UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Universitat Politècnica de València
Department of Computer Systems and Computing
Master's Degree in Software Systems Engineering and Technology

Analysis of different approaches to determine the importance of attributes in learning models

MASTER THESIS

AYA ALLAH ALI ABDELHAMED ELSAYED

Academic Supervisor

Prof. María José Ramírez Quintana

Academic Course 2019-2020

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Universitat Politècnica de València

Universitat Politècnica de València
Department of Computer Systems and Computing
Master's Degree in Software Systems Engineering and Technology

Analysis of different approaches to determine the importance of attributes in learning models

MASTER THESIS

AYA ALLAH ALI ABDELHAMED ELSAYED

Academic Supervisor

Prof. María José Ramírez Quintana

(Signature)	*(Signature)*	*(Signature)*
Prof. María José Ramírez Quintana	.....	.....

Academic Course 2019-2020

Abstract

This master thesis tries to work with black-box models assuming that the original data is not available with the aim to determine feature importance. So, the idea is to generate a dataset (by using normal distributions for randomly generating the values of the attributes), to label the dataset using the model (that acts as an oracle) and then to learn different models and extract features importance from them. Our approach is based on the following rational. Two models are similar in behavior then both of them may agree in feature importance. We use the kappa measure to identify which one of the surrogate models is the most similar. We prove our hypothesis experimentally over a large collection of datasets and using different learning techniques to create the surrogate models.

Keywords

Feature Importance, machine learning families, black-box model.

Abstracta

Esta tesis de maestría intenta trabajar con modelos de caja negra suponiendo que los datos originales no son disponible con el objetivo de determinar la importancia de la característica. Entonces, la idea es generar un conjunto de datos (mediante el uso de distribuciones normales para generar aleatoriamente los valores de los atributos), para etiquetar el conjunto de datos utilizando el modelo (que actúa como un oráculo) y luego aprender diferentes modelos y extraer presenta importancia de ellos. Nuestro enfoque se basa en lo siguiente racional. Dos modelos son similares en comportamiento, entonces ambos pueden estar de acuerdo en la importancia de la característica. Usamos el kappa mida para identificar cuál de los modelos sustitutos es el más similar. Probamos nuestra hipótesis experimentalmente a través de una gran colección de conjuntos de datos y utilizando diferentes técnicas de aprendizaje para crear Los modelos sustitutos.

Keywords

Importancia de la característica, familias de aprendizaje automático, modelo de caja negra.

Abstracta

Aquesta tesi de Master intenta treballar amb models de caixa negra que assumeixen que la dada original no és disponible. Todo això amb l'objectiu de determinar importància de característica. La idea és generar un dataset per etiquetar el dataset usant el model (que actua com un oracle) i distribucions normals per a l'atzar generant els valors dels atributs. i aleshores aprendre models diferents i extraure d'ells les seues propietats i característiques. El nostre apropament està basat en erl seguent rational: Dos models són similars en comportament llavors tots dos poden acordar la importància d'una característica. Ací usem la mesura kappa per a identificar quin dels models subrogats són el més similar i aportem evidència experimental de la nostra hipòtesi sobre una gran col·lecció de dades usant tècniques de aprenentatge diferents per crear els models subrogats.

To my Family

Acknowledgements

First of all I would like to thank my tutors in this thesis project *María José Ramírezfor* sharing her ideas and knowledge in the area, which allowed this research work to be carried out successfully.
To my family who always supported me from a distance, especially my mother and father who were always aware of me and ready to encourage me to follow my dreams.
To my husband and my kids who ware With me every second of this adventure.

Valencia, 2019

Aya Allah Ali Abdelhamed Elsayed

Contents

Chapter 1

Introduction

This chapter presents an introduction to the research problem that it will be discussed in the rest of the chapters, the objectives pursued and the factors that motivated the completion of this master's thesis

1.1 Motivation

Because of new computing technologies, machine learning (ML) today is not like machine learning of the past. It was born from pattern recognition and the theory that computers can learn without being programmed to perform specific tasks; researchers interested in artificial intelligence wanted to see if computers could learn from data. The iterative aspect of machine learning is important because as models are exposed to new data, they are able to independently adapt. They learn from previous computations to produce reliable, repeatable decisions and results. It's a science that's not new but one that has gained fresh momentum. ML models usually perform really well for predictions but are not always interpretable. There are different reasons why a model is not interpretable. On the one hand, there are ML techniques that generate models that behave as black-box models. That is, we can know the decisions made by the model but we can't know why these decisions were taken. Techniques based on Neural networks are some of these ML techniques. On the other hand, even if the model could be comprehensible (such as a decision tree), ML models may be considered confidential due to their sensitive training data, commercial value, or use in security applications such as spam or fraud detection, malware classification, ... Increasingly often, confidential ML models are being deployed with publicly accessible query interfaces, e.g., FICO/Credit score models, Health/Car/Life Insurance application models, IoT Systems Security...). That means that the user can only interact with the model through queries that are labeled by the model, which acts as an oracle.

There is an extensive literature on the topic of learning from queries. For instance, it has been used to explain an incomprehensible model by mimicking the behavior of a black-box model by obtaining an equivalent one that is expressed as a set of comprehensible rules (Domingos, 1998; Blanco-Vega et al., 2004; Ferri et al., 2002). The idea is to generate input data and query the model to label those data. The result is used as the training set to build a new comprehensible model (the surrogate, or mimetic, model) that captures the behaviour of the original model (the oracle). More recently, [1] has used surrogate models to determine the machine learning technique family that was used for training a model that is presented as a black-box model.

A different way of analyzing machine learning models is to study how input attributes affect output values (especially in supervised learning). The most classical insight is to determine those attributes that are the most relevant and derive feature importance from the model. Knowing which attributes are crucial for some predictions is key to understand how the model works, its attribute costs, and also its robustness to some of the attributes.

Different methods have been introduced to determine feature importance. Most of them make use of the original training dataset and/or the model. Generally, importance provides a score that indicates how useful or valuable each feature of the training data was in the construction of the model. The more an attribute is used to make key decisions with the model, the higher its relative importance. This importance is calculated explicitly for each attribute in the dataset, allowing attributes to be ranked and compared to each other.

With the growing interest in extracting information from no interpretable models, several proposals have arose to determine feature importance from black-box models. Some of them are specific for models built with a certain ML technique such as neural networks (Olden et a. 2004). A more general approach is presented in (Fisher et al. 2018). In this paper, the authors proposed a general method able to determine feature importance for any black-box model, whatever the ML technique is applied to learn the model. However, the method needs the training dataset to compute attribute importance.

This research project seeks to extract feature importance form a black-box model assuming that neither the original data nor the model are available. To our knowledge, there have not been proposed methods whatever the ML technique is used to learn the model that are not based on models nor training data.

1.2 Objectives

The main objective of this work is to determine feature importance for black-box models. Given that our starting point is that we can't use the original training data to this end, the idea is to construct models that capture or imitate the behavior of the original model (making use of surrogate models trained with an invented dataset labeled by the original black-box model, the oracle) and use those models to forecast feature importance.

The hypothesis is that if there are models that are similar in behavior then all of them may agree in feature importance. Based on this rational, we propose to identify which surrogate model SM is the most similar in behavior to the oracle model O, and then assume that feature importance for O is the same than for SM.

1.2.1 Specific Objectives

The general objective is decomposed in the following specific objectives:

1. Review of surrogate models' construction.

2. Review of methods for extracting feature importance from black-box models.

3. Present our approach to determine feature importance.

4. Perform some experiments using large collection datasets and applying different ML techniques to generate the surrogate models in order to evaluate our approach.

1.3 Document Structure

This document is structured as 5 chapters. Chapter 2 introduces preliminary concepts of machine learning, Supervised techniques for classification tasks, Black-box model, feature importance and the R language used in this work for the experiments and the description of the implementation in R of the method proposed in the literature to measure feature importance of black-box models.
Chapter 3 summaries surrogate models construction and how it can be used to approximate the behavior of a black-box models.
Chapter 4 defines our approach and present the experimental evaluation conducted by this work.
Finally Chapter 5 establishes the conclusions obtained from this research work and relates new research lines as future works.

Chapter 2

Preliminary Concepts

In this chapter we will review basic concepts related to Machine Learning, Supervised techniques for classification tasks, black-box model and feature importance. We also introduce the programming language we will use for the implementation of our approach and its experimental evaluation.

2.1 Machine Learning ML

Machine learning (ML) is a sub field of artificial intelligence and is a field within computer science. **What Is Machine Learning? ML** is a set of methods that computers use to make and improve predictions or behaviors based on data. It is concerned with enabling computer programs automatically to learn from data and improve their performance at some tasks through experience. According to [2] **ML** is depicted in Fig.2.1

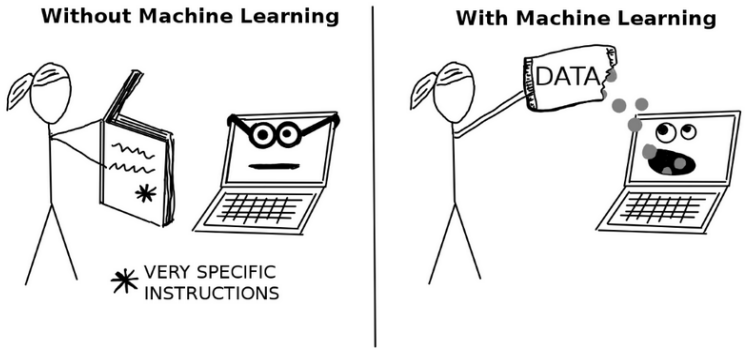

Figure 2.1: *Definition of machine learning .*

The main goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people. **ML** technique can be categorized into three different groups which are

1. Supervised Machine Learning
2. Unsupervised Machine Learning
3. Semi-Supervised Machine Learning

Supervised Machine Learning

Supervised Machine Learning aims to infer the function that maps inputs into outputs. Formally, given a dataset $D = (x, y)$, where x is a tuple of input attributes (also called features) $x = x_1, x_2, \ldots . x_n$, and y is the output attribute, the objective is to learn the mapping function from the input to the output as in Equation 2.1.

$$y = f(x) \tag{2.1}$$

The different supervised machine learning techniques approximate the real mapping function f by using D. For this reason, the learned function is usually denoted as $\hat{f}$ and is called the model. For inferring the model, it is needed that all the data in D be labeled with the output value (this is the reason why this collection of techniques are named supervised). Once the model has been trained, we can apply it to new data in order to predict the output value for such data.

Supervised learning can be applied to solve regression and classification tasks. **Classification**: A classification task is when the output variable is a category, such as "red" or "blue" or "disease" and "no disease".

Regression: A regression task is when the output variable is a real value, such as "dollars" or "weight".

Unsupervised Machine Learning

Unsupervised Machine Learning comprises methods and techniques for describing the data. All data is unlabeled, and the algorithms learn the inherent structure from the input data. The goal of unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data. It is grouped into clustering and association problems.

Clustering: A clustering problem is where you want to discover the inherent groups in the data, such as grouping customers by purchasing behavior.

Association: An association rule learning problem is where you want to discover rules that describe large portions of your data.

Semi-Supervised Machine Learning

In **Semi-Supervised Machine Learning** some data is labeled but most of it is unlabeled and a mixture of supervised and unsupervised techniques can be used. A good example is a photo archive where only some of the images are labeled, (e.g. dog, cat, person) and most of them are unlabeled [3] (Jason Brownlee, Machine Learning Algorithms).

2.2 Supervised Techniques for Classification Tasks

We briefly summarize the characteristics of the main families of supervised techniques we are going to use in this research.

2.2.1 Decision Trees (DT)

A basic decision tree partitions the training data into homogeneous subgroups (i.e., groups with similar response values) and then fits a simple constant (the class label) in each subgroup. The subgroups (also called nodes) are formed recursively using simple yes-or-no questions about each feature (e.g., is age < 18?). This is done a number of times until a suitable stopping criterion is satisfied. Variants of this (basic algorithm have been proposed to allow non-binary partitions and conditions on numerical attributes.

2.2.2 Ensembles (EN)

Ensemble methods are learning algorithms that construct a set of classifiers and then classify new data points by taking a (weighted) vote of their predictions. The set of classifiers can be constructed by acting on the training data (in order to use different sets for training each component of the ensemble) or using different techniques to train each classifier.

2.2.3 Neural Network (NNET)

Neural networks is a computational learning system that uses a network of neurons interconnected among them and organized in layers.There is an input layer, an output layer, and several hidden layers, NNET estimates the weight of the connections using the training data.

2.2.4 Naive Bayes (NB)

It is a method that assume attribute conditional independence in the training dataset. The assumption of conditional independence states that, given random variables X, Y and Z, we say X is conditionally independent of Y given Z, if and only if the probability distribution governing X is independent of the value of Y given Z. Naive Bayes assigns as class of an instance the label that maximizes the posterior probability, that is, the product of the class probability and the probabilities of each attribute given the class.

2.2.5 Nearest Neighbors (NN)

The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. It captures the idea of similarity by calculating the distance between points. To classify a new instance e the k items more close to e are selected and its class labels are used to determine the class of instance e.

2.2.6 Generalized Linear Models (GLM)

It is a flexible generalization of ordinary linear regression that allows for response variables that have error distribution models other than a normal distribution. GLM generalizes lineal regression by allowing the lineal model to be related to the response variable via link function.

2.2.7 Partial Least Squares and Regression (PLSR)

In its simplest form,a linear model specifies the (linear) relationship between a dependent (response) variable Y and a set of predictor variables, the X's. PLSR finds a linear regression model by measuring the input variables X and the output variable Y to a new domain space. In PLSR, prediction functions are represented by factors extracted from the Y'XX'Y matrix.The number of such prediction functions that can be extracted typically will exceed the maximum of the number of Y and X variables.

2.2.8 Logistic and Multinominal Regression (LMR)

It measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function, which is the cumulative logistic distribution.

2.2.9 Discriminant Analysis (DA)

Discriminant analysis is a way to build classifiers: that is, the algorithm uses labeled training data to build a predictive model of group membership which can then be applied to new cases. It makes the assumptions that the variables are distributed normally, and that the within-group covariance matrices are equal.

2.2.10 Multivariate Adaptive Regression Splines(MARS)

The MARS algorithm is an extension of linear models that makes no assumptions about the relationship between the response variable and the predictor variables.

2.3 Black-box Model

A **Black -Box Model** is a model that does not reveal its internal mechanisms. In machine learning, "black-box" describes models that cannot be understood by looking at their parameters or internal structure (chapter 10 in [2]). As depicted in fig.2.2

Figure 2.2: *Working with black-box model.*

ML algorithms such as neural networks, ensembles or support vector machines (SVM) are often considered to produce **black-box models** because they do not provide any direct explanation

for their predictions. However, these methods often outperform simple linear models or decision trees in predictive performance as they can model complex relationships in the data. Nevertheless, such simple models are still preferred in areas such as life sciences and social sciences due to their simplicity and interpretability [4]. Many researchers have therefore developed and implemented several model-agnostic interpretability tools, which quantify or visualize feature effects or feature importance [5], [6], chapter 5 in [2], [7].

2.4 Feature Importance

Here in this section we are going to talk about the feature importance which is the thesis's objective. We are going to know what feature is and why knowing its importance is interesting.

Features are the term used for denoting the inputs used for predicting the output. Thus, a feature is a column in the dataset. The interpretability of the features is a big assumption. But if it is hard to understand the input features, it is even harder to understand what the model does.

From the definition above we can see that a **feature** is "**important**" if shuffling its values increases the model error, because in this case the model relied on the feature for the prediction. A feature is "unimportant" if shuffling its values leaves the model error unchanged, because in this case the model ignored the feature for the prediction. Feature importance tools describe how much input attribute contribute to a prediction model's accuracy. We note that in statistics, a covariate is any characteristic that affect the outcome in a study. Thus, in many feature importance methods,input attributes are denoted covariates.Let us include an example. In Random Forests, **FI** is measured by the decrease in prediction accuracy when a covariate is permuted (Breiman, 2001, et al). Strobl et al., 2008; Altmann et al., 2010; Zhu et al., 2015; Gregoruttiet al., 2015; Datta et al., 2016; Gregorutti et al., 2017). A similar "Perturb" FI measure has been used for neural networks, where noise is added to covariates (Recknagel et al., 1997; Yao et al., 1998; Scardi and Harding, 1999; Gevrey et al., 2003). Such tools can be useful for identifying covariates that must be measured with high precision, for improving the transparency of a "black-box" prediction model (see also Rudin, 2019), or for determining what scenarios may cause the model to fail.

Why knowing feature importance is helpful?

By knowing feature importance, we can get a better understanding of the model's logic: you can not only verify it being correct but also work on improving the model by focusing only on the important features and by this way it can be used for feature selection — you can remove features that are not that significant in the sense that a model trained without those features obtains similar or better performance in much shorter training time than using them. In some business cases it makes sense to sacrifice some accuracy for the sake of interpretability. And to do this we need to select the feature and for this reason we have a process called features selection which is the process of selecting a subset of relevant feature for use in model construction.

Feature selection methods can be used to identify and remove unneeded, irrelevant and redundant features from data that do not contribute to the accuracy of a predictive model or may in fact decrease the accuracy of the model.

The objective of feature selection is three-fold: improving the prediction performance of the predictors, providing faster and more cost-effective predictors, and providing a better understanding of the underlying process that generated the data.

2.5 R Language

R is a programming language and free software environment for statistical computing and graphics supported by the R Foundation for Statistical Computing. The R language is widely used among statisticians and data miners for developing statistical software and data analysis. Polls, data mining surveys, and studies of scholarly literature databases show substantial increases in popularity; as of July 2019, R ranks 20th in the TIOBE index, a measure of popularity of programming languages.

A GNU package, source code for the R software environment is written primarily in C, Fortran and R itself, and is freely available under the GNU General Public License. Pre-compiled binary versions are provided for various operating systems. Although R has a command line interface, there are several graphical user interfaces, such as RStudio, an integrated development environment.

2.5.1 Packages of R

The capabilities of the R language are extended through user-created packages, which allow specialized statistical techniques, graphical devices, import/export capabilities, reporting tools (Rmarkdown, knitr, Sweave), etc.

These packages are developed primarily in R, and sometimes in Java, C, C++, and Fortran. The R packaging system is also used by researchers to create compendia to organize research data, code and report files in a systematic way for sharing and public archiving.

A core set of packages is included with the installation of R, with more than 15,000 additional packages (as of September 2018) available at the Comprehensive R Archive Network (CRAN) Bioconductor, Omegahat GitHub, and other repositories.

In this master thesis we use the **caret** library to learn the ML models both the oracle and the surrogate models. The **caret** package (short for Classification and Regression Training) contains functions to streamline the model training process for complex regression and classification problems. The package utilizes a number of R packages but tries not to load them all at package start-up (by removing formal package dependencies, the package startup time can be greatly decreased). The package "suggests" field includes 30 packages. **Caret** loads packages as needed and assumes that they are installed. If a modeling package is missing, there is a prompt to install it. And we use the **iml**(Interpretable Machine Learning) library to extract the feature importance. The iml package is probably the most robust ML interpretability package available.

The advantages of iml:

1. It provides both global and local model-agnostic interpretation methods. It can be used for any supervised ML model (many feature are only relevant to regression and binary classification problems) which is specifically related to our work.

2. feature importance: uses a permutation-based approach for feature importance, which is model agnostic, and accepts any loss function to assess importance.

2.6 How To measure Feature Importance: using iml package

We use the iml package to extract the feature importance. So we create a Predictor object that holds the surrogate model and the data. By using the function $predictor = Predictor$new(rf, data = X, y = Boston$medv$). Then, we can measure how important each feature was for the predictions with the function **FeatureImp** from the package **iml** [5]. Basically, the idea is to generate permutations of the attributes and to measure the performance of the obtained model with respect to a given loss function.

2.6.1 Description

FeatureImp computes feature importance for prediction models. The importance is measured as the factor by which the model's prediction error increases when the feature is shuffled.

2.6.2 Usage

```
imp = FeatureImp$new(predictor, loss)

imp$plot()

imp$resultsprint(imp)
```

2.6.3 Arguments

For FeatureImp $new(): predictor: (Predictor) The object (created with Predictor$new()) holding the machine learning model and the data. loss: ('character (1)' | function)

The loss function. Either the name of a loss (e.g. **"ce" for classification**) or a function. See Details for allowed loss.

2.6.4 Details

To compute the feature importance for a single feature, the model prediction loss (error) is measured before and after shuffling the values of the feature. By shuffling the feature values, the association between the outcome and the feature is destroyed. The larger the increase in prediction error, the more important the feature was. The shuffling is repeated to get more accurate results, since the permutation feature importance tends to be quite unstable.

2.6.5 Fields

original.error: ('numeric (1)') The loss of the model before perturbing features. predictor: (Predictor) The prediction model that was analyzed. results: (data.frame) data.frame with the results of the feature importance computation. One row per feature with the following columns: importance.05 (5 importance (median importance), importance.95 (95 is also visualized as a bar in the plots, the median importance over the repetitions as a point.

