square (R^2), Mean Square Error (MSE)/Root Mean Square Error (RMSE) and Mean Absolute Error (MAE). The R^2 measures how much the model

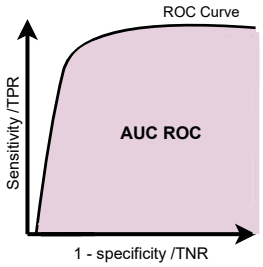


Figure 2.6: Illustration with AUC ROC

can explain the variability of a dependent variable. This measure is calculated by the sum of the squared prediction error divided by the total sum of the square of mean, shown in Equation 2.12. The MSE computes the prediction error sum, which is the expected value minus the output value given by the classifier, Equation . The RMSE is the square root of MSE, allowing for ease of interpretability of the model, giving numbers at the same level of accuracy error. The MAE works in the same way as the MSE, but the sum is of the absolute error value, calculated as in Equation . MAE, unlike MSE, treats errors all the same without giving large penalties to large errors.

$$\mathrm{R}^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2} \tag{2.12}$$

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2.13}$$

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{2.14}$$

N = Number of observations;

y_i = Expected value for an observation i;

$\hat{y}_i$ = Predicted value for an observation i;

$\overline{y}$ = Mean of all observations.

2.1.8.2 Statistical Tests

When we want to compare different machine learning model approaches and observe if there is a statistically significant difference, i.e., the difference between the models is not just due to the presence of noise in the data. There are several types of statistical tests, and depending on the model type and the data distribution, the test that should be chosen varies.

The tests can be divided into two major groups: parametric and non-parametric. The parametric ones include Paired T-Test and Correlation Coefficient, among others, and the non-parametric ones are Wilcoxon Signed Rank Test and Spearman Rank Correlation, among others. This broad division is because parametric is based on statistical data distributions, assuming that the population data is typically distributed.

Paired T-Test is the most common form, and this checks whether the mean difference of the models' performance from the data sets is significantly different from 0. d being the difference by actual values and n being the number of examples in the sample:

$$t = \frac{\sum d}{\sqrt{\frac{n(\sum d^2) - (\sum d)^2}{n-1}}} \tag{2.15}$$

The Correlation Coefficient or Pearson's correlation coefficient checks the relationship between two continuous variables. It is the most used and suitable method for this association because it is based on their covariance. The number lies between -1 and 1; the closer the absolute value 1 is found, the more significant the correlation between the two variables. When it is close to 1, it means that when one changes, the other also changes in the same direction of the sign. If it is close to -1, it indicates that if one changes, the other changes in the opposite direction.

$$r = \frac{\sum_{0 \leq i \leq n-1}(x_i - \overline{x})(y_i - \overline{y}))}{\sqrt{\sum_{0 \leq i \leq n-1}(x_i - \overline{x})^2 \sum_{0 \leq i \leq n-1}(y_i - \overline{y})^2}} \tag{2.16}$$

where,

x = set 1 of observations;

y = set 2 of observations;

n = number of observations.

Wilcoxon Signed Rank compares the ranks for the positive and negative differences of the metrics of the two models for each sample set, ignoring the signs. These differences are ranked according to their absolute values. Their averages are used in case of a tie in rank. Spearman Rank Correlation is the same as Person Correlation, but with the particularity that it evaluates monotonic relationships. That is, how well can a monotonic function represent the relationship between the chosen variables.

2.2 Data Leakage

Data leakage [3] occurs when information about the target variable exists in a predictive variable. That is knowledge in a variable that would only be discovered after a target is discovered.

An example given by Kaufman, S. et al. [3] is simple to understand: "A leaky attribute is "session length", which is the total number of pages viewed by the user during this visit to the website. This attribute is added to each page view record at the end of the session."

Leakage is a significant problem because it causes overfitting to occur in the model; therefore, when faced with a confirmed case, it fails.

2.3 Causality

Many machine learning models focus on creating correlations between variables to make predictions. However, according to Reichenbach [26] such correlations always result from causal relationships and are thus a statistical dependency.

The correlation coefficient between variables tells us they are linearly related and change together. However, it does not tell us why or how, unlike causality. Causality is the relationship between two events/variables, where each variable causes the effect on another variable. For example, if a survey says that there is a positive correlation between the price of tobacco and coffee consumption, thus meaning that as the price of tobacco increases, so does the consumption of coffee. Nevertheless, this does not mean that an increase in the price of tobacco causes an increase in coffee consumption. Causality adds real-world context and meaning to correlation.

Causality refers to the relationship between two variables with a valid explanation, which turns possibility into actuality. For example, to say that something causes an effect on another variable means that the outcome of one variable is directly influenced by the other: either the cause precedes the effect, or the effect changes when the cause changes.

To reason about causality in SSL, the causal structural model is used. A Structural Causal Model (SCM) is based on a causal mechanism that generates X_i as all its parents and its edges in a structural equation, [27]:

$$X_i := f(A, B, ...) \tag{2.17}$$

where f is the function that computes X_i of the causal variables, their parents $A, B, ...$

In an SCM (Fig. 2.7) we have several structural equations represented by Fig. 2.7(a), for each of them, we take the right-hand side variables and make them parents of the left-hand side variable, Fig. 2.7(b).

In semi-supervised learning, it is defined by the following structural equation:

$$X_i := f_i(PA_i, N_i) \text{for} i = 1, ..., n \tag{2.18}$$

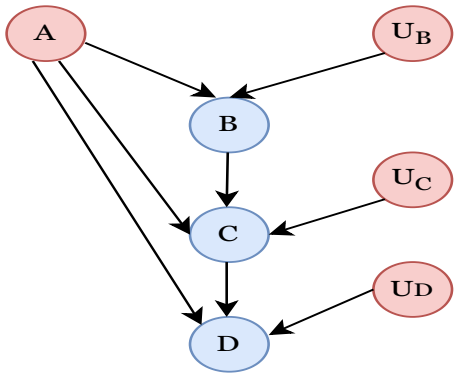
where f_i is the function that computes X_i from its parent $PA_i \subseteq X_1, ..., X_n$ and its exogeneous

$$B := f_B(A, U_B)$$

$$C := f_C(A, B, U_C)$$

$$D := f_D(A, C, U_D)$$

(a) Structural Equations



(b) SCM graph - Endogeneous variables, represented by the blue colour, influence modulating the causal mechanisms. The exogenous variables, defined by the red colour, are those with no progenitor and are not modulating how they are caused.

Figure 2.7: Representation of SCM: M

noise variable N_i. The mutually independent noise assumption implies that all common causes for any pair of observed variables are included in the model (i.e. there is no hidden information), and is referred to as causal sufficiency.

2.4 Causal and Anti-causal Learning

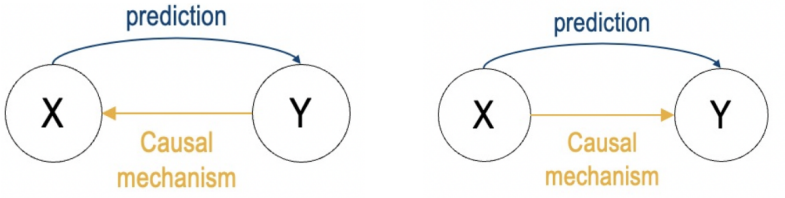
Two types of tasks can distinguish learning because of the direction of prediction, i.e. the direction of the cause: causal learning and anti-causal learning.

Causal learning says that the feature is a cause of the target variable. $P(X)$ and $P(Y|X)$ are algorithmically independent, i.e., they do not share information.

Anticausal learning is when the target variable is a cause of the feature under study. Formally, the independence relation is between $P(Y)$ and $P(X|Y)$, whereby $P(X)$ may contain information from $P(Y|X)$.

2.5 Study of the relationship between Causality and Semi-supervised Learning

Semi-supervised learning is considered where it is given a small labelled sample of X and Y from some joint distribution and a typically sizeable unlabelled sample from $P(X)$. The goal of semi-supervised learning is generally to improve the estimate of the conditional $P(X|Y)$ from additional information about the $P(X)$. Therefore, linking these two distributions with some



(a) Causal mechanism: $P(X|Y)$ (b) Causal mechanism: $P(Y|X)$

Figure 2.8: Causal mechanisms

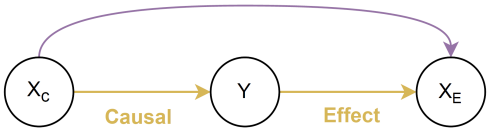


Figure 2.9: Features are divided into two sets: potential causes X_C and potential effects X_E. For example, breast cancer data where Y is diagnosis with causal features, X_C: Sex, Diet and Genetics and with effect features, X_E: Clinical symptoms and Test Results.

additional assumptions is necessary. The two common ones are the cluster assumption which posits that points in the same cluster of $P(X)$ have the same label Y, and the low-density separation assumption, which states that class boundaries of $P(Y|X)$ given line in an area where $P(X)$ is small.

