UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

AUGUSTO EXENBERGER BECKER

Assessing Data Leakage Effects on the Performance Estimates of Machine Learning Classifiers

Work presented in partial fulfillment
of the requirements for the degree of
Bachelor in Computer Engineering

Advisor: Prof[a]. Dr[a]. Mariana Recamonde Mendoza

Porto Alegre
September 2023

“Perfecting oneself is as much unlearning as it is learning.”

— EDSGER WYBE DIJKSTRA

AGRADECIMENTOS

Agradeço aos meus pais, Denis Becker e Raquel Exenberger Becker, por todo o apoio, carinho e incentivo durante esses anos. Aos meus irmão Thales Exenberger Becker e Pedro Henrique Exenberger Becker, que trilharam esse caminho antes de mim, por compartilharem suas dificuldades e aprendizados. À minha namorada Letícia Cattai Contino, pelo apoio e incentivo durante essa trajetória. Agradeço também à minha orientadora, Profª. Mariana Recamonde Mendoza, por aceitar essa posição e pelo conhecimento, indicações e ensinamentos que foram fundamentais para este trabalho.

ABSTRACT

Data Leakage (DL) in the context of Machine Learning (ML) refers to the inadvertent introduction of information from test data into the training process. This contamination can occur in various forms, including subtle ones such as during data pre-processing stages (*e.g.*, normalization, handling missing values, and feature selection) and hyperparameter tuning. When DL takes place, it can result in an inflated estimation of model performance due to overfitting, which fails to translate into real-world predictions. In this study, we investigate the impact of DL on the performance assessment of classification models using a set of 30 carefully chosen data sets. These data sets were sourced from the Penn Machine Learning Benchmark repository, with an emphasis on varying their metadata (*e.g.*, number of instances and number of features) but keeping a relative balance between classes. The assessment encompasses six distinct ML algorithms: K-Nearest Neighbors, Support Vector Machine, Decision Trees, Naïve Bayes, Random Forest, and Logistic Regression. Performance evaluation is carried out using variations of the k-fold cross-validation, and the balanced accuracy and F1 score metrics. The findings of this research reveal a consistent pattern of performance overestimation when DL is present. Notably, the effect of DL is particularly pronounced in the context of hyperparameter tuning and feature selection. Moreover, our analysis indicates a relatively higher susceptibility of the Support Vector Machine algorithm to DL, whereas the impact on Logistic Regression is comparatively less significant. Categorizing the data sets based on the impact of DL on their performance, we identified three groups: one group of 10 data sets experiences a substantial increase in performance estimates, another group of 9 shows a marginal increase, and the remaining 11 either maintain or exhibit decreased performance estimates in the presence of DL. In specific cases, we observed notable improvement in performance estimates, with average scores increasing up to 8 percentile points upon DL insertion, particularly for certain preparation tasks and data sets. Overall, our results suggest that the effects of DL should not be neglected since it tends to positively affect model performance, generating results that may be hard to replicate with real-world data.

Keywords: Data Leakage. Data Contamination. Overfitting. Performance Estimate. Machine Learning.

Avaliação dos Efeitos do *Data Leakage* na Estimativa de Desempenho de Classificadores baseados em Aprendizado de Máquina

RESUMO

Data Leakage (DL) refere-se à introdução inadvertida de informações de dados de teste no processo de treinamento de modelos com aprendizado de máquina (AM) . Esse vazamento de dados pode ocorrer de formas sutis, como durante os estágios de pré-processamento de dados (*e.g.,*, normalização, tratamento de valores ausentes e seleção de atributos) e ajuste de hiperparâmetros, e pode resultar em uma estimativa inflacionada do desempenho do modelo devido ao *overfitting*, que não se traduz em previsões do mundo real. Neste estudo, investigamos o impacto do DL na avaliação de desempenho de classificadores usando 30 conjuntos de dados do repositório Penn Machine Learning Benchmark, escolhidos com ênfase na variação de seus metadados (*e.g.*, número de instâncias e número de atributos) e no equilíbrio relativo entre as classes. A avaliação abrange seis algoritmos de AM: K-Nearest Neighbours, Support Vector Machine (SVM), Árvores de Decisão, Naïve Bayes, Florestas Aleatórias e Regressão Logística. O desempenho foi avaliado com variações da validação cruzada k-fold e com as métricas de acurácia balanceada e F1-score. Os resultados obtidos revelam um padrão consistente de superestimação de desempenho quando há vazamento de dados. Notavelmente, o efeito de DL é particularmente pronunciado no contexto do ajuste de hiperparâmetros e seleção de atributos. Observou-se uma susceptibilidade relativamente maior do algoritmo SVM ao DL, enquanto o impacto na regressão logística foi menos significativo. Com base no impacto do DL na estimativa de desempenho, identificamos três grupos dentre os conjuntos de dados: (i) 10 conjuntos de dados exibiram um aumento substancial nas estimativas de desempenho, (ii) 9 conjuntos de dados tiveram um aumento marginal, (iii) 11 conjuntos de dados mantiveram os reduziram a estimativa de desempenho na presença de DL. Para determinadas combinações de tarefas e conjuntos de dados, observamos uma melhoria notável nas estimativas de desempenho, com pontuações médias aumentando em até 8 pontos percentuais após a inserção de DL. No geral, os nossos resultados sugerem que os efeitos de DL não devem ser negligenciados, uma vez que tendem a afetar positivamente o desempenho dos modelos, gerando resultados que podem ser difíceis de replicar com dados do mundo real.

Palavras-chave: Data Leakage, Contaminação de Dados, Aprendizado de Máquina, Overfitting, Estimativa de Desempenho.

LIST OF ABBREVIATIONS AND ACRONYMS

CV	Cross-Validation
DL	Data Leakage
DT	Decision Tree
KNN	K-Nearest Neighbors
LR	Logistic Regression
ML	Machine Learning
NB	Naive Bayes
PMLB	Penn Machine Learning Benchmarks
RF	Random Forest
SVM	Support Vector Machine

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

Machine Learning (ML) algorithms are being used in a wide range of research and applications, many showing promising results. However, when the process of training and validation of ML models is not done correctly, in a methodological point of view, it can lead to Data Leakage (DL). Data contamination through DL generates models that over-perform validation tests, creating false expectations and unreproducible results (KAPOOR; NARAYANAN, 2022).

Data leakage is defined as the contamination of training data with privileged information from the independent validation set (or test set) to which the algorithm or method should not have access during model development. This can originate in the mishandling of the data set during pre-processing and data preparation steps, by subtle mistakes in normalization, missing values imputation, feature selection, or hyperparameter tuning, for instance. It is essential to understand the extent of DL effects better once it has been cited as one of the "Top 10 Data Mining Mistakes" (NISBET; ELDER; MINER, 2009).

Recently, a broad spread of research has been conducted focusing on DL. Efforts were placed on understanding, detecting, and preventing the issue, providing formulations (KAUFMAN et al., 2012), a taxonomy of leakage types (KAPOOR; NARAYANAN, 2022), and static code analysis approaches to detect leakage signs in the code (DROBNJAKOVIĆ; SUBOTIĆ; URBAN, 2022; YANG et al., 2022). Additionally, some studies investigated the consequences of leakage in specific applications (SAMALA et al., 2020; SHIM; LEE; HWANG, 2021; YAGIS et al., 2021).

Nonetheless, there is still a lack of studies that analyze the effects of unintentionally inserting DL in different pre-processing steps. This work aims to propose a study of the effects of DL on the performance estimate of ML models developed for classification tasks to understand how different data sets and algorithms are affected by each kind of pre-processing contamination: normalization, missing values imputation, feature selection, and hyperparameter optimization.

This work is organized as follows. Chapter 2 provides a theoretical background, describing the key concepts for this work. Chapter 3 summarizes existing efforts based on related works. Next, Chapter 4 presents the methods used, including the selection of data and the experimental methodology adopted. The experimental results are unveiled in Chapter 5, alongside five research questions that our work aimed to address. Finally, conclusions and possible future works are presented in Chapter 6.

2 THEORETICAL BACKGROUND

2.1 Machine Learning

Machine Learning is a field of Artificial Intelligence focused on approximating models to represent or describe data collections. This is accomplished by using optimization or learning algorithms to approximate a target function using example data or past experience (ALPAYDIN, 2014). ML applications exist in various fields, such as bioinformatics (TAN et al., 2021), medicine (NAQA; MURPHY, 2015), smart cities (ULLAH et al., 2020), education (LUAN; TSAI, 2021), and finance (DIXON; HALPERIN; BILOKON, 2020).

ML algorithms are divided into two major categories: supervised learning and unsupervised learning. Supervised learning incorporates techniques based on the "mapping" of given inputs and outputs and uses this generated mapping to predict the output of unseen input data (CUNNINGHAM; CORD; DELANY, 2008). The output can be either categorical for classification tasks or numerical for regression tasks. In contrast, unsupervised learning techniques work with data that does not have a previously defined output, with the goal of clustering or organizing these values (GREENE; CUNNINGHAM; MAYER, 2008), for example.

In what follows, we revise selected supervised learning algorithms commonly applied for classification tasks, which is the interest of the current work.

2.2 Supervised Learning Algorithms

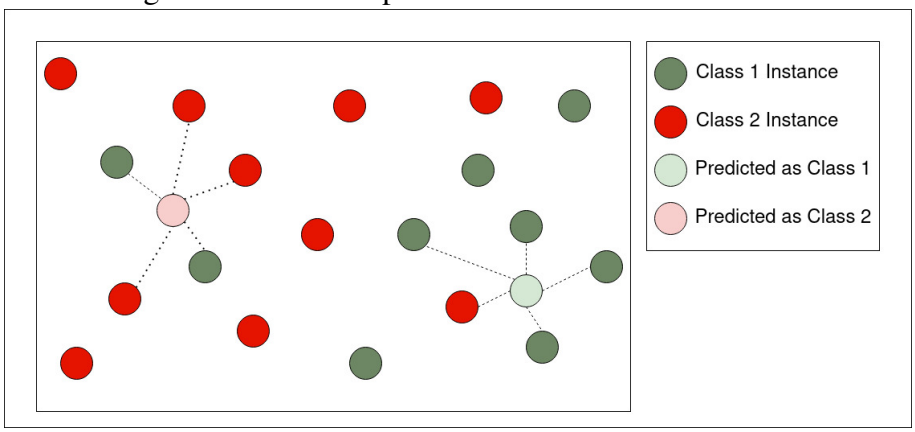
2.2.1 K-Nearest Neighbors

The K-Nearest Neighbors (KNN) algorithm relies on the premise that data instances with similar values should have the same classification. To predict the class of a new instance, the KNN algorithm uses the known classes for the K nearest data instances from the provided point. The predicted class selected for this new instance is the most frequent class among its K neighbors (KRAMER, 2013). Figure 2.1 shows an example of how two new instances would be classified according to their nearest neighbors, considering a model based on a 5-Nearest Neighbors approach.

Despite being simple, the KNN algorithm can provide a very complex decision

boundary since every data instance is classified according to the surrounding values. However, the algorithm is also highly sensitive to the setting of its three main hyperparameters: the number of nearest Neighbors (K), the distance function used to select the nearest neighbors, and the weighting function (in case of using weighted voting among nearest neighbors) (KANG, 2021).

Figure 2.1: An example of KNN-based classification.



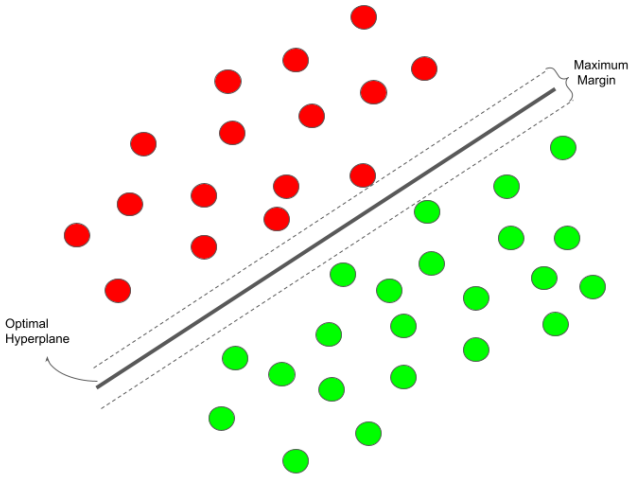
Source: The Author

2.2.2 Support Vector Machine

Support Vector Machine (SVM) is an ML algorithm based on splitting the decision space with a hyperplane that best divides the instances while maximizing the margin between the decision surface and the nearest training data instances from distinct classes (NOBLE, 2006). Figure 2.2 shows an example of a possible SVM classifier splitting instances in a two-dimensional domain (i.e., two features). The dotted lines show the margins from the splitting line and the nearest data instances, also known as the support vectors.

The traditional SVM is able to deal only with linearly separable data. However, different kernels and techniques used in SVMs allow the algorithm to conduct non-linear classification. The kernel trick, for instance, consists of transforming the data points by projecting them into higher dimensional spaces, adding extra features created as a function of the original ones to obtain linear separability. Then, the algorithm generates a splitting hyperplane in this new feature space and transforms it back to the original vector space (SUTHAHARAN; SUTHAHARAN, 2016).

Figure 2.2: An example of SVM decision boundary for binary classification.

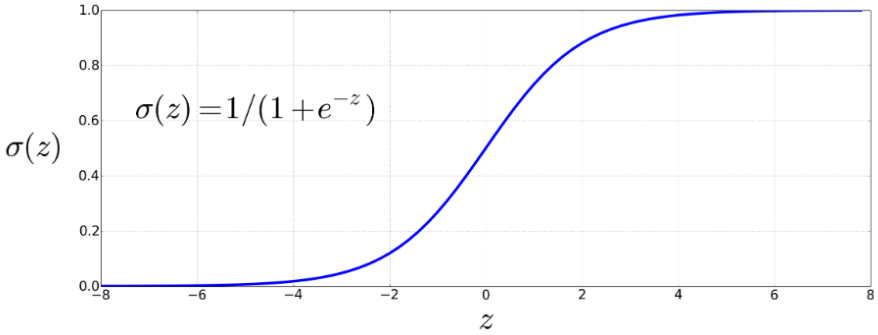


Source: The Author

2.2.3 Logistic Regression

The Logistic Regression (LR) algorithm is an alternative to model probabilities for n classes as a function of the feature vector in question (HASTIE et al., 2009). For a binary problem, the LR can use the sigmoid function (Figure 2.3) to make the binary decision about a new observation (JURAFSKY et al., 2014). For multiclass problems, different approaches can be applied. The implementation from scikit-learn (PEDREGOSA et al., 2011), adopted in our work, uses the One versus Rest approach to select the resulting class for multiclass classification. Consequentially, the sigmoid function can be used to decide for each class against the remainder. The class with the largest predicted score is then used for the prediction.

Figure 2.3: Plot of the Sigmoid function, with the characteristic "S"-shaped curve.



Source: Jurafsky et al. (2014)

2.2.4 Naive Bayes

As the name suggests, the Naive Bayes (NB) algorithm is based on the Bayes theorem. It uses the strong assumption of conditional independence between features given a class to calculate the conditional probability of each feature to each class (ZHANG; LI, 2007). The conditional probabilities along with the *a priori* probability for each class are then used to calculate *a posteriori* probability for each class of a new instance, given its features (BERRAR, 2018).

The following equation is used to calculate the likelihood of a y_i class for a given data instance x with d features. As mentioned, this value is calculated by the product of the probability of the value of each x_j feature for the respective class. It also takes into consideration the *a prior* probability of the class, $P(y_\mathrm{i})$. The predicted class is the one that maximizes the posterior probability $P(y_i|x)$.

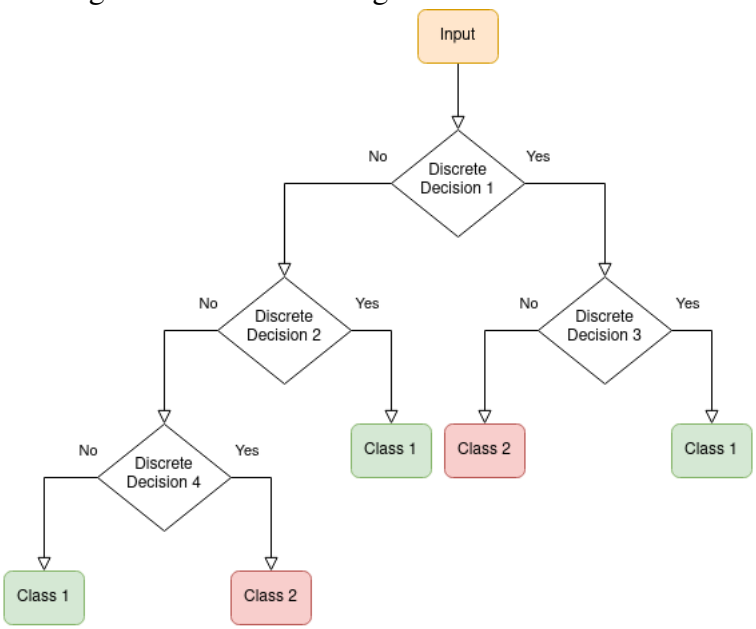
$$P(y_i|x) = P(y_i) \prod_{j=1}^{d} P(x_j|y_i) \tag{2.1}$$

2.2.5 Decision Trees

The Decision Tree (DT) is a classifier trained based on a sequence of recursive divisions of the input space. For each level, starting on the root of the tree, the DT directs the model's decision according to a discrete function defined based on a given input feature (ROKACH; MAIMON, 2005). The algorithms chooses the feature whose split provides the maximum impurity decrease in the training data set. The successive application of these discrete functions is used to guide the classification of a given instance. The leaf nodes are called terminal nodes and are related to the final classification response. Figure 2.4 shows a diagram with the structure for a DT.