2.6.6 Methods

loss (actual, predicted). The loss function. Can also be applied to data: object$loss(actual, predicted). plot () method to plot the feature importance. See plot.FeatureImp

2.6.7 Example

```
imp = FeatureImp$new(mod, loss = "ce")
```

mod is the Predictor object, that holds the model and the data. We put in the loss function **ce** because we are working with classification.

```
plot(imp)
```

we use plot in the beginning to see the order of the feature importance and visualize it. But After that it becomes not very important because we made two methods to know the most feature important.

Chapter 3

Surrogate Models

In this chapter we will go deeper into the field of surrogate models. We are going to talk about:

1. The Definition of the Surrogate Model (**SM**).
2. The Importance of Using (**SM**)
3. The Process of Generating the surrogate Data.
4. The Way to Learn the Surrogates Models.
5. Evaluate SM to Oracle Model (O) According to the kappa.

3.1 The Definition of the Surrogate Model (SM)

As we said in the section of machine learning, ML algorithms are often considered to produce **black-box models**. A surrogate model is a model that tries to imitate the behavior of a black-box model. In order to generate such surrogate model we just use the black-box model as an oracle for labeling a synthetic dataset, generated by following a specific query strategy. This is then used to learn different models (using different ML techniques belonging to different learning families) that capture the oracle behavior.

3.2 The Importance of Using (SM)

The purpose of surrogate models is to approximate the predictions of the underlying model as accurately as possible and to be able to give insights about model characteristics at the same time. And in the field of feature importance a fundamental issue is the ability to fit the machine learning model. T

3.3 The Process of Generating the Surrogate Data

In order to generate surrogate datasets, we can query an oracle O (the black-box model to be attacked) with artificial examples so that O labels them (fig. 3.1 illustrates this) [1].

This allows us to build an artificial dataset labeled by O (what we call the surrogate dataset SD). As this dataset (the surrogate dataset SD) contains the output labels of O, it tends to capture the boundary patterns of O.

The input attributes can be generated following different strategies. For instance, if we have some knowledge about the real distribution of the attributes, we can use this knowledge to generate SD. If no knowledge is available, the simplest way is to assume that all the attributes follow a uniform distribution. Additionally, this strategy provides a good coverage of the whole feature space. Thus, this is the strategy we adopt for the implementation of our approach.

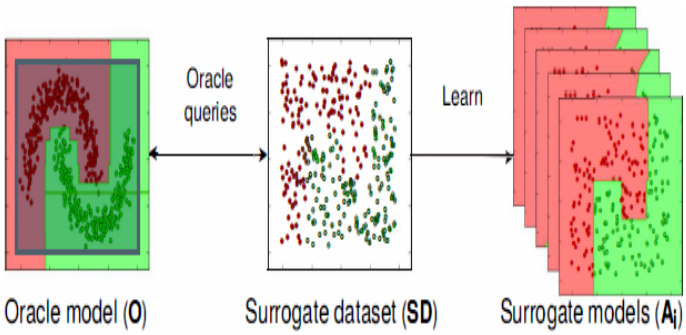

Figure 3.1: *Black-box models (oracles), trained over an unknown original dataset, are used to label synthetic surrogate datasets (generated following specific query strategies), which are then used to train surrogate models*

3.4 The Way to Learn the Surrogates Models

The basic (and effective) strategy that is used [1] consists in generating the artificial examples SD, the surrogate dataset) and learn the surrogate model using SD as training data. These surrogate models, denoted by SM_i, 1 <= i <= N, with N is the number of ML families of techniques we consider in this work. we use a different technique since it might provide a different characterization of SD and, indirectly, a characterization of O.

3.5 Evaluate SM with Respect to the Oracle Model (O)

We use the Cohen's kappa coefficient(k) [8] as a dissimilarity metric to estimate the degree of agreement for qualitative items. In order to measure(evaluate) the agreement between the oracle (that has been used to label SD) and the surrogate model SM_i when explaining the same dataset SD. We can evaluate the surrogate dataset SD (e.g., by using a train-test split) with the different surrogate models SM_i, so that we would obtain a k_i for each SM_i. In the top row known left to right we see the original data and the oracle (a decision tree trained with those data). then we see the surrogate dataset split into train and test. the training is cased to generate surrogate model(shown in the bottom row of the figure) and the test is used to calculate the kappa value.As seen some surrogate models capture very well the decision technique of the oracle(such as ,the decision tree and the NN) as reflect their high kappa value. Fig.3.2 illustrates this procedure [1].

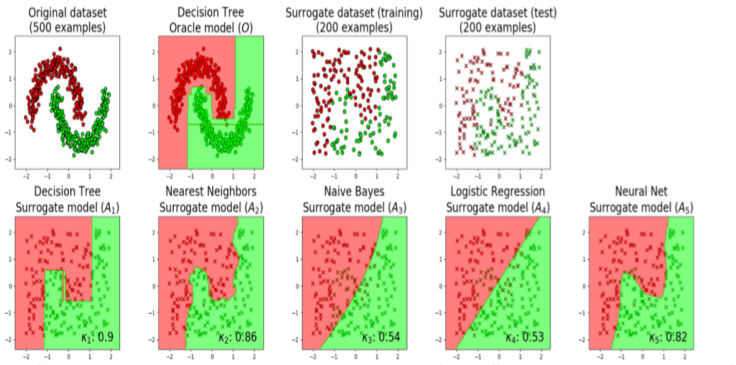

Figure 3.2: *Synthetic example to show the l measure for different surrogate models SM_i when compared to an specific oracle model O as a decision tree technique), which are then used to train surrogate models*

Chapter 4

Our Approach

This chapter explain our approach to extract features importance from the black-box model assuming that neither the original data nor the model are available. Also this chapter introduces our experimental setup and result discussion as we applied our approach over a large collection of datasets and a different set of Machine learning techniques.

4.1 The Hypothesis

Our hypothesis follow the assumption that a surrogate model which has the highest kappa value should have a behavior very similar to the behavior of the original oracle model, thus concluding that its features importance will be similar to the features importance of the original oracle model, as it is depicted in Fig. 4.1.

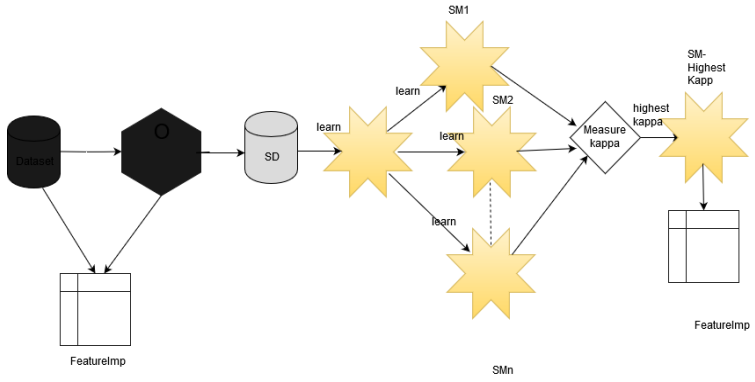

Figure 4.1: *Extract the Feature Importance of the surrogate model that has the highest kappa*

4.2 Measuring the similarity between the oracle model and surrogate model

As it is explained in chapter 3 to measure the similarity between the oracle model and the surrogate model, we use the Cohen's kappa coefficient [9], a dissimilarity metric that estimates the degree of agreement for qualitative items. As said in [1], "Unlike other evaluation measures, the kappa statistic is used not only to evaluate a single classifier, but also to evaluate classifiers between themselves (corrected by chance), and it is generally thought to be a more robust measure

than a simple percentage of the agreement". So we measure the kappa for every surrogate model. We use the test set (labeled by the oracle) to determine the kappa value of the surrogate model and take the surrogate model which has the highest kappa. Then we extract the feature importance from both the original model and the surrogate model that has the highest kappa as it is depicted in fig 4.1. Then we extract the features importance for the surrogate model of highest kappa in order to validate our hypothesis. We will compare the feature importance for the highest kappa surrogate model (hypothesis) with the feature importance of the oracle model.

4.3 Extracting Features Importance

To obtain the feature importance for the surrogate models we proceed as follows. Before extracting the features importance the models need to be prepared in order to apply the function FeatureImp$new() in the iml package as it has been explained in the chapter 2. Once the models have been obtained we needed to create a Predictor object, which holds the model and the data. And to create the predictor object there is a function inside the iml package which is called Predictor$new (). This function take two parameters. The first parameter we need to pass to it is the model that we have and the second is the data from which we want to extract the features importance (the training data).

4.3.1 Measuring Features Importance

Now we can measure how important each feature was for the predictions with the function FeatureImp$new(). This function works by shuffling each feature and measuring how much the performance drops. This function takes two parameters one of them is the object predictor that we just created and the other is the method loss that we assign **ce** as the value for it (because we are working on classification tasks). The result of this function is an object that contains the numerical value per attribute representing its importance. We can call the plot() function of the object or look at the results of the object as it is depicted in fig 4.3 and 4.2, respectively, for the balance-scale dataset and C5.0 oracle. Fig 4.2 shows the object resulting of applying the FeatureImp$new() function. Each row represents the information per feature with the following meaning: Columns importance.05 and importance.95 represent the interval of feature importance values obtained in the importance calculation using different attribute permutations, column importance shows the median importance, and the last column shows the prediction error of the model performing the attribute permutation. The plot (fig 4.3) shows importance.05 and importance.95 as a bar and the point in it is the median importance. We see that the feature are ordered by feature importance being "safety" the most important and "doors" the least important.

4.4 The Hypothesis Validation

In order to validate our hypothesis we use a correlation measure to calculate the relationship between the rank of attributes when we sort them by ordered on features importance obtained for all each one of the surrogate models and the oracle model as depicted in fig. 4.4 this allow us to see if the surrogate model with the highest kappa is also the model with importance attribute

```
> imp$results
   feature importance.05 importance importance.95 permutation.error
1   safety     4.9546392   5.309278      5.412371        0.29803241
2  persons     4.1381443   4.484536      4.552577        0.25173611
3   buying     3.4494845   3.567010      4.010309        0.20023148
4    maint     3.4618557   3.567010      3.758763        0.20023148
5 lug_boot     2.0082474   2.216495      2.408247        0.12442130
6    doors     0.8927835   1.010309      1.028866        0.05671296
>
```

Figure 4.2: *Extract the Feature Importance of the surrogate model that has the highest kappa*

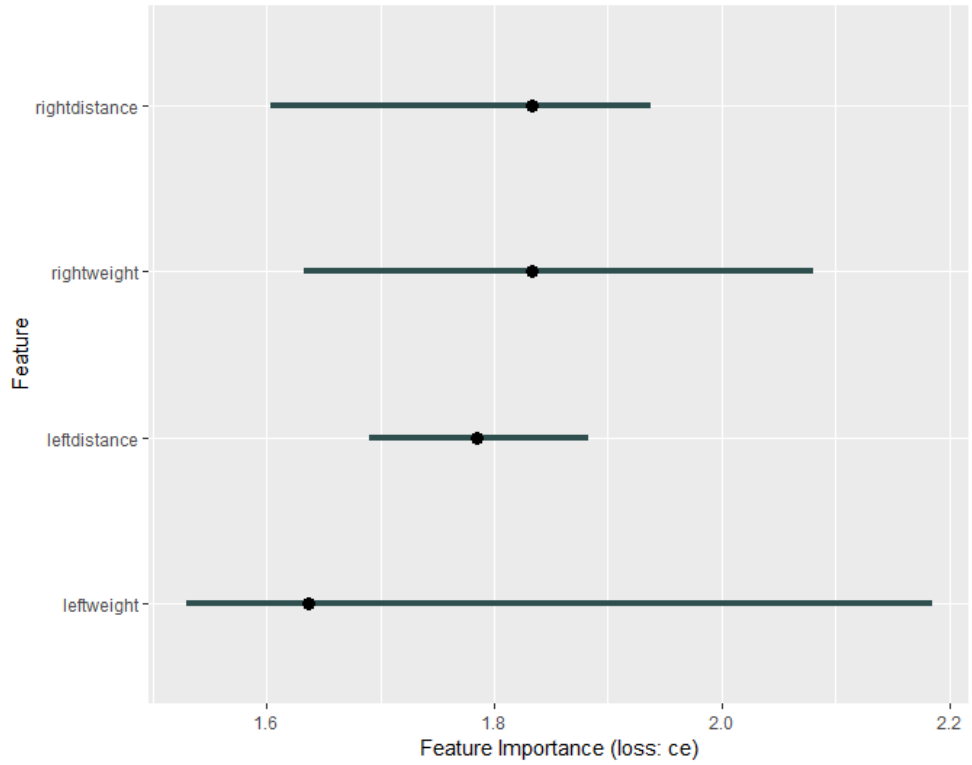

Figure 4.3: *Extract the Feature Importance of an oracle model trained with C5.0 technique for the balance-scale dataset.*

rank more connected to the oracle one. So, we have explored three correlation measures in R, namely spearman, Kendall and Pearson.

Correlation Measures :

- **Pearson** : Use the Pearson product-moment correlation coefficient when you can assume that two correlated data series follow a bivariate normal distribution.
- **Spearman**: Use the Spearman rank-order correlation coefficient when you cannot make a normality assumption about the two data series.
- **Kendall's Concordance**: Use Kendall's coefficient of concordance when you have more than two correlated data series and wish to capture the relationship between them in terms of a single number.

From the definitions above, Pearson and Kendall correlation are not applicable to our approach.

However, Spearman correlation, which is based on rank-order, is more suitable to our problem. Thus, we used Spearman correlation as a measure of similarity between the feature importance of Surrogate models and that of the Oracle model.

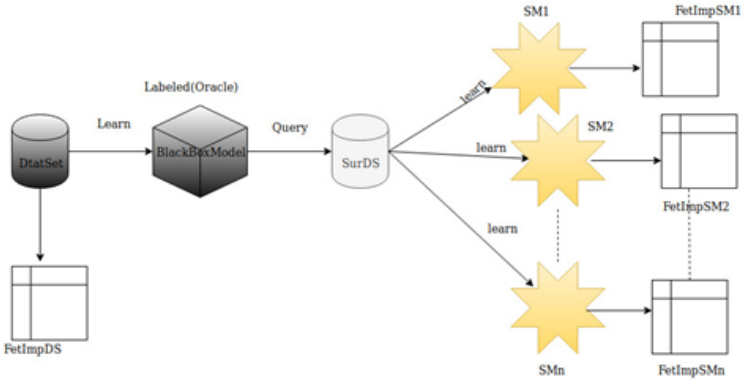

Figure 4.4: *Extract the Features Importance for all surrogate models to measure the correlation.*

After that we selected the surrogate model that has the highest correlation.Then we took the surrogate model with the highest kappa and the surrogate model with the highest correlation to see how many times our hypothesis holds. In the next section, we present the experiments we performed in order to validate the hypothesis.

4.5 Experimental Setting

All the experiments have been performed using R and, in particular, the packages ***Caret*** (to train the different ML models) and ***iml*** (to extract the feature importance). For each dataset $DS_d.\epsilon D, i = i...20$, we trained $N = 10$ different oracle models using the original dataset as training (O_i) belonging to different families as introduced in section 4.5.2. And for each O_i we trained $N = 10$ different surrogate models SM_j of surrogate dataset i.e. in total $|D| \times N = 20 \times 10 = 200$ oracle models and $|D| \times N \times N = 20 \times 10 \times 10 = 2000$ surrogate models, where d is an index over the input datasets and i, j are indices over the set of ML learning techniques.

4.5.1 Datasets

We conduct our experimental evaluation over 20 datasets from the UCI repository [10]. The selected datasets are large enough to generate a dataset of meta-features that can be used to evaluate our approach. Table 4.1 summaries the characterization of the different datasets used in the experiments: number of numerical (num) and discrete (disc) attributes, number of instances (inst), and number of classes (class).

4.5.2 ML learning techniques

In the experiments we selected 10 **ML** families using one technique as representative of each family. More concretely, the list of families and **techniques** used is: Discriminant Analysis (Regularized Discriminant Analysis, **RDA**), Ensembles (Random Forest, **RF**), Decision Trees (**5.0**), Neural Networks (**NNMLP**), Naive Bayes (**NB**), K-Nearest Neighbors (**KNN**), Generalized Linear Models (**GLM**), Partial Least Squares and Regression (**PLSR**), Penalized Logistic

	badges2	banknote-auth	balance-scale	car	cmc	credit-a	credit-g	diabetes	haberman	heart-c	heart-h	iris	labor	monks-1	phoneme	PhishingWebsites	tae	vote	wall-robot-navigation	waveform-500
num	10	4	4	-	2	6	7	8	2	6	6	4	8	-	5	-	3		25	40
disc	-	-	-	6	7	9	13	-	1	7	7	-	8	6	-	30	2	16	-	
inst	295	1372	626	1728	1473	690	1000	768	306	303	294	150	57	575	5405	11056	151	435	5458	5000
class	2	2	3	4	3	2	2	2	2	2	2	3	2	2	2	2	3	2	3	3

Table 4.1: *Description of the datasets used in the experimental of evaluation. We included the number of numerical and discrete of attributes,number of instances (inst), and number of classes (class).*

and Multinomial Regression (**PLMR**), and Multivariate Adaptive Regression Splines (**PLSR**). Table 4.2 shows the acronym and the name of the ML techniques (columns ML Techniques and Model, respectively), the name of the method in caret (column Method), the R package in that it is implemented and the parameters tuned (the rest of the parameters were left to their default values).

ML Techniques	Model	Method	R Package	Parameters
C50	C5.0 Tree-Based model	C5.0	C50	
PLSR	Partial Least Squares Regression	simpls	caret/pls	ncomp=4
NNMLP	Multi-Layer Perceptron	mlpML	caret/RSNNS	
GLM	generalized linear model	glmnet	caret/glmnet	
PLMR	Penalized Multinomial Regression	multinom	caret/nnet	
MARS	Multivariate Adaptive Regression Splines	gcvEarth	caret/earth	degree=3
KNN	k-Nearest Neighbors	knn	caret/knn	k = 5
RF	Random Forest	rf	caret/randomForest	
RDA	Regularized Discriminant Analysis	rda	caret/klaR	
NB	Naive Bayes	naive-bayes	naivebayes	laplace = 3

Table 4.2: *ML techniques used in the experiments implemented in the caret package. We include the name of ML techniques, the models of the ML method, the R package, and the parameters we set for the methods (the rest of parameters were left to their default values.*

4.5.3 Surrogate Models Construction

Surrogate models construction follows a two-steps process. The starting point is the R code of Raul Fabra developed for generating the surrogate data (SD) and learning the surrogate models (SM) [1]. Secondly we added to it the new R functions that implement our approach to approximate features importance of a black-box model. To construct the Surrogate Models, we proceed as follow. For each original dataset, we first generate a collection of artificial examples (the surrogate dataset SD) such that each example is generated at random following a uniform distribution. For each categorical attribute a random value is chosen from its possible values, and for each numerical attribute a random value is generated between its minimum and maximum values. This

generation strategy provides a good coverage of the feature space. Once SD is generated, the examples are labeled by the oracle model O, then we split the SD to train and test. But we didn't split the dataset as normal way to give train 70 percent and the test 30 percent instead of this we generate two sets (training and test) of equal size (calculated for each dataset as 100 multiplied by the number of attributes), using the following piece of code:

```
1
2 GridTrain<-generateUnifTest(Train, 100 * (ncol(datos)-1))
3 GridTest<-generateUnifTest(Train, 100* (ncol(datos)-1))
```

The training set is used to learn the surrogate models. As each surrogate model SM_i belongs to a different model family, then each SM_i might provide a different characterization of SD and, indirectly, a characterization of O (as it is explained in Chapter 3).

4.6 Evaluation

4.6.1 The Ground Truth

We set as the ground truth the real feature importance of the oracle, calculated by applying the iml function to the oracle over the original data (the data used to train the oracle), and extract the features importance rank. In this way we can measure the quality of our hypothesis by comparing the features importance of the surrogate model with the ground truth.

4.6.2 Calculation the Kappa Value for the Surrogate model

We calculate the similarity between the oracle model and the surrogate model by calculating the kappa value by using the function $confusionMatrix$ in the caret package. By passing the test of the surrogate dataset as one of the parameters that the function needs. Then we select the highest kappa by using the function **max()** in R language. And from that we identify the surrogate model that has the highest kappa.

As an example, the Kappa result for the Cmc dataset is shown in table 4.3. The table shows the pairwise kappa values between Oracle models as listed in the rows and Surrogate models as listed in the columns. Table cells are colored according kappa values where the color scales are red, yellow and green where red is the smallest value and green is the maximum value. Observe that depending on the oracle, not always the surrogate model with highest kappa has been trained using the same ML technique than the oracle. This happens, for instance, with the PLSR technique for which the GLM surrogate model obtains the highest kappa.

All the tables with the kappa results for the rest of the datasets are reported in the Appendix A.1.