Schölkopf B. et al. [1] not only formulated but also validated a hypothesis that semi-supervised learning does not work when there is a causal relation between the target and the attributes, specifically when the target variable is an effect of its predictors.

They built this hypothesis using the principle of Independent Causal Mechanism (ICM) [28] that assumes that $P(X)$ and mechanism $P(X|Y)$ are "independent" and the goal is to learn $X \to Y$, i.e., estimate $P(Y|X)$:

- If $P(X)$ changes, the mechanism $P(Y|X)$ is unaffected by the assumption described above. So semi-supervised learning doesn't work since $P(X)$ contains no information about $P(Y|X)$, 2.8(a).
- If $P(Y)$ or mechanism $P(X|Y)$ changes, $P(X)$ is influenced. So, semi-supervised learning is possible because $P(X)$ contains information about $P(Y|X)$,2.8(b).

This research is relevant to the approach because the relationship between variables' causality and SSL is the basis for solving the problem presented.

Kügelgen, J. et al. [29] asked what happens if we have both cause and effect features of a target Y, Fig. 2.9.

Analogous to the causal learning setting, they find that the distribution of causal features contains no useful additional information about $P(Y|X)$. On the other hand, the distribution of

effect features given causal features contains all the relevant information provided by additional unlabelled data about the target conditional of the interest. Therefore, SSL in this set of causal and effect features should link these two conditional distributions with some additional assumptions.

Kügelgen, J. et al. [29] proposed the conditional cluster assumption where points in the same cluster of effect characteristics given causal characteristics share the same Y label.

In addition to this, they presented two algorithms. The first algorithm is a semi-generative model. The idea is to fit a generative model only to the informative part of the distribution, meaning conditioning on the causal features and only modelling the distribution of Y and X_e. It requires assuming a particular parametric form, and the approach then proceeds by maximizing the conditional likelihood of the labelled and unlabelled data - using an expectation-maximization (EM) procedure. The second one, termed conditional self-learning, relies on the idea of using regression errors for a self-learning approach. It assumes a zero additive noise model, and the approach proceeds by initializing these two regressors from labelled data. Then getting predictions for all the unlabelled data points labelling that point with the slightest prediction error, retraining the two regressors and repeating this procedure until labelling all points.

Knowing that the target Y is an effect of the predictive variables X and given that semi-supervised learning should not work in this case; then, if it does, one could conclude that **there is leakage**, i.e. **X contains information of Y**. Formally, with cause X and effect Y, $P(X)$ and $P(Y|X)$ should be independent, but if semi-supervised learning works, it means that it is not.

Chapter 3

Detecting data leakage with semi-supervised learning

This chapter provides the methodology implemented to detect data leakage in twenty datasets: ten for regression and ten for classification tasks. These datasets were both used as static datasets and streaming datasets. The methodology is based on machine learning models, using the learning technique: semi-supervised. To employ semi-supervised learning three models were built: Model 1, Model 2, and Model 3. Model 1 is the model used as the upper bound, which helps us control the quality of the model; Model 2 is the down bound and Model 3 is where semi-supervised learning is applied. To verify that semi-supervised learning really works, Model 3 has to be in between the other 2 models.

The methodology can be divided into two parts, depending on whether the dataset will be treated as batch data or streaming data. For batch data, the library used was scikit-learn [30], an essential library for implementing machine learning models and the semi-supervised learning approach used was self-training.. For streaming data, the framework chosen was scikit-multiflow [31], a python package for online/streaming machine learning using co-training instead of self-training..

Since it was very difficult to find publicly available datasets with well-documented leakage, our strategy was to artificially introduce leakage in datasets without known leakage issues. Several different types of leakage were artificially introduced in the datasets. Then, we applied a semi-supervised learning approach to detect a target information input to each predictive variable.

3.1 Datasets

For this work, the only necessary concern for the choice of datasets was that they should be, in their entirety, causal. That is that the target variable is a effect of all the predictive variables. That is that the target variable must be an effect of all the predictive variables.

Classification datasets (Table [3.1]) can be binary or multi-class. Regression datasets (Table [3.2]) are of different scales.

Dataset	No. of Features	No. of Examples	No. of Classes	Class Distribution
bank-full	17	45211	2	39922/5289
churn_modelling	12	10000	2	7963/2037
cryotherapy	7	90	2	48/42
drug200	6	200	5	91/54/23/16/16
employee	9	4653	2	3053/1600
hr_data	17	8995	2	7313/1682
mobile_range_price	21	2000	4	500/500/500/500
predictive_maintenance	9	10000	6	9652/112/95/78/45/18
to_in_college	11	1000	2	500/500
water-quality	10	2011	2	1200/811
weatherAUS	23	56420	2	43993/12427

Table 3.1: Datasets for classification task

Dataset	No. of Features	No. of Examples	Mean of Target
auto-mpg	10	398	23.515
bias_correction_ucl	25	7588	22.911
bike_details	7	626	87958.714
car_purchasing_data	8	400	44032.713
concrete_data_yeh	9	1030	35.818
expenses	7	1337	13258.552
housing	13	545	4766729.248
laptop_price	12	1302	1124.359
sample-superstore	18	8295	30.702
second-hand-cars-sales	12	1000	308520.2425

Table 3.2: Datasets for regression task

3.2 Data Pre-processing

Some procedures should be followed to better deal with prediction problems. The pattern used for this implementation is shown in Fig.3.1

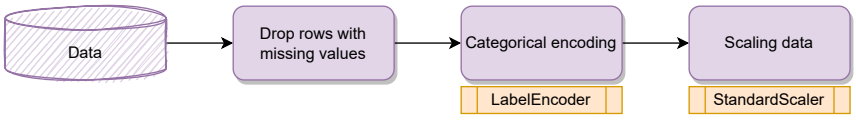


Figure 3.1: Diagram of the data pre-processing steps

The first step is to deal with missing values, which in this case is just deleting them. Next, we encode the categorical variables, using `LabelEncoder` from the library `sklearn.preprocessing`. Although this is not the best encoding method, depending on the number of different categories,

it can introduce certain importance to some categories and some machine learning models. For example, if we transform {water, wine, water, tea, wine}, using `LabelEncoder`, into {1, 2, 1, 3, 2}, we thus impose an ordinality where the average of wine and water is tea. However, since, in our case, the goal is to input information from the target into the predictive variable, it is preferable that the encoding done is then `LabelEncoder`, because if the target variable is numeric and the predictive variable is categorical, or vice versa, the input form has no non-ordinarily requirement, as will be shown in the 3.3.

The final preprocessing step involves placing the numerical values on the same scale and using the `StandardScaler` of `sklearn.preprocessing` removes the mean and scales the data to a variance equal to 1 ($\mu = 0$; $\sigma = 1$), thus removing a more significant influence of values with huge scales on the prediction of the target variable.

3.3 Leakage Introduction

As concluded in the section 2.5 from the research of Schölkopf, B. et al. [1] knowing that the target variable is an effect of the predictive variables, semi-supervised learning does not work; otherwise, then there are predictive variables that are causes of the target variable, i.e. contain information from it.

The chosen datasets are causal, that is, those in which the target is a causal effect of the predictive variables. To this end, we imputed information from the target to each predictive variable to see how the different models (Model 1, Model 2 and Model 3) would behave. This information differs according to the type of each variable (predictive and target) and add this in different percentages: 50%, 60%, 70%, 80%, 90% e 100%.

In this division, we will talk about further leakages introduced, which mathematical equations will represent. In these equations, the variables mean:

- *data*: all examples presents in dataset;
- m: Number of different values of target;
- n: Number of features;
- $V_i, ..., V_n$: predictive variables, where $0 \leq i \leq n-1$;
- Y: target variable and y_j, ... y_m: different values that the target can assume, where $0 \leq j \leq m-1$;
- p: percentage leakage introduced;
- $X \sim U(0,1)$: random number following a uniform distribution;
- μ_{y_j/V_i}: mean number of occurrences of a target value, given a specific category.