In general, DTs are good at dealing with categorical values, using discrete functions that split the decision plane according to the possible categories. When dealing with continuous numeric values, the decision plane is split using a function to discretize the original feature values (KINGSFORD; SALZBERG, 2008).

Figure 2.4: Decision diagram of a Decision Tree

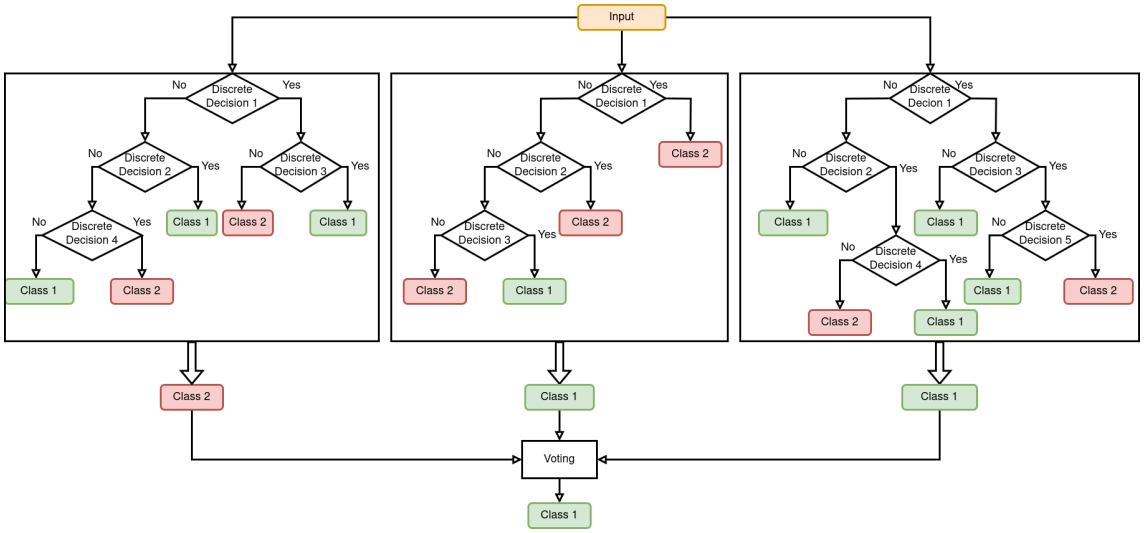


Source: The Author

2.2.6 Random Forests

Random Forest (RF) classifiers are ensembles of DTs. The plurality of the ensemble comes from training different DTs, using a random selection of instances (via bootstrap) and features (CUTLER; CUTLER; STEVENS, 2012). Figure 2.5 shows a diagram representing a simple RF with three DTs. The training process for each tree is the same as described in Section 2.2.5. In the classification step, the output class is selected based on a majority voting among all trees in the ensemble (BREIMAN, 2001).

Figure 2.5: Decision diagram of a Random Forest



Source: The Author

2.3 Model Approximation

Model approximation is considered here as the process of using the data available to train, evaluate, and select the best ML model for a given task. In this section, three concepts are presented. The data preparation steps are explained in 2.3.1. Section 2.3.2 describes the process of hyperparameter tuning. Finally, Section 2.3.3 presents two methods to evaluate model performance.

2.3.1 Pre-processing

In the context of ML model approximation, pre-processing refers to the set of techniques performed on raw data to prepare it for use in model training. This step can be composed of several tasks that change the input data somehow. Examples of pre-processing tasks are: missing value imputation, normalization, and feature selection.

Value imputation is a process used to deal with missing values. Value imputation aims to replace missing points in the data instances for reasonable values (YOZGATLIGIL et al., 2013). There are different strategies through which this goal can be achieved, from more straightforward methods that use the median or average of the existing values to more complex processes that use KNN algorithms or other ML approaches to estimate the probable values (GARCÍA; LUENGO; HERRERA, 2015).

Normalization is a form of data transformation and it aims to change the domain of raw data features to avoid those features with more considerable variations dominating over others in the learning process (GARCÍA; LUENGO; HERRERA, 2015). One of the most common methods of normalization is the Min-Max Normalization. This method uses the minimum and maximum values for the respective feature in the data set to calculate the normalized feature value.

Feature Selection is an strategy commonly applied to reduce the dimensionality of the data set. The goal of feature selection is to select the optimal subset of features according to a selection criteria (GARCÍA; LUENGO; HERRERA, 2015). Different criteria can be used to select the ideal features, including Dependence Measures, Separability Measures, Consistency Measures, Accuracy and Information Measures (GARCÍA; LUENGO; HERRERA, 2015). Therefore, this process removes irrelevant features while also reducing the complexity of the resulting model.

2.3.2 Hyperparameter tuning

Hyperparameter tuning is the process of adapting the hyperparameters of the algorithms, changing their values to achieve better performance (RASCHKA, 2018). These hyperparameters can be the number of nearest neighbors in the KNN algorithm 2.2.1 or the kernel of an SVM 2.2.2, for example. The process works by changing the hyperparameter values iteratively, usually based on search strategies such as grid search or randomized search, and selecting the configuration of values that produces the best results for a selected performance metric, such as accuracy (RASCHKA, 2018).

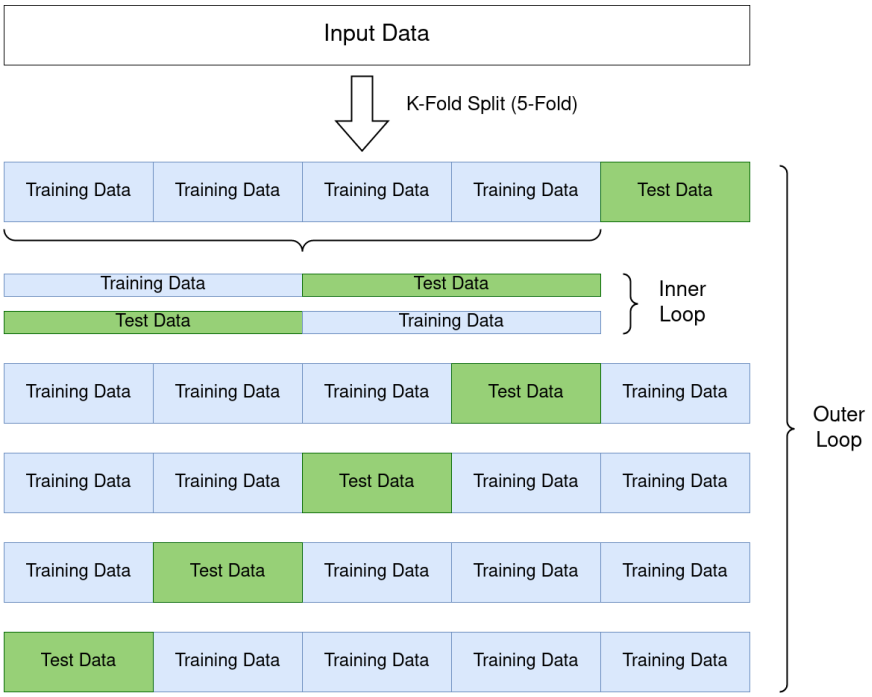
2.3.3 Cross-Validation

Cross-validation (CV) is a technique for model evaluation, also used as basis for model selection. The technique consists of sequential iterations of training and validation stages, allowing every data instance to be used for training and testing the model.

In K-fold CV the data is split into K parts of the same size, called folds. These folds are used for model training and evaluation in K different iterations. In each iteration, one of these folds is used for validation, and the other K-1 folds are used for training (RASCHKA, 2018). In Figure 2.6, the outer loop is equivalent to the process of K-Fold CV. After all iterations, the average performance across all test folds is used to determine the overall performance of the model.

Nested CV is a technique that uses multiple levels of CV. It can reduce bias compared to K-Fold CV when used for hyperparameter tuning and evaluation (VARMA; SIMON, 2006). The process consists of two loops of CV, the outer and inner loops. Figure 2.6 presents the process of Nested CV but shows the inner loop only in the first iteration of the outer loop. The outer loop is similar to a traditional K-Fold CV, where the folds are used to train and evaluate the model. The inner loop is run over the training folds in every iteration of the outer loop and executes a new round of CV each time. In the example of hyperparameter tuning, the training data is split into new folds that are used to select the best combination of hyperparameters. After the inner loop runs, a model is trained for the outer loop evaluation using the selected hyperparameters.

Figure 2.6: Structure of a nested cross-validation with inner and outer loops.



Source: The Author

2.4 Data Leakage

Data leakage refers to the phenomenon where training data becomes contaminated with information obtained from the validation or test set during the model development process. To be able to evaluate the model, we usually assume that part of the available data will be the real-world unseen data based on which we will further estimate model performance. Once we define a subset of the data as our test set, the algorithms should not have access to it during any step of model training and validation. When we accidentally use information from outside the training data to prepare this data set or guide the model development, by sharing information from the holdout test set, we have a *data leakage*.

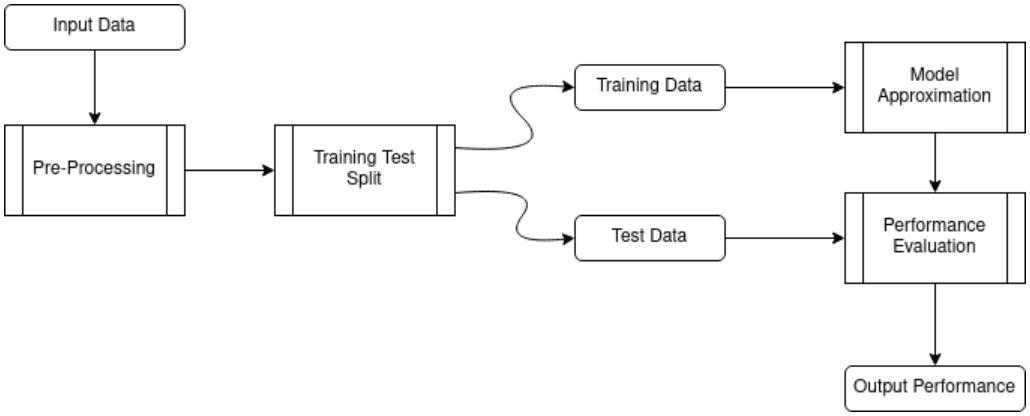
Data leakage has been classified as one of the "Top 10 Data Mining Mistakes" (NISBET; ELDER; MINER, 2009) and continues to be a critical issue[1]. At a minimum, DL leads to an inflated estimation of the performance of the trained model. However, this optimistic view of model performance often fails to translate into real-world predictions after deploying the model into production. This discrepancy arises due to overfitting and our limited capacity to estimate the generalization power of the model since it is being run on data that it had already seen in some extent during training. Consequently, mod-

[1] *Data Leakage in Machine Learning: How it can be detected and minimize the risk.* <https://tinyurl.com/2ptpub3h>, last accessed in August 26th, 2023.

els trained with DL tend to exhibit poorer generalization compared to models developed without such leakage (KAUFMAN et al., 2012).

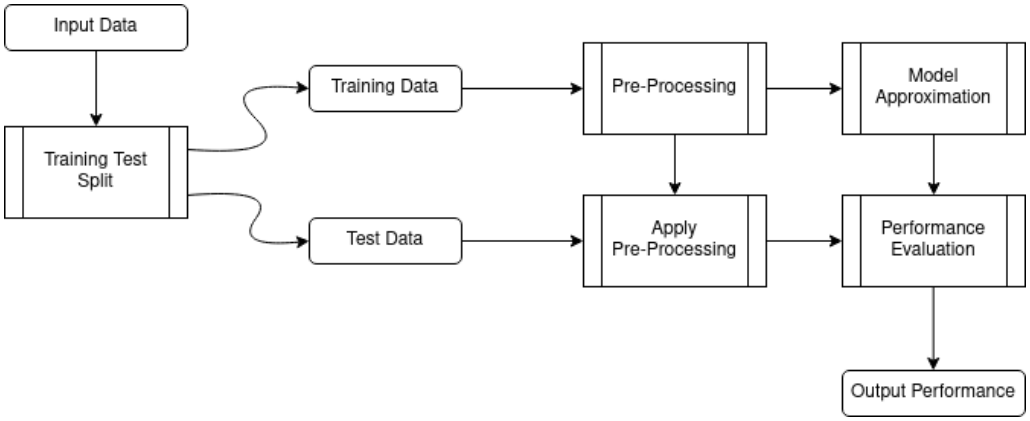
Although the most obvious cases of DL is accidentally including information about the target variable as a feature or failing to avoid intersection between training sets and test sets, this phenomenon can also happen in more subtle ways. In this work, our primary focus is on a form of DL that arises when the training and testing sets are inadequately separated throughout all pre-processing steps and model approximation processes (KAPOOR; NARAYANAN, 2022). These steps encompass normalization, value imputation, feature selection, and hyperparameter tuning. Figure 2.7 illustrates how the data can be improperly handled during pre-processing and model approximation, causing DL. Figure 2.8 exemplifies the correct way that data should be handled to prevent leaking information from the test set into training data. Both examples assume a two-way holdout division of the original data into training data and test data.

Figure 2.7: Diagram showing how data leakage can be inserted during data pre-processing.



Source: The Author

Figure 2.8: Diagram showing how to avoid data leakage during data pre-processing.



Source: The Author

In what follows, we summarize how the introduction of DL may happen in the four tasks addressed in the current work:

- **Normalization:** During normalization, DL occurs when the whole data set is normalized once before splitting into training and test sets. The information about the test set is leaked through normalization. To avoid contamination, the normalization parameters should be estimated using only the training set, and later applied separately to training and test data (DROBNJAKOVIĆ; SUBOTIĆ; URBAN, 2022).
- **Value imputation:** Similarly to normalization, DL during missing value imputation happens when this step is performed before the data set is split into training and test data. When this occurs, information from the test data set is used to define the imputed values or the imputation model that will be used for all data, including the training set. The solution would be to learn the parameters of the imputation strategy (or the imputed values themselves) using only training data and apply this knowledge, or the imputation model obtained, to impute values in the test set (KAPOOR; NARAYANAN, 2022).
- **Feature Selection** Since feature selection incites the need to train and evaluate models with different combinations of features, using the same train/test split that will be used for the final model evaluation would lead to DL since the features selected would be optimized for data they should not have access to (SAMALA et al., 2020). When using an iterative method of feature selection, an alternative to prevent DL in feature selection would be a method similar to the nested CV described in 2.3.3. In this setup, the inner loop could be used for feature selection and the outer loop to train and evaluate the performance of the models based on selected features. On the other hand, when a more straightforward method that only scores the importance of each feature is used, simply separating the test data before feature selection would prevent DL.
- **Hyperparameter Tuning** Similarly to feature selection, hyperparameter tuning also implicates the need to evaluate each hyperparameter configuration through strategies such as grid search. Therefore, using the original train/test split for tuning the hyperparameters would cause train-test contamination and overfitting (YANG et al., 2022). The solution would also be to use a nested CV approach, with the inner loop implementing the hyperparameter tuning and the outer loop evaluating the model performance.

3 RELATED WORKS

As discussed in Section 2.4 of the preceding chapter, DL is a severe problem in ML that can have significant consequences for the accuracy and reliability of predictive models. Over the past few years, substantial research efforts have emerged to comprehend, identify, preempt, and alleviate the impact of DL on ML models. In this chapter, we review some of the recent contributions within this domain.

3.1 Data Leakage

The concepts of DL and the forms with which it can affect performance evaluation have been the focus of some studies in recent years. In their study of leakage in data mining, Kaufman et al. (2012) present formulations to describe and define DL. They use these formulations as a basis for proposing a *Data Leakage Avoidance Methodology* and three techniques for leakage detection. This discussion was carried out by authors while also addressing the issue of how to proceed when leakage is, in fact, identified.

On the other hand, the research on DL and the reproducibility crisis (KAPOOR; NARAYANAN, 2022) presents an overview of how DL can lead to overoptimistic conclusions. The study presents a survey with evidence of the so-called "Reproducibility Crisis" in ML. They present 20 studies of 17 areas that identified pitfalls in adopting ML methods in scientific research. Collectively, these 20 studies identified 329 affected papers, many of which carry out pre-processing or feature selection steps on the train-test data set. The research also proposed a taxonomy of eight types of leakage and a model for detecting and preventing leakage. The authors also reproduced and corrected tests made by selected papers to show the extent of the damages caused by DL.

Regarding cognitive classification methods of forum transcripts, Farrow, Moore and Gašević (2019) demonstrate how pre-processing practices can lead to overoptimistic results. Their results indicate that the best way to ensure that the data used for validation is not contaminated is to split the data by processing session instead of only using the stratified folds.

The Static Analysis methods presented in Drobnjaković, Subotić and Urban (2022) and Yang et al. (2022) show a different approach to avoid the DL problem. The authors of both papers propose static methods to identify DL in the code before the models are even trained. They describe the problem in the context of code cells and notebooks and use the

cell structure to track how the data is being treated, looking for common signs of leakage.