4.6.3 Measuring the Features Importance Rank Correlation between the Oracle and the Surrogate Models.

We calculate the correlation between the oracle model and the surrogate model. This can be done after the features importance has been calculated by extracting the information from the object created as we mentioned in section 4.3. So, we take these two objects for both O and SM and order them by using the function **order()** in **R**. As shown in the following code:

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.823	0.327	0.335	0.322	0.339	0.598	0.419	0.698	0.481	0.338
PLSR	0.726	0.790	0.833	0.953	0.952	0.809	0.521	0.795	0.836	0.754
NNMLP	0.778	0.878	0.939	0.943	0.939	0.874	0.426	0.830	0.900	0.795
GLM	0.736	0.741	0.551	0.881	0.887	0.786	0.488	0.733	0.783	0.728
PLMR	0.762	0.805	0.504	0.967	0.978	0.804	0.532	0.796	0.834	0.776
MARS	0.868	0.157	0.324	0.281	0.280	0.856	0.301	0.846	0.392	0.330
KNN	0.305	0.157	0.000	0.215	0.220	0.270	0.266	0.318	0.307	0.222
RF	0.563	0.366	0.420	0.411	0.429	0.526	0.429	0.630	0.467	0.419
RDA	0.774	0.821	0.490	0.968	0.973	0.858	0.522	0.799	0.859	0.806
NB	0.524	0.000	0.328	0.502	0.547	0.518	0.279	0.392	0.350	0.337

Table 4.3: *Kappa value for* ***Cmc*** *Dataset. Rows are Oracle models and columns are Surrogate models.*

```
sm_imp$results[order(sm_imp$results$feature),]$importance
# which is the object of the features importance of the surrogate model.
# as one of the parameter of the correlation function.
# And we passed
om_imp$results[order(om_imp$results$feature),]$importance
```

The correlation is calculated using the core() function in R. It takes three parameters: the first and second parameters are the ranks generated for oracle and surrogate model (respectively), and the third parameter is the correlation method that we use **spearman** as it said in the section 4.4. We proceed in the same way for the rest of surrogate models and the oracle model. To illustrate this process, we list in table 4.4 the correlation result for the Cmc dataset. As stated before, table cells are colored according correlation values where the color scales are red, yellow and green where red is the smallest value and green is the maximum value. All correlations results for the rest of datasets are listed in the Appendix A.2.

Then we obtain the highest correlation by using the function **max()** in R language and from that we identify the surrogate model that has the highest correlation. Observe that there are oracles for which more than one surrogate model obtain a high correlation being the differences among them very small. For instance, for the C50 oracle model, the surrogate models GLM and MARS get the maximum correlation value but the correlation obtained by the RDA surrogate model is also quite close.

4.7 Comparing the Highest kappa with the Highest Correlation Obtaining the Global Accuracy.

As we mentioned before to validate our hypothesis, we need to asses if the surrogate model with the highest kappa is the same surrogate model that has the highest rank features importance correlation. To accomplish that, a comparison between **SM** with the highest kappa and **SM** with the highest correlation is conducted for each dataset to find if they match or not. For each dataset and oracle we select the surrogate models with the highest correlation and the surrogate model

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.417	0.450	-0.176	0.450	0.400	0.450	0.150	0.333	0.433	0.250
PLSR	0.904	0.711	0.795	0.921	0.937	0.921	0.569	0.904	0.879	0.937
NNMLP	0.833	0.962	0.917	0.933	0.870	0.695	0.609	0.824	0.862	0.243
GLM	0.745	0.895	0.828	0.929	0.912	0.912	0.845	0.912	0.870	0.803
PLMR	0.750	0.795	0.867	0.845	0.778	0.683	0.661	0.728	0.733	0.733
MARS	0.917	0.067	0.403	0.333	0.233	0.857	0.385	0.883	0.452	0.333
KNN	0.417	0.202	-0.402	0.033	0.433	0.217	0.600	0.217	-0.075	-0.008
RF	0.533	0.283	0.417	0.502	0.433	0.450	0.383	0.600	0.617	0.378
RDA	0.667	0.617	0.343	0.667	0.653	0.517	0.633	0.717	0.617	0.517
NB	0.633	0.176	0.317	0.117	0.233	0.567	0.243	0.550	0.283	0.561

Table 4.4: *Correlation between the rank of the attributes obtained from feature importance for the oracle model and each one of the surrogate models for the Cmc Dataset. Rows are oracle models and Columns are the surrogate models.*

with the highest kappa each time matched it is considered as a success.

For selecting models with the highest Kappa or correlation, we consider a restrict mode of selection where we only select models that has exactly same value of the maximum values. However this ignore small differences between very close models (as indicated in the previous section). For example, in tables 4.3 and 4.4 for dataset *Cmc*, we notice that for oracle *PLSR*, surrogate models with highest Kappa are *GLM* and *PLMR* with values (0.953,0.952) respectively, and surrogate models with highest correlation are *PLMR* and *GLM* with values (0.937,0.921), respectively. Following a strict selection mode, the surrogate model with highest Kappa (*GLM*) is not the same as the surrogate model with highest correlation (*PLMR*). However, in this case with only a difference in values less than 0.01, we can consider both models as SM with the highest Kappa or Correlation. For this reason we apply two selection criteria using a threshold T such that we set $T = 0$ in the strict selection mode (selecting the highest kappa and correlation surrogate models) and $T \leq 0.05$ in the soft selection mode (selecting the surrogate models that are the T-most close to the highest values).

4.7.1 Accuracy

To measure the global accuracy of our hypothesis we average the information of the success per oracle and dataset. To facilitate the accuracy calculation, we have generated two auxiliary tables showing partial results. Thus, table 4.5 includes the highest kappa surrogate model per datasets and oracle models. As described in section 4.7 models selected based on strict selection mode are highlighted in bold font, and models selected with soft selection mode with $T \leq 0.05$ are decorated with an asterisk (*). Also the last row in the table shows (in percentage) the number of times the surrogate model with the highest kappa has been generated using the same ML technique used to generate the oracle.

We observe that is some cases the cell has more than one ML technique indicating that several surrogate models here obtained the same (highest) value. For instance, this is shown in table 4.5.

That is the case for the dataset badge2 and oracle trained with C50 for which the surrogate models trained with the ML techniques C50, PLSR, NNMLP, GLM, MARS, RF, and RDA obtain the same kappa value). Additionally, for the oracles trained with some ML techniques such as PLSR, the surrogate models with highest kappa have been trained with a different ML technique is almost all datasets. All those results are consistent with the results in [1]. On the other hand, Table 4.6 shows the surrogate model with highest feature importance rank correlation per datasets (rows) and oracle(columns). As observed in table 4.6 some cells contains more than one ML technique. Newly, this is because several surrogate models give the same rank of attributes presenting their importance. For the accuracy calculation, we consider that a success occurs when the surrogate model with the highest kappa is contained in the list of the highest correlation surrogate model and vice verse.

Using both tables the calculation of the (global) accuracy can be performed as follows. Let us denote 4.5 and 4.6 as K and C, respectively. For the strict selection K[i,j] and C[i,j] are assumed to be the surrogate models highlighted in bold (in tables 4.5 and 4.6), whereas for the soft selection K[i,j] and C[i,j] are assumed to be the surrogate models highlighted with asterisk. Then the accuracy can be calculated as in Eqs 4.1 and 4.2.

$$accuracy = \frac{1}{N+D}\sum_{i=1}^{|D|}\sum_{j=1}^{N} I(K[i,j], C[i,j]) \tag{4.1}$$

$$I(a,b) = \begin{cases} 1, & \text{if } a = b \text{ (solving the ties as explained before)} \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

The following table shows the surrogate models with the highest kappa per dataset(Rows) and oracle (columns).

	C50	**PLSR**	**NNMLP**	**GLM**	**PLMR**	**MARS**	**KNN**	**RF**	**RDA**	**NB**
Car	**C50** - - - - - - - - -	- - NNMLP * GLM * **PLMR** - - - - -	C50 * - **NNMLP** GLM * PLMR * - - - - -	- - - **GLM** PLMR * - - - - -	- - - **GLM** PLMR * - - - - -	C50 * - - GLM * PLMR * MARS * - **RF** - -	- - - - - - - **RF** - -	**C50** - - GLM * PLMR * - - - - -	C50 * - - **GLM** PLMR * - - - - -	- - - **GLM** PLMR * - - - - -
Heart-c	**C50** - NNMLP * - - **MARS** - **RF** - -	- - NNMLP * GLM * **PLMR** - - - - -	- - NNMLP * GLM * **PLMR** - - - RDA * -	- - NNMLP * GLM * **PLMR** - - - - -	- - NNMLP * GLM * **PLMR** - - - - -	- - - - - **MARS** - - - -	- - NNMLP * GLM * PLMR * MARS * - RF * **RDA** -	- - NNMLP * GLM * PLMR * **MARS** - - RDA * -	- - NNMLP * **GLM** **PLMR** - - - - -	- - - GLM * PLMR * **MARS** - - - -

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
Badges2	**C50** **PLSR** **NNMLP** **GLM** **PLMR** **MARS** **KNN** **RF** **RDA** **NB**	**C50** **PLSR** **NNMLP** **GLM** **PLMR** **MARS** **KNN** **RF** **RDA** **NB**	**C50** **PLSR** **NNMLP** **GLM** **PLMR** **MARS** **KNN** **RF** **RDA** **NB**	**C50** **PLSR** **NNMLP** **GLM** **PLMR** **MARS** **KNN** **RF** **RDA** **NB**	C50 * PLSR * NNMLP * GLM * **PLMR** MARS * KNN * RF * RDA * NB *	**C50** **PLSR** **NNMLP** **GLM** **PLMR** **MARS** **KNN** **RF** **RDA** **NB**	C50 * - - - - - - **RF** - -	**C50** **PLSR** **NNMLP** **GLM** **PLMR** **MARS** **KNN** **RF** **RDA** **NB**	**C50** **PLSR** **NNMLP** **GLM** **PLMR** **MARS** **KNN** **RF** **RDA** **NB**	C50 * - NNMLP * GLM * PLMR * MARS * - **RF** RDA * NB *
Balance-scale	C50 * - - - - **MARS** - RF * - -	- PLSR * **NNMLP** **GLM** PLMR * MARS * - - RDA * NB *	- PLSR * NNMLP * **GLM** **PLMR** MARS * - RF * RDA * -	- - - **GLM** **PLMR** MARS * - - RDA * -	- - NNMLP * GLM * **PLMR** MARS * - - RDA * -	- PLSR * NNMLP * GLM * **PLMR** MARS * - RF * RDA * NB *	- - - - - **MARS** - - - NB *	- PLSR * NNMLP * GLM * PLMR * **MARS** - RF * RDA * NB *	- - **NNMLP** **GLM** **PLMR** MARS * - - RDA * -	- **PLSR** **NNMLP** GLM * PLMR * MARS * - RF * RDA * NB *
Banknote-Authentication	C50 * - - - - MARS * - **RF** - -	- PLSR * NNMLP * **GLM** PLMR * - - - RDA * -	- - **NNMLP** GLM * PLMR * MARS * - - RDA * -	- - **NNMLP** **GLM** **PLMR** MARS * - - RDA * -	- - NNMLP * **GLM** **PLMR** - - - - -	- - - - - MARS * **KNN** RF * - -	C50 * - - - - MARS * **KNN** RF * - -	C50 * - - - - MARS * **KNN** RF * - -	- - - - - **MARS** - - RDA * -	C50 * PLSR * NNMLP * GLM * PLMR * MARS * KNN * RF * **RDA** -
Credit-g	**C50** - - - - - - - - -	- - NNMLP * **GLM** PLMR * - - - - -	- PLSR * **NNMLP** GLM * PLMR * MARS * - - RDA * -	- PLSR * **NNMLP** GLM * PLMR * - - - RDA * -	- - NNMLP * GLM * **PLMR** - - - - -	- - - - - **MARS** - - - -	- - NNMLP * GLM * PLMR * - - - **RDA** -	- - - - - MARS * - **RF** - -	- PLSR * NNMLP * GLM * **PLMR** - - - RDA * -	- PLSR * - GLM * PLMR * **MARS** - - RDA * -
Diabetes	**C50** - - - - - - RF * - -	- - NNMLP * GLM * **PLMR** - - - - -	- **PLSR** NNMLP * GLM * PLMR * MARS * - - **RDA** NB *	- PLSR * **NNMLP** GLM * PLMR * MARS * - - RDA * -	- PLSR * NNMLP * GLM * **PLMR** - - - RDA * -	- - - - - **MARS** - - - -	- - - - - MARS * **KNN** RF * - -	- - - - - - - **RF** - -	- - **NNMLP** GLM * PLMR * - - - - -	- - NNMLP * GLM * PLMR * - - - **RDA** -
Haberman	C50 * - NNMLP * - PLMR * **MARS** - RF * - -	- - NNMLP * GLM * **PLMR** - - - - -	- - NNMLP * GLM * - **MARS** - - - NB *	- PLSR * - - - - - - **RDA** NB *	- - **NNMLP** GLM * PLMR * - - RF * - -	- - - - - **MARS** - RF * - -	**C50** - NNMLP * - - - KNN * - - -	C50 * - - - - - - **RF** - -	- - NNMLP * GLM * **PLMR** - - RF * - -	C50 * - NNMLP * **GLM** - MARS * - RF * - -

	C50	**PLSR**	**NNMLP**	**GLM**	**PLMR**	**MARS**	**KNN**	**RF**	**RDA**	**NB**
Iris	**C50**	-	-	-	-	C50 *	-	**C50**	-	-
	-	-	-	-	-	-	-	-	-	-
	-	-	NNMLP *	-	-	-	-	-	-	-
	-	GLM *	GLM *	**GLM**	GLM *	-	**GLM**	-	GLM *	-
	-	**PLMR**	PLMR *	PLMR *	**PLMR**	-	**PLMR**	-	**PLMR**	-
	-	-	-	-	-	MARS *	MARS *	-	-	**MARS**
	-	-	-	-	-	-	-	-	-	-
	RF *	-	-	-	-	**RF**	-	RF *	-	-
	-	-	**RDA**	-	-	RDA *	RDA *	-	RDA *	-
	-	-	-	-	-	-	-	-	-	-
Labor	C50 *	-	-	-	-	**C50**	-	C50 *	-	-
	-	-	-	-	-	-	-	-	-	-
	-	NNMLP *	**NNMLP**	NNMLP *	**NNMLP**	-	NNMLP *	-	**NNMLP**	NNMLP *
	-	**GLM**	GLM *	GLM *	GLM *	-	-	-	GLM *	GLM *
	-	PLMR *	PLMR *	**PLMR**	**PLMR**	-	PLMR *	-	PLMR *	PLMR *
	MARS *	-	-	-	-	-	-	-	-	**MARS**
	-	-	-	-	-	-	-	-	-	-
	RF	-	-	-	-	-	-	**RF**	-	-
	-	-	-	-	-	-	**RDA**	-	-	-
	-	-	-	-	-	-	-	-	-	-
Monks1	**C50**	**C50**	**C50**	**C50**	**C50**	**C50**	C50 *	**C50**	**C50**	**C50**
	PLSR	**PLSR**	-	**PLSR**	**PLSR**	-	-	-	-	**PLSR**
	NNMLP	**NNMLP**	**NNMLP**	**NNMLP**	**NNMLP**	-	-	-	-	**NNMLP**
	GLM	**GLM**	**GLM**	**GLM**	**GLM**	-	-	-	-	**GLM**
	PLMR	**PLMR**	**PLMR**	**PLMR**	**PLMR**	**MARS**	MARS *	**MARS**	MARS *	**PLMR**
	MARS	**MARS**	**MARS**	**MARS**	**MARS**	-	-	-	-	**MARS**
	-	-	-	-	-	RF *	**RF**	**RF**	RF *	-
	RF	**RF**	RF *	**RF**	**RF**	-	-	-	RDA *	**RF**
	RDA	**RDA**	RDA *	**RDA**	**RDA**	-	-	-	-	**RDA**
	NB	**NB**	-	**NB**	**NB**	-	-	-	-	**NB**
Phoneme	**C50**	-	-	-	-	C50 *	-	C50 *	-	-
	-	PLSR *	-	-	PLSR *	-	-	-	-	-
	-	NNMLP *	NNMLP *	NNMLP *	NNMLP *	-	-	-	-	-
	-	**GLM**	GLM *	GLM *	GLM *	-	-	-	-	-
	-	PLMR *	**PLMR**	**PLMR**	**PLMR**	-	-	-	-	-
	-	MARS *	-	-	-	**MARS**	-	MARS *	-	**MARS**
	-	-	-	-	-	-	**KNN**	-	**KNN**	-
	RF *	-	-	-	-	RF *	-	**RF**	-	-
	-	RDA *	RDA *	RDA *	RDA *	-	-	-	-	-
	-	-	-	-	-	-	-	-	-	-
Tae	**C50**	C50 *	-	**C50**	-	C50 *	-	C50 *	C50 *	C50 *
	-	-	-	-	-	-	-	PLSR *	-	-
	-	-	NNMLP *	-	-	-	-	-	-	-
	-	GLM *	GLM *	GLM *	GLM *	-	-	**GLM**	-	-
	-	**PLMR**	PLMR *	PLMR *	**PLMR**	-	-	PLMR *	-	-
	-	-	**MARS**	MARS *	-	**MARS**	-	MARS *	**MARS**	**MARS**
	-	KNN *	KNN *	KNN *	-	-	**KNN**	KNN *	KNN *	KNN *
	-	-	-	RF *	-	-	-	-	-	-
	-	RDA *	-	RDA *	-	-	-	RDA *	-	-
	-	-	-	NB *	-	-	-	NB *	-	-
Vote	**C50**	**C50**	-	**C50**	-	**C50**	-	C50 *	**C50**	-
	PLSR	**PLSR**	-	**PLSR**	-	-	PLSR *	-	**PLSR**	-
	NNMLP	**NNMLP**	NNMLP *	**NNMLP**	NNMLP *	-	NNMLP *	NNMLP *	**NNMLP**	NNMLP *
	GLM	**GLM**	GLM *	**GLM**	GLM *	-	GLM *	GLM *	**GLM**	**GLM**
	PLMR	**PLMR**	**PLMR**	**PLMR**	**PLMR**	-	PLMR *	PLMR *	**PLMR**	PLMR *
	MARS	**MARS**	-	**MARS**	-	**MARS**	MARS *	MARS *	**MARS**	MARS *
	KNN *	KNN *	-	KNN *	-	-	-	-	KNN *	-
	RF	**RF**	-	**RF**	-	**RF**	RF *	**RF**	**RF**	-
	RDA	**RDA**	-	**RDA**	-	-	**RDA**	-	**RDA**	-
	NB	**NB**	-	**NB**	-	-	-	-	**NB**	-

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
PhishingWebsites	**C50**	-	-	-	-	-	-	C50 *	-	-
	PLSR *	-	PLSR *	PLSR *	-	-	-	PLSR *	-	-
	NNMLP *	NNMLP *	NNMLP *	NNMLP *	NNMLP *	-	-	NNMLP *	-	NNMLP *
	GLM *	GLM *	**GLM**	**GLM**	GLM *	-	-	GLM *	-	GLM *
	PLMR *	**PLMR**	PLMR *	PLMR *	**PLMR**	-	-	PLMR *	-	**PLMR**
	MARS *	-	MARS *	MARS *	-	**MARS**	**MARS**	MARS *	MARS *	MARS *
	-	-	-	-	-	-	-	-	-	-
	RF *	-	-	-	-	-	RF *	**RF**	-	-
	RDA *	-	RDA *	RDA *	-	-	RDA *	RDA *	**RDA**	-
	NB *	-	-	NB *	-	-	-	NB *	-	-
Wall-Robot-Navigation	C50 *	-	C50 *	-	-	C50 *	-	C50 *	-	-
	-	-	**PLSR**	-	-	-	-	-	-	PLSR *
	-	-	NNMLP *	-	-	-	-	-	-	-
	-	GLM *	GLM *	GLM *	GLM *	-	GLM *	-	-	GLM *
	-	**PLMR**	PLMR *	**PLMR**	**PLMR**	-	PLMR *	-	-	PLMR *
	MARS *	-	MARS *	-	-	**MARS**	**MARS**	MARS *	MARS *	**MARS**
	-	-	-	-	-	-	-	-	-	-
	RF	-	RF *	-	-	RF *	RF *	**RF**	-	-
	-	-	**RDA**	-	-	-	RDA *	-	**RDA**	RDA *
	-	-	NB *	-	-	-	NB *	-	NB *	NB *
Waveform-5000	**C50**	-	C50 *	-	-	-	-	-	-	-
	-	-	PLSR *	-	-	-	PLSR *	PLSR *	-	-
	-	-	NNMLP *	-	-	-	-	-	-	-
	-	GLM *	GLM *	**GLM**	GLM *	-	GLM *	GLM *	GLM *	-
	-	**PLMR**	PLMR *	PLMR *	**PLMR**	-	PLMR *	PLMR *	**PLMR**	-
	-	-	MARS *	-	-	**MARS**	MARS *	**MARS**	-	**MARS**
	-	-	**KNN**	-	-	-	-	-	-	-
	RF *	-	RF *	-	-	-	-	-	-	-
	-	-	RDA *	-	-	-	**RDA**	RDA *	-	-
	-	-	NB *	-	-	-	NB *	-	-	-
Cmc	**C50**	-	-	-	-	**C50**	C50 *	-	-	C50 *
	-	-	-	-	-	-	-	-	-	-
	-	-	NNMLP *	-	-	-	-	-	-	-
	-	**GLM**	**GLM**	GLM *	GLM *	-	-	-	GLM *	GLM *
	-	PLMR *	PLMR *	**PLMR**	**PLMR**	-	-	-	**PLMR**	**PLMR**
	-	-	-	-	-	MARS *	MARS *	-	-	MARS *
	-	-	-	-	-	-	-	-	-	-
	-	-	-	-	-	RF *	**RF**	**RF**	-	-
	-	-	RDA *	-	-	-	RDA *	-	-	-
	-	-	-	-	-	-	-	-	-	-
Heart-h	**C50**	-	-	-	-	C50 *	-	C50 *	-	-
	-	-	-	-	-	-	PLSR *	-	-	-
	-	-	NNMLP *	**NNMLP**	NNMLP *	-	**NNMLP**	-	**NNMLP**	NNMLP *
	-	GLM *	GLM *	GLM *	**GLM**	-	GLM *	-	-	GLM *
	-	**PLMR**	**PLMR**	PLMR *	PLMR *	-	PLMR *	-	-	PLMR *
	MARS *	-	-	-	-	**MARS**	MARS *	MARS *	-	MARS *
	-	-	-	-	-	-	-	-	-	-
	RF *	-	-	-	-	RF *	RF *	**RF**	-	-
	-	-	-	-	-	-	RDA *	-	-	**RDA**
	-	-	-	-	-	-	NB *	-	-	NB *
Credit-a	**C50**	-	-	-	-	C50 *	C50 *	C50 *	-	-
	PLSR	-	-	-	-	-	-	PLSR *	-	-
	NNMLP	NNMLP *	NNMLP *	-	NNMLP *	NNMLP *	NNMLP *	NNMLP *	-	NNMLP *
	GLM	**GLM**	-	**GLM**	**GLM**	-	**GLM**	**GLM**	-	**GLM**
	PLMR	PLMR *	**PLMR**	-	PLMR *	PLMR *	PLMR *	PLMR *	**PLMR**	PLMR *
	MARS	-	-	-	-	**MARS**	MARS *	MARS *	-	MARS *
	-	-	-	-	-	-	-	-	-	-
	RF	-	-	-	-	RF *	RF *	RF *	-	RF *
	RDA	-	-	-	-	-	-	RDA *	-	-
	NB	-	-	-	-	-	-	NB *	-	-
%	**1.00**	**0.3**	**1.00**	**0.95**	**1.00**	**0.95**	**0.25**	**0.8**	**0.45**	**0.25**

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB

Table 4.5: *Summary table showing the surrogate models with the highest kappa highlighted in bold (strict selection) and the approximate kappa highlighted with an asterisk (soft selection) per dataset(Rows) and oracle (columns).*

The following table shows the surrogate model with the max features importance rank correlation for the dataset (Rows) and oracle (columns).