When we have the predictive variable and the target variable of the numerical type, the method used was shown in Equation 3.1. It transforms each value in the chosen column with complementary percentages of that cell's value and the target's corresponding value.

$$data[V_i] := (1 - p) * data[V_i] + p * data[Y] \tag{3.1}$$

Suppose we have the categorical predictive variable and the numerical target variable. In that case, n intervals are initially created, where n is the number of different values present in the predictive variable column. Each interval is classified with a category present in the predictive column. After this, depending on a random number with uniform distribution (0 - 1), the value of the cell in the column of the predictive variable is changed to the category corresponding to the interval that the numerical value of the target is, as shown in Equation 3.2.

$$data[V_i] := \begin{cases} interval\ category & \text{if } X \sim (0,1) >= (1-p) \\ data[V_i] & \text{otherwise} \end{cases} \tag{3.2}$$

In case both variables are categorical, we used the `TargetEncoder` function of the `sklearn.preprocessing` library, which inputs the leakage we are trying to decipher in the dataset. Using this method that uses the probability of the target variable to encode the different features, we feed them with information from the variable we are encoding. As shown in Equation 3.3, depending on a random number with uniform distribution (0 - 1), the cell in the predictive variable column is changed by the mean number of occurrences of a target value, given a specific category.

$$data[V_i] := \begin{cases} \mu_{y_j/V_i} & if\ \ X \sim (0,1) >= (1-p) \\ data[V_i] & \text{otherwise} \end{cases} \tag{3.3}$$

In the case where the target variable is categorical, and the predictor variable is numerical, Equation 3.2 is used. However, before the leakage is entered, the function `LabelEncoder` is applied to the categorical variable so that the target is an absolute numeric value, according to the different target values.

Before running the models, two significant steps are required: pre-processing and leakage input. Both are automatic, being changed according to the type of task desired. The main goal of this automatic configuration is no user intervention. In the end, the models are not only evaluated on their performance in predicting the target but also statistically among themselves.

3.4 Model Evaluation

Once the model is built, it is necessary to evaluate its performance. In this study, we used two different measures for each prediction task. For regression, Mean Absolute Error (MAE) was used and for classification, F1.

The MAE, given by the Equation 2.14, allows having a perception of the average magnitude of the errors in a set of predictions without considering their direction and is considered a negatively-oriented score, which means that the lower the values, the better the MAE.

The F1 evaluates the errors caused by false positives and false negatives, which are in general undesirable, thus maximizing precision and recall, represented by Equation 2.9. This is used in binary classification problems. Our datasets are multi-class problems, and we will have to choose the averaging we use: micro, macro or weighted. The micro-average aggregates all classes to compute the average metric, and the macro-average computes the metric's value for each class independently and then averages it. Weighted-average computes the average metric of all classes, considering the number of examples each class has. Typically, if in the dataset there is no significant disparity between the number of examples in each class, the average should be used as micro.

Moreover, the datasets used in the study are imbalanced classes, so we have to choose either macro-average or weighted-average. Macro-average is used when we want all classes to have the same importance, regardless of the number of examples each class contains. However, according to the number of examples, we should use weighted-average if we want classes to have different weights. The chosen one for evaluation in our model was weighted because the leakage introduced is information of the target value, and depending on the number of examples each target had, it influences. Both metrics range between 0 and 1, and the closer to 1, the better the model performance.

3.5 Batch

3.5.1 Methodology

In this section, we discuss the architecture implemented in our methodology. The methodology used in batch data is based on the self-training approach. We implemented three machine learning models with the Random Forest algorithm. The test data was the same in the three models corresponding to 20% of the entire dataset. The change in their implementation was only in the data used for training, Figure 3.2:

- **Model 1**: 80% of the examples of the complete dataset;
- **Model 2**: 5% of the examples of the complete dataset;

- **Model 3** `(Semi-supervised Learning)`: same data as Model 1, but with 5% of the actual data plus 75% of the data with label predicted by Model 2.

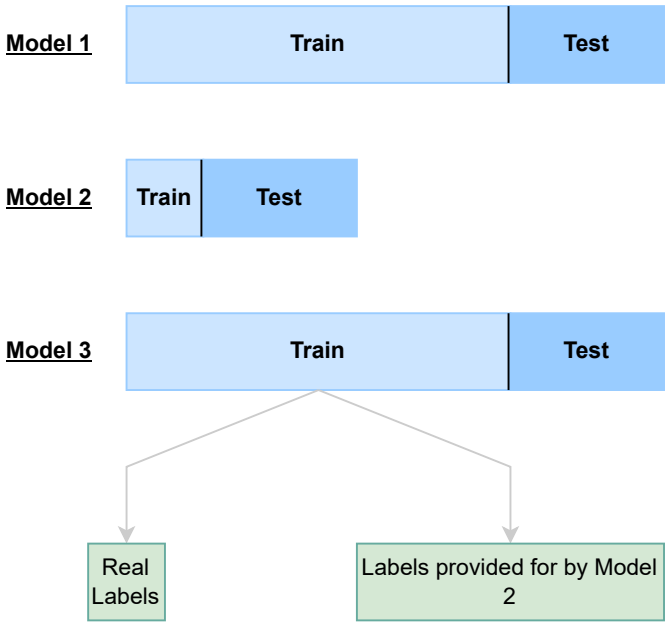


Figure 3.2: Batch: Representation of the division of the data for the execution of the different models

3.6 Streaming

3.6.1 Methodology

In this procedure, the approach chosen was co-training since some authors say it is the technique that works best on data streams. We also implemented three models, similar to the batch data methodology but with the Adaptive Random Forest algorithm. The test data is the same in all three models, about 20% of the total data. The data used in training are, therefore, shown in Figure 3.3:

- **Model 1:** 80% of the examples of the entire dataset;
- **Model 2 and Model 3 `(Semi-supervised Learning)`:** 20% of the examples of the entire dataset.

The pre-processing of data and the imputation of leakage is done offline, something that should be changed to online in future work.

To implement co-training, the algorithm developed by Sousa and Gama [32] was used as the basis for our implementation. However, this algorithm uses overlapping attributes in the views, and we chose to operate without this overlapping but with each view having different attributes.

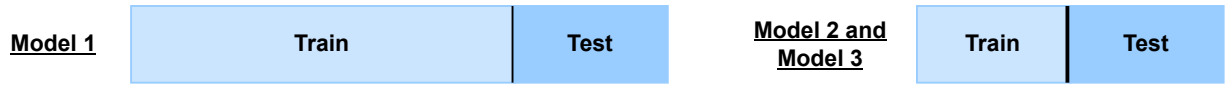


Figure 3.3: Streaming: Representation of the division of the data for the execution of the different models

When we implement co-training, we came across into a problem, which is still under study, which is the optimal division of the feature set into two subsets that follow the assumptions defined by the creators of this co-training technique, Blum and Michell [2]. So, we developed an algorithm, with the help of Professor Ana Paula Tomás, intending to create a balanced feature division minimizing the dependency between them but maximizing each one individually.

3.6.2 Method for splitting features for the co-training technique

In our research, we found that the techniques present in Table 2.1 to check whether the splitting of the two views made them sufficient to predict the target variable ran the model several times. However, this approach implies a long run and does not guarantee that it is the optimal solution to the problem.

One way to guarantee that these two assumptions were present, the programming model we chose because it was adequate was Mixed-Integer Programming. This type of programming allows us to guarantee the two necessary conditions: the Inter-Conditional Mutual Information (InterCMI) be minimum and Intra-Conditional Mutual Information (IntraCMI) be maximum, taking into account that these views must be sufficient to predict the class, that is, the class-conditional mutual information must be balanced.

This algorithm uses the minimum cut algorithm of the graph as a basis, where the nodes are the attributes of the dataset and the edges are the conditional mutual information between the attributes that are in the nodes. The cut will be done in order to divide a graph into two, one being View 1 and the other View 2.

3.6.2.1 SplitFeat-MIP algorithm

The developed algorithm starts with the pre-processed dataset, which calculates the Class-conditional Mutual Information (CondMI) between all pairs of features, thus building a matrix with the values corresponding to each feature. In this matrix the lines and the columns are the dataset features, being represented by `matrixCDMI`. Our algorithm is based on the already existing [12] technique, taking advantage of a graph and then cutting it.

The essence of this problem is to choose each feature in which views, View1 and View2, will be. Moreover, since the question is whether it is in the view or not, it can be considered as an integer problem, where the decision variables are:

$$x_i := \begin{cases} 1 & \text{if feature i is in View1,} \\ 0 & \text{if feature is not in View1;} \end{cases}$$

$$e_{ij} := \begin{cases} 1 & \text{if feature i and feature j is in same view,} \\ 0 & \text{if feature i and feature j is not in same view;} \end{cases}$$

$$y_{ij} := \begin{cases} 1 & \text{if feature i and feature j is in View1,} \\ 0 & \text{if feature i and feature j is not in View1 (may one belong and another not).} \end{cases}$$

Other relevant variables for defining this problem are:

$P_c :=$ Weight of the graph cut;

$P_1 :=$ Weight of partition 1;

$W :=$ Value to balance the weight of each partition.