Vabalas et al. (2019) study ML validation when there are limited instances in the data set. The objective of the study was to verify if bias could be introduced in this low sample context by using inadequate validation methods. The study simulated the effects using five different validation approaches: Train/Test Split, K-fold, Nested, and two types of partially nested CV. It uses data originating from a synthetic data set generated by values from a Gaussian distribution. Simulation results showed that using only a K-Fold CV produced strongly biased results, even when increasing the number of samples. Additionally, the study showed that with pooled training and testing data, feature selection contributed more to the bias than hyperparameter tuning.

3.2 Effects of Data Leakage

Different studies focused on understanding and measuring the effects of DL in specific ML applications. Here, we summarize some representative examples.

Regarding the use of Convolutional Neural Networks for image classification, several studies analyze the effect of DL in model generalization. Effects of using correlated slices of 3D MRI data are presented in Yagis et al. (2021); the authors claim that the CV accuracy results could be erroneously boosted from 29% up to 55%. Similarly, the study of DL in digital pathology investigates the reproducibility issues of ML solutions in this area of knowledge (BUSSOLA et al., 2021). The authors claim that, when using different tiles from the same subject in both training and testing sets, predictive scores can be inflated by up to 41%.

Some studies investigate the consequences of DL in feature handling. In Samala et al. (2020), the authors tested the effect of DL on feature selection in an application for breast cancer classification. They inserted leakage by performing feature selection with influence from the validation group. The results show that values for Area Under the ROC curve with the leaked model estimated performances of 75% to 99% when the independent test showed that the values could only reach 72%. The study also showed that when reducing the sample size, the effects of DL were increased, with higher performance estimates and lower performance on independent tests.

In Shim, Lee and Hwang (2021), the authors investigate the effects of DL during feature selection in applications for diagnosing neuropsychiatric diseases and predicting patients' treatment. The study analyses the impact of DL in two test cases: using random

variable features created with Gaussian distributions for 2-class predictions and using actual clinical data. The results showed an overestimation of the performance in both cases when feature selection was not done properly.

The study presented by Cawley and Talbot (2010) focuses on introducing bias that occurs when not dealing with hyperparameter tuning correctly. The authors discuss examples of biased evaluation on model selection. The bias during model selection is introduced when the same data is seen in validation set used for hyperparameter tuning and in the test set. The paper also provides an unbiased performance evaluation methodology that, according to authors, "*correctly accounts for any overfitting that may occur in model selection*".

The work by Kuncheva and Rodríguez (2018) studies the bias introduced in feature selection when the same data is used for the pre-processing step and the model evaluation. The focus of the study is on very low-sample data, which incites the reuse of data. The authors also proposed an alternative protocol to avoid contamination using CV. The results show that their protocol achieved estimates closer to the ones seen on the independent test set compared to the biased method.

3.3 Summary

This section summarizes the related works presented. Table 3.1 contains a relation of the topics approached by each work mentioned in sections 3.1 and 3.2, alongside the topics of this work. As we may note, previous works did not evaluate the effects of DL in different stages of pre-processing and model development as extensively as proposed in the current work.

Table 3.1: Summary of the related work and expected contributions of this study.

	Define and Understand DL	Propose Strategy to Avoid DL	Asses Effects of Data Reuse	Asses Effects on Feature Selection	Asses Effects on Hyperparameter Tuning	Asses Effects on Normalization	Asses Effects on Value Impute	Investigate Effects variation among Tasks	Investigate Effects variation among Algorithms	Investigate Effects on different Data sets
Cawley and Talbot (2010)		X			X					
Kaufman et al. (2012)	X	X								
Kuncheva and Rodríguez (2018)			X	X						
Farrow, Moore and Gašević (2019)		X								
Vabalas et al. (2019)		X								
Samala et al. (2020)				X						
Bussola et al. (2021)		X	X							
Shim, Lee and Hwang (2021)				X						
Yagis et al. (2021)			X							
Drobnjaković, Subotić and Urban (2022)		X								
Kapoor and Narayanan (2022)	X	X		X						X
Yang et al. (2022)		X								
This Work				X	X	X	X	X	X	X

Source: The Author

4 METHODOLOGY

This chapter will present the methodology used to select the data sets for our experiments and conduct model training with and without DL. Our experiments are designed to assess if DL affects performance estimates in classification models. Section 4.1 covers how the data sets were selected and presents metadata about them. Section 4.2 details how the experiments were conducted for each type of DL evaluated in our work, outlining the approach to insert and avoid DL in each case, and also covering the parameters used in our experiments.

4.1 Data Sets

Aiming to use data sets with standardized formats, no pre-processing, and no missing values, we selected as source for this work data set from the Penn Machine Learning Benchmarks (PMLB)(ROMANO et al., 2021). The benchmark data provided by this curated collection comes with metadata associated with each data set. Besides that, PMLB also provides an API to easily retrieve the benchmark data sets, facilitating the batch analyses required for our experiments.

Prior to the selection of the subset of data sets used for the experiments, a few filters were applied to guarantee they comply with the following guidelines: all selected data sets must be for a classification task; the class imbalance of the chosen data set must be lower than 0.2 according to the imbalance index provided by PMLB; the selected data sets must have no more than 10,000 instances. These guidelines were defined to guarantee that the models trained would all be for the same type of task, to remove the influence of larger class imbalance in the experimental results, and to ensure that the model training time would not be overwhelming.

In the scope of this work, a total of 30 data sets were selected from PMLB. To make this selection, a series of Principal Component Analysis (PCA) were applied over data sets metadata, also shown in Table 4.1. A PCA consists of reducing the dimensions of a data set while retaining most of the variability from the data (RINGNÉR, 2008). The selection process was run in six steps. Each step consisted of selecting the two main metadata with PCA and selecting the five data sets farthest from the remaining data sets when considering the two selected columns. Table 4.1 contains the 30 selected data sets, alongside their metadata, in the order they were selected.

Table 4.1: Selected data sets with their respective metadata.

Data set	*Inst.*	*Feats*	*Bin. Feats*	*Categ. Feats*	*Cont. Feats*	*Classes*	*Imbalance*	*Feat/Class*	*Inst/Class*	*Feat/Inst*
GAMETES_Epistasis_2_Way_1000atts_0.4H_EDM_1_EDM_1_1	1600	1000	31	969	0	2	0	500	800	1.60
agaricus_lepiota	8145	22	5	16	1	2	0	11	4072.5	370.23
mushroom	8124	22	5	16	1	2	0	11	4062	369.27
ring	7400	20	0	0	20	2	0	10	3700	370.00
twonorm	7400	20	0	0	20	2	0	10	3700	370.00
clean1	476	168	0	0	168	2	0.02	84	238	2.83
dna	3186	180	180	0	0	3	0.08	60	1062	17.70
phoneme	5404	5	0	0	5	2	0.17	2.5	2702	1080.80
mfeat_pixel	2000	240	0	240	0	10	0	24	200	8.33
banana	5300	2	0	0	2	2	0.01	1	2650	2650.00
mfeat_factors	2000	216	0	0	216	10	0	21.6	200	9.26
spambase	4601	57	0	0	57	2	0.04	28.5	2300.5	80.72
Hill_Valley_with_noise	1212	100	0	0	100	2	0	50	606	12.12
Hill_Valley_without_noise	1212	100	0	0	100	2	0	50	606	12.12
waveform_40	5000	40	0	0	40	3	0	13.33	1666.67	125.00
waveform_21	5000	21	0	0	21	3	0	7	1666.67	238.10
movement_libras	360	90	0	0	90	15	0	6	24	4.00
satimage	6435	36	0	0	36	6	0.03	6	1072.5	178.75
chess	3196	36	35	1	0	2	0	18	1598	88.78
kr_vs_kp	3196	36	35	1	0	2	0	18	1598	88.78
optdigits	5620	64	3	10	51	10	0	6.4	562	87.81
splice	3188	60	0	60	0	3	0.08	20	1062.67	53.13
texture	5500	40	0	0	40	11	0	3.636	500	137.50
sonar	208	60	0	0	60	2	0	30	104	3.46
molecular_biology_promoters	106	57	0	57	0	2	0	28.5	53	1.86
mfeat_fourier	2000	76	0	0	76	10	0	7.6	200	26.31
analcatdata_authorship	841	70	0	3	67	4	0.08	17.5	210.25	12.01
tokyo1	959	44	0	2	42	2	0.08	22	479.5	21.79
soybean	675	35	1	34	0	18	0.04	1.94	37.5	19.29
mfeat_karhunen	2000	64	0	0	64	10	0	6.4	200	31.25

Source: The Author

4.2 Proposed Experiments

This section details how the experiments were executed to obtain performance estimates for ML models with and without DL. The subsections 4.2.1 to 4.2.4 cover the experimental procedure for each of the four DL possibilities presented in Section 2.4: normalization, value imputation, feature selection, and hyperparameter tuning.

All experiments were conducted with six repetitions of a 5-fold stratified CV, with each repetition using a different, randomized folds split. However, the same configuration of folds split is used across different experiments, to allow comparisons of DL effects among different tasks. The performance estimates obtained for each repetition are given by the average across the corresponding 5-fold CV execution. During our experiments, we monitored two selected metrics:

- **Balanced Accuracy**: Average accuracy by class, used to deal with imbalanced data sets.
- **F1-score**: Harmonic mean of Precision (i.e., positive predictive value) and Recall (i.e., true positive rate), representing both values in one metric.

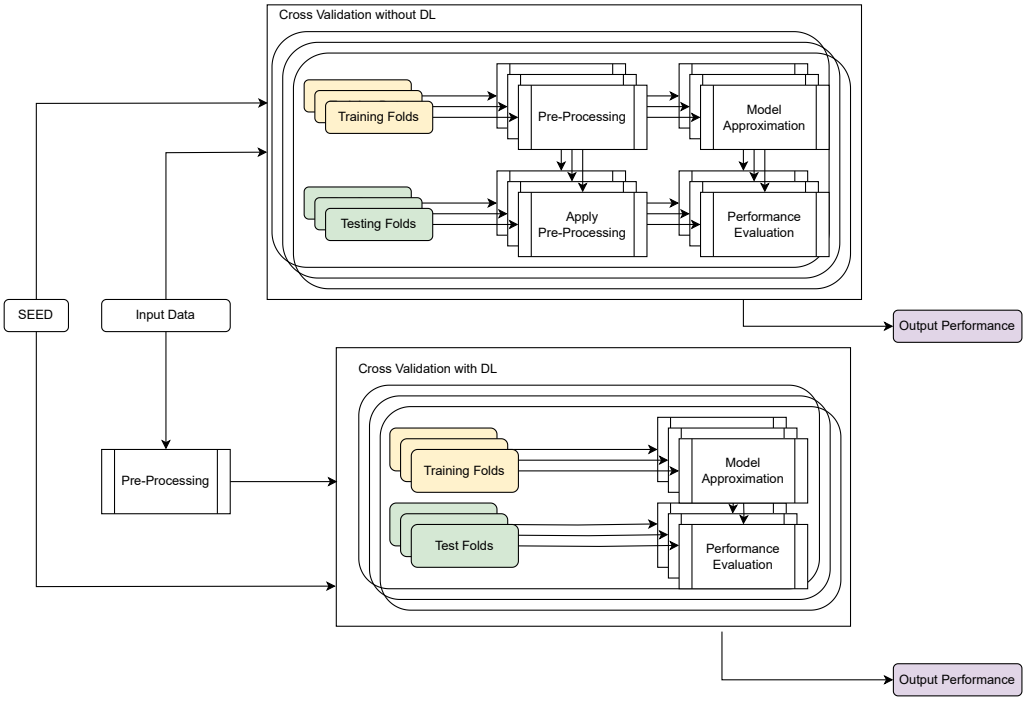
4.2.1 Experiments for normalization

The experiments focused on normalization aimed to obtain performance estimates for models with and without DL in the normalization stage. With that in mind, the diagram presented in Figure 4.1 shows the experimental process for one repetition of the CV process, considering the "Pre-Processing" step to be the normalization task. To simulate the DL case, all data is normalized before the CV. In the DL-free case, the normalization occurs internally within every iteration of the CV process, estimating the normalization parameters only based on the training data.

4.2.2 Experiments for value imputation

Since the data sets provided by PMLB do not have missing values, as explained in Section 4.1, the experiments for value imputation required the systematic insertion of missing values into the data sets. For this purpose, we used the Missing at Random approach to artificially generate missing values. This strategy inserts missing data randomly

Figure 4.1: Diagram detailing one repetition of the cross-validation for the experiments with data pre-processing tasks.



Source: The Author

into each of the data columns selected without taking into consideration any information about the instances. We used the implementation provided by the r-miss-tastic (MAYER et al., 2019) project. To enable a broader range of analysis, the experiments were executed for four variations of percentage of missing values: 5%, 10%, 20%, and 30%. Nevertheless, to guarantee equality of the number of experiments considered when analyzing the general effects of DL across all tasks here considered, we used only the results for 20% missing values as the representative for value imputation experiments. Moreover, our experiments also included three imputation techniques: Average value, Median value, and KNN-based imputation.

To conduct the experiments, at first, the missing values are randomly inserted, generating a new data set that will be used in the analyses with and without DL. Figure 4.1 depicts the testing process for each repetition after the insertion of missing values. For the task of value imputation, the "Pre-Processing" step can be interpreted as the step of missing values imputation. Following that, the DL experiment imputes the values using the entirety of the data set. In other words, all instances are used to learn the parameters for the imputation (i.e., or fit the imputer). On the other hand, the model without DL uses only the training folds on each iteration to fit the imputer, which is then used to impute values on the training folds and test fold, separately.

4.2.3 Experiments for feature selection

The experiments conducted for feature selection follow the same basic steps previously presented in Figure 4.1, except that the "Pre-Processing" step is replaced by a feature selection analysis.

To select a subset of features, we used the *SelectPercentile* function from the scikit-learn Python package (PEDREGOSA et al., 2011). This method uses a scoring function (scikit-learn's *f_classif* in the scope of this work) to evaluate the relevance of each feature and selects a subset of features according to a percentile of the highest scores. The experiments with DL use all instances to evaluate and select the target features with the *SelectPercentile* function. On the other hand, in the experiments without DL, only the instances selected for training on each iteration of the CV are used for feature selection.

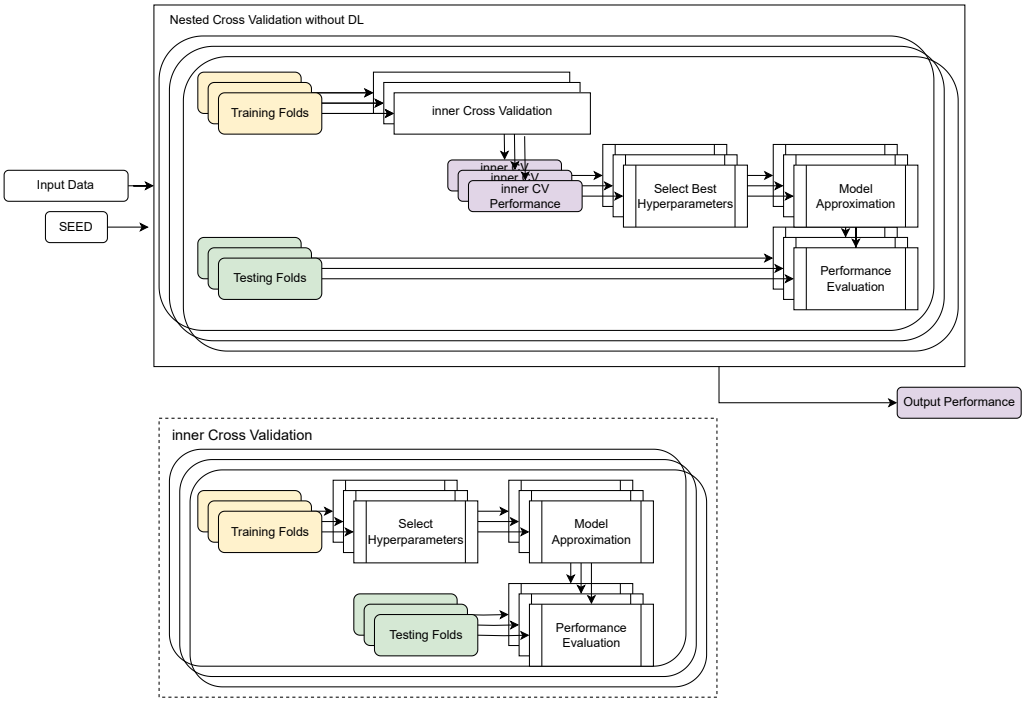
These experiments were run for multiple configurations of the parameter that defines the percentile of features to keep in the analysis: 1%, 5%, 10%, and 20%. Similarly to the experiments described in Section 4.2.2, to guarantee equality of the number of experimental data points when analyzing the general effects of DL, only results for 10% were used in the analysis.

4.2.4 Experiments for hyperparameter tuning

The experiments for hyperparameter tuning were implemented with a different approach. The experiments without DL used a Nested CV, detailed in Section 2.3.3, to ensure there would be no leakage during the process of hyperparameter tuning. The experiments with DL used a standard k-fold CV, which means employing the same CV to do both the hyperparameter tuning and the model performance evaluation.