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
Car	-	-	-	-	-	**C50**	C50 *	**C50**	-	**C50**
	-	-	-	-	-	-	-	-	**PLSR**	**PLSR**
	-	-	-	-	-	-	-	**NNMLP**	-	**NNMLP**
	-	**GLM**	-	**GLM**	**GLM**	**GLM**	-	-	-	**GLM**
	PLMR	-	-	-	**PLMR**	**PLMR**	-	**PLMR**	-	**PLMR**
	-	-	**MARS**	**MARS**	-	**MARS**	**MARS**	-	-	**MARS**
	-	-	-	-	-	**KNN**	-	-	-	-
	RF	-	-	-	-	**RF**	-	-	-	**RF**
	RDA	**RDA**	-	**RDA**	**RDA**	**RDA**	-	**RDA**	-	**RDA**
	-	**NB**	**NB**	-	-	**NB**	-	-	**NB**	**NB**
Heart-c	-	C50 *	C50 *	-	-	C50 *	-	-	-	-
	-	-	-	-	-	-	-	-	-	-
	-	NNMLP *	**NNMLP**	-	-	NNMLP *	-	-	**NNMLP**	-
	-	**GLM**	GLM *	GLM *	-	-	GLM *	-	-	-
	-	-	PLMR *	-	-	-	PLMR *	-	-	-
	-	-	-	-	-	-	**MARS**	-	-	-
	-	KNN *	-	**KNN**	-	**KNN**	-	-	KNN *	-
	RF	RF *	-	-	-	-	-	**RF**	-	-
	-	-	-	-	-	-	RDA *	-	-	**RDA**
	-	NB *	-	-	**NB**	-	NB *	-	-	-
Badges2	**C50**	**C50**	**C50**	**C50**	C50 *	**C50**	-	**C50**	**C50**	C50 *
	PLSR	**PLSR**	**PLSR**	**PLSR**	PLSR *	**PLSR**	-	**PLSR**	**PLSR**	-
	NNMLP	**NNMLP**	**NNMLP**	**NNMLP**	NNMLP *	**NNMLP**	-	**NNMLP**	**NNMLP**	**NNMLP**
	GLM	**GLM**	**GLM**	**GLM**	GLM *	**GLM**	-	**GLM**	**GLM**	GLM *
	PLMR	**PLMR**	**PLMR**	**PLMR**	PLMR *	**PLMR**	-	**PLMR**	**PLMR**	**PLMR**
	MARS	**MARS**	**MARS**	**MARS**	MARS *	**MARS**	-	**MARS**	**MARS**	MARS *
	-	**KNN**	-	-	KNN *	-	-	**KNN**	**KNN**	**KNN**
	RF	**RF**	**RF**	**RF**	**RF**	**RF**	-	**RF**	**RF**	-
	RDA	**RDA**	**RDA**	**RDA**	RDA *	**RDA**	RDA *	**RDA**	**RDA**	-
	NB	**NB**	**NB**	**NB**	NB *	**NB**	**NB**	**NB**	**NB**	-
Balance-scale	**C50**	-	-	-	-	**C50**	**C50**	**C50**	-	-
	-	-	**PLSR**	-	**PLSR**	**PLSR**	-	-	-	-
	-	**NNMLP**	-	-	-	-	**NNMLP**	-	**NNMLP**	-
	-	-	**GLM**	-	-	-	**GLM**	-	-	-
	-	-	**PLMR**	-	-	**PLMR**	**PLMR**	-	-	-
	-	-	-	-	-	**MARS**	-	-	-	-
	KNN	-	**KNN**	-	**KNN**	**KNN**	-	-	**KNN**	-
	-	**RF**	**RF**	**RF**	-	**RF**	-	-	-	**RF**
	-	-	**RDA**	**RDA**	**RDA**	**RDA**	-	-	-	-
	-	**NB**	-	-	**NB**	-	-	-	-	-
Banknote-Authentication	**C50**	**C50**	-	-	-	-	**C50**	**C50**	-	**C50**
	-	**PLSR**	**PLSR**	**PLSR**	-	-	**PLSR**	-	-	**PLSR**
	NNMLP	-	-	**NNMLP**	-	-	**NNMLP**	**NNMLP**	-	**NNMLP**
	-	-	-	-	-	-	**GLM**	**GLM**	-	**GLM**
	-	**PLMR**	-	-	-	-	**PLMR**	**PLMR**	**PLMR**	**PLMR**
	MARS	-	-	-	-	**MARS**	**MARS**	**MARS**	-	**MARS**
	KNN	-	-	-	**KNN**	**KNN**	**KNN**	**KNN**	-	**KNN**
	RF	-	-	-	-	**RF**	**RF**	**RF**	-	**RF**
	-	-	-	-	-	**RDA**	-	**RDA**	-	**RDA**
	-	**NB**	-	**NB**	-	-	-	**NB**	**NB**	-

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
Credit-g	-	-	-	C50 *	-	-	-	-	**C50**	-
	-	-	-	**PLSR**	-	-	-	-	-	-
	-	**NNMLP**	**NNMLP**	NNMLP *	NNMLP *	-	-	**NNMLP**	-	NNMLP *
	GLM	GLM *	GLM *	GLM *	GLM *	-	-	-	-	**GLM**
	-	PLMR *	PLMR *	PLMR *	PLMR *	-	-	-	-	-
	-	MARS *	-	-	-	MARS *	MARS *	-	-	-
	-	-	-	-	-	-	**KNN**	-	-	-
	-	-	-	-	RF *	**RF**	-	-	-	-
	RDA *	RDA *	-	RDA *	-	-	-	-	-	-
	-	-	NB *	-	**NB**	-	-	-	-	-
Diabetes	**C50**	-	**C50**	-	-	-	-	C50 *	-	-
	-	PLSR *	-	PLSR *	-	PLSR *	-	**PLSR**	-	-
	-	NNMLP *	-	NNMLP *	-	**NNMLP**	-	-	-	-
	-	GLM *	-	GLM *	-	-	-	-	-	-
	-	PLMR *	-	**PLMR**	-	-	-	-	-	-
	MARS *	-	-	-	**MARS**	-	**MARS**	-	-	-
	-	**KNN**	-	-	-	-	-	-	-	-
	RF	-	-	RF *	-	-	-	-	-	-
	-	RDA *	-	-	-	RDA *	-	-	-	-
	-	NB *	-	-	NB *	-	-	-	**NB**	**NB**
Haberman	**C50**	**C50**	**C50**	**C50**	**C50**	**C50**	-	-	**C50**	**C50**
	-	**PLSR**	**PLSR**	**PLSR**	**PLSR**	-	-	-	**PLSR**	**PLSR**
	NNMLP	**NNMLP**	**NNMLP**	**NNMLP**	**NNMLP**	-	-	-	**NNMLP**	**NNMLP**
	GLM	**GLM**	**GLM**	**GLM**	**GLM**	-	-	**GLM**	**GLM**	**GLM**
	PLMR	**PLMR**	**PLMR**	**PLMR**	**PLMR**	-	-	-	**PLMR**	**PLMR**
	MARS	**MARS**	**MARS**	**MARS**	**MARS**	-	-	-	**MARS**	**MARS**
	KNN	**KNN**	**KNN**	**KNN**	**KNN**	-	-	-	**KNN**	**KNN**
	RF	**RF**	**RF**	**RF**	**RF**	**RF**	-	-	**RF**	**RF**
	-	**RDA**	**RDA**	**RDA**	**RDA**	-	**RDA**	**RDA**	**RDA**	**RDA**
	NB	**NB**	**NB**	**NB**	**NB**	-	-	-	**NB**	**NB**
Iris	-	**C50**	**C50**	**C50**	**C50**	-	-	**C50**	-	**C50**
	PLSR	**PLSR**	**PLSR**	-	-	-	-	-	-	PLSR *
	NNMLP	-	**NNMLP**	**NNMLP**	**NNMLP**	**NNMLP**	-	**NNMLP**	-	NNMLP *
	GLM	**GLM**	**GLM**	**GLM**	**GLM**	**GLM**	**GLM**	-	-	GLM *
	PLMR	**PLMR**	**PLMR**	**PLMR**	**PLMR**	-	**PLMR**	-	-	PLMR *
	-	**MARS**	**MARS**	**MARS**	**MARS**	-	-	**MARS**	-	MARS *
	-	**KNN**	**KNN**	**KNN**	**KNN**	-	-	-	-	KNN *
	-	**RF**	**RF**	**RF**	**RF**	-	-	**RF**	-	RF *
	RDA	**RDA**	**RDA**	**RDA**	**RDA**	-	**RDA**	-	**RDA**	RDA *
	NB	**NB**	**NB**	**NB**	**NB**	**NB**	-	-	**NB**	NB *
Labor	**C50**	-	-	-	-	**C50**	-	-	-	-
	-	-	-	-	-	-	-	-	-	-
	-	**NNMLP**	-	**NNMLP**	-	-	-	-	-	-
	-	**GLM**	GLM *	-	-	-	-	-	-	-
	-	-	-	-	-	-	-	-	-	-
	MARS	-	**MARS**	-	-	-	-	-	**MARS**	**MARS**
	-	-	-	-	KNN *	-	**KNN**	**KNN**	-	-
	RF	**RF**	RF *	-	-	**RF**	-	**RF**	-	-
	-	**RDA**	-	-	**RDA**	**RDA**	-	-	-	-
	-	NB *	-	-	NB *	-	-	NB *	-	-
Monks1	**C50**	**C50**	-	**C50**	**C50**	-	C50 *	-	C50 *	-
	PLSR	**PLSR**	-	**PLSR**	**PLSR**	-	-	-	-	-
	NNMLP	**NNMLP**	**NNMLP**	**NNMLP**	**NNMLP**	-	-	-	**NNMLP**	-
	GLM	**GLM**	-	**GLM**	**GLM**	-	-	-	-	-
	PLMR	**PLMR**	-	**PLMR**	**PLMR**	-	-	-	-	-
	MARS	**MARS**	-	**MARS**	**MARS**	MARS *	-	**MARS**	MARS *	-
	-	-	-	-	-	-	**KNN**	-	**KNN**	**KNN**
	RF	**RF**	RF *	**RF**	**RF**	-	-	**RF**	-	-
	RDA	**RDA**	-	**RDA**	**RDA**	**RDA**	-	-	**RDA**	-
	NB	**NB**	-	**NB**	**NB**	-	-	-	**NB**	-

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
Phoneme	-	-	**C50**	**C50**	**C50**	**C50**	-	-	**C50**	-
	PLSR	-	**PLSR**	**PLSR**	**PLSR**	-	-	**PLSR**	-	-
	-	-	**NNMLP**	**NNMLP**	**NNMLP**	-	-	-	-	**NNMLP**
	-	-	**GLM**	-	**GLM**	-	**GLM**	-	-	**GLM**
	-	-	**PLMR**	-	-	-	**PLMR**	-	-	-
	-	-	**MARS**	-	-	-	-	-	**MARS**	-
	-	-	**KNN**	-	-	-	-	-	**KNN**	-
	-	-	**RF**	-	-	-	-	-	**RF**	-
	-	**RDA**	-	-	**RDA**	-	-	-	**RDA**	-
	-	-	-	-	NB *	-	-	-	**NB**	-
Tae	-	**C50**	-	-	**C50**	-	**C50**	-	-	-
	-	-	-	-	-	-	-	-	-	-
	-	NNMLP *	-	**NNMLP**	-	-	-	-	-	-
	GLM	**GLM**	-	-	**GLM**	**GLM**	-	-	-	**GLM**
	-	-	-	-	**PLMR**	-	-	-	-	-
	-	**MARS**	-	**MARS**	**MARS**	-	-	-	**MARS**	-
	-	**KNN**	-	-	**KNN**	-	**KNN**	-	-	-
	-	-	**RF**	-	-	-	-	**RF**	-	-
	-	**RDA**	-	-	**RDA**	**RDA**	-	-	-	-
	-	**NB**	**NB**	NB *	-	**NB**	-	-	-	-
Vote	**C50**	**C50**	-	**C50**	-	**C50**	C50 *	**C50**	**C50**	**C50**
	PLSR	-	**PLSR**	**PLSR**	PLSR *	-	-	-	**PLSR**	PLSR *
	NNMLP	-	NNMLP *	**NNMLP**	NNMLP *	-	-	-	**NNMLP**	-
	GLM	-	GLM *	**GLM**	GLM *	-	-	-	**GLM**	-
	PLMR	-	-	**PLMR**	**PLMR**	-	-	-	**PLMR**	-
	MARS	-	MARS *	**MARS**	-	**MARS**	-	-	**MARS**	-
	KNN *	-	-	-	-	-	-	-	-	KNN *
	RF	-	-	**RF**	-	RF *	-	-	**RF**	RF *
	RDA	-	RDA *	**RDA**	RDA *	-	**RDA**	RDA *	**RDA**	RDA *
	NB	-	NB *	**NB**	-	-	NB *	-	**NB**	NB *
PhishingWebsites	-	-	-	-	C50 *	-	-	-	-	-
	-	-	PLSR *	-	-	-	-	-	-	-
	-	**NNMLP**	NNMLP *	-	NNMLP *	NNMLP *	-	-	NNMLP *	NNMLP *
	-	GLM *	**GLM**	-	**GLM**	GLM *	-	-	-	GLM *
	-	PLMR *	PLMR *	-	PLMR *	-	-	-	-	PLMR *
	-	MARS *	MARS *	-	-	-	-	-	-	**MARS**
	-	-	-	-	-	-	-	-	-	-
	RF	-	RF *	-	-	**RF**	**RF**	**RF**	**RF**	-
	-	-	RDA *	-	-	-	-	-	-	-
	-	-	-	**NB**	-	-	-	-	-	NB *
Wall-Robot-Navigation	-	-	-	-	C50 *	-	-	C50 *	-	-
	-	-	PLSR *	-	-	-	PLSR *	-	-	-
	-	-	-	-	-	-	-	-	-	-
	-	-	-	-	-	-	-	-	**GLM**	-
	-	-	**PLMR**	PLMR *	-	-	-	-	-	-
	-	MARS *	-	MARS *	**MARS**	**MARS**	-	MARS *	-	-
	-	-	-	-	-	-	-	KNN *	-	-
	RF	-	-	-	-	RF *	-	**RF**	-	**RF**
	-	RDA *	-	-	-	-	**RDA**	-	-	-
	-	**NB**	-	**NB**	-	-	-	-	-	-
Waveform-5000	**C50**	-	-	C50 *	-	**C50**	-	**C50**	C50 *	**C50**
	-	PLSR *	-	-	-	-	**PLSR**	-	-	-
	-	-	-	NNMLP *	-	-	-	-	-	-
	-	GLM *	-	-	GLM *	-	-	-	-	-
	-	PLMR *	-	-	**PLMR**	-	-	-	-	-
	-	MARS *	-	-	-	MARS *	MARS *	-	-	-
	-	-	-	-	-	-	-	-	-	-
	RF *	**RF**	-	**RF**	-	RF *	RF *	RF *	**RF**	-
	-	RDA *	**RDA**	-	RDA *	-	-	-	-	-
	-	NB *	-	-	-	NB *	-	-	-	-

	C50	**PLSR**	**NNMLP**	**GLM**	**PLMR**	**MARS**	**KNN**	**RF**	**RDA**	**NB**
Cmc	C50 *	C50 *	-	-	-	**C50**	-	-	C50 *	**C50**
	PLSR	-	**PLSR**	PLSR *	-	-	-	-	-	-
	-	-	NNMLP *	-	**NNMLP**	-	-	-	-	-
	GLM	GLM *	GLM *	**GLM**	GLM *	-	-	-	GLM *	-
	PLMR *	**PLMR**	-	PLMR *	-	-	-	-	-	-
	MARS	MARS *	-	MARS *	-	-	-	-	-	-
	-	-	-	-	-	-	**KNN**	-	-	-
	-	RF *	-	RF *	-	RF *	-	RF *	**RF**	-
	RDA *	-	-	-	-	-	-	**RDA**	-	-
	-	**NB**	-	-	-	-	-	-	-	-
Heart-h	**C50**	-	-	-	-	C50 *	-	C50 *	-	-
	-	PLSR *	-	-	PLSR *	-	**PLSR**	-	-	-
	-	NNMLP *	**NNMLP**	NNMLP *	**NNMLP**	-	NNMLP *	-	-	-
	-	GLM *	-	GLM *	GLM *	GLM *	GLM *	-	-	-
	PLMR *	PLMR *	PLMR *	PLMR *	PLMR *	-	PLMR *	-	-	**PLMR**
	-	**MARS**	-	-	MARS *	MARS *	MARS *	-	-	-
	-	-	-	**KNN**	-	**KNN**	-	-	**KNN**	-
	RF *	-	-	-	-	-	-	RF *	-	-
	RDA *	RDA *	-	RDA *	RDA *	-	RDA *	**RDA**	RDA *	-
	-	-	-	-	NB *	-	NB *	NB *	-	-
Credit-a	-	-	-	-	-	-	-	-	-	-
	-	**PLSR**	-	-	-	-	-	-	-	-
	-	-	-	NNMLP *	-	-	-	-	-	-
	-	-	-	**GLM**	-	-	**GLM**	-	-	-
	-	-	-	PLMR *	-	-	PLMR *	-	-	PLMR *
	-	-	-	-	-	-	-	-	-	-
	-	-	-	-	**KNN**	-	-	-	**KNN**	-
	-	-	-	-	-	**RF**	-	**RF**	-	**RF**
	RDA	-	RDA *	-	-	-	-	-	-	-
	-	-	**NB**	-	-	-	-	-	-	-
%	**0.55**	**0.45**	**0.50**	**0.6**	**0.55**	**0.50**	**0.3**	**0.65**	**0.35**	**0.25**

Table 4.6: *Summary table showing the surrogate model with the max features importance rank correlation highlighted in bold (strict selection) and the approximate correlation highlighted with an asterisk (soft selection) for the dataset (Rows) and oracle (columns).*

4.8 Results

In this section we discuss the results obtained from the experiments performed to validate our method for extracting the features importance of a black-box model.