There are eleven types of constraints for the model developed:

(i) being n the number of features cut the split equal;

(ii) to ensure that if either feature i or feature j does not belong in View1 ($x_i - x_j = 1$ *or* $x_j - x_i = 1$), we must also ensure that both are not in the same view ($e_{ij} = 1$). It should also be implied that if e_{ij} is equal to 1, the features are in the same set.

(iii) to ensure that if we have that both features i and j belong to View1 ($y_{ij} = 1$) then $x_i + x_j$ is necessarily equal to 2. But we must also fix that if $x_i + x_j = 2$, then y_{ij} is equal to 1.

(iv) P_c is the sum of all the CondMI where the features are in different views;

(v) P_1 is the sum of all the CondMI of the features that are present in View1;

(vi) The value of W will try to be maximum, but must be limited by the weights of View1 and View2, i.e., Partitions 1 and 2.

The model is:

$$\text{Maximize } W \tag{3.4}$$

subject to:

$$\sum_{i=1}^{n} x_i \quad = n/2 \tag{3.5}$$

$$e_{ij} >= x_i - xj \qquad (i = 0, ..., n-1; \;\; j = 0, ..., i-1) \tag{3.6}$$

$$e_{ij} >= x_j - xi \qquad (i = 0, ..., n-1; \;\; j = 0, ..., i-1) \tag{3.7}$$

$$x_i + x_j + e_{ij} <= 2 \qquad (i = 0, ..., n-1; \;\; j = 0, ..., i-1) \tag{3.8}$$

$$x_i + x_j >= e_{ij} \qquad (i = 0, ..., n-1; \;\; j = 0, ..., i-1) \tag{3.9}$$

$$x_i + x_j >= 2 * y_{ij} \qquad (i = 0, ..., n-1; \;\; j = 0, ..., i-1) \tag{3.10}$$

$$x_i + x_j - 1 <= y_{ij} \qquad (i = 0, ..., n-1; \;\; j = 0, ..., i-1) \tag{3.11}$$

$$\sum_{i=0}^{n} e_{ij} * \texttt{matrixCDMI}_{ij} \quad = P_c \qquad (j = 0, ..., i-1) \tag{3.12}$$

$$\sum_{i=0}^{n} y_{ij} * \texttt{matrixCDMI}_{ij} \quad = P_1 \qquad (j = 0, ..., i-1) \tag{3.13}$$

$$P_1 >= W \tag{3.14}$$

$$\left(\sum_{i}^{n} \texttt{matrixCDMI}_{ij} \right) /2 - P_c - P_1 <= W \tag{3.15}$$

The objective function is only to maximize the value of W, that is, the value to balance the two views. We chose this maximization because we intend that there is a balance between the two views when using the two classifiers. The goal is that there is not one of the models with greater significant sufficiency than the other and thus may influence the algorithm's performance in the co-training environment in real data. However, this value will never be greater than the different weights since it is limited by the weight of view1 and the weight of view2.

The solver chosen was Gurobi, which solves mathematical programming problems, as is the case of our Mixed-Integer Programming problem. Besides Mixed-Integer Programming also allows for solving problems like Linear Programming and Quadratic Programming, among others.

The solver can explore modern architectures and be used in different programming languages such as Python, C, and Java, among others.

3.6.2.2 Results

To check whether our approach would be better than some already developed approaches demonstrated earlier. The approaches used for this comparison were:

- Equal Split: splitting the feature set into two subsets where the first n/2 will be in the first subset and the remainder in the second subset, where n is the feature set size;
- Random Split: splitting the set of two subsets, of equal size, in a random way [14];
- Entropy Split: division of the set of features into two subsets using a heuristic - ordering in a descending way of entropy calculation - the features in the odd positions go to the first subset, and the ones in the even positions go to the other subset, [14];
- MaxInd Split: splitting the set of features using the minimum graph cut, minimizing the independence between the features present in the two different subsets [12].

The data used in this experiment has its description in Appendix A.1.

EqualSplit and Random Split are unreliable methods for the reason that: (1) `Equal Split`, the attributes can be in a favourable order for its division, but in some datasets can bring a bad performance; (2) `Random Split`, the division made randomly, and as it is known, can manage to return an optimal solution with as much probability as return a completely useless solution for the problem.

As shown in Table 3.3 we can observe that the algorithm we developed is the one that maintains a balanced split at the intra-view dependency level, thus maintaining the assumption that both should be sufficient for the models to be able to predict the label. However, the view dependency cannot achieve as good values as `MaxInd Split`. As both use the minimum graph cut heuristic, we can conclude that this heuristic is an approach with immense potential for feature splitting, to do the splitting according that the edges that connect the two subgraphs will have a smaller weight, i.e., that the conditional mutual information between the features of the two subgraphs, be as small as possible.

The `Entropy Split` that despite achieving lower dependency values than SplitFeat-MIP presents an imbalance in intra-views dependencies, which provides that one of the views can predict better than the other.

To prove that this influences the Co-Training behaviour, we created an experiment using different splitting methods with the same datasets. The algorithm applied was the Random Forest. The Co-Training approach used was the same that was done for the streams methodology,

Datasets	Splitting Methods	IntraView 1	IntraView 2	InterViews
agaricus-lepiota	Equal Split	12.887	12.905	28.043
	Random Split	14.0	11.561	12.423
	Entropy Split	12.867	11.434	29.533
	MaxInd Split	12.887	12.905	28.043
	SplitFeat-MIP	11.494	12.425	29.916
breast-cancer	Equal Split	1.241	0.528	1.823
	Random Split	0.815	0.819	1.793
	Entropy Split	1.097	0.858	1.637
	MaxInd Split	1.241	0.529	1.823
	SplitFeat-MIP	0.536	0.541	2.515
breast-cancer-wiscosin	Equal Split	8.983	1.631	11.438
	Random Split	2.154	7.474	12.423
	Entropy Split	8.209	1.557	12.286
	MaxInd Split	2.193	6.605	13.254
	SplitFeat-MIP	1.862	6.488	13.703
flare1	Equal Split	1.266	0.258	1.159
	Random Split	0.498	0.671	1.514
	Entropy Split	1.066	0.342	1.274
	MaxInd Split	1.266	0.258	1.159
	SplitFeat-MIP	0.547	0.426	1.711
flare2	Equal Split	0.500	0.083	0.447
	Random Split	0.220	0.201	0.610
	Entropy Split	0.299	0.196	0.537
	MaxInd Split	0.500	0.083	0.447
	SplitFeat-MIP	0.174	0.151	0.706

Datasets	Splitting Method	IntraView 1	IntraView 2	InterViews
glass	Equal Split	46.616	20.810	80.581
	Random Split	32.919	31.513	83.576
	Entropy Split	35.193	29.626	83.189
	MaxInd Split	43.322	23.3	81.386
	SplitFeat-MIP	31.513	32.919	83.576
hepatitis	Equal Split	5.392	40.893	36.374
	Random Split	19.583	17.177	45.900
	Entropy Split	21.656	15.252	45.752
	MaxInd Split	2.167	56.723	23.771
	SplitFeat-MIP	18.698	16.412	47.55
house-votes-84	Equal Split	2.828	1.688	4.970
	Random Split	1.967	2.295	5.224
	Entropy Split	1.641	2.665	5.180
	MaxInd Split	2.828	1.688	4.970
	SplitFeat-MIP	2.064	1.952	5.471
import-85	Equal Split	46.139	160.671	192.460
	Random Split	138.187	56.412	200.081
	Entropy Split	102.517	88.172	213.231
	MaxInd Split	20.498	228.833	150.562
	SplitFeat-MIP	101.945	89.471	221.087
primary-tutor	Equal Split	1.656	1.012	2.562
	Random Split	1.296	1.006	2.928
	Entropy Split	1.33	1.088	2.812
	MaxInd Split	1.656	1.012	2.562
	SplitFeat-MIP	1.058	1.078	3.094

Table 3.3: Results of the dependency of the views using the different methods. IntraView 1 is the dependency between features in view 1, IntraView 2 is the dependency between features in view 2 and InterViews is the dependency between features in the two views.

but in static form, that is, knowing the entire dataset. This influence is demonstrated in Figure 3.4.