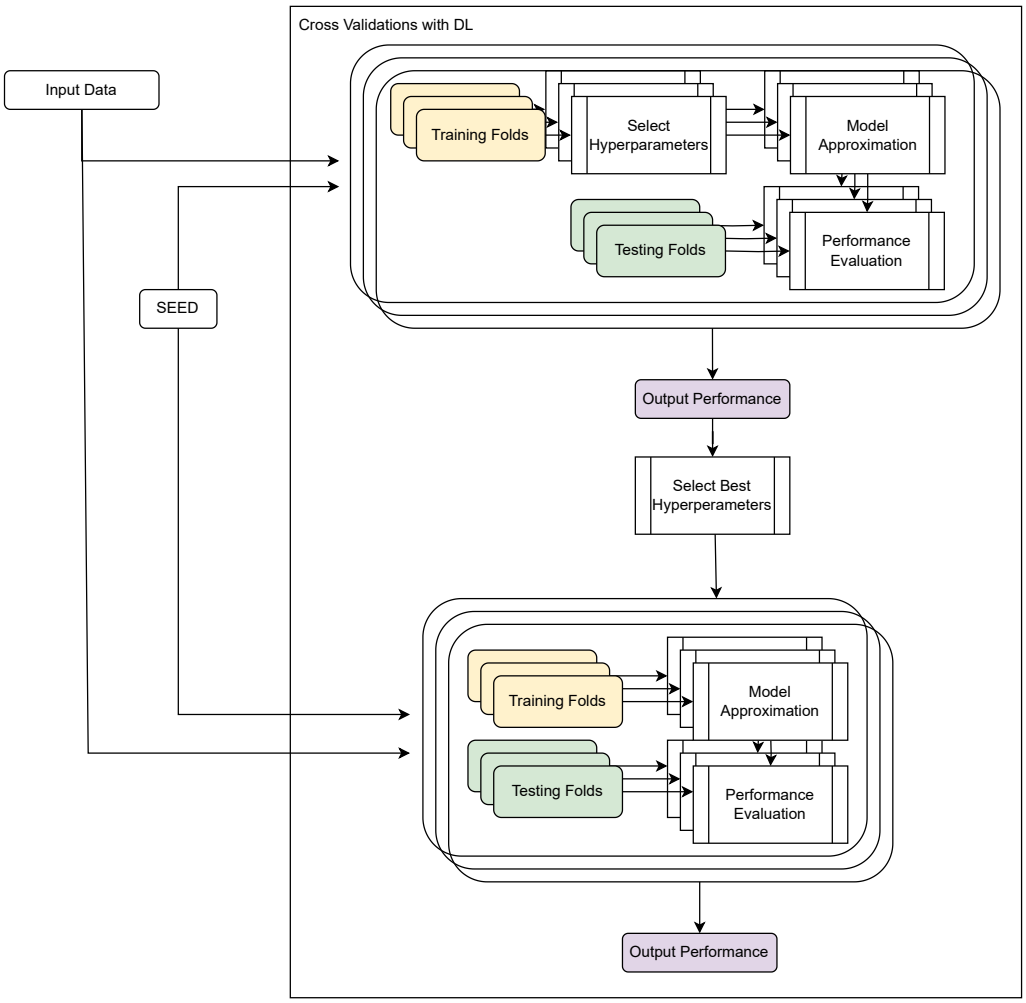
Figure 4.2 shows a diagram explaining the steps involved in each repetition of the experiments without DL. This diagram contains the outer CV and a box detailing the inner CV. Figure 4.3 presents the diagram for the experiments that suffer of DL during hyperparameter tuning. It is possible to notice that the whole data is used to select the best hyperparameters and evaluate the performance. Appendix A has the complete list of hyperparameters used for each algorithm.

Figure 4.2: Diagram showing one repetition of the experiments for hyperparameter tuning with nested CV to avoid data leakage.



Source: The Author

Figure 4.3: Diagram showing one repetition of the experiments for hyperparameter tuning with standard CV, causing data leakage.



Source: The Author

5 RESULTS

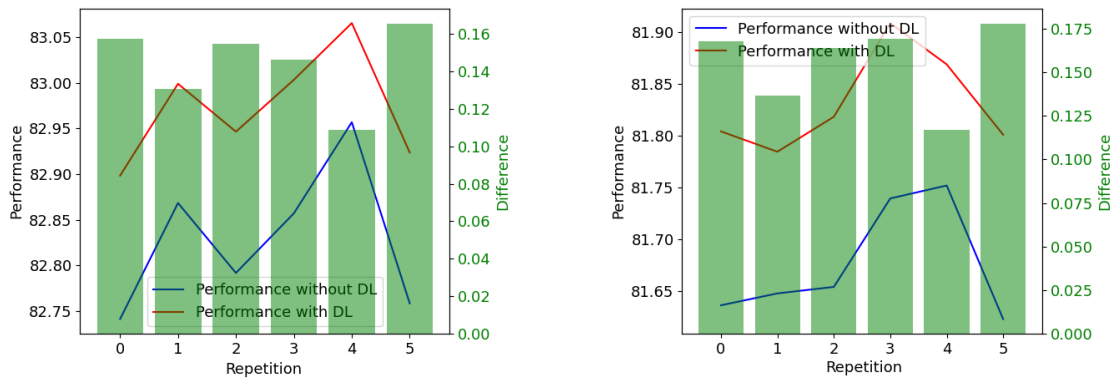
This chapter is dedicated to presenting and analyzing the results obtained for the four different experiments detailed in the previous chapter (Section 4.2), involving DL in the tasks of normalization, value imputation, feature selection, and hyperparameter tuning. All performance estimates reported here are shown in a range of 0 to 100% for the respective metric. By the end of this chapter, we intend to have answered the following research questions using data acquired from our experiments:

- Does data leakage impact performance evaluation?
- Is the effect of data leakage more prominent in a specific task?
- Do data leakage effects change for different data sets?
- Is there a machine learning algorithm more sensitive to data leakage effects?
- Is there a correlation between the metadata of the data sets and the consequences of data leakage?

5.1 Does data leakage impact performance evaluation?

This section explores the possible impacts of DL on performance evaluation. Using a combined analysis of all results from the experiments described in Section 4.2, we aim to assess if data leakage has any consequences in the model evaluation and, if so, how it affects performance estimate.

Figure 5.1: Overall analysis of the average scores and difference in scores considering all experiments conducted.



(a) Balanced accuracy scores

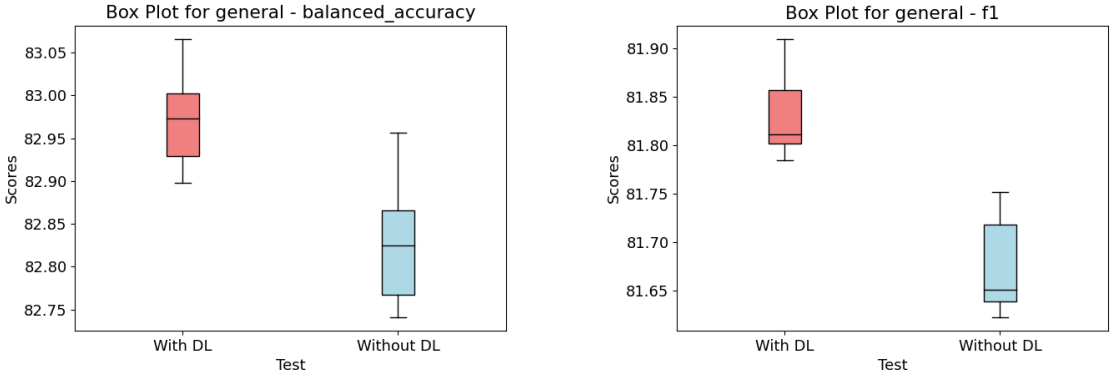
(b) F1 scores

Source: The Author

Figure 5.1 shows the first evidence to be analyzed. The plots show the average scores of the experiments on each of the six repetitions of the 5-fold CV. In red, we present the results for models with DL, while the results for models without DL are shown in blue. The difference between the results for models with and without DL in each repetition is depicted in the green bars.

As we can see in Figure 5.1a, the balanced accuracy scores for the models with accidental data contamination are consistently higher than those for the uncontaminated models. It is possible to note that the differences in the average scores for this metric range from 0.1 to around 0.16 in favor of DL, indicating a slight average increase in every repetition of the repeated CV. Similarly, Figure 5.1b shows consistently higher F1 scores for the models with DL. The increases in the average repetition scores range from 0.125 to 0.175.

Figure 5.2: Distributions of average scores for all experiments conducted.



(a) Average balanced accuracy scores (b) Average F1 scores

Source: The Author

Figure 5.2 presents the results using a different type of visualization based on box plots. We compare the distribution of the average results across all experiments conducted. In Figure 5.2a, the distribution and the median values of the balanced accuracy scores in the contaminated models (shown in red) are clearly above the values for the uncontaminated models (shown in blue). Although the differences among the average values are not so expressive, it still demonstrates overoptimistic results. Regarding the box plots for the F1 score (Figure 5.2b), we also note performance advantages for the models with DL. One interesting aspect is the apparent smaller variation of results for DL compare to those without DL. Besides that, again, we can see slightly bigger values for the leaked models.

Thus, we conclude that DL indeed causes impact in the performance evaluation of

predictive models. Given that the overall analysis provided here encompasses all experiments, some of the most significant effects can be diluted by looking only at the average. Therefore, the next sections will investigate the experiments in more detail, noting the effects associated with variations in tasks, datasets, and algorithms.

5.2 Is the effect of data leakage more prominent in a specific task?

This section is dedicated to presenting and analyzing the results of the experiments by type of task (i.e., data pre-processing or hyperparameter tuning), aiming to investigate if any of these tasks seems to be more susceptible to the effect of DL. Subsection 5.2.1 focuses on presenting and analyzing the results on a task basis, discussing our findings for feature selection, hyperparameter tuning, missing values imputation, and normalization. The comparisons between DL effects on different tasks are presented in Subsection 5.2.2.

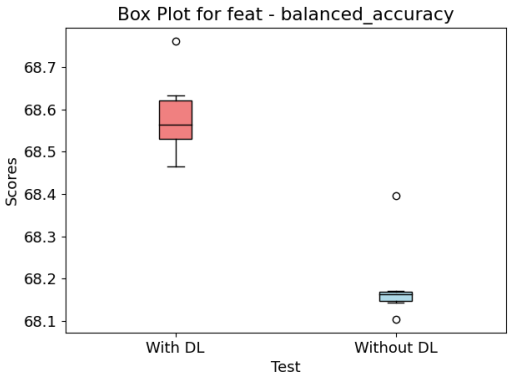
5.2.1 Results evaluation per task

This section comprehensively analyzes the CV scores for each selected task. These scores provide insights into the effects of DL according to the stage in the model development cycle in which it is inserted.

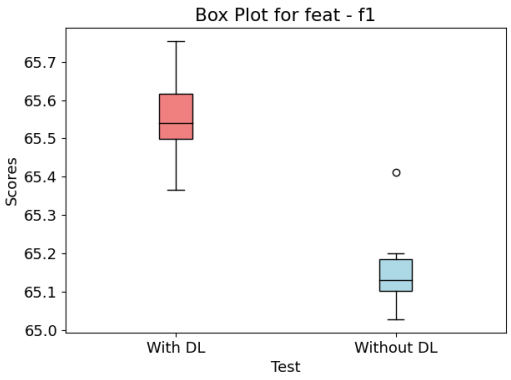
The first task to be analyzed is feature selection. Figure 5.3 presents the summary of the results for this pre-processing step. These plots present results from all experiments with different percentages of selected features (1%, 5%, 10%, and 20%). The box plot of the distribution of balanced accuracy scores (Figure 5.3a) shows that the models with DL achieved higher performance than models without DL, even when we consider the top-performing outlier for models without DL. We also notice that the median for the experiments with accidental data contamination is close to 68.6, while the results for the experiments without contamination show a median close to 68.2. A similar pattern is observed for the F1 scores (Figure 5.3b), with the DL experiments having a much higher median than the experiments without DL, and also showing overall superior results.

Another interesting analysis can be done based on the plots of average results and score differences for balanced accuracy and F1 scores considering the six CV repetitions, as shown in Figures 5.3c and 5.3d, respectively. The results for both metrics show an advantage for the models with DL, with the average values being consistently higher at

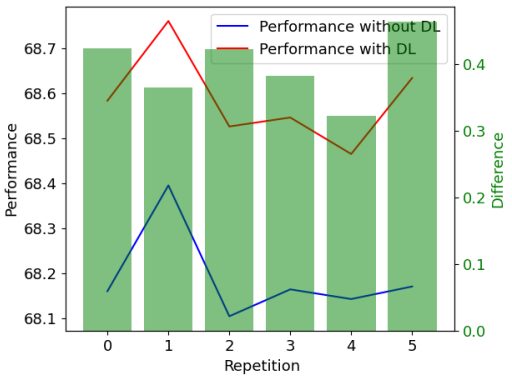
Figure 5.3: Summary of the results for feature selection experiments.



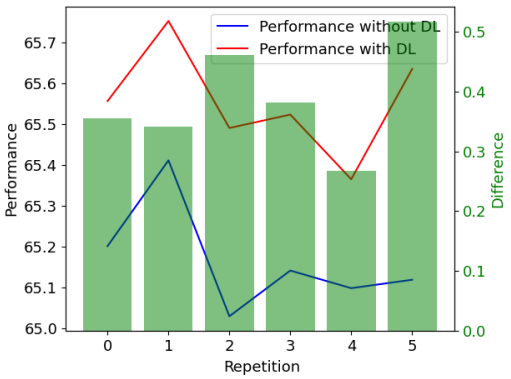
(a) Distribution of balanced accuracy scores



(b) Distribution of F1 scores



(c) Average values and performance differences for balanced accuracy scores



(d) Average values and performance differences for F1 scores

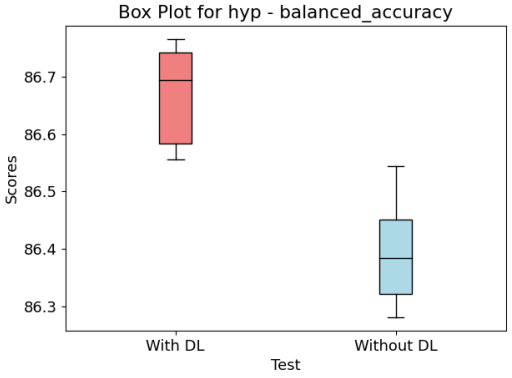
Source: The Author

every repetition. Moreover, differences in performance range from 0.3 to 0.5 in balanced accuracy (and similar results in F1 scores), meaning an overoptimistic evaluation. Results for each different percentage of selected features are shown in Appendix B. We note that the findings are very similar, with positive effects over performance observed for almost all repetitions of CV, especially considering the results for 10% and 20% selected features.

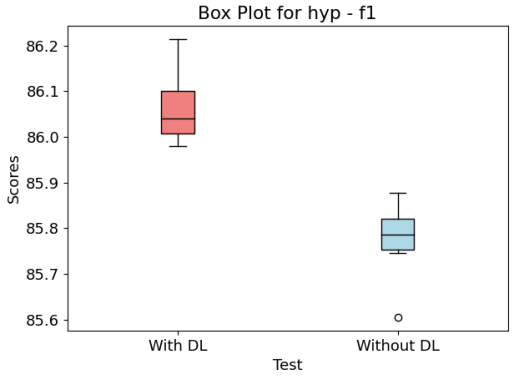
Regarding the experiments for hyperparameter tuning, Figure 5.4 provides a summary of our results, structured similarly to the plots analyzed for feature selection. The box plot for balanced accuracy (Figure 5.4a) shows higher scores for the models with DL. The median value of the results for DL is around 0.3 higher than the median for the pipeline without DL. A similar trend is observed when analyzing the distribution for F1 scores (Figure 5.4b. Even though the experiments without data contamination show smaller variations, the experiments with DL generated slightly higher results.

The comparisons among average values and score differences per repetition are

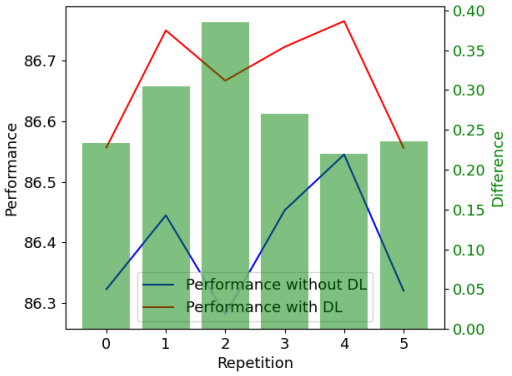
Figure 5.4: Summary of the results for hyperparameter tuning experiments.