4.8.1 Final Discussion

Table 4.7 shows the accuracy for the strict and soft selection mods. Each row represents the dataset and each column represents the oracle model. The cell value has the following measuring: a value equal to 1 means the hypothesis success, and 0 otherwise. We also use color for better understanding the results, with green color meaning that the surrogate model that has highest kappa is the same as highest features importance rank correlation according to the restricted selection mode as we mentioned in the section 4.7.1 , the cell with yellow color means that it is a fail for the strict selection criterion but it is a success for the soft criterion, as it mentioned in section 4.7.1, and the cell with red color it means they not matched and the value always for that cell is 0. The column with name average shows the accuracy average per dataset (the sum of the row divided by 10, the number of oracle models) according to the strict selection criterion. And the last column with name average* shows the same as the column average but according to the soft selection criterion. The row with name average shows as well the accuracy average but per oracle (the sum of the column divided by 20, the number of the datasets) according to the strict selection mode. And the last row with name average* shows the same as the row average but according to the soft selection mode. We observe that there is a big difference between the accuracy result obtained for each one of the selection criteria. The difference in accuracy obtained (from 0.365 to 0.710) shows that the strict selection criteria is too much restrictive because the differences in kappa and rank correlation among surrogate models and oracle is almost negligible in most cases. So our assumption of using the soft selection criterion seems to be suitable for the problem at hand. Regarding the result for the soft case, we can say that an accuracy of 0.71 is a good result that validates our hypothesis, taking into account that estimating the feature importance of a black-box oracle without using the training data is a very complex and difficult problem.

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB	Average	Average*
Car	0	1	0	1	1	1	0	1	0	1	0.50	0.60
Heart-c	1	1	1	1	0	0	1	0	1	0	0.10	0.60
Badges2	1	1	1	1	1	1	0	1	1	1	0.70	0.90
Balance-scale	1	1	1	1	1	1	0	0	1	1	0.40	0.80
Banknote-Authentication	1	1	0	1	0	1	1	1	0	1	0.60	0.70
Credit-g	0	1	1	1	1	1	0	0	0	1	0.10	0.60
Diabetes	1	1	0	1	0	0	1	0	0	0	0.10	0.40
Haberman	1	1	1	1	1	1	0	0	1	1	0.70	0.80
Iris	0	1	1	1	1	0	1	1	1	1	0.60	0.80
Labor	1	1	1	1	0	1	0	1	0	1	0.50	0.70
Monks1	1	1	1	1	1	1	1	1	1	0	0.60	0.90
Phoneme	0	1	1	1	1	1	0	0	1	0	0.20	0.60
Tae	0	1	0	1	1	0	1	0	1	0	0.30	0.50
Vote	1	1	1	1	1	1	1	1	1	0	0.70	0.90
PhishingWebsites	1	1	1	1	1	0	1	1	0	1	0.20	0.80
Wall-Robot-Navigation	1	0	1	1	0	1	1	1	0	0	0.30	0.60
Waveform-5000	1	1	1	0	1	1	1	0	0	0	0.20	0.60
Cmc	1	1	1	1	1	1	0	1	1	1	0.10	0.90
Heart-h	1	1	1	1	1	1	1	1	0	1	0.10	0.90
Credit-a	1	0	0	1	0	1	1	1	0	1	0.30	0.60
Average	0.60	0.35	0.40	0.40	0.35	0.40	0.25	0.40	0.30	0.20	0.365	
Average*	0.75	0.90	0.75	0.95	0.70	0.75	0.60	0.60	0.50	0.60	0.710	

Table 4.7: *Count of number of times hypothesis holds or not for each oracle and datasets for the strict and soft selection criteria.*

Chapter 5

Conclusion

5.1 Conclusion

Machine Learing models usually perform really well for predictions but are not always interpretable, as most ML techniques generate models that behave as black-box models. ML model interpretability attracts a lot of research interest in the recent years with a growing interest in extracting information from non-interpretable models. There is an extensive literature that performed studies on features selection and feature importance, however most of those studies do not work directly with black-box models and this issue still has not been investigated in depth.

Our focus in this TFM was to work with black-box models assuming that the original data nor the model are available, with the aim to determine their feature importance. Surrogate models trained with an artificially generated dataset labeled by the original black-box model was used to construct models that capture or imitate the behavior of the original model and those models were used to forecast feature importance. Our main assumption was that two models that are similar in behavior, also will agree in their feature importance. We identify surrogate/oracle model similarity in behaviour and feature importance by utilizing two measures: Kappa statistics and Spearman correlation, respectively. We proved our hypothesis experimentally over a large collection of datasets and using different learning techniques to create oracles and surrogate models.

From the results obtained in the experiments, we conclude that using the black-box model (oracle) to label a set of examples for training the surrogate models allows us to approximate feature importance through the surrogate models. We have presented two different criteria to select the surrogate model with highest kappa and correlation. The results show that the strict selection mode (in that we test whether the highest kappa surrogate model coincides with the highest correlation surrogate model) is too much restrictive since the differences in kappa and spearman correlation among several surrogate models are too small for some combinations oracle-dataset. Thus, we have introduced another selection criterion that considers all surrogate models that differ in their kappa and/or correlation value from the highest values in a threshold we set to 0.05. The accuracy results confirm that the soft criterion is more suitable than the strict criterion for determining feature importance from surrogate models. The global accuracy obtaining is 0.71 which can be considered as a good result given the difficulty in estimating feature importance without using the original model nor the original data.

5.2 Future Works

This work can be extended in different ways. First, more experiments using more datasets and other ML techniques would be performed to provide more evidences of correctness of our hypothesis, and using more extensive evaluation procedure by applying, for instance, cross validation. Also, we can further analyse the impact of the size of the surrogate dataset for feature importance determining. We also plan to extend this work by defining a new feature importance approach that uses the feature importance given by all the surrogate models instead of selecting only one surrogate model.

Bibliography

[1] R. Fabra-Boluda, C. Ferri, J. Hernández-Orallo, F. Martínez-Plumed, and M. J. Ramírez-Quintana, "Identifying the machine learning family from black-box models," in *Advances in Artificial Intelligence*, F. Herrera, S. Damas, R. Montes, S. Alonso, Ó. Cordón, A. González, and A. Troncoso, Eds. Cham: Springer International Publishing, 2018, pp. 55–65.

[2] C. Molnar, *Interpretable Machine Learning*, 2019, https://christophm.github.io/interpretable-ml-book/.

[3] J. Brownlee, *Master Machine Learning Algorithm*, 2016, https://books.google.es/books?hl=en&lr=&id=n--oDwAAQBAJ&oi=fnd&pg=PP1&dq=++++Jason+Brownlee+on+March+16,+2016+in+Machine+Learning+Algorithms.+&ots=3jmz02rxv9&sig=VlvUkEluE45Ic6MJmWDiJHH5d4c&redir_esc=y#v=onepage&q&f=false.

[4] Z. C. Lipton, "The mythos of model interpretability," 2016.

[5] A. Fisher, C. Rudin, and F. Dominici, "Model class reliance: Variable importance measures for any machine learning model class, from the "rashomon" perspective," 01 2018.

[6] A. Goldstein, A. Kapelner, J. Bleich, and E. Pitkin, "Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation," 2013.

[7] G. Casalicchio, C. Molnar, and B. Bischl, "Visualizing the feature importance for black box models," 2018.

[8] J. R. Landis and G. G. Koch, "An application of hierarchical kappa-type statistics in the assessment of majority agreement among multiple observers." *Biometrics*, vol. 33 2, pp. 363–74, 1977.

[9] ——, "The measurement of observer agreement for categorical data," *Biometrics*, vol. 33, no. 1, pp. 159–174, 1977. [Online]. Available: http://www.jstor.org/stable/2529310

[10] D. Dua and C. Graff, "UCI machine learning repository," 2019. [Online]. Available: http://archive.ics.uci.edu/ml

Abbreviations

ML	Machine learning
EN	Ensembles
DT	Decision Trees
NNET	Neural Network
NB	Naive Bayes
NN	Nearest Neighbours
GLM	Generalized Linear Models
PLSR	Partial Least Squares and Regression
LMR	Logistic and Multinominal Regression
DA	Discriminant Analysis
MARS	Multivariate Adaptive Regression Splines
FeatureImp	Features importance
D	Dataset
SM	Surrogate Models
OM	Oracle model
O	Qracler
SD	Surrogate dataset

Appendices

Appendix A

Table

A.1 Kappa

Kappa Result Tables

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.924	0.652	0.738	0.819	0.801	0.706	0.547	0.804	0.755	0.671
PLSR	0.927	0.868	0.965	0.970	0.991	0.854	0.627	0.866	0.857	0.877
NNMLP	0.956	0.840	0.993	0.964	0.964	0.878	0.682	0.914	0.877	0.883
GLM	0.878	0.721	0.810	0.951	0.930	0.635	0.513	0.792	0.828	0.728
PLMR	0.871	0.745	0.799	0.926	0.910	0.666	0.532	0.795	0.834	0.734
MARS	0.990	0.784	0.817	0.959	0.959	0.962	0.638	0.993	0.888	0.815
KNN	0.760	0.524	0.752	0.736	0.709	0.625	0.420	0.817	0.746	0.527
RF	0.881	0.687	0.746	0.837	0.841	0.697	0.539	0.793	0.779	0.669
RDA	0.908	0.733	0.790	0.940	0.925	0.793	0.623	0.858	0.862	0.773
NB	0.841	0.836	0.915	0.976	0.956	0.862	0.563	0.839	0.905	0.825

Table A.1: *Kappa result for* ***Car*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.982	0.775	0.949	0.913	0.915	0.982	0.755	0.982	0.845	0.840
PLSR	0.536	0.819	0.964	0.982	0.982	0.790	0.494	0.589	0.901	0.653
NNMLP	0.312	0.498	0.548	0.546	0.548	0.496	0.365	0.367	0.499	0.486
GLM	0.493	0.710	0.949	0.948	0.970	0.742	0.454	0.573	0.883	0.632
PLMR	0.528	0.714	0.959	0.976	0.979	0.845	0.432	0.593	0.871	0.656
MARS	0.703	0.437	0.491	0.424	0.443	0.827	0.483	0.721	0.596	0.415
KNN	0.570	0.614	0.670	0.665	0.668	0.674	0.613	0.648	0.691	0.588
RF	0.535	0.462	0.624	0.651	0.647	0.671	0.394	0.609	0.625	0.449
RDA	0.497	0.762	0.942	0.968	0.968	0.668	0.532	0.595	0.852	0.618
NB	0.601	0.083	0.716	0.749	0.755	0.766	0.297	0.590	0.716	0.613

Table A.2: *Kappa result for* ***Heart-c*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
PLSR	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
NNMLP	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
GLM	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
PLMR	0.950	0.964	0.986	0.974	0.986	0.962	0.960	0.964	0.964	0.964
MARS	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
KNN	0.788	0.690	0.724	0.732	0.730	0.752	0.782	0.834	0.746	0.666
RF	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
RDA	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
NB	0.899	0.836	0.901	0.899	0.897	0.933	0.865	0.938	0.919	0.908

Table A.3: *Kappa result for* ***Badges2*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.878	0.759	0.778	0.783	0.799	0.907	0.798	0.895	0.823	0.823
PLSR	0.825	0.985	0.995	0.995	0.990	0.970	0.790	0.900	0.985	0.970
NNMLP	0.693	0.788	0.808	0.818	0.818	0.803	0.756	0.768	0.778	0.717
GLM	0.651	0.781	0.891	0.992	0.992	0.983	0.682	0.754	0.992	0.780
PLMR	0.674	0.754	0.988	0.988	0.996	0.992	0.653	0.791	0.975	0.754
MARS	0.746	0.869	0.884	0.894	0.895	0.851	0.779	0.890	0.884	0.865
KNN	0.676	0.840	0.840	0.825	0.825	0.910	0.751	0.835	0.845	0.870
RF	0.789	0.819	0.829	0.829	0.829	0.860	0.689	0.830	0.829	0.845
RDA	0.688	0.751	1.000	1.000	1.000	0.996	0.662	0.771	0.992	0.756
NB	0.725	0.930	0.930	0.925	0.925	0.925	0.815	0.885	0.910	0.920

Table A.4: *Kappa result for* ***Balance-scale*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.902	0.708	0.715	0.706	0.706	0.871	0.859	0.912	0.708	0.691
PLSR	0.858	0.955	0.995	1.000	0.995	0.950	0.874	0.858	0.970	0.904
NNMLP	0.813	0.908	0.995	0.984	0.979	0.953	0.850	0.876	0.959	0.879
GLM	0.849	0.923	0.990	0.990	0.990	0.943	0.850	0.865	0.943	0.916
PLMR	0.817	0.923	0.979	0.995	0.995	0.885	0.850	0.870	0.933	0.889
MARS	0.709	0.451	0.608	0.451	0.451	0.768	0.811	0.766	0.742	0.479
KNN	0.794	0.749	0.727	0.738	0.754	0.834	0.839	0.829	0.753	0.724
RF	0.815	0.739	0.707	0.739	0.734	0.819	0.845	0.840	0.744	0.734
RDA	0.629	0.785	0.802	0.745	0.750	0.866	0.775	0.742	0.835	0.709
NB	0.899	0.865	0.875	0.886	0.880	0.895	0.868	0.905	0.912	0.814

Table A.5: *Kappa result for* ***Banknote-Authentication*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.938	0.706	0.726	0.736	0.723	0.811	0.467	0.869	0.705	0.741
PLSR	0.468	0.918	0.962	0.979	0.976	0.881	0.454	0.648	0.924	0.845
NNMLP	0.418	0.643	0.652	0.650	0.641	0.634	0.363	0.494	0.643	0.579
GLM	0.500	0.830	0.855	0.847	0.852	0.768	0.422	0.616	0.827	0.718
PLMR	0.494	0.904	0.959	0.968	0.980	0.872	0.461	0.661	0.914	0.856
MARS	0.564	0.420	0.426	0.428	0.416	0.813	0.360	0.705	0.486	0.427
KNN	0.269	0.387	0.431	0.418	0.423	0.390	0.335	0.379	0.455	0.345
RF	0.582	0.585	0.513	0.568	0.588	0.664	0.298	0.666	0.545	0.573
RDA	0.532	0.931	0.959	0.980	0.980	0.849	0.443	0.627	0.936	0.894
NB	0.683	0.822	0.791	0.826	0.832	0.853	0.253	0.727	0.826	0.776

Table A.6: *Kappa result for* ***Credit-g*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.951	0.742	0.747	0.721	0.742	0.792	0.672	0.914	0.738	0.703
PLSR	0.745	0.910	0.975	0.977	0.985	0.925	0.747	0.842	0.910	0.825
NNMLP	0.687	0.798	0.758	0.772	0.772	0.753	0.719	0.719	0.798	0.774
GLM	0.748	0.925	0.950	0.945	0.940	0.915	0.748	0.830	0.925	0.830
PLMR	0.717	0.940	0.980	0.987	0.987	0.882	0.745	0.842	0.940	0.865
MARS	0.644	0.429	0.455	0.449	0.446	0.867	0.552	0.676	0.582	0.499
KNN	0.476	0.463	0.461	0.467	0.468	0.510	0.558	0.543	0.502	0.484
RF	0.696	0.586	0.595	0.601	0.594	0.701	0.648	0.761	0.634	0.628
RDA	0.730	0.902	0.997	0.982	0.992	0.860	0.760	0.845	0.902	0.805
NB	0.507	0.000	0.699	0.724	0.714	0.660	0.632	0.341	0.739	0.664

Table A.7: *Kappa result for* ***Diabetes*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.952	0.840	0.952	0.904	0.939	0.966	0.883	0.945	0.835	0.868
PLSR	0.819	0.863	0.948	0.955	0.992	0.889	0.851	0.864	0.863	0.807
NNMLP	0.598	0.084	0.669	0.650	0.611	0.685	0.206	0.625	0.560	0.636
GLM	0.741	0.866	0.806	0.791	0.799	0.745	0.668	0.768	0.866	0.857
PLMR	0.807	0.895	0.977	0.954	0.961	0.891	0.799	0.930	0.888	0.817
MARS	0.797	0.216	0.317	0.157	0.203	0.861	0.388	0.848	0.246	0.193
KNN	0.799	0.586	0.754	0.626	0.620	0.678	0.793	0.740	0.605	0.632
RF	0.806	0.513	0.666	0.506	0.507	0.746	0.593	0.806	0.464	0.612
RDA	0.779	0.870	0.937	0.936	0.952	0.866	0.768	0.904	0.878	0.826
NB	0.949	0.888	0.967	0.975	0.910	0.966	0.745	0.949	0.888	0.905

Table A.8: *Kappa result for* ***Haberman*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	1.000	0.631	0.555	0.805	0.797	0.942	0.746	0.996	0.888	0.884
PLSR	0.766	0.879	0.434	0.985	0.989	0.879	0.849	0.823	0.887	0.766
NNMLP	0.713	0.664	0.937	0.933	0.933	0.805	0.792	0.758	0.957	0.744
GLM	0.711	0.747	0.588	0.987	0.987	0.780	0.853	0.819	0.881	0.763
PLMR	0.750	0.728	0.609	0.976	0.988	0.845	0.847	0.813	0.933	0.739
MARS	0.943	0.646	0.694	0.869	0.878	0.943	0.828	0.951	0.904	0.847
KNN	0.633	0.638	0.514	0.817	0.817	0.813	0.730	0.762	0.808	0.710
RF	0.809	0.479	0.340	0.526	0.546	0.723	0.629	0.787	0.600	0.556
RDA	0.518	0.509	0.438	0.630	0.637	0.536	0.541	0.527	0.612	0.545
NB	0.803	0.652	0.678	0.804	0.800	0.886	0.791	0.798	0.814	0.762

Table A.9: *Kappa result for* ***Iris*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.996	0.530	0.771	0.522	0.536	0.996	0.671	1.000	0.622	0.577
PLSR	0.660	0.127	0.948	0.956	0.948	0.694	0.548	0.454	0.775	0.552
NNMLP	0.526	0.167	0.952	0.931	0.906	0.661	0.656	0.479	0.781	0.591
GLM	0.550	0.174	0.898	0.913	0.916	0.650	0.620	0.457	0.748	0.590
PLMR	0.479	0.189	0.962	0.953	0.962	0.665	0.643	0.452	0.772	0.650
MARS	0.904	0.000	0.000	0.000	0.125	0.774	0.437	0.807	0.479	0.080
KNN	0.548	0.138	0.735	0.686	0.741	0.630	0.716	0.550	0.776	0.528
RF	0.695	0.191	0.590	0.569	0.569	0.638	0.515	0.720	0.607	0.469
RDA	0.536	0.000	0.884	0.840	0.851	0.612	0.623	0.471	0.740	0.514
NB	0.663	0.000	0.861	0.843	0.843	0.881	0.517	0.650	0.751	0.806

Table A.10: *Kappa result for* ***Labor*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	1.000	1.000	1.000	1.000	1.000	1.000	0.668	1.000	1.000	1.000
PLSR	1.000	1.000	1.000	1.000	1.000	1.000	0.823	1.000	1.000	1.000
NNMLP	1.000	0.943	1.000	1.000	1.000	1.000	0.679	0.983	0.961	0.949
GLM	1.000	1.000	1.000	1.000	1.000	1.000	0.813	1.000	1.000	1.000
PLMR	1.000	1.000	1.000	1.000	1.000	1.000	0.816	1.000	1.000	1.000
MARS	1.000	0.285	0.565	0.291	0.274	1.000	0.506	0.996	0.419	0.555
KNN	0.987	0.264	0.315	0.254	0.254	0.987	0.713	0.993	0.659	0.282
RF	1.000	0.241	0.311	0.241	0.227	1.000	0.700	1.000	0.690	0.266
RDA	0.997	0.829	0.868	0.829	0.829	0.987	0.804	0.986	0.949	0.829
NB	1.000	1.000	1.000	1.000	1.000	1.000	0.810	1.000	1.000	1.000

Table A.11: *Kappa result for* ***Monks1*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.580	0.016	0.000	0.020	0.155	0.458	0.464	0.566	0.163	0.163
PLSR	0.748	0.977	0.991	1.000	0.995	0.963	0.867	0.843	0.986	0.857
NNMLP	0.672	0.920	0.940	0.964	0.972	0.884	0.812	0.844	0.924	0.872
GLM	0.694	0.935	0.977	0.968	0.991	0.913	0.834	0.777	0.945	0.770
PLMR	0.693	0.940	0.964	0.963	0.973	0.871	0.834	0.781	0.950	0.776
MARS	0.490	0.000	0.000	0.000	0.042	0.529	0.428	0.487	0.334	0.240
KNN	0.472	0.363	0.449	0.396	0.392	0.475	0.604	0.514	0.480	0.421
RF	0.622	0.294	0.000	0.407	0.424	0.575	0.569	0.624	0.340	0.381
RDA	0.530	0.188	0.000	0.188	0.316	0.493	0.584	0.450	0.429	0.284
NB	0.767	0.263	0.455	0.352	0.375	0.861	0.699	0.677	0.561	0.496

Table A.12: *Kappa result for* ***Phoneme*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.941	0.437	0.538	0.455	0.461	0.786	0.744	0.628	0.604	0.465
PLSR	0.942	0.904	0.824	0.976	0.985	0.932	0.938	0.844	0.943	0.888
NNMLP	0.855	0.555	0.930	0.904	0.915	0.942	0.894	0.858	0.859	0.868
GLM	0.814	0.648	0.662	0.804	0.809	0.809	0.803	0.797	0.809	0.813
PLMR	0.905	0.817	0.732	0.952	0.968	0.910	0.872	0.861	0.883	0.868
MARS	0.933	0.442	0.384	0.533	0.509	0.981	0.690	0.678	0.623	0.575
KNN	0.507	0.330	0.370	0.382	0.395	0.426	0.635	0.367	0.536	0.371
RF	0.667	0.677	0.550	0.716	0.712	0.706	0.716	0.620	0.698	0.698
RDA	0.847	0.655	0.570	0.745	0.745	0.874	0.857	0.808	0.809	0.721
NB	0.871	0.761	0.604	0.842	0.846	0.911	0.874	0.735	0.800	0.816