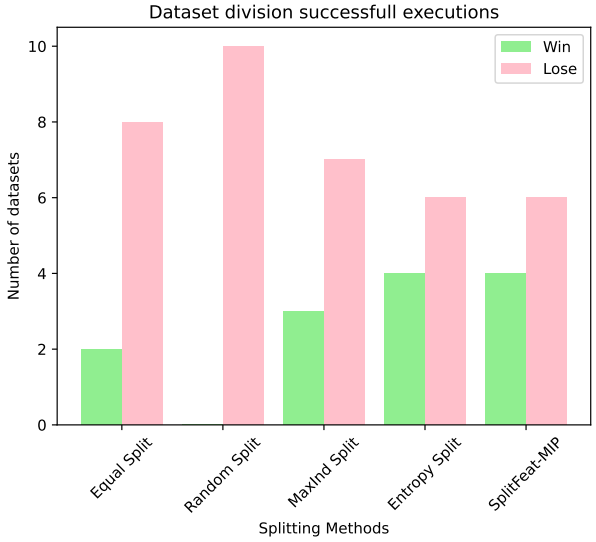


Figure 3.4: Number of datasets where co-training worked for each different method. When co-training works it is set to `Win`, otherwise it is set to `Lose`

Looking at Figure 3.4, as expected, none of the methods already developed is sufficiently capable of doing an optimal splitting of the feature set. However, two stand out positively: Entropy Split and SplitFeat-MIP. An essential aspect concerning SplitFeat-MIP is that it works well with datasets where the number of attributes is even, that is, when it is possible to do a perfect split for the two sets, as seen in Figure A.1. Another thing to retain from this experiment is the confirmation of what we said before regarding the Equal Split and Random Split methods.

Once, both did not have outstanding performances, including Random Split being unable to get Co-Training working in any of the chosen datasets.

About MaxInd Split it has very similar behaviour to SplitFeat-MIP, failing in one, which leads us to believe that the fact of not taking care to check the intra-views dependency influences negatively.

In conclusion, the two algorithms that have the same performance in the way co-training works are Entropy Split and SplitFeat-MIP. However, Entropy Split takes a small advantage in the difference between Model 3, the one where Co-Training is applied and Model 2, when it does not work. That is, Model 3 manages to achieve values very close to Model 2 when it fails ((Figure A.1(a), Figure A.1(d), Figure A.1(h)).

The reasons why our algorithm still shows some deficit in the positive influence on Co-Training are:

- The number of attributes in the dataset being odd;
- The sum of the conditional mutual information is negative, that is, less than 0.

Although we believe that these are the main reasons for the most visible flaws, we still do not know how to mitigate them, so the further study would still be needed.

Chapter 4

Results

In this chapter, we discuss our experiments and the results obtained. First, we performed a small preliminary experiment to swap the target variable for the other variables in each dataset. Next, we explore the leakage behaviour of each variable in the dataset. In the end, we present a brief discussion on the results. We will also make a case study, present in A, of one of the datasets for a more detailed analysis.

4.1 Preliminary Experiments

As mentioned before, the chosen datasets have the main requirement to be causal; the target is the effect of the other variables. Figure 4.1 demonstrates how the different models predict the target variable. Therefore, we can conclude that semi-supervised learning does not improve (Model 3) over the other two models. So this confirms what Schölkopf B et al. [1] over argue, i.e., semi-supervised learning does not work in the causal direction.

In a first step, we swap the target variable for the other variables in the dataset. It should be noted that in some cases, semi-supervised learning works [Figure 4.2], thus reinforcing the statement [1]. In the cases in which semi-supervised learning does not work, this could be attributable to the fact that the variable is neither cause or effect of the target, or simply that it as noisy and/or uninformative variable.

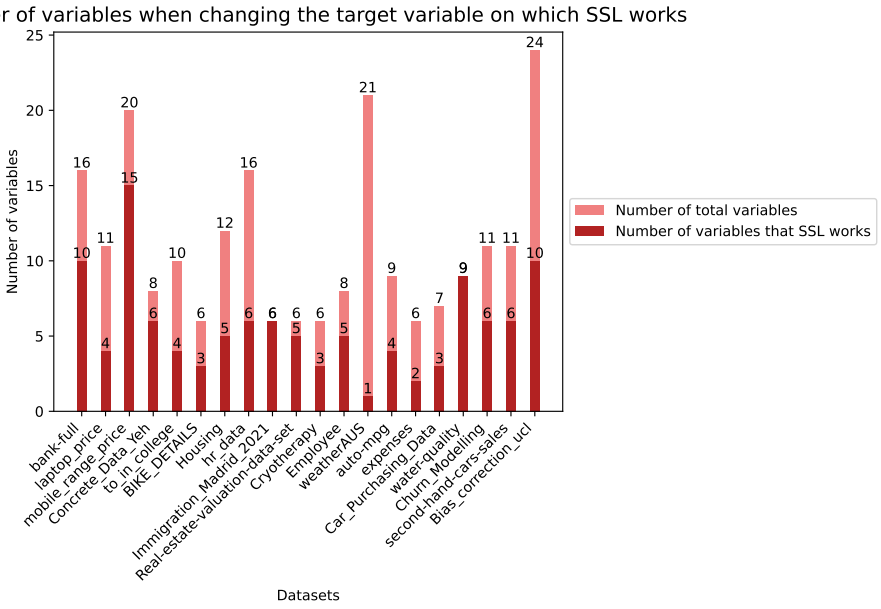


Figure 4.2: For each dataset under study, the number of variables when swapping the target variable for each predictive variable and SSL works

4.2 Experiments

To validate our method, we measure Mean Absolute Error (MAE) and F1 Score (F1) in the regression and classification tasks, respectively, in each model. To test this, as previously mentioned we built three models for each configuration. In batch, Model 1 is trained with 80% of the data, Model 2 with only 5% of the data, and Model 3 has the same data as Model 1, but only 5% of the data contains the true label, and the others are predicted by Model 2. In stream, Model 1 then has 80% of the data and Model 2 and Model 3 have about 20% of the data. In all models, the dataset used for testing is the same, ie 20% of the initial data.

Model 1 is the base model. In principle Models 2 and 3 should not outperform it, since they are trained with less true labels. If Model 3 is superior to Model 2, this means that SSL works, which means we have detected leakage; if Model 2 is at least as good as Model 3, then no leakage is detected.

In Regression task, the target variable is numerical, either integers or continuous. However, the target variables can be numeric or categorical; therefore, we have to use different methods of introducing leakage, shown in Equation 3.1 and 3.2.

The ten datasets were then used, totalling 121 predictive variables where the leakage was introduced in different percentages.

In Figure 4.3, we observe the number of variables with the introduction of the leakages in different percentages in the regression task. This graphic allows us to analyze two behaviours

with the same data processing characteristics but in different settings: batch and streaming. In a first analysis, we can conclude that, in general, the leakage types developed for regression work significantly better in batch environments (Figure 4.3(a)) than in streaming environments (4.3(b)).

When considering Figure 4.3(b), we see that it peaks when it is at the 80% introduction percentage and then drops significantly, when it is the opposite of what we expected.

In the classification case, the target variable is categorical. However, as in the other type of task, the predictive variables can be categorical and numerical, thus using the equations 3.2 and 3.3 , respectively.

To study this, we collected ten datasets with the categorical target variable. thus introducing the different types of leakage, having in total 142 attributes.

In the classification task, in all cases of the amount of leakage, we observe that the numbers are not disparate, as shown in Figure 4.4, unlike what happened in the regression case (Figure 4.3. Furthermore, and beyond this big difference, the behaviour of these leakage types in these two environments is very identical, which leads us to conclude that the way our method looks at these imputations is the same, regardless of the approach we are in.

In this type of task, essentially in batch, as should be expected, when we introduce leakage in all the data of that attribute (Percentage of leakage introduced = 1. 0), we should see a significant increase, as we see in Figure 4.3(a); however, this does not happen, and the reason is straightforward: when we introduce leakage, in general, the three models improve, since the model has learned these changes and with this, many times Model 2 reaches a F1 equal to 1.0, which in turn will also be the value of Model 1. That is, Model 3, the model that uses the Semi-supervised Learning (SSL) approach, cannot improve on Model 2, so we cannot retain anything from there.

In all the leakage percentage differences introduced, we observe that it fails to detect leakage in more than half of the variables. The reason is that when we introduce leakage in a specific variable, its importance increases. However, in some cases, it fails to have enough potential to allow SSL to work as it is supposed to, as we can observe in the case study presented below.