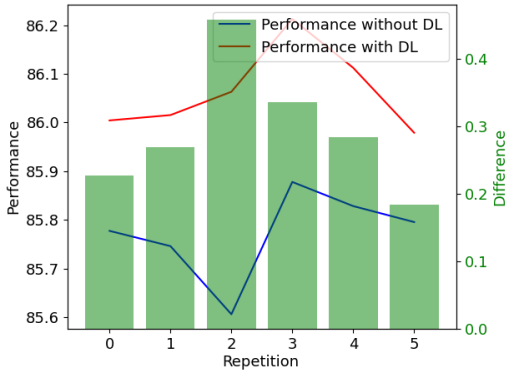
(a) Distribution of balanced accuracy scores



(b) Distribution of F1 scores



(c) Average values and performance differences for balanced accuracy scores



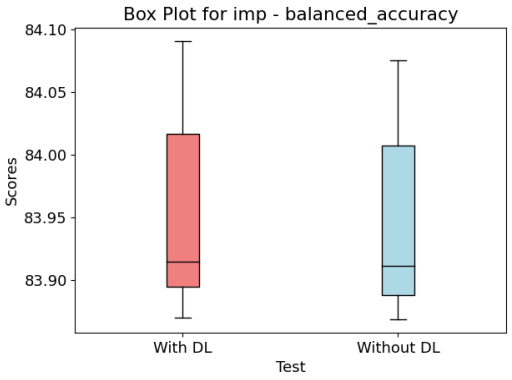
(d) Average values and performance differences for F1 scores

Source: The Author

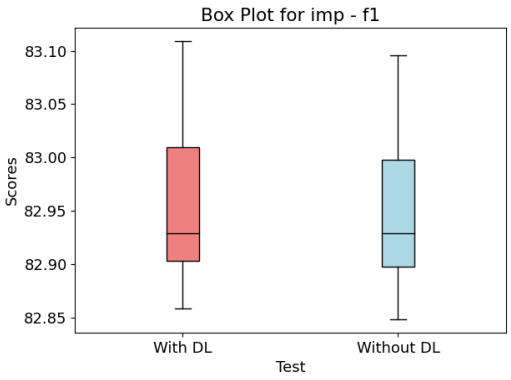
shown in Figures 5.4c and 5.4d. Both plots show that the differences are consistently positive, showing a possibly unnatural increase in performance estimates when DL is accidentally introduced. The average performances per repetition were around 0.2 to 0.4 higher for balanced accuracy and approximately 0.2 to 0.45 more elevated for F1 scores.

The summary of our results for experiments focused on missing value imputation (Figure C.2) shows a different scenario. This depiction of results for all value imputation experiments, considering all selected percentiles of missing data (5%, 10%, 20%, and 30%) and imputers (Average, Mean, and KNN), indicate lower influence from DL. Taking into consideration the distribution of performance between models with DL and models without DL during value imputation (Figures C.2a and C.2b), we notice that there are no significant differences in performance introduced by DL. The medians for these experiments were very close among both scenarios and both metrics analyzed. Besides that, the variations in performance are also very similar. Thus, we can not observe a clear

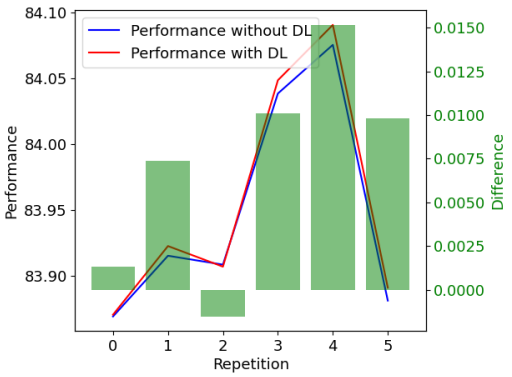
Figure 5.5: Summary of the results for missing value imputation experiments.



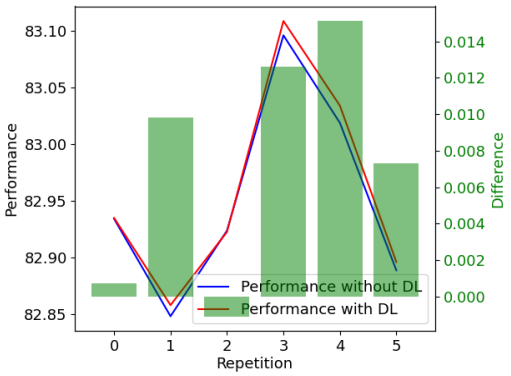
(a) Distribution of balanced accuracy scores



(b) Distribution of F1 scores



(c) Average values and performance differences for balanced accuracy scores



(d) Average values and performance differences for F1 scores

Source: The Author

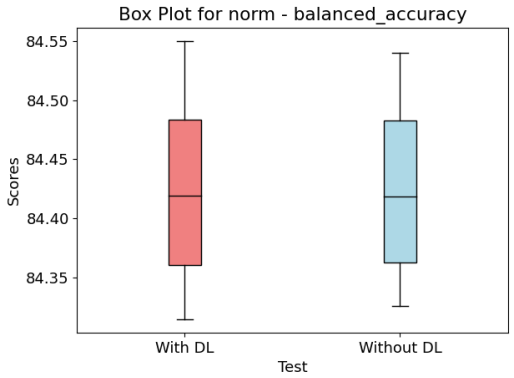
influence of DL over performance estimates.

Analyzing the average values and score differences across distinct repetitions (Figures C.2c and C.1d), we can see that the curves associated with the models with and without DL are practically superimposed and that in cases where some difference in performance is observed, this difference is insignificant. In general, the performance impact appears in a much smaller scale than the ones perceived for hyperparameter tuning and feature selection. The highest different for balanced accuracy and F1 score is around 0.015. The analysis of effects for different percentages of missing values and different strategies of data imputation indicated similar results for all values. The detailed results are presented in Appendix C.

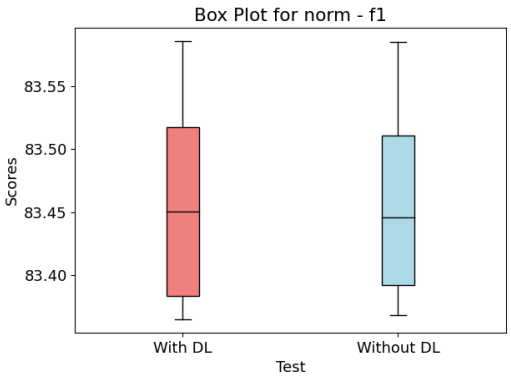
Likewise, the results of the experiments for data normalization also diverge from the results for hyperparameter tuning and feature selection tasks. The box plots depicted in Figures 5.6a and 5.6b are practically identical for models trained with and without DL

during normalization. Additionally, while the analysis of score differences per repetition show oscillations of positive and negative effects over performance (Figures 5.6c and 5.6d), without surpassing the absolute difference of 0.01, the comparison among average scores suggest a very small impact of DL in performance estimates since the curves tend to be superimposed.

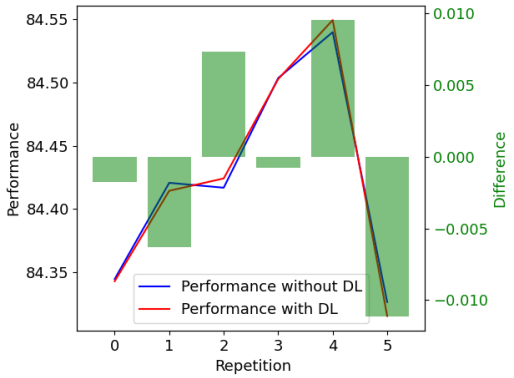
Figure 5.6: Summary of the results for normalization experiments.



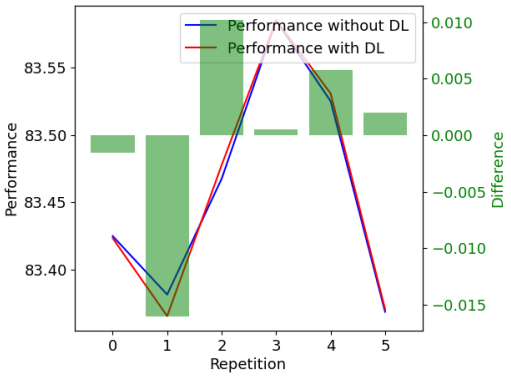
(a) Distribution of balanced accuracy scores



(b) Distribution of F1 scores



(c) Average values and performance differences for balanced accuracy scores



(d) Average values and performance differences for F1 scores

Source: The Author

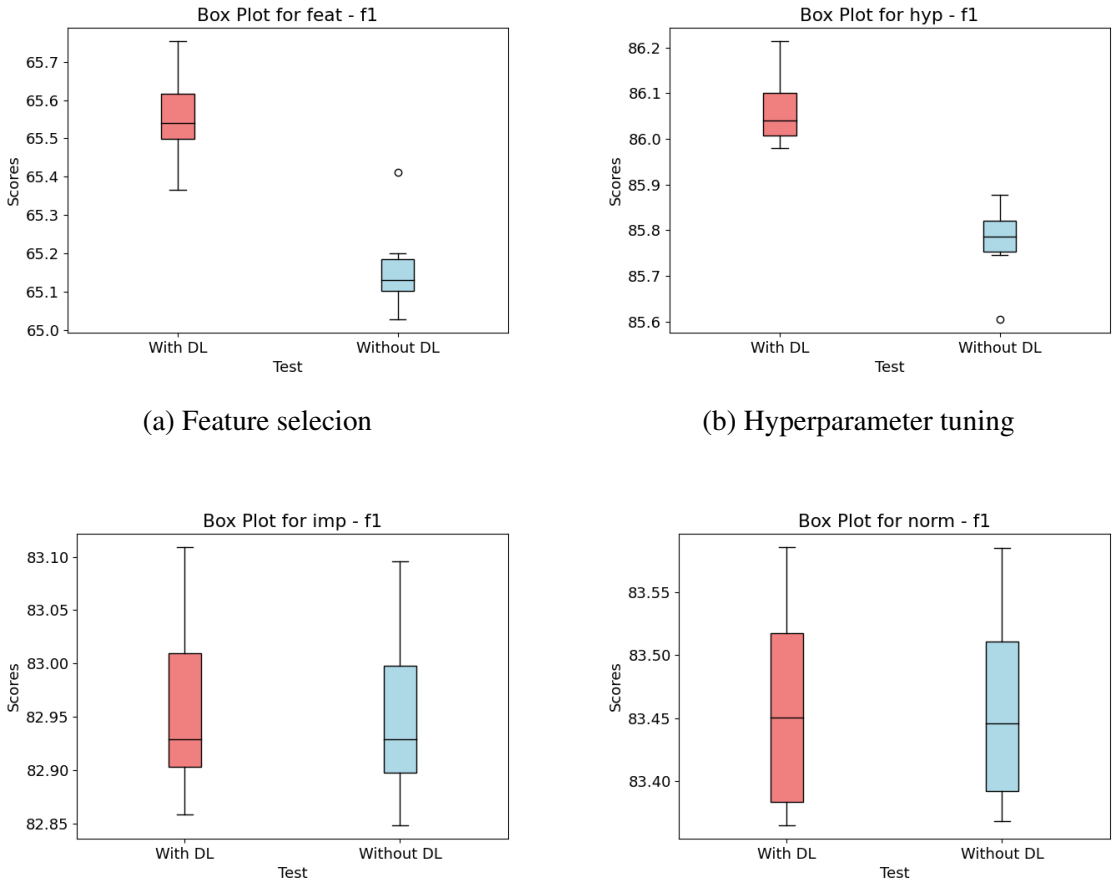
5.2.2 Comparison among tasks

This subsection aims to review the results presented previously in a comparative fashion, with the intention to assess how the insertion of DL in each task affects performance evaluation. To summarize results and compare the effects of DL among distinct tasks addressed, Figure 5.7 unites all F1 scores box plots presented in Subsection 5.2.1.

As we may note through this direct comparison, the effects of DL in fact vary when

data contamination is accidentally inserted in different tasks. Models with DL have, on average, higher scores than their counterparts without DL in the hyperparameter tuning (Figure 5.7b) and feature Selection experiments (Figure 5.7a). In both cases, the average increase in performance estimates was close to 0.3. On the other hand, experiments for missing value imputation (Figure 5.7c) and normalization (Figure 5.7d) did not show evidence of a significant impact of DL on model performance evaluation. Different explanations for the fact that DL inserted during Feature Selection and Hyperparameter Tuning had more impact on the performance estimates are possible. One interpretation is that both tasks can have high impact on the final model: Feature Selection actively selects which data will be used for the training, while Hyperparameter Tuning adjusts the algorithms' parameters to maximize performance. Another possible reason is that both tasks also have a high capacity to improve the model's performance, causing higher increases when DL is inserted.

Figure 5.7: Comparison of F1 score results with and without DL for different tasks



(a) Feature selecion

(b) Hyperparameter tuning

(c) Missing value imputation

(d) Normalization

Source: The Author

5.3 Do data leakage effects change for different data sets?

This section aims to assess and discuss the effects of DL on different data sets. We first provide graphic examples for three groups of data sets defined based on the results obtained (Table 5.1). We aim to use selected data sets from each group to illustrate the general picture of all tested data sets. Next, we emphasize specific cases that showed more significant DL impact based on the combined results of Section 5.2 and Subsection 5.3.1.

Table 5.1: Classification of data sets according to experimental analysis of DL effects.

Data set	*Classification*
GAMETES_Epistasis_2_Way_1000atts_0.4H_EDM_1_EDM_1_1	Significantly Increased
agaricus_lepiota	Decreased or Not Changed
mushroom	Decreased or Not Changed
ring	Decreased or Not Changed
twonorm	Significantly Increased
clean1	Slightly Increased
dna	Decreased or Not Changed
phoneme	Decreased or Not Changed
mfeat_pixel	Decreased or Not Changed
banana	Slightly Increased
mfeat_factors	Slightly Increased
spambase	Decreased or Not Changed
Hill_Valley_with_noise	Decreased or Not Changed
Hill_Valley_without_noise	Significantly Increased
waveform_40	Slightly Increased
waveform_21	Significantly Increased
movement_libras	Significantly Increased
satimage	Slightly Increased
chess	Slightly Increased
kr_vs_kp	Slightly Increased
optdigits	Significantly Increased
splice	Decreased or Not Changed
texture	Decreased or Not Changed
sonar	Significantly Increased
molecular_biology_promoters	Significantly Increased
mfeat_fourier	Slightly Increased
analcatdata_authorship	Significantly Increased
tokyo1	Slightly Increased
soybean	Significantly Increased
mfeat_karhunen	Decreased or Not Changed

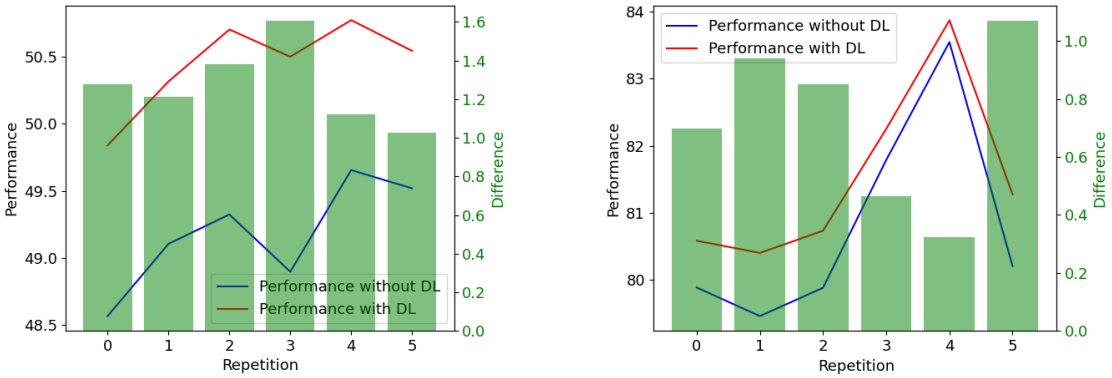
Source: The Author

5.3.1 Overall results for grouped data sets

We analyzed the overall results per data set considering all experiments conducted (i.e., with variations in the addressed task and in the learning algorithm) and defined three groups: (i) data sets for which the performance significantly increase upon DL; (ii) data sets for which the performance slightly increased upon DL; and (iii) data sets for which the performance decreased or did not change upon DL. Table 5.1 summarizes data sets classification according to this analysis.

The first group, depicted in Figure 5.8, is composed of data sets that had a relevant average increase in performance estimates for the average of all four experiments described in Section 4.2. As an example of data set for this group, we emphasize the results for the Gametes Epistasis Data set in Figure 5.8a, which showed the biggest difference in average performances. In this case, we observed F1 score differences ranging from 1.0 to 1.6 across the CV repetitions. Another data set with a significant average increase was the Molecular Biology Promoter data set, shown in Figure 5.8b, for which the presence of DL caused performance boosts ranging from 0.4 to 1.0 in F1 scores.

Figure 5.8: Average F1 scores and performance differences for data sets with significantly increased estimates upon DL.



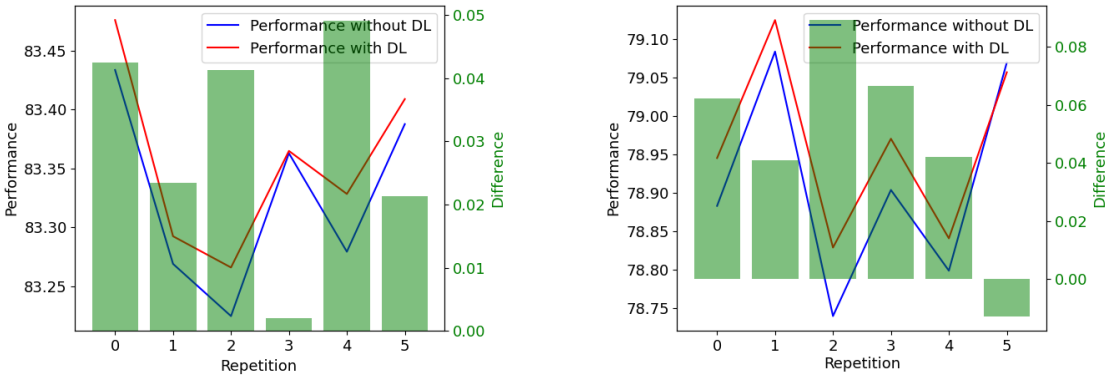
(a) Gametes Epistasis data set

(b) Molecular Biology Promoter data set

Source: The Author

The second group consists of data sets for which models had a slight increase in performance as a consequence of DL. Figure 5.9 provides two examples of this behavior. The results for Satimage (Figure 5.9a) and Mfeat Fourier (Figures 5.9b) data sets show a increase in the average performance estimates that does not exceeds 0.05 and 0.08 for each data set. Moreover, in the experiments with Mfeat Fourier there is one repetition where performance estimates were a little bit higher for models trained wit DL. Still, the

Figure 5.9: Average F1 scores and performance differences for data sets with slightly increased estimates upon DL.