Table A.13: *Kappa result for* ***Tae*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	1.000	1.000	1.000	1.000	1.000	1.000	0.992	1.000	1.000	1.000
PLSR	0.999	0.999	0.999	0.999	0.999	0.999	0.989	0.999	0.999	0.999
NNMLP	0.818	0.897	0.980	0.986	0.992	0.881	0.765	0.882	0.916	0.871
GLM	1.000	1.000	1.000	1.000	1.000	1.000	0.992	1.000	1.000	1.000
PLMR	0.828	0.940	0.962	0.978	0.992	0.891	0.705	0.866	0.936	0.808
MARS	1.000	0.877	0.930	0.930	0.888	1.000	0.845	1.000	0.934	0.851
KNN	0.646	0.745	0.753	0.745	0.745	0.729	0.597	0.712	0.755	0.692
RF	0.884	0.814	0.868	0.888	0.889	0.862	0.835	0.897	0.840	0.814
RDA	1.000	1.000	1.000	1.000	1.000	1.000	0.991	1.000	1.000	1.000
NB	0.672	0.940	0.970	0.993	0.989	0.980	0.664	0.772	0.941	0.807

Table A.14: *Kappa result for* ***Vote*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.941	0.902	0.926	0.911	0.909	0.903	0.616	0.937	0.913	0.902
PLSR	0.756	0.783	0.972	0.969	0.987	0.917	0.649	0.788	0.805	0.776
NNMLP	0.619	0.723	0.736	0.740	0.735	0.725	0.486	0.671	0.722	0.679
GLM	0.746	0.946	0.949	0.960	0.960	0.948	0.552	0.810	0.946	0.923
PLMR	0.830	0.845	0.967	0.981	0.987	0.906	0.594	0.875	0.881	0.896
MARS	0.849	0.877	0.891	0.892	0.892	0.975	0.537	0.864	0.880	0.791
KNN	0.405	0.458	0.424	0.448	0.448	0.520	0.384	0.480	0.503	0.437
RF	0.918	0.882	0.910	0.894	0.895	0.907	0.625	0.922	0.893	0.874
RDA	0.648	0.578	0.652	0.656	0.651	0.678	0.375	0.622	0.719	0.536
NB	0.862	0.931	0.972	0.977	0.983	0.935	0.587	0.904	0.931	0.911

Table A.15: *Kappa result for* ***PhishingWebsites*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.995	0.260	0.210	0.276	0.322	0.949	0.149	0.997	0.304	0.379
PLSR	0.471	0.782	0.789	0.959	0.972	0.812	0.546	0.641	0.889	0.762
NNMLP	0.663	0.708	0.705	0.697	0.697	0.695	0.510	0.689	0.708	0.672
GLM	0.479	0.809	0.695	0.959	0.968	0.824	0.531	0.613	0.886	0.768
PLMR	0.470	0.802	0.704	0.951	0.967	0.780	0.530	0.601	0.883	0.749
MARS	0.870	0.351	0.409	0.424	0.432	0.893	0.170	0.880	0.414	0.433
KNN	0.278	0.302	0.306	0.387	0.399	0.431	0.247	0.399	0.417	0.394
RF	0.935	0.274	0.213	0.281	0.318	0.911	0.140	0.942	0.318	0.365
RDA	0.416	0.520	0.405	0.605	0.606	0.645	0.434	0.592	0.674	0.633
NB	0.537	0.735	0.694	0.745	0.746	0.762	0.491	0.708	0.761	0.749

Table A.16: *Kappa result for* ***Wall-Robot-Navigation*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.743	0.377	0.207	0.375	0.381	0.544	0.225	0.720	0.413	0.379
PLSR	0.370	0.888	0.522	0.971	0.986	0.854	0.421	0.666	0.918	0.835
NNMLP	-0.028	-0.031	-0.027	-0.029	-0.029	-0.027	-0.018	-0.031	-0.030	-0.037
GLM	0.438	0.894	0.546	0.983	0.980	0.882	0.445	0.676	0.931	0.831
PLMR	0.400	0.874	0.595	0.972	0.979	0.867	0.423	0.649	0.916	0.806
MARS	0.355	0.451	0.277	0.461	0.469	0.587	0.238	0.523	0.494	0.469
KNN	0.274	0.583	0.350	0.586	0.584	0.558	0.300	0.475	0.586	0.573
RF	0.537	0.690	0.404	0.696	0.696	0.738	0.359	0.664	0.692	0.672
RDA	0.363	0.900	0.509	0.981	0.984	0.877	0.444	0.665	0.927	0.839
NB	0.406	0.403	0.483	0.644	0.643	0.715	0.262	0.389	0.611	0.571

Table A.17: *Kappa result for* ***Waveform-5000*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.823	0.327	0.335	0.322	0.339	0.598	0.419	0.698	0.481	0.338
PLSR	0.726	0.790	0.833	0.953	0.952	0.809	0.521	0.795	0.836	0.754
NNMLP	0.778	0.878	0.939	0.943	0.939	0.874	0.426	0.830	0.900	0.795
GLM	0.736	0.741	0.551	0.881	0.887	0.786	0.488	0.733	0.783	0.728
PLMR	0.762	0.805	0.504	0.967	0.978	0.804	0.532	0.796	0.834	0.776
MARS	0.868	0.157	0.324	0.281	0.280	0.856	0.301	0.846	0.392	0.330
KNN	0.305	0.157	0.000	0.215	0.220	0.270	0.266	0.318	0.307	0.222
RF	0.563	0.366	0.420	0.411	0.429	0.526	0.429	0.630	0.467	0.419
RDA	0.774	0.821	0.490	0.968	0.973	0.858	0.522	0.799	0.859	0.806
NB	0.524	0.000	0.328	0.502	0.547	0.518	0.279	0.392	0.350	0.337

Table A.18: *Kappa result for* ***Cmc*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.984	0.748	0.870	0.772	0.780	0.938	0.826	0.970	0.862	0.740
PLSR	0.718	0.759	0.919	0.986	0.996	0.820	0.556	0.749	0.898	0.776
NNMLP	0.579	0.501	0.954	0.967	0.971	0.747	0.555	0.586	0.831	0.664
GLM	0.697	0.806	0.970	0.959	0.959	0.855	0.632	0.742	0.883	0.780
PLMR	0.674	0.789	0.962	0.962	0.959	0.848	0.564	0.754	0.909	0.782
MARS	0.886	0.581	0.659	0.558	0.617	0.897	0.668	0.886	0.717	0.572
KNN	0.593	0.682	0.694	0.684	0.684	0.684	0.593	0.676	0.688	0.648
RF	0.663	0.517	0.590	0.610	0.611	0.668	0.460	0.684	0.573	0.497
RDA	0.540	0.141	0.829	0.747	0.757	0.691	0.377	0.376	0.557	0.435
NB	0.696	0.178	0.753	0.742	0.756	0.745	0.484	0.682	0.761	0.725

Table A.19: *Kappa result for* ***Heart-h*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.999	0.999	0.999	0.999	0.999	0.999	0.540	0.999	0.999	0.999
PLSR	0.591	0.413	0.914	0.957	0.946	0.757	0.189	0.714	0.812	0.631
NNMLP	0.125	0.000	0.482	0.423	0.520	0.347	0.042	0.042	0.176	0.232
GLM	0.355	0.000	0.703	0.838	0.774	0.438	0.045	0.285	0.266	0.412
PLMR	0.689	0.056	0.836	0.875	0.868	0.692	0.192	0.661	0.602	0.644
MARS	0.973	0.842	0.942	0.934	0.937	0.986	0.538	0.963	0.934	0.882
KNN	0.589	0.258	0.590	0.606	0.595	0.594	0.224	0.574	0.506	0.533
RF	0.948	0.926	0.935	0.953	0.941	0.932	0.497	0.948	0.928	0.926
RDA	0.524	0.000	0.757	0.754	0.841	0.285	0.000	0.112	0.200	0.409
NB	0.837	0.228	0.849	0.896	0.858	0.858	0.093	0.884	0.726	0.829

Table A.20: *Kappa result for* ***Credit-a*** *dataset*

A.2 Correlation

Correlation Result Tables.

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.943	0.943	0.943	0.943	1.000	0.943	0.829	1.000	1.000	0.886
PLSR	0.943	0.943	0.943	1.000	0.943	0.943	0.486	0.943	1.000	1.000
NNMLP	0.943	0.886	0.943	0.943	0.943	1.000	0.943	0.943	0.943	1.000
GLM	0.943	0.943	0.943	1.000	0.943	1.000	0.714	0.943	1.000	0.886
PLMR	0.943	0.943	0.943	1.000	1.000	0.943	0.771	0.943	1.000	0.943
MARS	0.943	0.829	0.829	0.943	0.943	0.943	0.943	0.943	0.943	0.943
KNN	0.754	0.543	0.314	0.714	0.429	0.771	0.377	0.714	0.543	0.486
RF	1.000	0.943	1.000	0.943	1.000	0.943	0.829	0.943	1.000	0.943
RDA	0.943	1.000	0.943	0.943	0.943	0.943	0.714	0.943	0.943	1.000
NB	0.943	0.943	0.943	0.943	0.943	0.943	0.714	0.943	0.943	0.943

Table A.21: *Spearman Correlation result for* ***Car*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.732	0.499	0.642	0.725	0.597	0.730	0.613	0.868	0.659	0.311
PLSR	0.618	0.521	0.579	0.626	0.571	0.546	0.594	0.582	0.546	0.585
NNMLP	0.834	0.751	0.852	0.847	0.839	0.685	0.715	0.614	0.789	0.500
GLM	0.411	0.450	0.472	0.525	0.472	0.450	0.548	0.243	0.475	0.378
PLMR	0.447	0.584	0.503	0.609	0.556	0.658	0.269	0.423	0.559	0.717
MARS	0.837	0.492	0.824	0.554	0.695	0.738	0.852	0.789	0.702	0.487
KNN	0.349	0.460	0.460	0.529	0.522	0.537	0.230	0.282	0.533	0.526
RF	0.654	0.628	0.702	0.696	0.727	0.779	0.530	0.832	0.736	0.473
RDA	0.254	0.368	0.535	0.357	0.407	0.194	0.513	0.316	0.288	0.429
NB	0.353	0.288	0.389	0.366	0.337	0.298	0.420	0.361	0.573	0.282

Table A.22: *Spearman Correlation result for* ***Heart-c*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	1.000	1.000	1.000	1.000	1.000	1.000	0.745	1.000	1.000	1.000
PLSR	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
NNMLP	1.000	1.000	1.000	1.000	1.000	1.000	0.745	1.000	1.000	1.000
GLM	1.000	1.000	1.000	1.000	1.000	1.000	0.745	1.000	1.000	1.000
PLMR	0.525	0.527	0.527	0.522	0.524	0.524	0.527	0.539	0.532	0.525
MARS	1.000	1.000	1.000	1.000	1.000	1.000	0.745	1.000	1.000	1.000
KNN	0.604	0.474	0.567	0.679	0.557	0.525	0.594	0.647	0.775	0.801
RF	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
RDA	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
NB	0.019	-0.003	0.064	0.032	0.064	0.016	0.064	-0.035	-0.041	0.000

Table A.23: *Spearman Correlation result for* ***Badges2*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.800	0.600	-0.400	0.600	-0.738	0.000	0.800	0.600	0.400	0.400
PLSR	0.600	0.800	1.000	0.800	0.800	0.800	0.800	1.000	0.800	1.000
NNMLP	0.800	1.000	0.800	1.000	1.000	0.800	1.000	1.000	1.000	0.800
GLM	0.800	0.600	0.800	0.800	-0.400	0.105	0.600	0.949	0.949	0.400
PLMR	-0.258	0.775	-0.258	0.258	-0.544	-0.258	0.775	0.258	0.775	0.775
MARS	1.000	1.000	0.632	0.800	1.000	1.000	1.000	1.000	1.000	0.800
KNN	-0.400	-1.000	-0.400	-0.400	-0.400	-0.738	-1.000	-0.738	-0.800	-0.800
RF	0.738	0.600	0.000	-0.400	0.600	-0.600	-0.200	0.600	-0.200	-0.400
RDA	-0.632	-0.056	0.500	-0.211	0.105	-0.105	0.500	-0.389	-0.105	-0.632
NB	-0.400	-0.400	-0.400	-0.400	-0.632	-0.632	0.400	0.600	0.000	-0.200

Table A.24: *Spearman Correlation result for* ***Balance-scale*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	1.000	0.800	1.000	0.800	0.800	1.000	1.000	1.000	0.800	0.800
PLSR	0.400	0.400	0.200	0.200	0.400	0.200	0.200	0.200	0.200	0.400
NNMLP	0.400	1.000	0.949	0.400	0.200	0.400	0.400	0.400	0.632	0.400
GLM	0.400	0.800	0.800	0.200	0.316	0.400	0.200	0.200	0.400	0.800
PLMR	0.400	0.400	0.400	0.200	0.400	0.316	0.800	0.400	0.400	0.400
MARS	0.800	0.800	0.800	0.800	0.800	1.000	1.000	1.000	1.000	0.632
KNN	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.800	0.800
RF	1.000	0.800	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
RDA	0.200	0.400	0.400	0.200	0.949	0.400	0.400	0.200	0.800	0.949
NB	0.400	0.400	0.400	0.400	0.400	0.400	0.400	0.400	0.400	0.200

Table A.25: *Spearman Correlation result for* ***Banknote-Authentication*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.579	0.589	0.599	0.732	0.515	0.429	0.222	0.658	0.710	0.412
PLSR	0.648	0.680	0.733	0.731	0.729	0.699	0.461	0.659	0.713	0.642
NNMLP	0.358	0.430	0.524	0.504	0.489	0.423	0.218	0.173	0.447	0.475
GLM	0.389	0.424	0.402	0.415	0.393	0.365	0.192	0.334	0.394	0.327
PLMR	0.400	0.463	0.552	0.555	0.568	0.505	0.452	0.544	0.468	0.591
MARS	0.729	0.558	0.522	0.310	0.428	0.796	0.623	0.842	0.716	0.741
KNN	0.031	0.193	0.207	0.067	0.202	0.327	0.338	0.067	0.157	0.198
RF	0.701	0.711	0.849	0.723	0.643	0.640	0.540	0.750	0.666	0.703
RDA	0.505	0.352	0.289	0.276	0.316	0.332	0.369	0.202	0.340	0.349
NB	0.498	0.593	0.725	0.733	0.564	0.517	0.308	0.505	0.606	0.676

Table A.26: *Spearman Correlation result for* ***Credit-g*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.904	0.707	0.810	0.337	0.645	0.898	0.635	0.904	0.647	0.455
PLSR	0.755	0.790	0.814	0.790	0.814	0.778	0.830	0.777	0.790	0.794
NNMLP	0.862	0.738	0.762	0.790	0.762	0.762	0.690	0.778	0.762	0.810
GLM	0.826	0.886	0.886	0.886	0.916	0.850	0.611	0.873	0.850	0.850
PLMR	0.826	0.826	0.826	0.826	0.826	0.898	0.587	0.707	0.743	0.850
MARS	0.755	0.789	0.838	0.719	0.614	0.778	0.755	0.753	0.802	0.639
KNN	0.452	0.619	0.452	0.575	0.405	0.714	0.286	0.333	0.476	0.491
RF	0.731	0.755	0.343	0.323	0.551	0.443	0.623	0.620	0.359	0.659
RDA	0.833	0.857	0.786	0.857	0.786	0.857	0.833	0.862	0.810	0.976
NB	0.193	-0.099	0.084	0.048	0.180	0.036	-0.096	0.228	0.323	0.419

Table A.27: *Spearman Correlation result for* ***Diabetes*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.500	-0.500	0.500	0.500	0.500	0.500	0.500	0.500	-0.500	0.500
PLSR	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
NNMLP	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
GLM	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
PLMR	0.866	0.866	0.866	0.866	0.866	0.866	0.866	0.866	0.866	0.866
MARS	1.000	-1.000	0.500	-1.000	-1.000	0.500	0.000	1.000	-0.500	-0.500
KNN	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	1.000	0.500
RF	-0.866	-0.866	-0.866	0.000	-0.866	-0.866	-0.866	-0.866	0.000	-0.866
RDA	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500
NB	0.866	0.866	0.866	0.866	0.866	0.866	0.866	0.866	0.866	0.866

Table A.28: *Spearman Correlation result for* ***Haberman*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.949	1.000	1.000	1.000	1.000	0.949	0.949	0.949	1.000	1.000
PLSR	0.800	0.800	0.400	0.800	0.800	0.800	0.800	0.800	0.800	0.800
NNMLP	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
GLM	0.800	0.600	0.800	0.800	0.800	0.800	0.800	0.800	0.800	0.800
PLMR	1.000	0.800	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
MARS	0.949	0.632	1.000	1.000	0.949	0.949	0.949	0.949	0.949	1.000
KNN	0.316	0.400	0.200	1.000	1.000	0.800	0.800	0.800	1.000	0.400
RF	1.000	0.400	1.000	0.800	0.800	1.000	0.800	1.000	0.800	0.400
RDA	0.949	0.800	0.800	0.800	0.800	0.800	0.800	0.800	1.000	1.000
NB	0.778	0.738	0.738	0.738	0.738	0.738	0.738	0.738	0.738	0.738

Table A.29: *Spearman Correlation result for* ***Iris*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.935	0.880	0.273	0.880	0.880	0.935	0.759	0.935	0.759	0.880
PLSR	0.928	-0.348	0.986	0.986	0.928	0.812	0.721	0.986	0.986	0.956
NNMLP	0.829	0.754	0.771	0.943	0.829	0.986	0.657	0.943	0.829	0.829
GLM	0.899	0.435	0.986	0.899	0.750	0.667	0.899	0.812	0.899	0.754
PLMR	0.829	0.091	0.829	0.771	0.771	0.771	0.943	0.829	0.986	0.943
MARS	0.928	0.015	0.377	0.162	0.638	0.870	0.812	0.928	0.928	-0.203
KNN	0.441	0.224	0.177	0.265	0.265	0.000	0.706	0.177	0.441	0.224
RF	0.772	0.000	0.772	0.617	0.833	0.772	0.926	0.926	0.772	0.892
RDA	0.618	0.545	0.567	0.500	0.567	0.925	0.736	0.618	0.736	0.500
NB	0.794	0.224	0.794	0.706	0.706	0.940	0.794	0.716	0.706	0.706

Table A.30: *Spearman Correlation result for* ***Labor*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.664	0.664	0.664	0.664	0.664	0.664	0.406	0.664	0.664	0.664
PLSR	0.674	0.674	0.674	0.674	0.674	0.674	0.224	0.674	0.674	0.674
NNMLP	0.893	0.406	0.955	0.893	0.893	0.893	0.899	0.928	0.841	0.348
GLM	0.655	0.655	0.655	0.655	0.655	0.655	-0.143	0.655	0.655	0.655
PLMR	0.696	0.696	0.696	0.696	0.696	0.696	0.339	0.696	0.696	0.696
MARS	0.880	0.257	0.829	0.086	0.486	0.941	0.657	0.714	0.986	0.714
KNN	0.986	0.371	-0.486	-0.086	0.257	0.754	1.000	0.829	0.771	-0.543
RF	0.935	-0.030	-0.154	-0.091	0.152	1.000	0.880	1.000	0.941	-0.516
RDA	0.706	0.074	0.754	0.464	-0.232	0.706	0.754	0.580	0.754	0.754
NB	0.655	0.655	0.655	0.655	0.655	0.655	0.736	0.655	0.655	0.655

Table A.31: *Spearman Correlation result for* ***Monks1*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.700	0.800	-0.051	0.700	0.300	0.300	-0.200	0.300	0.300	0.154
PLSR	0.600	0.300	0.564	0.300	0.527	0.564	0.700	0.300	0.900	0.000
NNMLP	0.800	0.800	0.800	0.800	0.800	0.800	0.800	0.800	0.500	0.667
GLM	0.600	0.600	0.600	0.300	0.462	0.300	0.300	-0.200	0.500	0.100
PLMR	0.700	0.700	0.700	0.700	0.400	0.600	0.600	0.500	0.700	0.667
MARS	0.900	-0.154	0.103	-0.738	-0.205	0.800	0.700	0.800	0.700	0.154
KNN	0.359	0.821	0.600	0.975	0.975	0.500	0.500	0.500	0.100	0.700
RF	0.600	0.900	-0.462	0.600	0.359	0.667	0.600	0.800	0.300	0.667
RDA	0.900	-0.051	0.500	-0.051	0.300	0.900	0.900	0.900	0.900	0.900
NB	0.000	-0.200	0.100	0.100	-0.103	0.000	-0.400	-0.564	0.000	-0.400