4.2.0.1 Case Study: Mobile Range Price

This dataset consists of 2000 data examples with 20 attributes: [$'battery\_power'$,$'blue'$,$'clock\_speed'$,$'dual\_sim$ $'int\_memory'$,$'m\_dep'$,$'mobile\_wt'$,$'n\_cores'$,$'pc'$,$'px\_height'$,$'px\_width'$,$'ram'$,$'sc\_h'$,$'sc\_w'$,$'talk\_time$ $'touch\_screen'$,$'wifi'$,$'price\_range'$] The target variable is price range.

Analyzing the importance of each feature, in Figure 4.5 in the objective of finding out what the price range of that product is, we conclude that the ram variable has a great influence (0.4735) on it, inversely, the blue (0.0064) and three_g (0.0053) variables are completely irrelevant.

When we introduce 50% leakage into a variable like three_g, its importance increases, as shown in Figure 4.8 , thus proving that the leakage was introduced. However, looking at Figure 4.9, we can see that this leakage is not easily detected, and semi-supervised learning does not work, because the ram variable still has great importance.

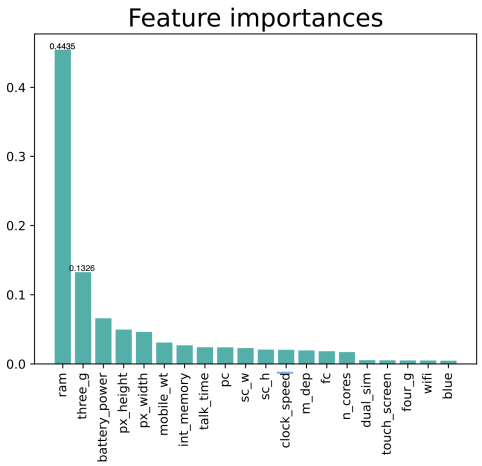


Figure 4.6: Feature importances after imputing 50% of leakage in predictive variable: three_g

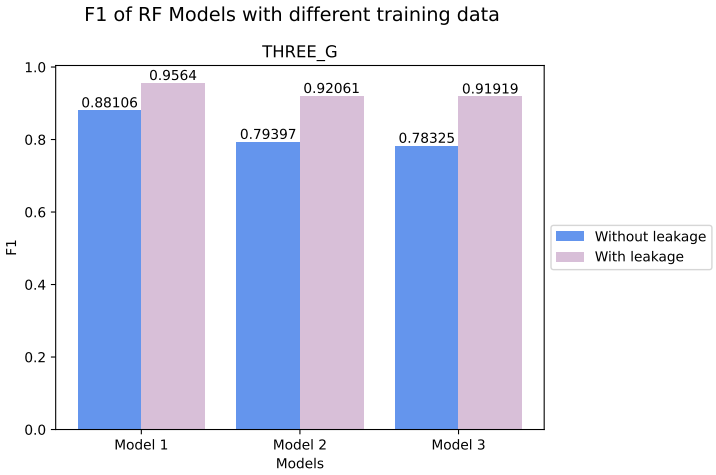


Figure 4.7: F1 Score of the three models after imputing 50% leakage in the predictive variable: three_g

However, when we introduce leakage in all examples, the importance of the three_g variable becomes higher than the ram variable, which leads the semi-supervised learning to conclude that there is leakage in this variable.

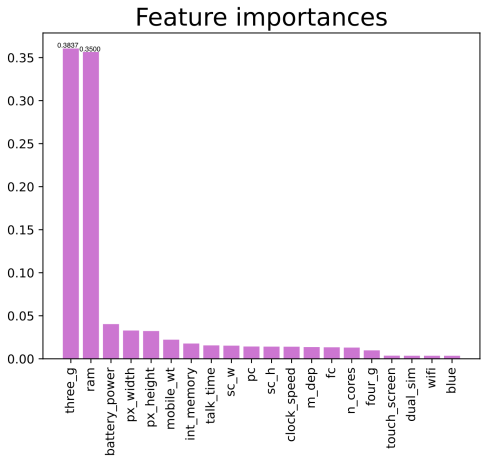


Figure 4.8: Feature importances after imputing 100% of leakage in predictive variable: three_g

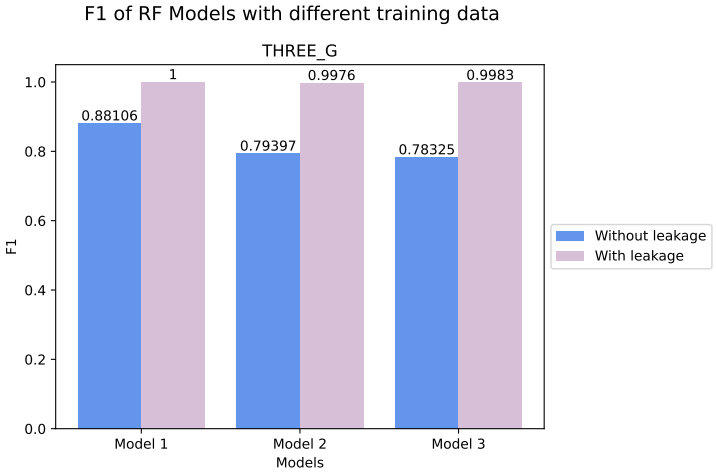


Figure 4.9: F1 Score of the three models after imputing 100% leakage in the predictive variable: three_g

In short, depending on the characteristics of each attribute and each dataset, the method

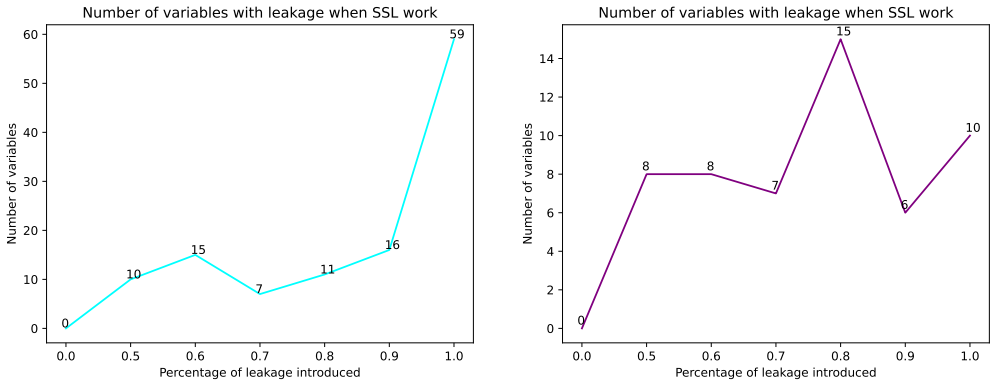
developed has some limitations, but it is an important step for this area of interest.

In the next step, we would apply our method to datasets in which we know there is leakage to be able to detect which of the predictive variables contain target information.

(a) Regression Task: MAE for Causal datasets

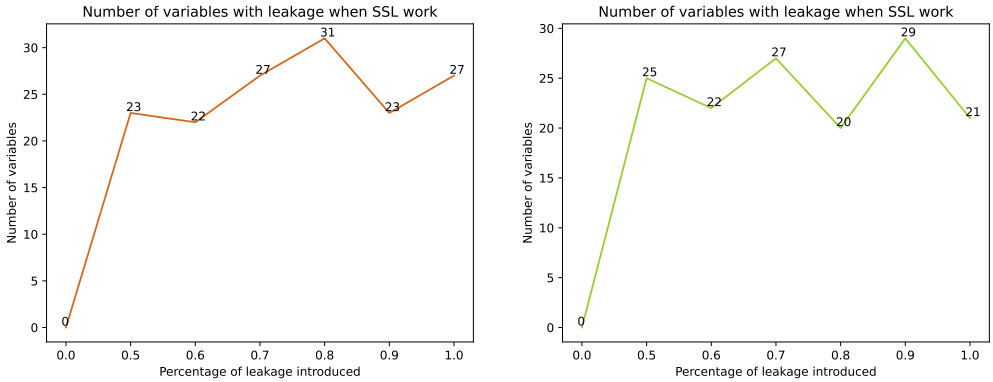
(b) Classification Task: F1 for Causal datasets

Figure 4.1: The behaviour of the different models on causal datasets



(a) Batch Setting (b) Streaming Setting

Figure 4.3: Regression Task: Number of variables, in each dataset, where the leakage (Equation 3.1 and Equation 3.2 was detected after imputation



(a) Batch Setting (b) Streaming Setting

Figure 4.4: Classification Task: Number of variables, in each dataset, where the leakage (Equation 3.3 and Equation 3.2 was detected after imputation