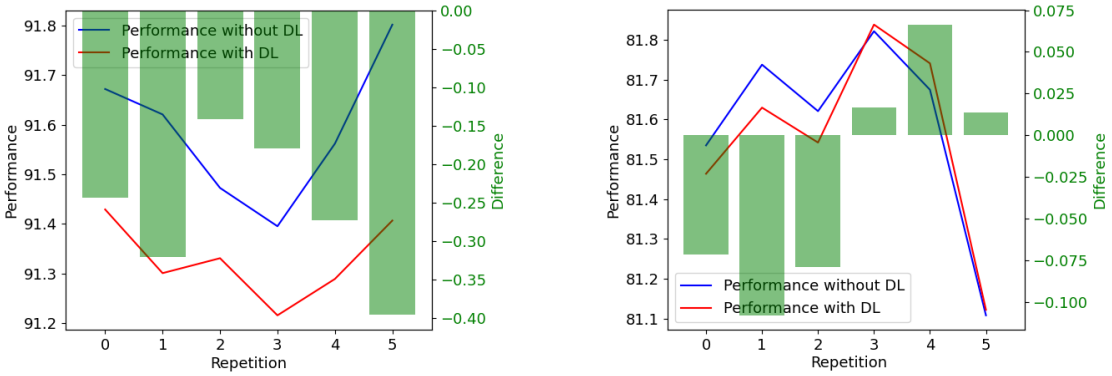


(a) Satimage data set (b) Mfeat Fourier data set

Source: The Author

overall results indicate that DL artificially boosted their results to a slight extent.

Figure 5.10: Average F1 scores and performance differences for data sets with decreased or unaffected estimates upon DL.



(a) Mfeat Pixel data set (b) Splice data set

Source: The Author

The final group of results is presented in Figure 5.10. These results show data sets with decreased average values or no apparent changes in average values for the performance estimates when data contamination is present in model development. The plots of Mfeat Pixel (Figure 5.10a) show the most significant decrease in average performance when DL is introduced. The Splice experiments (Figure 5.10b), on the other hand, show a scenario where models with DL have smaller performances on a few repetitions and slightly higher on others, with positive differences of at most 0.075 and maximum negative difference of 0.1.

5.3.2 Analysis of relevant combinations of data sets and tasks

Considering the results presented in Section 5.2 and Subsection 5.3.1, this subsection aims to analyze combinations of algorithms and tasks that showed the biggest impacts from DL. We provide results of data sets included in the "Significantly Increased" group, according to Table 5.1, and the tasks with the most meaningful effects of DL, hyperparameter tuning and feature selection.

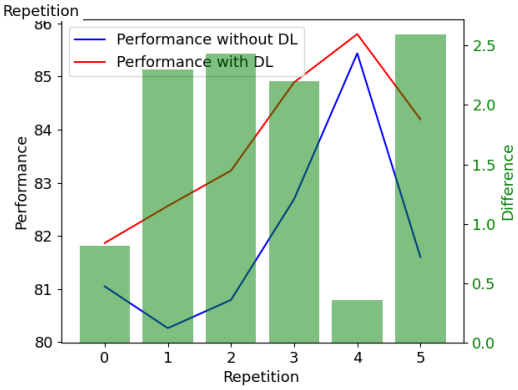
The results presented in Figure 5.11 exemplify relevant cases where DL effects over the average performance were very prominent for the hyperparameter tuning task. Molecular Biology Promoter results (Figure 5.11a) show average increases of up to 2.5 points in the F1 score metrics. Similarly, Sonar and Gametes results (Figures 5.11c and 5.11d) show increases of up to 1.75 and 3.0 in the performance estimates. Finally, despite our overall analysis per data set indicating that the Hill Valley With Noise data set presented decreased or unaffected performance estimates upon DL, the results considering solely the hyperparameter tuning experiments for this data set (Figure 5.11b) show an average increase of up to 4.0 in one of its CV repetitions. These results indicate the possibility of more impactful DL effects than the averages presented previously.

Results for the selected data sets analysed for the feature selection task are presented in Figure 5.12. Similar to hyperparameter tuning, Feature Selection experiments also indicated a high influence of DL in some data sets, including very notable increases for the Gametes data set (Figure 5.12a). This data set had significant increases ranging from 6.0 to 8.0 points in the performance percentages averages when DL was inserted. The feature selection results also indicate artificially increased performance estimates of up to 3.0 points for the Molecular Biology Promoter data set (Figure 5.12c); up to 1.4 points on the Twonorm data set (Figure 5.12d); and up to 2.5 points on the Sonar data set results (Figure 5.12b).

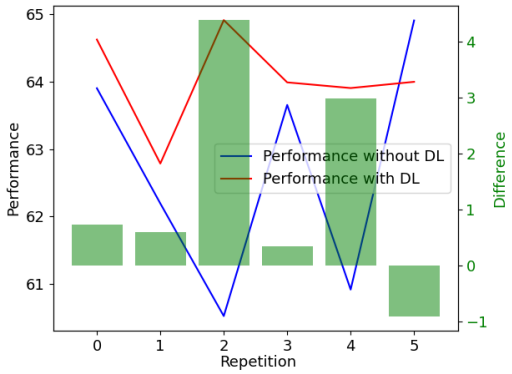
5.4 Is there a machine learning algorithm more sensitive to data leakage effects?

Investigating the distinct impact that DL can have on different algorithms is essential to better understanding this phenomenom. This section aims to provide evidence about how each of the algorithms depicted in Section 2.2 is affected by the presence of DL. The plots presented here for each algorithm consider results for every data set and every task anlyzed in the experiments.

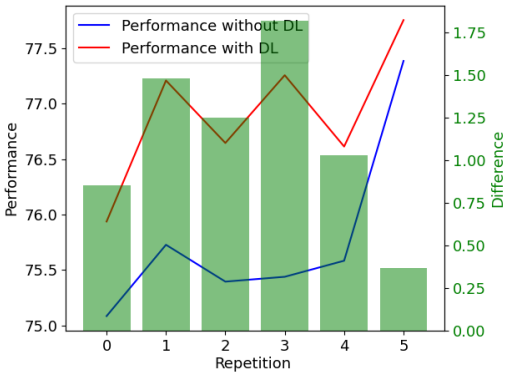
Figure 5.11: F1 score results for selected data sets with significant increase in performance upon DL during hyperparameter tuning.



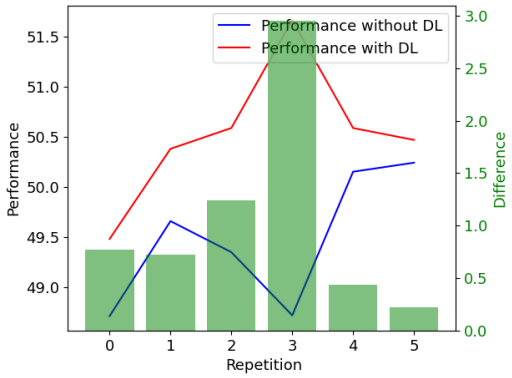
(a) Scores averages and differences for Molecular Biology Promoter data set



(b) Scores averages and differences for Hill Valley with noise data set



(c) Scores averages and differences for Sonar data set



(d) Scores averages and differences for Gametes Epistasis data set

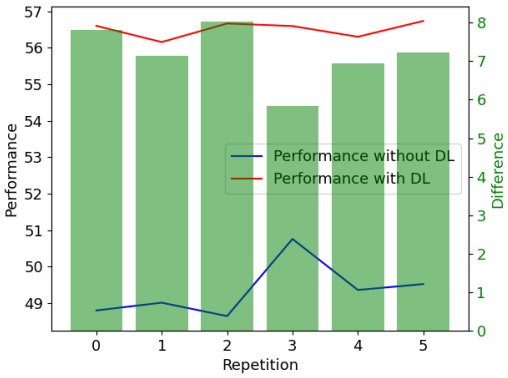
Source: The Author

The results for the KNN algorithm (Figure 5.13) show that the average scores per repetition for models with DL have higher values than the performance estimates without DL. The comparison among F1 scores (Figure 5.13a reach differences of up to 0.2, close to the average for all algorithms, presented in section 5.1. The same behavior is seen for the balanced accuracy results (Figure 5.13b).

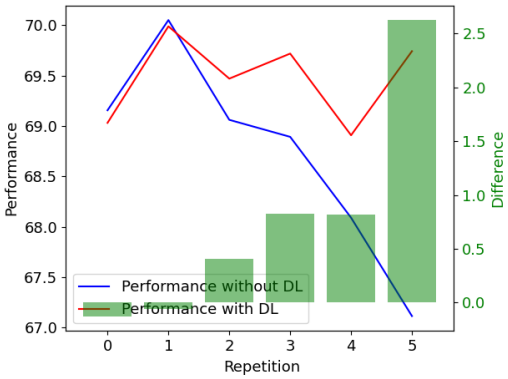
The results for DTs (Figure 5.14) are slightly different as they show a larger variation in the score differences among models with DL and without DL. The F1 scores (Figure 5.14a) have a maximum difference in average results of around 0.16 and a minimum difference close to 0.02. Similarly, balanced accuracy results (Figure 5.14b) show higher values for scores associated to DL experiments and higher variation in the score differences for each repetition.

Another similar result is presented in Figure 5.15 and concerns the Naïve Bayes

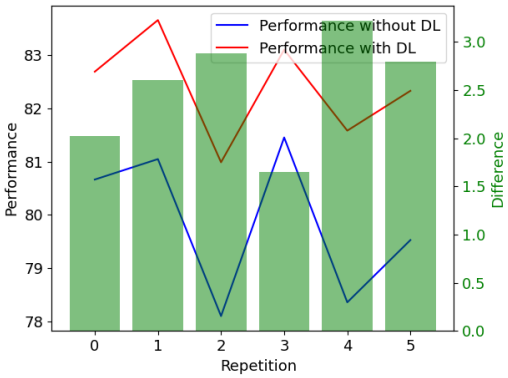
Figure 5.12: F1 score results for selected data sets with significant increase in performance upon DL during feature selection.



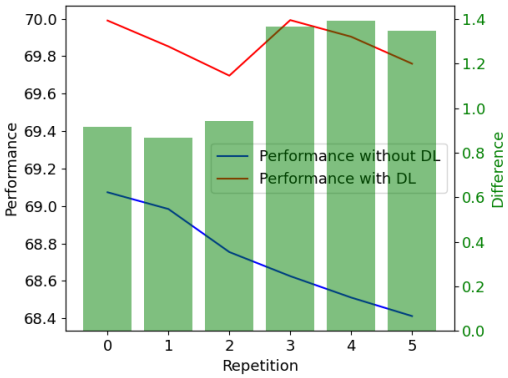
(a) Scores averages and differences for Gametes on Feature Selection



(b) Scores averages and differences for Sonar on Feature Selection



(c) Scores averages and differences for Molecular Biology Promoter data set



(d) Scores averages and differences for Twonorm data set

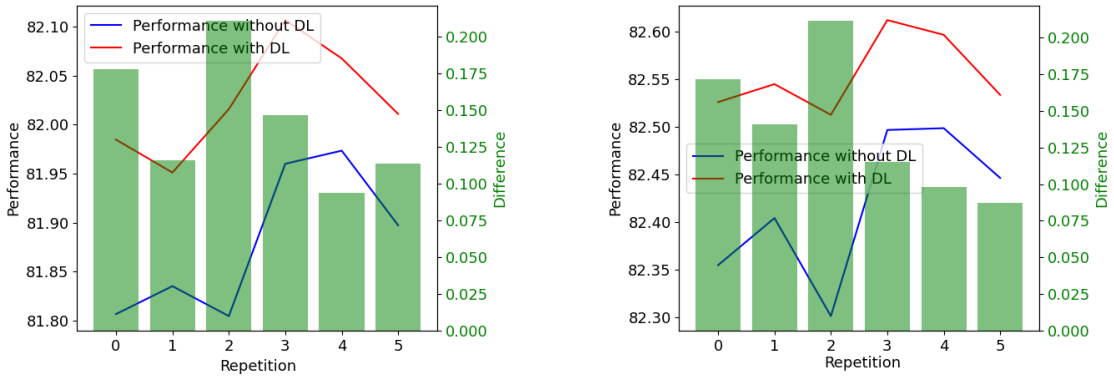
Source: The Author

algorithm. The average results for Naïve Bayes considering both the F1 score (Figure 5.15a) and balanced accuracy (Figure 5.15b) align with the observations for KNN and DTs, with a slight increase in the maximum score differences for F1 score. It is possible to note a difference of up to 0.25 in favor of DL models when considering the F1 score, and an increase of up to 0.20 for balanced accuracy.

Along the same lines, the results for the RF algorithm, presented in Figure 5.16 has score differences close to the average observed for the overall results and for the previous three algorithms. The same pattern of an increase of approximately 0.2 in performance estimate under the presence of DL is seen in F1 score (Figure 5.16a) and balanced accuracy (Figure 5.16b).

The results presented so far indicate similar results in KNN, DT, NB, and RF experiments. Nonetheless, according to the experimental data obtained, not all algorithms

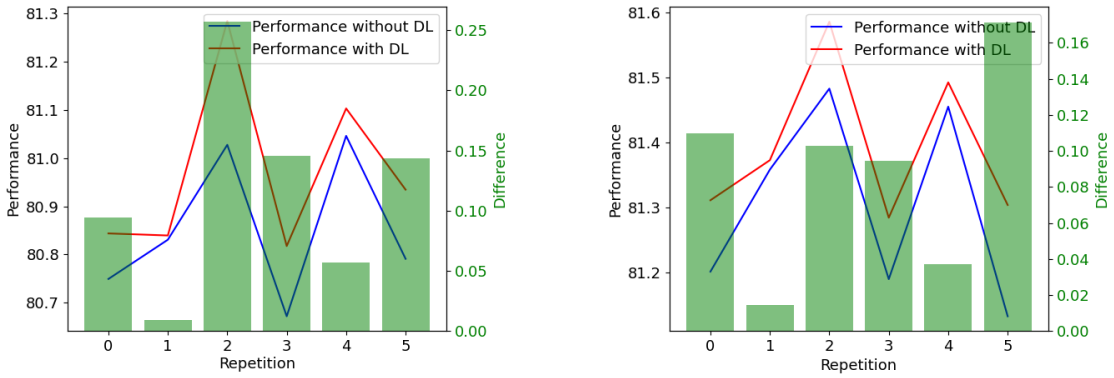
Figure 5.13: Average performance estimates and score differences per repetition for KNN.



(a) F1 score (b) Balanced accuracy

Source: The Author

Figure 5.14: Average performance estimates and score differences per repetition for Decision Trees.



(a) F1 score (b) Balanced accuracy

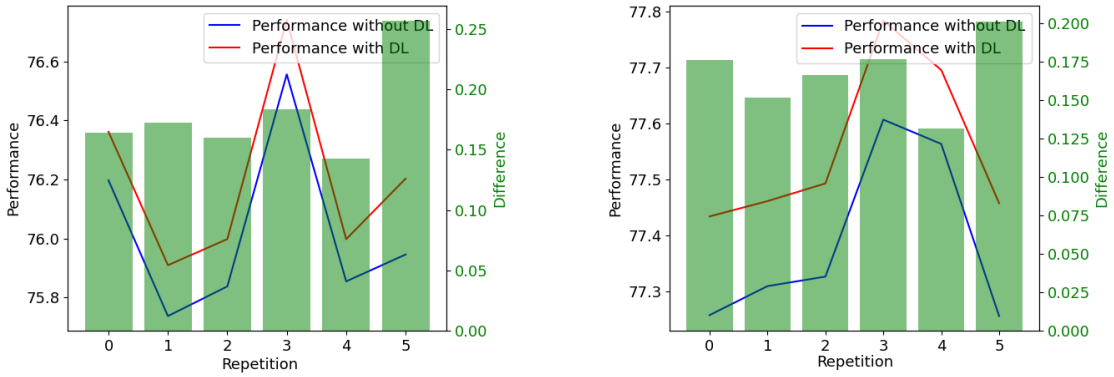
Source: The Author

presented the same level of influence of DL over performance estimates. The two remaining results presented in this section, for SVM and LR experiments, differ among themselves and also from the standard results presented so far.

The first difference in behavior is seen in the experiments for SVM (Figure 5.17). Differently from the previous experiments, the results for SVM show higher levels of influence of DL in F1 scores and balanced Accuracy (Figures 5.17a and 5.17b). The F1 scores show increases of up to 0.35 in models trained with DL in contrast to models trained without DL, while the differences in balanced accuracy varies from around 0.15 to 0.25. This could indicate a slightly higher sensitivity of this algorithm to the phenomenon of DL.