Table A.32: *Spearman Correlation result for* ***Phoneme*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.100	0.200	0.100	0.600	0.200	0.100	0.100	0.462	0.100	0.300
PLSR	0.700	0.600	0.667	0.700	0.600	0.700	0.700	0.600	0.700	0.700
NNMLP	0.800	0.667	0.900	0.800	0.800	0.800	0.900	1.000	0.900	1.000
GLM	0.500	0.600	0.700	0.500	0.500	0.700	0.500	0.600	0.600	0.667
PLMR	0.700	0.600	0.359	0.700	0.700	0.700	0.700	0.600	0.700	0.600
MARS	0.667	0.564	0.103	0.872	0.789	0.632	0.821	0.821	0.872	0.872
KNN	0.821	-0.344	0.308	0.667	0.410	0.667	0.821	0.564	0.667	0.541
RF	-0.200	-0.200	-0.600	0.100	0.100	-0.200	-0.200	0.300	-0.200	0.100
RDA	0.526	0.462	-0.308	0.616	0.616	0.975	0.564	0.616	0.462	0.462
NB	0.600	0.600	0.000	0.700	0.600	0.600	0.600	0.600	0.600	0.600

Table A.33: *Spearman Correlation result for* ***Tae*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.431	0.431	0.431	0.431	0.431	0.431	0.427	0.431	0.431	0.431
PLSR	0.461	-0.046	0.080	0.235	0.068	0.172	0.072	0.329	-0.200	0.273
NNMLP	0.755	0.967	0.929	0.934	0.896	0.919	0.860	0.894	0.950	0.955
GLM	0.428	0.428	0.428	0.428	0.428	0.428	-0.178	0.428	0.428	0.428
PLMR	0.827	0.947	0.935	0.950	0.952	0.795	0.806	0.859	0.950	0.881
MARS	0.839	0.747	0.671	0.681	0.716	0.839	0.694	0.835	0.589	0.509
KNN	0.736	0.689	0.721	0.719	0.579	0.703	0.673	0.701	0.774	0.727
RF	0.812	0.446	0.631	0.705	0.743	0.572	0.672	0.754	0.773	0.265
RDA	0.441	0.441	0.441	0.441	0.441	0.441	0.216	0.441	0.441	0.441
NB	0.337	0.297	0.246	0.254	0.228	0.194	0.297	0.317	0.292	0.296

Table A.34: *Spearman Correlation result for* ***Vote*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.392	0.125	0.256	0.007	0.432	0.143	0.307	0.584	0.166	0.199
PLSR	0.619	0.006	0.864	0.839	0.817	0.827	0.723	0.807	0.382	0.435
NNMLP	0.707	0.825	0.822	0.838	0.835	0.808	0.553	0.801	0.835	0.784
GLM	0.799	0.834	0.835	0.851	0.857	0.823	0.498	0.849	0.860	0.916
PLMR	0.903	0.554	0.898	0.909	0.906	0.832	0.342	0.806	0.855	0.784
MARS	0.702	0.673	0.797	0.799	0.669	0.671	0.613	0.814	0.717	0.731
KNN	0.574	0.578	0.507	0.544	0.393	0.329	0.410	0.662	0.597	0.598
RF	0.617	0.172	0.553	0.080	0.518	0.544	0.315	0.716	0.427	0.437
RDA	0.682	0.527	0.697	0.627	0.407	0.513	0.638	0.740	0.680	0.285
NB	0.383	0.450	0.664	0.654	0.659	0.691	0.233	0.439	0.630	0.650

Table A.35: *Spearman Correlation result for* ***PhishingWebsites*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.672	0.679	0.411	0.607	0.381	0.494	0.399	0.772	0.694	0.558
PLSR	0.442	0.384	0.365	0.505	0.494	0.521	0.153	0.394	0.541	0.570
NNMLP	0.384	0.748	0.701	0.664	0.756	0.315	0.100	0.573	0.558	0.627
GLM	0.426	0.264	0.050	0.471	0.473	0.497	0.171	0.397	0.416	0.523
PLMR	0.562	0.361	0.177	0.492	0.525	0.581	0.315	0.513	0.490	0.521
MARS	0.602	0.461	0.174	0.415	0.181	0.676	0.124	0.638	0.451	0.399
KNN	0.172	0.245	0.070	0.088	0.060	0.029	0.151	0.122	0.245	0.056
RF	0.731	0.625	0.628	0.642	0.479	0.738	0.769	0.781	0.393	0.572
RDA	0.498	0.190	-0.002	0.594	0.422	0.277	0.341	0.336	0.370	0.411
NB	0.456	0.187	-0.013	0.491	0.442	0.464	0.452	0.622	0.335	0.374

Table A.36: *Spearman Correlation result for* ***Wall-Robot-Navigation*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.815	0.538	0.420	0.690	0.493	0.632	0.560	0.801	0.577	0.639
PLSR	0.729	0.791	0.705	0.791	0.775	0.782	0.747	0.823	0.777	0.818
NNMLP	0.656	0.695	0.734	0.717	0.708	0.562	0.642	0.691	0.821	0.484
GLM	0.782	0.717	0.767	0.684	0.711	0.715	0.742	0.793	0.687	0.737
PLMR	0.755	0.765	0.662	0.829	0.856	0.793	0.699	0.795	0.840	0.775
MARS	0.682	0.531	0.354	0.486	0.616	0.661	0.500	0.677	0.573	0.679
KNN	0.783	0.846	0.734	0.773	0.719	0.810	0.677	0.798	0.757	0.685
RF	0.859	0.784	0.547	0.794	0.755	0.793	0.560	0.834	0.766	0.766
RDA	0.760	0.687	0.611	0.709	0.715	0.717	0.646	0.809	0.705	0.707
NB	0.703	0.591	0.193	0.649	0.612	0.600	0.387	0.643	0.612	0.508

Table A.37: *Spearman Correlation result for* ***Waveform-5000*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.417	0.450	-0.176	0.450	0.400	0.450	0.150	0.333	0.433	0.250
PLSR	0.904	0.711	0.795	0.921	0.937	0.921	0.569	0.904	0.879	0.937
NNMLP	0.833	0.962	0.917	0.933	0.870	0.695	0.609	0.824	0.862	0.243
GLM	0.745	0.895	0.828	0.929	0.912	0.912	0.845	0.912	0.870	0.803
PLMR	0.750	0.795	0.867	0.845	0.778	0.683	0.661	0.728	0.733	0.733
MARS	0.917	0.067	0.403	0.333	0.233	0.857	0.385	0.883	0.452	0.333
KNN	0.417	0.202	-0.402	0.033	0.433	0.217	0.600	0.217	-0.075	-0.008
RF	0.533	0.283	0.417	0.502	0.433	0.450	0.383	0.600	0.617	0.378
RDA	0.667	0.617	0.343	0.667	0.653	0.517	0.633	0.717	0.617	0.517
NB	0.633	0.176	0.317	0.117	0.233	0.567	0.243	0.550	0.283	0.561

Table A.38: *Spearman Correlation result for* ***Cmc*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.874	0.524	0.756	0.793	0.832	0.710	0.669	0.856	0.848	0.657
PLSR	0.618	0.726	0.695	0.726	0.726	0.738	0.616	0.610	0.713	0.646
NNMLP	0.607	0.468	0.708	0.632	0.669	0.615	0.644	0.522	0.656	0.362
GLM	0.330	0.324	0.385	0.434	0.399	0.330	0.434	0.232	0.385	0.263
PLMR	0.691	0.822	0.838	0.823	0.823	0.799	0.732	0.783	0.835	0.801
MARS	0.811	0.705	0.742	0.796	0.662	0.782	0.829	0.729	0.726	0.657
KNN	0.665	0.811	0.787	0.774	0.774	0.783	0.462	0.701	0.774	0.768
RF	0.799	0.710	0.714	0.566	0.677	0.585	0.756	0.821	0.831	0.800
RDA	0.654	-0.285	0.747	0.747	0.661	0.766	0.827	0.594	0.796	0.766
NB	0.630	0.153	0.520	0.580	0.734	0.519	0.361	0.560	0.468	0.634

Table A.39: *Spearman Correlation result for* ***Heart-h*** *dataset*

	C50	PLSR	NNMLP	GLM	PLMR	MARS	KNN	RF	RDA	NB
C50	0.037	0.364	0.207	0.484	0.036	0.156	0.102	0.004	0.633	0.229
PLSR	0.367	0.642	0.270	0.261	0.220	0.402	0.221	0.391	0.328	0.383
NNMLP	0.117	0.094	0.305	0.338	0.232	0.258	0.165	0.194	0.417	0.418
GLM	0.054	0.147	0.529	0.566	0.541	0.473	0.400	0.275	0.317	0.442
PLMR	0.039	0.660	0.451	0.360	0.367	0.127	0.775	0.041	0.569	0.188
MARS	0.385	0.164	0.200	0.245	0.112	0.370	-0.032	0.653	0.290	0.084
KNN	-0.288	-0.102	-0.066	0.020	-0.004	-0.348	-0.239	-0.351	-0.106	-0.112
RF	0.384	0.041	0.352	0.248	0.251	0.386	0.339	0.605	0.387	-0.175
RDA	0.074	-0.068	0.512	0.526	0.511	0.313	0.577	0.357	0.284	0.246
NB	0.273	0.005	0.285	0.336	0.486	0.300	-0.077	0.506	0.348	0.427

Table A.40: *Spearman Correlation result for* ***Credit-a*** *dataset*

Appendix B

Source Code Used for Experiments

B.1 Source code for Learning Surrogate Model

This code of learning surrogate model with the modification that we did to prepare the surrogate model for extracting the feature importance

```
library(caret)
library("mlr")
library("iml")
options(warn=0)

load_install <- function(lib_name)
{
  if ((lib_name %in% installed.packages()) == FALSE)
  {
    install.packages(lib_name)
  }
  suppressPackageStartupMessages(require(lib_name, character.only = TRUE, quietly = TRUE))
}
runmethods <-function (mymethod, train, test)
{

  # For testing
  # mymethod = om
  # train = Train
  # test = GridTrain

  ERROR <- FALSE
  model <- NA
  preds <- list()
  preds$error <- NULL
  preds$fail <- FALSE
  # print(paste("Dataset: ",datasets[selected],", Method: ", mymethod,", fold: ",ik))



#################### For run methods in Caret
#########################
###  Decision Trees  ###
#########################


set.fail <- function(preds){
  print(paste("Model creation failed for ",mymethod))
  #preds$error <- e
  preds$fail <- TRUE
  preds$crisp <- rep(FALSE, nrow(test))
  preds$probs <- rep(FALSE, nrow(test))
  preds
}

if (mymethod=="c5.0") {
  load_install("C50")
  preProc <- preProcess(train, method=c("center", "scale"))
  train <- predict(preProc, train)
  test <- predict(preProc, test)
  ERROR <-  tryCatch(model <- C5.0(Class ~., data = train), error = function(e) { print(e);return(TRUE)})

  if (is.logical(ERROR)){
    preds <- set.fail(preds)
```

```
  }else{
    preds$crisp <- tryCatch(predict(model, newdata = test, type="class"), error = function(e) {return(rep(FALSE, nrow(
        test)))})
    preds$probs <- tryCatch(predict(model, newdata = test, type="prob"), error = function(e) {return(rep(FALSE, nrow(
        test)))})
    # tree = rpart(Class ~ ., data= train)
    # y = train$Class
    # X = train[-which(names(train) == "Class")]
    # mod1 = Predictor$new(tree, data = X, y = y, type = "prob")
    # preds_mod <-tryCatch(mod1, error = function(e) {return(rep(FALSE, nrow(test)))})

  }
  tryCatch(detach("package:C50", unload=TRUE), error = function(e){print(e)})
}

if (mymethod == "gcvEarth_d3") { #135
    load_install("earth")
    ERROR <-  tryCatch(model <- caret::train(Class ~ ., data = train, preProc = c("center", "scale"), method = "
        gcvEarth",tuneGrid=data.frame(degree = 3),  trControl = trainControl(method="none", classProbs = TRUE)), error
         = function(e) {return(TRUE)})

    if (is.logical(ERROR)){
      preds <- set.fail(preds)
    }else{
      preds$crisp <- tryCatch(predict(model, newdata = test, type="raw"), error = function(e) {return(rep(FALSE, nrow(
          test)))})
      preds$probs <- tryCatch(predict(model, newdata = test, type="prob"), error = function(e) {return(rep(FALSE, nrow(
          test)))})
      # tree = rpart(Class ~ ., data= train)
      # y = train$Class
      # X = train[-which(names(train) == "Class")]
      # mod1 = Predictor$new(tree, data = X, y = y, type = "prob")
      # preds_mod <-tryCatch(mod1, error = function(e) {return(rep(FALSE, nrow(test)))})
    }
    tryCatch(detach("package:earth", unload=TRUE), error = function(e){print(e)})
  }
  #Model: Penalized Multinomial Regression
  # Method: multinom
    if (mymethod == "multinom") {
              load_install("nnet")
     ERROR <-  tryCatch(model <- caret::train(Class ~ ., data = train, preProc = c("center", "scale"), method = "
        multinom", trControl = trainControl(method="none", classProbs=TRUE)), error = function(e) {print(e); return(
        TRUE)})

      if (is.logical(ERROR)){
        preds <- set.fail(preds)
      } else {
        preds$crisp <- tryCatch(predict(model, newdata = test, type="raw"), error = function(e) {return(rep(FALSE, nrow(
            test)))})
        preds$probs <- tryCatch(predict(model, newdata = test, type="prob"), error = function(e) {return(rep(FALSE, nrow
            (test)))})
        # tree = rpart(Class ~ ., data= train)
        # y = train$Class
        # X = train[-which(names(train) == "Class")]
        # mod1 = Predictor$new(tree, data = X, y = y, type = "prob")
        # preds_mod <-tryCatch(mod1, error = function(e) {return(rep(FALSE, nrow(test)))})
      }
                  tryCatch(detach("package:nnet", unload = TRUE), error = function(e){print(e)})
    }

 if (mymethod == "simpls_ncomp4") {
   load_install("pls")
   ERROR <-  tryCatch(model <- caret::train(Class ~ ., data = train, preProc = c("center", "scale"), method = "simpls",
        tuneGrid=data.frame(ncomp = 4),  trControl = trainControl(method="none", classProbs = TRUE)), error = function(e
        ) {print(e);return(TRUE)})

   if (is.logical(ERROR)){
    # preds <- set.fail(preds)
   }else{
     preds$crisp <- tryCatch(predict(model, newdata = test, type="raw"), error = function(e) {return(rep(FALSE, nrow(
         test)))})
     preds$probs <- tryCatch(predict(model, newdata = test, type="prob"), error = function(e) {return(rep(FALSE, nrow(
         test)))})
     # tree = rpart(Class ~ ., data= train)
     # y = train$Class
     # X = train[-which(names(train) == "Class")]
     # mod1 = Predictor$new(tree, data = X, y = y, type = "prob")
     # preds_mod <-tryCatch(mod1, error = function(e) {return(rep(FALSE, nrow(test)))})
   }
   tryCatch(detach("package:pls", unload=TRUE), error = function(e){print(e)})
 }