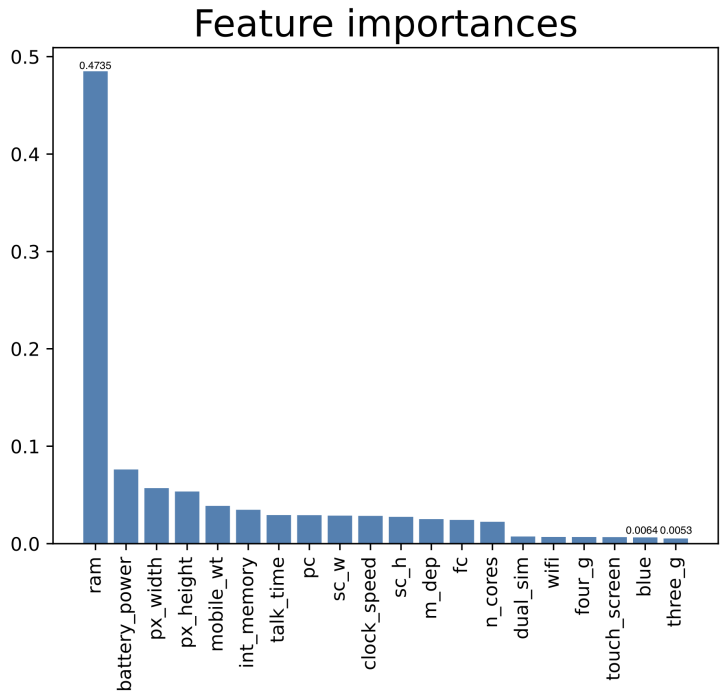


Figure 4.5: Importance of Mobile Range Price dataset features

Chapter 5

Conclusion

In this work, we introduce mechanisms for leakage detection in a dataset in order to extract the most reliable information from the data in it. This research takes a key concept in the literature: the hypothesis validated by Schölkopf et al. [1], in which semi-supervised learning doesn't works in the causal sense.

Due to the use of semi-supervised learning, one of its techniques was explored, Co-training. Therefore raised the interest to study more about this technique, which is still very much under research. Therefore, we created an algorithm in Mixed-Integer Programming (MIP) to comply with the assumptions made by the creators of Co-training, Blum and Michell [2] about the dependency between the two subsets of features.

5.1 Key findings

Our main focus was to be able, through the causal framework, to distinguish features with leakage and features without leakage in two environments: batch and streaming. For this, we evaluated two perspectives:

1. Batch: Semi-supervised Learning (SSL) using the self-training approach;
2. Streaming: SSL using the co-training approach.

From the results obtained from the batch environment, we can state that using the Schölkopf, B. search, we could detect the presence of features with leakage because when target information is added to a specific feature, it significantly increases its importance. However, despite its importance increasing significantly, it does not indicate that it is powerful enough to detect leakage. On the other hand, the streaming results also show the ease semi-supervised learning has in detecting target information in predictive variables. Nevertheless, in this case, we did not analyse the growth/decrease of the importance of the features, so we are not able to draw such a conclusion.

With the implementation of co-training in streams, there was interest in creating an algorithm; as already mentioned, despite obtaining good results compared to some presented in the state-of-art, due to achieving a balance between the two assumptions of Blum and Michell should be explored in detail. Nevertheless, we believe it will be an essential step in this field of research.

5.2 Future work

We believe that this developed work is a significant step to be applied in data pre-processing in the future. However, it has the potential to be improved. As such, the improvements we outlined, being that they were not carried out due to the time constraint. One of the improvements is the imputation of leakage on streams, in an online way, instead of offline, thus making the whole process incremental and not only the execution of the model.

Finally, we consider that we should work on constructing a method to detect which predictive variable(s) contains information about the target, thus significantly improving the reliability and accuracy of the models in a real case.

Appendix A

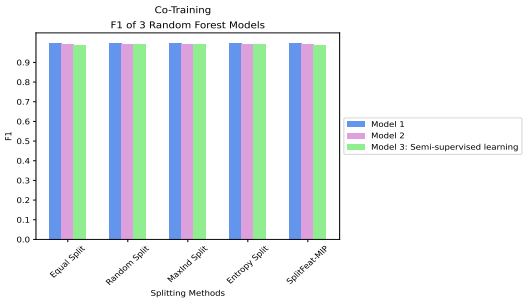
SplitFeat-MIP

A.1 Description of the datasets used for comparison of the different view splitting approaches

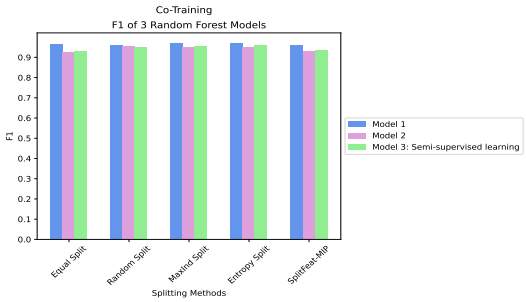
Datasets	No of Features	No of Examples
agaricus-lepiota	23	8123
breast-cancer	10	285
breast-cancer-wiscosin	11	698
flare1	13	322
flare2	13	1065
glass	11	243
hepatitis	20	154
house-votes-84	17	434
import-85	26	204
primary-tumor	18	338

Table A.1: Datasets used for splitting methods

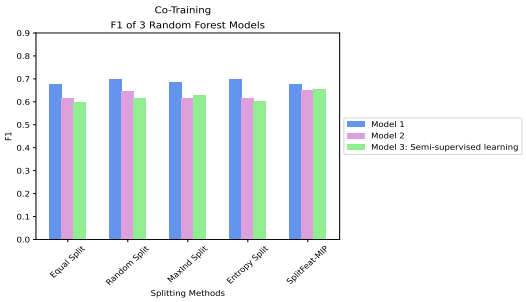
A.2 Results of co-training applied with splitting methods