On the other hand, differing from the previous examples, the results for LR (Figure 5.18) show little influence of DL on the performance indicators. The balanced accuracy

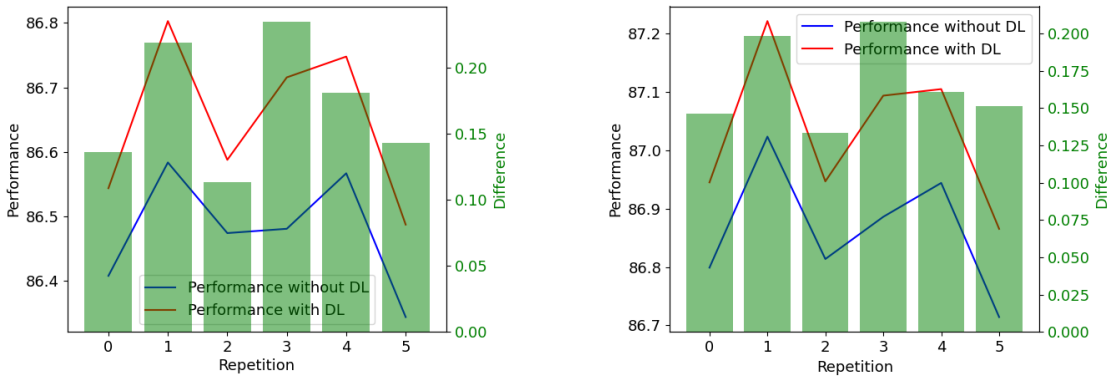
Figure 5.15: Average performance estimates and score differences per repetition for Naïve Bayes.



(a) F1 score (b) Balanced accuracy

Source: The Author

Figure 5.16: Average performance estimates and score differences per repetition for Random Forests.



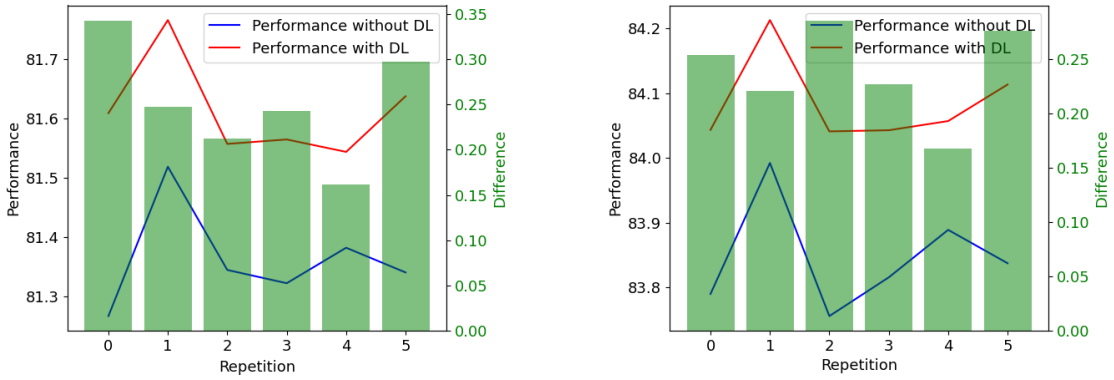
(a) F1 score (b) Balanced accuracy

Source: The Author

results seem to be barely affected by the introduction of DL, with a maximum increase of 0.08 on the average results, as seen in Figure 5.18a. The F1 score also does not seem to be significantly affected by DL, with a maximum score difference close to 0.1 across all repetitions.

Figure 5.19 shows a summary of results per algorithm. It is possible to notice that the results for LR (Figure 5.19a show smaller levels of influence from DL. The KNN, RF, NB, and DT results (Figures 5.19b, 5.19c, 5.19d, and 5.19e) show similar results with slightly higher impacts from DL. Lastly SVM results (Figure 5.19f) show impact form DL with increases higher than the ones seen on the previous group.

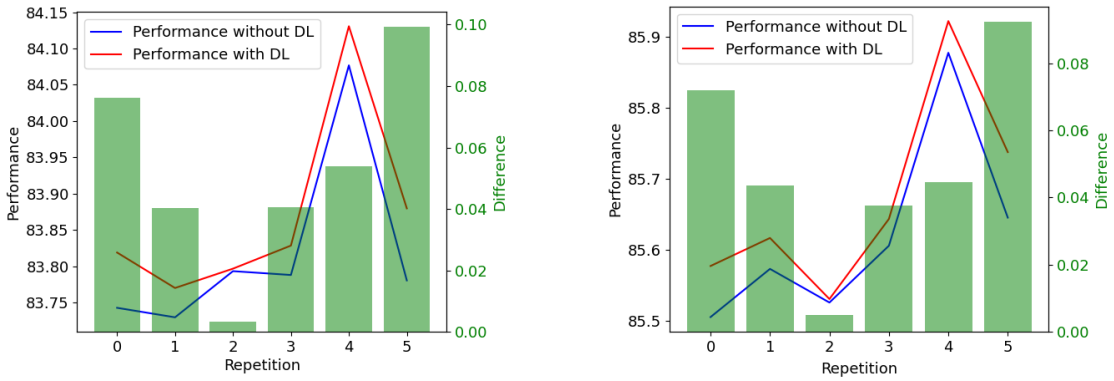
Figure 5.17: Average performance estimates and score differences per repetition for SVM.



(a) F1 score (b) Balanced accuracy

Source: The Author

Figure 5.18: Average performance estimates and score differences per repetition for Logistic Regression.



(a) F1 score (b) Balanced accuracy

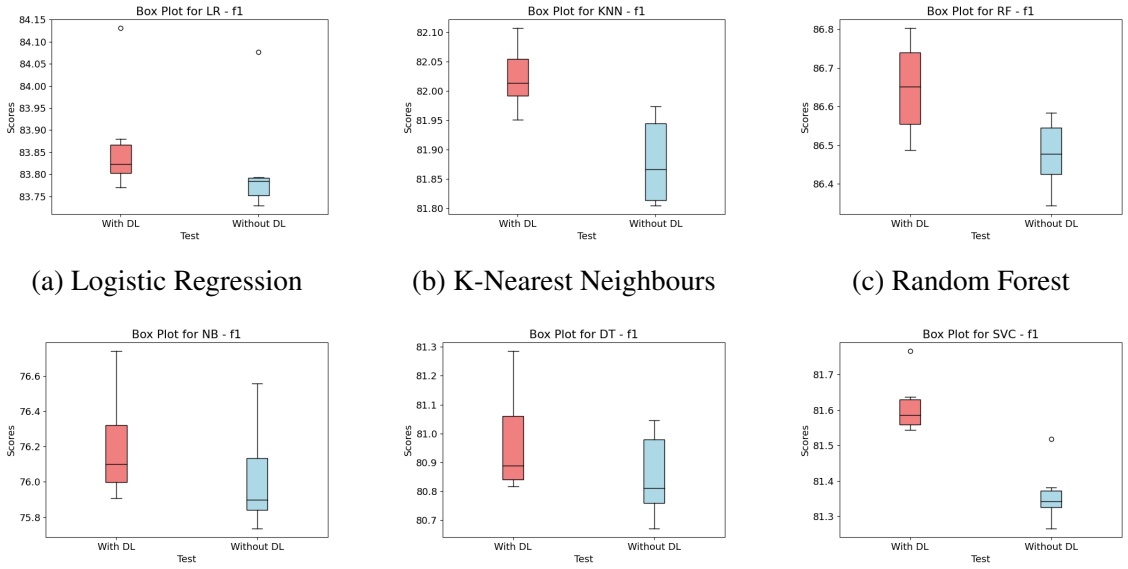
Source: The Author

5.5 Is there a correlation between the metadata of the data sets and the effects of data leakage?

Finally, we aim to investigate the possible correlation between data sets metadata and the effects of DL observed through our experiments. With this in mind, we computed the Kendall Correlation among data sets metadata and the average differences in performance obtained for each task, each algorithm, and considering all results. To evaluate the relevance of the correlation, we adopted the thresholds presented by Wechsler (1997). Therefore, absolute correlation values from 0 to 0.19 are regarded as very weak, from 0.2 to 0.39 as weak, from 0.4 to 0.59 as moderate, from 0.6 to 0.79 as strong, and from 0.8 to 1 as very strong.

Table 5.2 presents the Kendall Correlation results for each task addressed in our

Figure 5.19: Summary of F1-score results per algorithm



(a) Logistic Regression (b) K-Nearest Neighbours (c) Random Forest

(d) Naive Bayes (e) Decision Trees (f) Support Vector Machine

Source: The Author

Table 5.2: Correlation analysis between data sets metadata and performance differences per task.

Task	Inst.	Bin. Feat.	Categ. Feat.	Cont. Feat.	Feats	Classes	Imbalance	Feats/Class	Inst/Class	Feats/Inst.
All	-0.054	0.009	0.022	0.017	0.001	0.0166	-0.007	-0.0004	-0.049	-0.046
Hyperparameter tuning	-0.228	0.105	-0.079	0.054	0.043	0.048	-0.002	0.068	-0.195	-0.199
Value imputation	0.003	0.007	0.012	-0.0004	0.007	0.005	0.001	-0.0007	-0.007	-0.003
Feature selection	-0.057	0.001	0.018	0.093	-0.075	-0.124	-0.052	0.063	0.001	-0.081
Normalization	-0.054	0.009	0.022	0.017	0.001	0.016	-0.007	-0.0004	-0.049	-0.046

Source: The Author

experiments. There is no evidence of significant correlation in this analysis. Most values indicate a very weak correlation, except for the correlation between the results of the hyperparameter tuning and the number of instances in the data set, which shows a weak negative correlation of -0.228. Similarly, the results presented for each algorithm (Table 5.3) show little evidence of correlation. All values indicate a very weak correlation.

Table 5.3: Correlation analysis between data sets metadata and performance differences per algorithm.

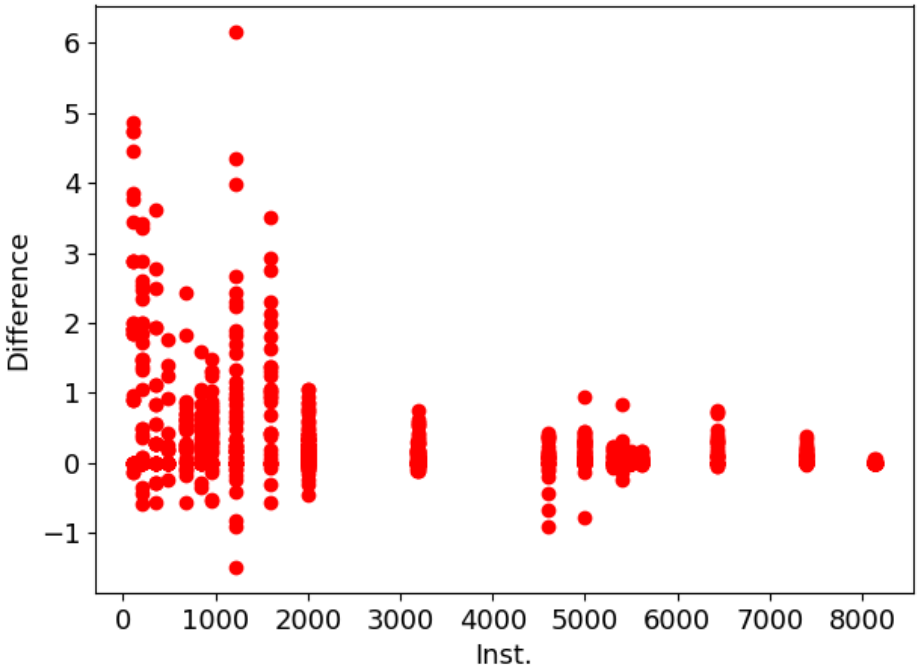
Alg	Inst.	Bin. Feat.	Categ. Feat.	Cont. Feat.	Feats	Classes	Imbalance	Feats/Class	Inst/Class	Feats/Inst.
KNN	-0.073	-0.006	0.066	0.026	-0.022	0.01	-0.009	-0.015	-0.055	-0.044
SVM	-0.041	0.006	-0.002	0.007	0.013	-0.013	-0.025	0.025	-0.035	-0.05
LR	-0.01	-0.007	-0.034	-0.027	0.006	0.023	0.009	-0.026	-0.013	-0.004
NB	-0.051	0.046	0.089	0.098	-0.04	0.055	-0.011	0.025	-0.063	-0.069
DT	-0.049	-0.001	0.014	-0.032	0.027	0.016	-0.012	-0.013	-0.044	-0.032
RF	-0.087	0.024	0.008	0.036	0.01	0.018	0.006	0.003	-0.073	-0.073

Source: The Author

Figure 5.20 shows a scatter plot comparing the distributions of the score differences according to the number of instances in the data set for the experiments with hyperparameter tuning. It is possible to notice a small negative correlation between the

variables, according to what was suggested by our analysis with the Kendall Correlation (Table 5.2). Additionally, there is a pattern of higher variation in the results for a smaller number of instances and smaller variation for a higher number of instances. This behavior corroborate the results reported in the study by Samala et al. (2020), which indicated a higher influence of DL when decreasing the sample size.

Figure 5.20: Distribution of score differences according to number of instances in the data set for hyperparameter tuning experiments.



Source: The Author

6 CONCLUSION & FUTURE WORK

This work proposed a systematic study of the effects of data leakage on the performance estimates of machine learning classifiers. According to our literature review, such evaluation encompassing several important tasks and a good variation in data sets and algorithms was not conducted before. Our experiments were guided by five research questions and allowed us to extract interesting findings, which we summarize next.

Does data leakage impact performance evaluation? The evidence presented in Section 5.1 describes the influence of DL on model performance evaluation. Although the average score differences observed for models with DL in relation to models without DL were small through to the six CV repetitions, the results showed a tendency of optimistic evaluation when accidental data contamination is present. Consequentially, there are reasons to be conscious about avoiding the introduction of DL when designing pipelines to train predictive models.

Is the effect of data leakage more prominent in a specific task? Regarding the sensitivity of different tasks, with results presented in Section 5.2, it is possible to notice distinct effects. It is clear that both feature selection and hyperparameter tuning showed a more significant impact from DL. We observed a tendency for performance overestimation in both tasks when the pipeline is built with accidental data contamination. Even though the average differences presented low values of up to 0.5 of increase, we highlight that this is the overall result for a experimental analysis covering 30 data sets and six algorithms, which may masks the specificities of each scenario. On the other hand, results for missing value imputation and normalization did not suggest a clear impact from DL over performance estimates.

Do data leakage effects change for different data sets? Some of the most interesting results can be seen when analyzing the influence of DL in each data set (Section 5.3). The three data sets categories created show the different ways the selected data sets were affected. Table 5.1 showed that 10 data sets had a significant increase in the performance estimates, 9 had a slight influence from DL to increase the performance evaluation results, and 11 had no consistent influence or even a negative influence from DL. Additionally, results for combinations of data sets and tasks showed the extent to which DL may impact on model performance evaluation. Differences in average performances of up to 8 percentile points were perceived in the results presented in Section 5.3.2.

Is there a machine learning algorithm more sensitive to data leakage effects?

From Section 5.4, the analysis of effects on an algorithm basis provided an interesting perspective. According to the results from the selected data sets, most algorithms had similar results, with KNN, DT, NB, and RF all having a small and very close performance evaluation increases when DL was introduced. The most differently affected algorithms were SVM and LR. SVM had slightly higher performance estimate increases than the other algorithms, while LR had smaller increases.

Is there a correlation between the metadata of the data sets and the consequences of data leakage? Our final analysis concerning the possible correlation between data sets metadata and the effects of DL, as presented in Tables 5.2 and 5.3, showed little to no correlation between these aspects. The largest absolute correlation was observed when analyzing the number of instances in the datasets and the difference in performance of the hyperparameter tuning, for which a correlation of -0.228 was found. We note, however, that the absence of more correlations may be a limitation of the selected data sets and the lack of enough sample variability.

While our conclusions are limited to the 30 selected data sets from the PMLB repository, our findings indicated a tendency of overestimation when accidental data contamination is introduced in the model development pipeline. Therefore, we emphasize the importance of adequately treating and manipulating data to avoid DL when training ML models in order to prevent unreproducible experimental results. For future works, it would be interesting to expand the scope of the analysis by increasing the number and variability of data sets, to conduct a more robust investigation about the influence of metadata in the impact of DL, as well as investigating the effects of DL in regression tasks.

REFERENCES

ALPAYDIN, E. Introduction to machine learning. In: ____. [S.l.: s.n.], 2014.

BERRAR, D. Bayes' theorem and naive bayes classifier. **Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics**, Elsevier Science Publisher Amsterdam, The Netherlands, v. 403, p. 412, 2018.

BREIMAN, L. Random forests. **Machine learning**, Springer, v. 45, p. 5–32, 2001.

BUSSOLA, N. et al. Ai slipping on tiles: Data leakage in digital pathology. In: SPRINGER. **Pattern Recognition. ICPR International Workshops and Challenges: Virtual Event, January 10–15, 2021, Proceedings, Part I**. [S.l.], 2021. p. 167–182.

CAWLEY, G. C.; TALBOT, N. L. C. On over-fitting in model selection and subsequent selection bias in performance evaluation. **J. Mach. Learn. Res.**, v. 11, p. 2079–2107, 2010. Available from Internet: <https://dl.acm.org/doi/10.5555/1756006.1859921>.