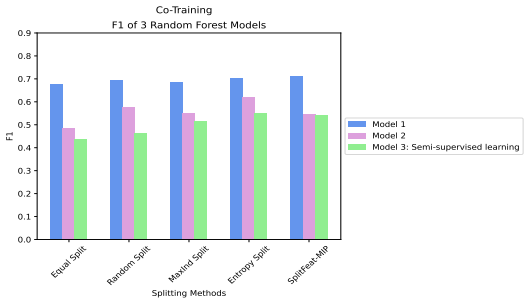
(a) Agaricus-lepiota



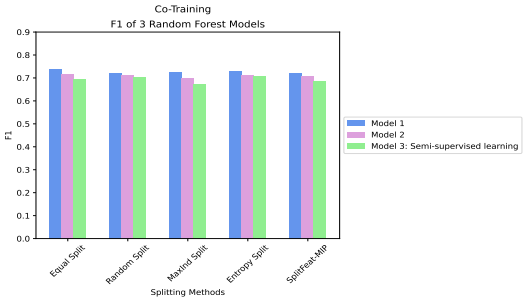
(b) Breast-cancer-wisconsin



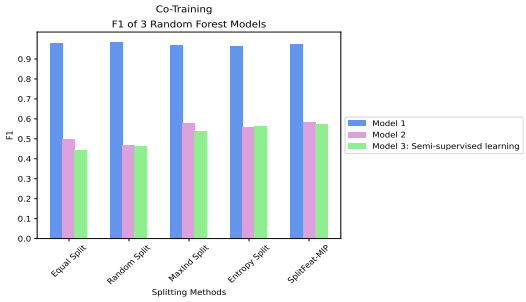
(c) Breast-cancer



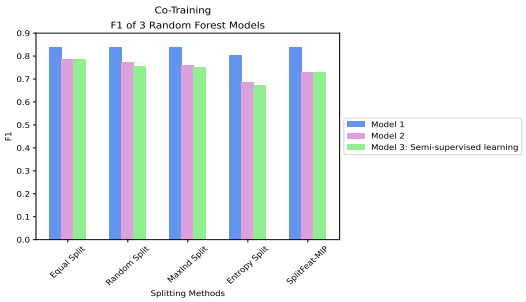
(d) Flare1



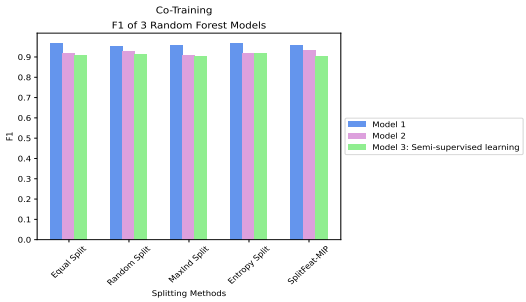
(e) Flare2



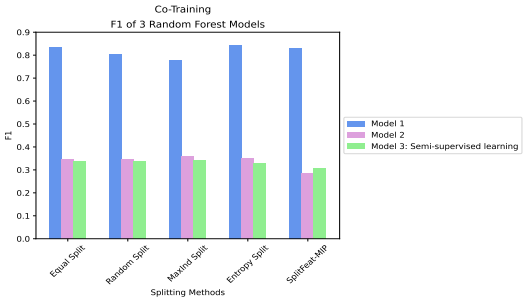
(f) Glass



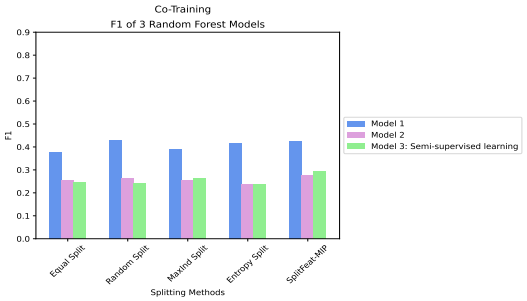
(g) Hepatitis



(h) House-votes-84



(i) Import-85



(j) Primary-tumor

Figure A.1: F1 Score of three Random Forests models, with the Co-Training technique, in different types of feature splitting

Bibliography

[1] Bernhard Schölkopf, Dominik Janzing, Jonas Peters, Eleni Sgouritsa, Kun Zhang, and Joris Mooij. On causal and anticausal learning. *arXiv preprint arXiv:1206.6471*, 2012. https://arxiv.org/pdf/1206.6471.pdf.

[2] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100, 1998. https://dl.acm.org/doi/pdf/10.1145/279943.279962.

[3] Shachar Kaufman, Saharon Rosset, Claudia Perlich, and Ori Stitelman. Leakage in data mining: Formulation, detection, and avoidance. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(4):1–21, 2012. https://dl.acm.org/doi/pdf/10.1145/2382577.2382579.

[4] Andriy Burkov. *The hundred-page machine learning book*, volume 1. Andriy Burkov Quebec City, QC, Canada, 2019.

[5] Pádraig Cunningham, Matthieu Cord, and Sarah Jane Delany. Supervised learning. In *Machine learning techniques for multimedia*, pages 21–49. Springer, 2008. https://link.springer.com/chapter/10.1007/978-3-540-75171-7_2.

[6] Mary K. Pratt. What is unsupervised learning, 2020. Available at https://www.techtarget.com/searchenterpriseai/definition/unsupervised-learning. Last Access: 2022-07-09.

[7] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009. http://www.acad.bg/ebook/ml/MITPress-%20SemiSupervised%20Learning.pdf.

[8] Matthias Seeger. Learning with labeled and unlabeled data. Technical report, 2000. https://infoscience.epfl.ch/record/161327.

[9] Xiaojin Jerry Zhu. Semi-supervised learning literature survey. 2005. https://minds.wisconsin.edu/bitstream/handle/1793/60444/TR1530.pdf?sequence=1&isAllowed=y.

[10] Alex Gammerman, Volodya Vovk, and Vladimir Vapnik. Learning by transduction. *arXiv preprint arXiv:1301.7375*, 2013. https://arxiv.org/pdf/1301.7375.pdf.

[11] Kamal Nigam and Rayid Ghani. Understanding the behavior of co-training. In *Proceedings of KDD-2000 workshop on text mining*, pages 15–17, 2000. http://www.cs.columbia.edu/~dplewis/candidacy/understanding-the-behavior-of.pdf.

[12] Felix Feger and Irena Koprinska. Co-training using rbf nets and different feature splits. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 1878–1885. IEEE, 2006. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1716339.

[13] Masahiro Terabe and Kazuo Hashimoto. Evaluation criteria of feature splits for co-training. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 2008, 2008. https://www.researchgate.net/profile/Masahiro-Terabe-2/publication/44261621_Evaluation_Criteria_of_Feature_Splits_for_Co-Training/links/0912f50ecd5041c68a000000/Evaluation-Criteria-of-Feature-Splits-for-Co-Training.pdf.

[14] Jun Du, Charles X Ling, and Zhi-Hua Zhou. When does cotraining work in real data? *IEEE Transactions on Knowledge and Data Engineering*, 23(5):788–799, 2010. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5560662.

[15] Brian W Kernighan and Shen Lin. An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, 49(2):291–307, 1970.

[16] Zhu Xiaojin and Ghahramani Zoubin. Learning from labeled and unlabeled data with label propagation. *Tech. Rep., Technical Report CMU-CALD-02–107, Carnegie Mellon University*, 2002. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.14.3864&rep=rep1&type=pdf.

[17] Heitor M Gomes, Albert Bifet, Jesse Read, Jean Paul Barddal, Fabrício Enembreck, Bernhard Pfharinger, Geoff Holmes, and Talel Abdessalem. Adaptive random forests for evolving data stream classification. *Machine Learning*, 106(9):1469–1495, 2017. https://link.springer.com/content/pdf/10.1007/s10994-017-5642-8.pdf.

[18] Nikunj Oza and Stuart Russell. Online bagging and boosting. *Proceedings of Artificial Intelligence and Statistics*, 2001.

[19] Dinesh Yadav. Categorical encoding using label-encoding and one-hot-encoder, 2019. Available at https://towardsdatascience.com/categorical-encoding-using-label-encoding-and-one-hot-encoder-911ef77fb5bd. Last Accessed: 2022-07-12.

[20] Vinicius Trevisan. Target-encoding categorical variables, 2022. Available at https://towardsdatascience.com/dealing-with-categorical-variables-by-using-target-encoder-a0f1733a4c69. Last Accessed: 2022-07-12.

[21] Rukshan Pramoditha. Encoding categorical variables: One-hot vs dummy encoding, 2021. Available at https://towardsdatascience.com/encoding-categorical-variables-one-hot-vs-dummy-encoding-6d5b9c46e2db. Last Accessed: 2022-07-12.

[22] Kumar Vishwesh. Dealing with categorical features with high cardinality: Feature hashing, 2020. Available at https://medium.com/flutter-community/dealing-with-categorical-features-with-high-cardinality-feature-hashing-7c406ff867c. Last Accessed: 2022-07-12.

[23] Baijayanta Roy. All about feature scaling, 2020. Available at https://towardsdatascience.com/all-about-feature-scaling-bcc0ad75cb35. Last Accessed: 2022-07-12.

[24] Rahul Agarwal. The 5 classification evaluation metrics every data scientist must know, 2019. Available at https://towardsdatascience.com/the-5-classification-evaluation-metrics-you-must-know-aa97784ff226. Last Accessed: 2022-07-29.

[25] Songhao Wu. 3 best metrics to evaluate regression model?, 2020. Available at https://towardsdatascience.com/what-are-the-best-metrics-to-evaluate-your-regression-model-418ca481755. Last Accessed: 2022-07-29.

[26] Ernest H. Hutten. The direction of time. by h. reichenbach. (the university of california press 1956. pp. xi + 280. price 41s. 6d. net.). *Philosophy*, 34(128):65–, 1959. https://infoscience.epfl.ch/record/161327. doi:10.1017/S0031819100029806.

[27] Leland Gerson Neuberg. Causality: models, reasoning, and inference. *Econometric Theory*, 19(4):675–685, 2003.

[28] Povilas Daniusis, Dominik Janzing, Joris Mooij, Jakob Zscheischler, Bastian Steudel, Kun Zhang, and Bernhard Schölkopf. Inferring deterministic causal relations. *arXiv preprint arXiv:1203.3475*, 2012. https://arxiv.org/pdf/1203.3475.pdf.

[29] Julius Kügelgen, Alexander Mey, Marco Loog, and Bernhard Schölkopf. Semi-supervised learning, causality, and the conditional cluster assumption. In *Conference on Uncertainty in Artificial Intelligence*, pages 1–10. PMLR, 2020. http://proceedings.mlr.press/v124/kugelgen20a/kugelgen20a.pdf.

[30] Sklearn developers. scikit-learn: Machine learning in python, 2007. Available at https://scikit-learn.org/stable/index.html. Last Access: 2022-08-05.

[31] Scikit multiflow developers. scikit multiflow - a machine learning package for streaming data in python, 2019. Available at https://scikit-multiflow.github.io. Last Access: 2022-08-05.

[32] Ricardo Sousa and João Gama. Co-training study for online regression. pages 529–531, 04 2018.