CUNNINGHAM, P.; CORD, M.; DELANY, S. J. Supervised learning. In: ____. **Machine Learning Techniques for Multimedia: Case Studies on Organization and Retrieval**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 21–49. ISBN 978-3-540-75171-7. Available from Internet: <https://doi.org/10.1007/978-3-540-75171-7_2>.

CUTLER, A.; CUTLER, D. R.; STEVENS, J. R. Random forests. In: ____. **Ensemble Machine Learning: Methods and Applications**. Boston, MA: Springer US, 2012. p. 157–175. ISBN 978-1-4419-9326-7. Available from Internet: <https://doi.org/10.1007/978-1-4419-9326-7_5>.

DIXON, M. F.; HALPERIN, I.; BILOKON, P. **Machine learning in finance**. [S.l.]: Springer, 2020.

DROBNJAKOVIĆ, F.; SUBOTIĆ, P.; URBAN, C. Abstract interpretation-based data leakage static analysis. **arXiv preprint arXiv:2211.16073**, 2022.

FARROW, E.; MOORE, J.; GAsEVIć, D. Analysing discussion forum data: A replication study avoiding data contamination. In: **Proceedings of the 9th International Conference on Learning Analytics & Knowledge**. New York, NY, USA: Association for Computing Machinery, 2019. (LAK19), p. 170–179. ISBN 9781450362566. Available from Internet: <https://doi.org/10.1145/3303772.3303779>.

GARCÍA, S.; LUENGO, J.; HERRERA, F. **Data preprocessing in data mining**. [S.l.]: Springer, 2015.

GREENE, D.; CUNNINGHAM, P.; MAYER, R. Unsupervised learning and clustering. In: ____. **Machine Learning Techniques for Multimedia: Case Studies on Organization and Retrieval**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 51–90. ISBN 978-3-540-75171-7. Available from Internet: <https://doi.org/10.1007/978-3-540-75171-7_3>.

HASTIE, T. et al. **The elements of statistical learning: data mining, inference, and prediction**. [S.l.]: Springer, 2009.

JURAFSKY, D. et al. **Speech and Language Processing**. Pearson Education, 2014. ISBN 9780133252934. Available from Internet: <https://books.google.com.br/books?id=Cq2gBwAAQBAJ>.

KANG, S. k-nearest neighbor learning with graph neural networks. **Mathematics**, v. 9, n. 8, 2021. ISSN 2227-7390. Available from Internet: <https://www.mdpi.com/2227-7390/9/8/830>.

KAPOOR, S.; NARAYANAN, A. Leakage and the reproducibility crisis in ml-based science. **arXiv preprint arXiv:2207.07048**, 2022.

KAUFMAN, S. et al. Leakage in data mining: Formulation, detection, and avoidance. **ACM Transactions on Knowledge Discovery from Data (TKDD)**, ACM New York, NY, USA, v. 6, n. 4, p. 1–21, 2012.

KINGSFORD, C.; SALZBERG, S. L. What are decision trees? **Nature biotechnology**, Nature Publishing Group US New York, v. 26, n. 9, p. 1011–1013, 2008.

KRAMER, O. K-nearest neighbors. In: ____. **Dimensionality Reduction with Unsupervised Nearest Neighbors**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 13–23. ISBN 978-3-642-38652-7. Available from Internet: <https://doi.org/10.1007/978-3-642-38652-7_2>.

KUNCHEVA, L. I.; RODRíGUEZ, J. J. On feature selection protocols for very low-sample-size data. **Pattern Recognition**, v. 81, p. 660–673, 2018. ISSN 0031-3203. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S003132031830102X>.

LUAN, H.; TSAI, C.-C. A review of using machine learning approaches for precision education. **Educational Technology & Society**, JSTOR, v. 24, n. 1, p. 250–266, 2021.

MAYER, I. et al. R-miss-tastic: a unified platform for missing values methods and workflows. **arXiv preprint arXiv:1908.04822**, 2019.

NAQA, I. E.; MURPHY, M. J. What is machine learning? In: ____. **Machine Learning in Radiation Oncology: Theory and Applications**. Cham: Springer International Publishing, 2015. p. 3–11. ISBN 978-3-319-18305-3. Available from Internet: <https://doi.org/10.1007/978-3-319-18305-3_1>.

NISBET, R.; ELDER, J.; MINER, G. D. **Handbook of statistical analysis and data mining applications**. [S.l.]: Academic press, 2009.

NOBLE, W. S. What is a support vector machine? **Nature biotechnology**, Nature Publishing Group UK London, v. 24, n. 12, p. 1565–1567, 2006.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

RASCHKA, S. Model evaluation, model selection, and algorithm selection in machine learning. **CoRR**, abs/1811.12808, 2018. Available from Internet: <http://arxiv.org/abs/1811.12808>.

RINGNÉR, M. What is principal component analysis? **Nature biotechnology**, Nature Publishing Group US New York, v. 26, n. 3, p. 303–304, 2008.

ROKACH, L.; MAIMON, O. Decision trees. In: ____. **Data Mining and Knowledge Discovery Handbook**. Boston, MA: Springer US, 2005. p. 165–192. ISBN 978-0-387-25465-4. Available from Internet: <https://doi.org/10.1007/0-387-25465-X_9>.

ROMANO, J. D. et al. Pmlb v1.0: an open source dataset collection for benchmarking machine learning methods. **arXiv preprint arXiv:2012.00058v2**, 2021.

SAMALA, R. K. et al. Hazards of data leakage in machine learning: a study on classification of breast cancer using deep neural networks. In: SPIE. **Medical Imaging 2020: Computer-Aided Diagnosis**. [S.l.], 2020. v. 11314, p. 279–284.

SHIM, M.; LEE, S.-H.; HWANG, H.-J. Inflated prediction accuracy of neuropsychiatric biomarkers caused by data leakage in feature selection. **Scientific Reports**, Nature Publishing Group UK London, v. 11, n. 1, p. 7980, 2021.

SUTHAHARAN, S.; SUTHAHARAN, S. Support vector machine. **Machine learning models and algorithms for big data classification: thinking with examples for effective learning**, Springer, p. 207–235, 2016.

TAN, M. S. et al. A review on omics-based biomarkers discovery for alzheimer's disease from the bioinformatics perspectives: statistical approach vs machine learning approach. **Computers in biology and medicine**, Elsevier, v. 139, p. 104947, 2021.

ULLAH, Z. et al. Applications of artificial intelligence and machine learning in smart cities. **Computer Communications**, Elsevier, v. 154, p. 313–323, 2020.

VABALAS, A. et al. Machine learning algorithm validation with a limited sample size. **PloS one**, Public Library of Science San Francisco, CA USA, v. 14, n. 11, p. e0224365, 2019.

VARMA, S.; SIMON, R. Bias in error estimation when using cross-validation for model selection. **BMC bioinformatics**, BioMed Central, v. 7, n. 1, p. 1–8, 2006.

WECHSLER, S. Statistics at square one. ninth edition, revised by m. j. campbell, t. d. v. swinscow, bmj publ. group, london, 1996. no. of pages: 140. price: £11. isbn 0-7279-0916-9. **Statistics in Medicine**, 1997. Available from Internet: <https://onlinelibrary.wiley.com/doi/abs/10.1002/%28SICI%291097-0258%2819971130%2916%3A22%3C2629%3A%3AAID-SIM698%3E3.0.CO%3B2-Z>.

YAGIS, E. et al. Effect of data leakage in brain mri classification using 2d convolutional neural networks. **Scientific reports**, Nature Publishing Group UK London, v. 11, n. 1, p. 22544, 2021.

YANG, C. et al. Data leakage in notebooks: Static detection and better processes. In: **Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering**. [S.l.: s.n.], 2022. p. 1–12.

YOZGATLIGIL, C. et al. Comparison of missing value imputation methods in time series: the case of turkish meteorological data. **Theoretical and applied climatology**, Springer, v. 112, p. 143–167, 2013.

ZHANG, H.; LI, D. Naïve bayes text classifier. In: IEEE. **2007 IEEE International Conference on Granular Computing (GRC 2007)**. [S.l.], 2007. p. 708–708.

APPENDIX A — LIST OF HYPERPARAMETERS TESTED

A.1 Logistic Regression

- Solver: ['saga', 'lbfgs']
- C: [0.01, 0.1, 1, 10, 100]

A.2 Support Vector Classifier

- C: [0.01, 0.1, 1, 10, 100]
- Solver: ['linear', 'rbf', 'poly']/

A.3 K-Nearest Neighbors

- N Neighbors: [3, 5, 7, 9, 11]
- Metric: ['euclidean', 'manhattan']

A.4 Random Forest

- N Estimators: [100, 200, 500]
- Max Depth: [3, 5, 8, 10, None]

A.5 Decision Tree

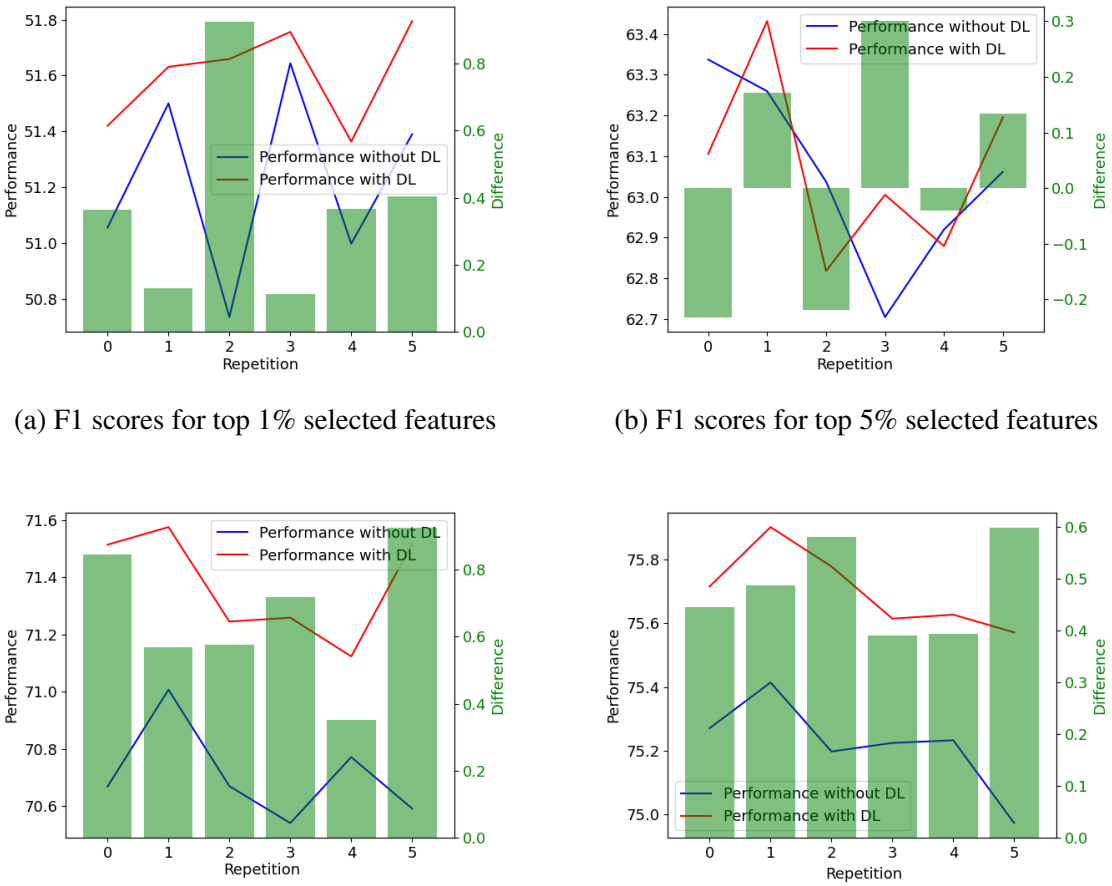
- Max Depth: [3, 5, 8, 10, None]

A.6 Naive Bayes

- Var Smoothing: [1e-9, 1e-8, 1e-7, 1e-6, 1e-5]

APPENDIX B — DETAILED RESULTS FOR FEATURE SELECTION

Figure B.1: Detailed F1-score results for feature selection considering different percentiles of top-selected features.



(a) F1 scores for top 1% selected features

(b) F1 scores for top 5% selected features

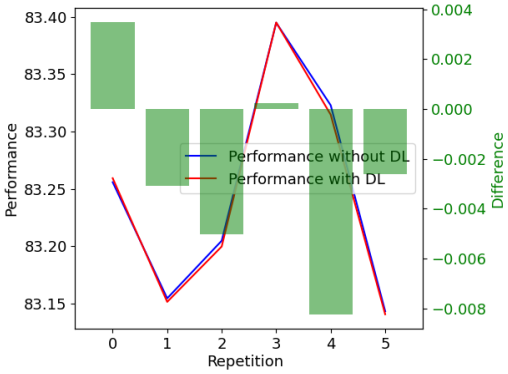
(c) F1 scores for top 10% selected features

(d) F1 Scores for top 20% selected features

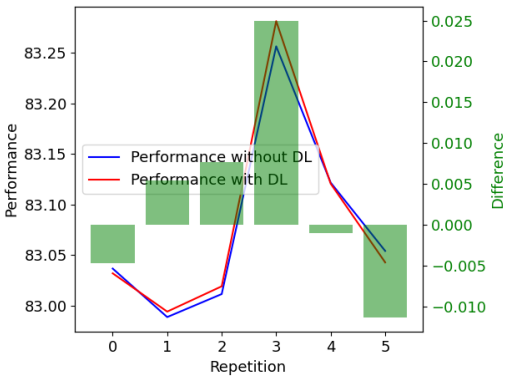
Source: The Author

APPENDIX C — DETAILED RESULTS FOR VALUE IMPUTATION

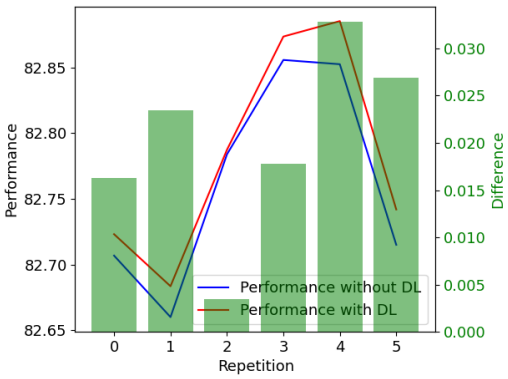
Figure C.1: Detailed F1-score results for value imputation considering different percentage of missing values.



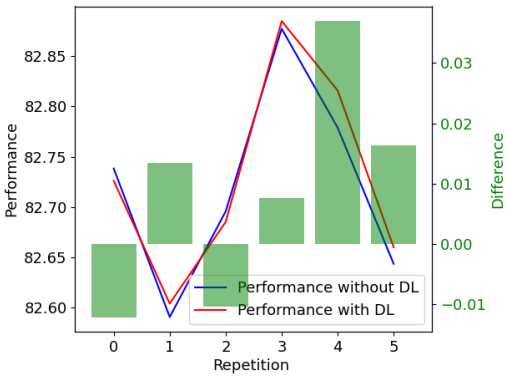
(a) F1 Scores for 5% missing values



(b) F1 Scores for 10% missing values



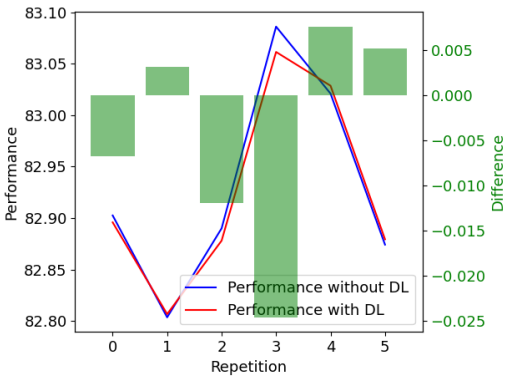
(c) F1 Scores for 20% missing values



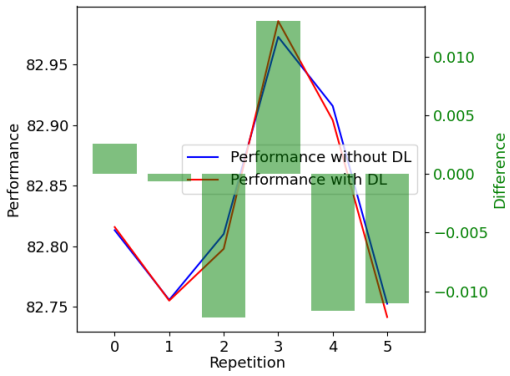
(d) F1 Scores for 30% missing values

Source: The Author

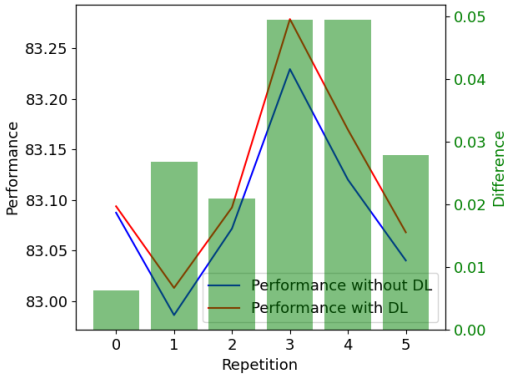
Figure C.2: Detailed F1-score results for value imputation considering different imputation strategies.



(a) F1 Scores for mean imputer



(b) F1 Scores for median imputer



(c) F1 Scores for KNN imputer

Source: The Author