```

```
123
124 if (mymethod == "glmnet") {
125   load_install("glmnet")
126   ######### drop classes of 1 or zero occurences
127   occurences<-table(unlist(train$Class))
128   #classesToDrop <- c()
129   for(iClass in names(occurences))
130   {
131
132     classCount <- occurences[iClass]
133     if(classCount <= 1 )
134      {
135          #classesToDrop <- c(classesToDrop,iClass)
136          train <- train[train$Class != iClass,]
137          warning(c("with Model : ",mymethod, ", Droping Class which have less than or equal one item in the train
                dataset : class = ",iClass))
138      }
139   }
140   train$Class <- droplevels(train$Class)
141   ##################
142
143   ERROR <-  tryCatch(model <- caret::train(Class ~ ., data = train, preProc = c("center", "scale"), method = "glmnet",
          trControl = trainControl(classProbs=TRUE)), error = function(e) {print(e);return(TRUE)})
144
145   if (is.logical(ERROR)){
146     preds <- set.fail(preds)
147   }else{
148     preds$crisp <- tryCatch(predict(model, newdata = test, type="raw"), error = function(e) {print(e); return(rep(FALSE
          , nrow(test)))})
149     preds$probs <- tryCatch(predict(model, newdata = test, type="prob"), error = function(e) {print(e); return(rep(
          FALSE, nrow(test)))})
150     # tree = rpart(Class ~ ., data= train)
151     # y = train$Class
152     # X = train[-which(names(train) == "Class")]
153     # mod1 = Predictor$new(tree, data = X, y = y, type = "prob")
154     # preds_mod <-tryCatch(mod1, error = function(e) {return(rep(FALSE, nrow(test)))})
155   }
156   tryCatch(detach("package:glmnet", unload = TRUE), error = function(e){print(e)})
157 }
158 if (mymethod == "knn_k5") {
159   ERROR <-  tryCatch(model <- caret::train(Class ~ ., data = train, preProc = c("center", "scale"), method = "knn",
          tuneGrid=data.frame(k = 5),  trControl = trainControl(method="none", classProbs = TRUE)), error = function(e) {
          return(TRUE)})
160
161   if (is.logical(ERROR)){
162     preds <- set.fail(preds)
163   }else{
164     preds$crisp<- tryCatch(predict(model, newdata = test, type="raw"), error = function(e) {return(rep(FALSE, nrow(test
          ))})
165     preds$probs<- tryCatch(predict(model, newdata = test, type="prob"), error = function(e) {return(rep(FALSE, nrow(
          test)))})
166     # tree = rpart(Class ~ ., data= train)
167     # y = train$Class
168     # X = train[-which(names(train) == "Class")]
169     # mod1 = Predictor$new(tree, data = X, y = y, type = "prob")
170     # preds_mod <-tryCatch(mod1, error = function(e) {return(rep(FALSE, nrow(test)))})
171   }
172 }
173
174  if (mymethod == "NB_laplace") {
175    load_install("naivebayes")
176    preProc <- preProcess(train, method=c("center", "scale"))
177    train <- predict(preProc, train)
178    test <- predict(preProc, test)
179    ERROR <-  tryCatch(model <- naive_bayes(train[,-ncol(train)], train[,ncol(train)], laplace = 3), error = function(e)
          {print(e);return(TRUE)})
180    #ERROR <-  tryCatch(model <- caret::train(Class ~ ., data = train, method = "naive_bayes", tuneGrid=data.frame(
          laplace=3, usekernel=NA, adjust=NA)), error = function(e) {print(e);return(TRUE)})
181
182    if (is.logical(ERROR)){
183      preds <- set.fail(preds)
184    }else{
185      preds$crisp<- tryCatch(predict(model, test, type = 'class'), error = function(e) {return(rep(FALSE, nrow(test)))})
186      preds$probs<- tryCatch(predict(model, test , type = 'prob'), error = function(e) {return(rep(FALSE, nrow(test)))})
187
188      # tree = rpart(Class ~ ., data= train)
189      # y = train$Class
190      # X = train[-which(names(train) == "Class")]
191      # mod1 = Predictor$new(tree, data = X, y = y, type = "prob")
192      # preds_mod <-tryCatch(mod1, error = function(e) {return(rep(FALSE, nrow(test)))})
193    }
194    tryCatch(detach("package:naivebayes", unload=TRUE), error = function(e){print(e)})
```

```
  }

if (mymethod == "mlpML") {
  load_install("RSNNS")

  ERROR <-  tryCatch(model <- caret::train(Class ~ ., data = train, preProc =  c("center", "scale"),method = "mlpML",
       trControl = trainControl(method="none", classProbs=TRUE)), error = function(e) {return(TRUE)})

  if (is.logical(ERROR)){
    preds <- set.fail(preds)
  }else{
    preds$crisp <- tryCatch(predict(model, newdata = test, type = "raw"), error = function(e) {return(rep(FALSE, nrow(
         test)))})
    preds$probs <- tryCatch(predict(model, newdata = test, type = "prob"), error = function(e) {return(rep(FALSE, nrow(
         test)))})
    # tree = rpart(Class ~ ., data= train)
    # y = train$Class
    # X = train[-which(names(train) == "Class")]
    # mod1 = Predictor$new(tree, data = X, y = y, type = "prob")
    # preds_mod <-tryCatch(mod1, error = function(e) {return(rep(FALSE, nrow(test)))})
  }
  tryCatch(detach("package:RSNNS", unload = TRUE), error = function(e){print(e)})
}
########## there is a problem in this model
if (mymethod == "svmRadialCost_C2_-5") {
  load_install("kernlab")
  ERROR <-  tryCatch(model <- caret::train(Class ~ ., data = train, preProc = c("center", "scale"), method = "
       svmRadialCost", tuneGrid=data.frame(C=2^(-5)), trControl = trainControl(method="none" , classProbs = TRUE)),
       error = function(e) {print(e);  return(TRUE)})
  if (is.logical(ERROR)){
    preds <- set.fail(preds)
  }else{
    preds$crisp <- tryCatch(predict(model, newdata = test, type="raw"), error = function(e) {return(rep(FALSE, nrow(test
         ))})
    preds$probs <- tryCatch(predict(model, newdata = test, type="prob"), error = function(e) {return(rep(FALSE, nrow(
         test)))})

    ###### in this model doesn't have preds$probs and gives me error because we ask it in the end of the function
         runmethods so I add
    #### it in the line below ##########
    #preds$probs<- tryCatch(predict(model, newdata = test, type="prob"), error = function(e) {return(rep(FALSE, nrow(
         test)))})

     preds_mod <-tryCatch(mod1, error = function(e) {return(rep(FALSE, nrow(test)))}) ##############################
          ************
  }
  tryCatch(detach("package:kernlab", unload=TRUE), error = function(e){print(e)})
}
  ###################################
  # if (mymethod == "svmRadialCost_C2_-3") {
  #   load_install("kernlab")
  #   ERROR <-  tryCatch(model <- caret::train(Class ~ ., data = train, preProc = c("center", "scale"), method = "
       svmRadialCost", tuneGrid=data.frame(C=2^(-3)), trControl = trainControl(method="none")), error = function(e) {
       return(TRUE)})
  #   if (is.logical(ERROR)){
  #     preds <- set.fail(preds)
  #   }else{
  #     preds<- tryCatch(predict(model, newdata = test, type="prob"), error = function(e) {return(rep(FALSE, nrow(test))
       )})
  #   }
  #   tryCatch(detach("package:kernlab", unload=TRUE), error = function(e){print(e)})
  # }
  ########################################
  if (mymethod == "rf") {
    load_install("randomForest")
    ERROR <-  tryCatch(model <- caret::train(Class ~ ., data = train, preProc = c("center", "scale"), method = "rf",
         trControl = trainControl(method="none")), error = function(e) {return(TRUE)})

    if (is.logical(ERROR)){
      preds <- set.fail(preds)
    }else{
      preds$crisp<- tryCatch(predict(model, newdata = test, type="raw"), error = function(e) {return(rep(FALSE, nrow(
           test)))})
      preds$probs<- tryCatch(predict(model, newdata = test, type="prob"), error = function(e) {return(rep(FALSE, nrow(
           test)))})
      # tree = rpart(Class ~ ., data= train)
      # y = train$Class
      # X = train[-which(names(train) == "Class")]
      # mod1 = Predictor$new(tree, data = X, y = y, type = "prob")
      # preds_mod <-tryCatch(mod1, error = function(e) {return(rep(FALSE, nrow(test)))})
    }
    tryCatch(detach("package:randomForest", unload=TRUE), error = function(e){print(e)})
```

```
  }


  ## this is a duplicate
  if (mymethod == "simpls_ncomp4") {
    load_install("pls")
    ERROR <-  tryCatch(model <- caret::train(Class ~ ., data = train, preProc = c("center", "scale"), method = "simpls",
         tuneGrid=data.frame(ncomp = 4),  trControl = trainControl(method="none", classProbs = TRUE)), error = function(
         e) { print(e);return(TRUE)})

    if (is.logical(ERROR)){
      # preds <- set.fail(preds)
      preds <- set.fail(preds)
    }else{
      preds$crisp <- tryCatch(predict(model, newdata = test, type="raw"), error = function(e) {return(rep(FALSE, nrow(
           test)))})
      preds$probs <- tryCatch(predict(model, newdata = test, type="prob"), error = function(e) {return(rep(FALSE, nrow(
           test)))})
      # tree = rpart(Class ~ ., data= train)
      # y = train$Class
      # X = train[-which(names(train) == "Class")]
      # mod1 = Predictor$new(tree, data = X, y = y, type = "prob")
      # preds_mod <-tryCatch(mod1, error = function(e) {return(rep(FALSE, nrow(test)))})
    }
    tryCatch(detach("package:pls", unload=TRUE), error = function(e){print(e)})
  }
  if (mymethod == "rda") {
    load_install("klaR")
    ERROR <-  tryCatch(model <- caret::train(Class ~ ., data = train, preProc = c("center", "scale"), method = "rda"),
         error = function(e) {print(e); return(TRUE)})

    if (is.logical(ERROR)){
      preds <- set.fail(preds)
    } else {
      preds$crisp <- tryCatch(predict(model, newdata = test, type="raw"), error = function(e) {return(rep(FALSE, nrow(
           test)))})
      preds$probs <- tryCatch(predict(model, newdata = test, type="prob"), error = function(e) {return(rep(FALSE, nrow(
           test)))})
      # tree = rpart(Class ~ ., data= train)
      # y = train$Class
      # X = train[-which(names(train) == "Class")]
      # mod1 = Predictor$new(tree, data = X, y = y, type = "prob")
      # preds_mod <-tryCatch(mod1, error = function(e) {return(rep(FALSE, nrow(test)))})
    }
    tryCatch(detach("package:klaR", unload = TRUE), error = function(e){print(e)})
  }


  preds_mod <- NULL
  if(!preds$fail) {

    ## it could happen to the return does not throw an exception
    ## check if preds$crisp is logical
    if(class(preds$crisp) == "logical" && class(preds$probs) == "logical")
    {
      # preds$fail <- true
      preds <- set.fail(preds)
    }
    else {
      preds$probs <-  (t(apply(preds$probs, 1, function(x){x/sum(x)})))
      #tree = rpart(Class ~ ., data= train)
      y = train$Class
      X = train[-which(names(train) == "Class")]
      preds_mod <-tryCatch(Predictor$new(model, data = X, y = y, type = "prob"), error = function(e) {return(rep(FALSE,
           nrow(test)))})
    }

  }



  output <- list()
  output$preds_mod <-preds_mod
  output$preds <- preds
  return(output)
}
```

```
##############################################
### Artificial dataset generation functions ###
##############################################

### Generate random dataset follwing uniform distribution at feature level
```

```
# data: dataset
# n: number of artificial examples to generate
# returns: ntest artificial dataset
generateUnifTest<-function (data,n=1000)
{
  k<-n
  numcols<-ncol(data)-1
  ntest<-data.frame()
  # Iterate each feature f
  for (f in (1:numcols))
  {
    # Check the type of f, to generate the new examples
    if (!is.factor(data[,f]))
    {
      if (!is.integer(data[,f]))
      { # Generation of numeric (real) features
        # Generate at random between the minimum and maximum values of feature f
        if (f==1)
        {
          ntest<-data.frame("kk"=runif(k,min(data[,f]),max(data[,f])))
          names(ntest)[1]<-names(data)[f]
        }
        else {
          ntest[names(data)[f]] <- runif(k,min(data[,f]),max(data[,f]))
        }
      } else {
        # Generation of numeric (integer) features
        # Generate at random between the minimum and maximum values of feature f
        if (f==1)
        {
          ntest<-data.frame("kk"=round(runif(k,min(data[,f]),max(data[,f]))))
          names(ntest)[1]<-names(data)[f]
        }
        else {
          ntest[names(data)[f]]<-round(runif(k,min(data[,f]),max(data[,f])))
        }
      }
    }
    else
    { # Generation of factor (nominal) features
      # Generate at random between all the possible values that feature f may take
      if (f==1)
      {
        ntest<-data.frame("kk"=sample (levels(data[,f]),k, replace=TRUE))
        names(ntest)[1]<-names(data)[f]
      }
      else {
        ntest[[names(data)[f]]]<-sample (levels(data[,f]),k, replace=TRUE)
      }
      ntest[[names(data)[f]]]<-as.factor( ntest[[names(data)[f]]])
      levels( ntest[[names(data)[f]]])<-levels(data[,f])
    }
  }
  ntest
}

### Generate random dataset assuming normal distribution for numerical attributes
### and multinomial bernoulli for nominal attributes (i.e., sample from the given dataset).
# data: dataset
# n: number of artificial examples to generate
# returns: ntest artificial dataset
generateNormalTest<-function (data,n=1000)
{
  # data <- Train
  # n <- nrow(Train) * (ncol(datos)-1)

  k<-n
  numcols<-ncol(data)-1
  ntest<-data.frame()
  # Iterate each feature f
  for (f in (1:numcols))
  {
    # Check the type of f, to generate the new examples
    if (!is.factor(data[,f]))
    {
      if (!is.integer(data[,f]))
      { # Generation of numeric (real) features
        # Generate random number from the normal distribution inferred from the dataset
        if (f==1)
        {
          ntest<-data.frame("kk"=as.numeric(rnorm(k,mean = mean(data[,f]),sd = sd(data[,f]))))
          names(ntest)[1]<-names(data)[f]
```

```
        }
        else {
          ntest[names(data)[f]] <- as.numeric(rnorm(k, mean = mean(data[,f]),sd = sd(data[,f])))
        }
      } else {
        # Generation of numeric (integer) features
        # Generate random number from the normal distribution inferred from the dataset
        if (f==1)
        {
          ntest<-data.frame("kk"= as.integer(round(rnorm(k,mean = mean(data[,f]),sd = sd(data[,f])))))
          names(ntest)[1]<-names(data)[f]
        }
        else {
          ntest[names(data)[f]]<-as.integer(round(rnorm(k, mean = mean(data[,f]),sd = sd(data[,f]))))
        }
      }
    }
    else
    { # Generation of factor (nominal) features
      # Sample the values from the dataset, following its same distribution
      if (f==1)
      {
        ntest<-data.frame("kk"=sample (data[,f],k, replace=TRUE))
        names(ntest)[1]<-names(data)[f]
      }
      else {
        ntest[[names(data)[f]]]<-sample (data[,f],k, replace=TRUE)
      }
      ntest[[names(data)[f]]]<-as.factor( ntest[[names(data)[f]]])
      levels( ntest[[names(data)[f]]])<-levels(data[,f])
    }
  }
  ntest
}

# Load dataset in CSV format
load_dataset <- function(filename) {
  # Load dataset and column type
  dataset <- read.table(filename, sep=",", header=TRUE, check.names=FALSE)
  col_types <- read.table(gsub(".csv","_coltypes.csv", filename))

  # Assign column types to dataset
  for (c in 1:nrow(col_types)) {
    if (col_types[c,1] == "factor") {
      dataset[,c] <- as.factor(dataset[,c])
    }
    else if (col_types[c,1] == "integer")
    {
      dataset[,c] <- as.integer(dataset[,c])
    }
    else {
      dataset[,c] <- as.numeric(dataset[,c])
    }
  }
  # Class as factor
  dataset[,ncol(dataset)] <- as.factor(dataset[,ncol(dataset)])
  dataset$Class <- correct_class_names(dataset$Class)
  return(dataset)
}

# Esto es necesario para cuando se trabaja con probabilidades,
# si no, se puede ignorar
correct_class_names <- function(original_names) {
  return(
    as.factor(sapply(original_names, function(x) {
      if (x == '+') return('PLUS')
      else if (x == '-') return('MINUS')
      else return(make.names(x))
    }))
  )
}
```

B.2 Source code for extracting feature importance

This the code for extract feature importance and our experiment

```
setwd("D:/Master/FinalProject TFM/codigo_mjose")
source("utils.R")
```

```r
source("runmethodeswith11model.R")
library("mlr")
library(compareDF)

offscreenOutputFile <- file("all.output.Rout", open = "wt")
offscreenOutput = FALSE
VERBOSE = TRUE
print.msg <- function(msg){
  if(VERBOSE) {
    cat(msg,"\n")
  }
}

################################################
calc.corr <- function(om_imp,sm_imp){
  sm.featureImpValue <- sm_imp$results[order(sm_imp$results$feature),]$importance
  om.featureImpValue <- om_imp$results[order(om_imp$results$feature),]$importance

  correlation.spearman <- cor.test(sm.featureImpValue,om.featureImpValue,method = "spearman")
  correlation.kendall <- cor.test(sm.featureImpValue,om.featureImpValue,method = "kendall")
  correlation.pearson <- cor.test(sm.featureImpValue,om.featureImpValue,method = "pearson")

  list(spearman = correlation.spearman, kendall = correlation.kendall, pearson = correlation.pearson)
}


init.table <- function( row.list,clm.list , init.value = 0) {

  table_of_correlation<- matrix(c(init.value),ncol = length( clm.list ),nrow=length(row.list), byrow = TRUE)
  colnames(table_of_correlation)<-clm.list
  rownames(table_of_correlation)<-row.list
  table_of_correlation<-as.table(table_of_correlation)
  return (table_of_correlation)

}

## known Issues
## dataset credit-a.cs, om=svmRadialCost_C2_-5
### dataset=liver-disorders.csv,om=svmRadialCost_C2_-5
## the input of GridTrain only contain one class, causing the classier to fail learning
## cuase : GridTrain<-generateUnifTest(Train, 100 * (ncol(datos)-1)) >> does not generate balanced dataset between
      classes
######
## NOt Now TODO:: modeify generateUnifTest to ranomly generated datasets taking into acount clases balance
######
## TODO:: modify run methods to return NULL in fail cases and all proceding methods to check if NULL to put a kappa of 0
       and correlation of 0

calc.imp <- function(models,om,ds, Train, GridTrain,GridTest) {
  print.msg(  paste( "Working with Oracle Model : ",om))

  # these lists for storage the surgoate model and in every iteration we created again and fill withe the surgate model
  #list for storage the kapa of surogate model
  list_kapa_per_om <- list()
  #this list for storage the feature importance of the surogate models
  list_imp_per_om <-list()
  list_conf_mat <- list()

  # Aprendemos el oraculo y etiquetamos el dataset surrogado de training
  set.seed(3)
  if (offscreenOutput) sink(offscreenOutputFile, type = c("output", "message"));
  output <- runmethods(om, Train, GridTrain)
  sink(type = c("output", "message"));

  om.imp =FeatureImp$new(output$preds_mod,loss="ce")


  preds<-output$preds
  # Reestablecemos los levels por si se han perdido clases
  GridTrain$Class <- factor(preds$crisp, levels=unique(preds$crisp))

  # Etiquetamos el dataset surrogado de test con el oraculo
  output <- runmethods(om, Train, GridTest)
  oracle_test_preds<-output$preds
  # Aprendizaje de los surrogados

  ## Error
  ## with om=gcvEarth_d3, sm = rf and ds=banknote-authentication.csv
  # Error: $ operator is invalid for atomic vectors

  for (sm in models) {
    print.msg(  paste( "\t Working surrogate Model : ",sm))
```

```
    # if (sm == om)
    #  next
    ## next is like contiue mean skip the following code
    ## it is here to skip comparing same model with it selt if om == sm
    ## so it should be after if



    if (offscreenOutput) sink(offscreenOutputFile, type = c("output", "message"));
    output <- runmethods(sm,GridTrain, GridTest)
    sink(type = c("output", "message"));
    surrogate_test_preds<-output$preds
    surrogate_train_model<-output$preds_mod
    # A partir de aqui se puede comparar la salida del oraculo (oracle_test_preds$crisp)
    # con la del surrogado (surrogate_test_preds$crisp)
    # Por ejemplo:
    imp <- FALSE
    conf_mat <- FALSE
    kappa <- 0
    if(!surrogate_test_preds$fail) {
      conf_mat <- caret::confusionMatrix(reference=oracle_test_preds$crisp,
                                         data=surrogate_test_preds$crisp,
                                         mode="everything")
      kappa <- conf_mat$overall["Kappa"]



      # ds.models.kappa.table[om,sm] <- kappa

      imp = FeatureImp$new(surrogate_train_model,loss="ce")
    ## error is here due to NA in  surrogate_test_preds for model sm = rda and om= knn_k5 and ds=balance-scale.csv this
        seems to be fine with the current seed
    }
    # Error in checkPrediction(prediction, newdata) :
    #   Assertion on 'prediction' failed: Contains missing values (column 'R', row 1).
    list_imp_per_om[[sm]]<-imp
    list_conf_mat[[sm]] <- conf_mat
    list_kapa_per_om[[sm]] <- kappa

  }
  print.msg(  paste(paste( "######## Finish Working Oracle Model : ",om),"############"   ))

  list(
    oracle.model =  om,
    dataset.name = ds,
    featuresImp = om.imp,
    sm.featuresImp = list_imp_per_om,
    sm.kappas = list_kapa_per_om,
    sm.conf.matrix = list_conf_mat
  )
}

################################################




######################## Constuct summary KAPPA  tables ##########################
### sm with kappa max
calc.kappa.summary <- function(datasets_csv,models,all.ds.om.results) {
  summary.table.sm.kappa.max.name = init.table(datasets_csv,models,"NA")
  #### this code to create the tables that have the result of spearman corrlation the orignial model and the surogate
       model which has the highest kappa with the dataset
  summary.table.sm.kappa.max.corr.spearman  <- init.table(datasets_csv,models,0)
  summary.table.sm.kappa.max.corr.pearson    <- init.table(datasets_csv,models,0)
  summary.table.sm.kappa.max.corr.kendall   <- init.table(datasets_csv,models,0)
  for (ds in datasets_csv){
    ds.om.results <- all.ds.om.results[[ds]]
    for(om in models) {
      om.results   <- ds.om.results[[om]]

      kappa_MAX <- which.max(om.results$sm.kappas)
      kappa_Max_model_name <- names(kappa_MAX)
      kappa_Max_value <- om.results$sm.kappas[kappa_MAX]
      # print(paste("Max for ", om))
      # print(paste("sm : ", kappa_Max_model_name))
      # print(paste("value : ", kappa_Max_value))

      sm_imp <- om.results$sm.featuresImp[[kappa_Max_model_name]]
      om_imp <- om.results$featuresImp
```

```
163
164
165       summary.table.sm.kappa.max.name[ds,om] <- kappa_Max_model_name
166
167       if(is.logical(sm_imp)){
168         summary.table.sm.kappa.max.corr.spearman[ds,om] <- 0
169         summary.table.sm.kappa.max.corr.pearson[ds,om]  <- 0
170         summary.table.sm.kappa.max.corr.kendall[ds,om]  <- 0
171       }
172       else
173       {
174         om.sm.corr<-calc.corr(om_imp,sm_imp)
175         summary.table.sm.kappa.max.corr.spearman[ds,om] <- om.sm.corr$spearman$estimate
176         summary.table.sm.kappa.max.corr.pearson[ds,om]  <- om.sm.corr$pearson$estimate
177         summary.table.sm.kappa.max.corr.kendall[ds,om]  <- om.sm.corr$kendall$estimate
178       }
179     }
180   }
181
182   list (
183     sm.kappa.max.name = summary.table.sm.kappa.max.name ,
184     sm.kappa.max.corr.spearman = summary.table.sm.kappa.max.corr.spearman  ,
185     sm.kappa.max.corr.pearson = summary.table.sm.kappa.max.corr.pearson  ,
186     sm.kappa.max.corr.kendall =  summary.table.sm.kappa.max.corr.kendall
187   )
188 }
189
190 ###################### Constuct  KAPPA summary  tables ##########################
191
192 ###################### Constuct  Correlation tables ##########################
193 get.max.sm.corr <- function(corr.table, om , list_kappa_om) {
194   max_sm_value <-  which.max( corr.table[om,])
195   dup.max.sm <- corr.table[om, ] == corr.table[om,max_sm_value]
196   if(length(dup.max.sm) > 1) {
197     ## lets see which have max kappa
198     # list_kappa_om <- ds.models.kappa.table[[om]]
199     max_sm_value <- which.max(list_kappa_om[dup.max.sm])
200   }
201   max_sm_value
202 }
203
204 calc.all.corr <- function(datasets_csv,models,all.ds.om.results) {
205   ####CREATE TABLE THAT CONTAIN HIGHEST CORRELATION the name of surrogate model WITH THE THE DATA SET FOR EVERY ORIGINAL
          MODEL
206   summary.table.sm.max.corr.spearman.name   <- init.table(datasets_csv,models,"NA")
207   summary.table.sm.max.corr.kendall.name    <- init.table(datasets_csv,models,"NA")
208   summary.table.sm.max.corr.pearson.name     <- init.table(datasets_csv,models,"NA")
209   ### this for the   value
210   summary.table.sm.max.corr.spearman.value  <- init.table(datasets_csv,models,0)
211   summary.table.sm.max.corr.kendall.value   <-init.table(datasets_csv,models,0)
212   summary.table.sm.max.corr.pearson.value    <-init.table(datasets_csv,models,0)
213
214   ##this the table  that has the result of  spearmancorrelation of the orignial model with every  surrogate mode
215
216   #ds.models.corr.spearman.table<-init.table(models,models,0)
217   all.ds.sm.corr.spearman.tables  <- list()
218   ##this the table  that has the result of  personecorrelation of the orignial model with every  surrogate mode
219   all.ds.sm.corr.pearson.tables    <- list()
220   ##this the table  that has the result of  kadnellcorrelation of the orignial model with every  surrogate mode
221   all.ds.sm.corr.kendall.tables   <- list()
222
223   ## list of tables of kappa
224   all.ds.sm.kappa.tables          <- list()
225   for (ds in datasets_csv){
226     print.msg(c("Dataset : ", ds))
227     ds.models.kappa.table         <- init.table(models,models,0)
228     # table_of_person_of_every_surogateModel
229     ds.models.corr.pearson.table  <- init.table(models,models,0)
230     ds.models.corr.spearman.table <- init.table(models,models,0)
231     ds.models.corr.kendall.table  <- init.table(models,models,0)
232     ds.om.results <- all.ds.om.results[[ds]]
233     for(om in models) {
234       print.msg(c("       om : ", om))
235
236       om.results   <- ds.om.results[[om]]
237       ## first loop iterate over oracle models (all modesl)
238       om_imp <- om.results$featuresImp
239
240       ## then for each om model you get list of surogate models featureImp and you get the names of those models in ss
            list
241       ## then you iterate over ss list
242       for(sm in models){
```

```
243         print.msg(c("              sm : ", sm))
244
245         ## here we get the FeatureImp of sm models
246         om.sm_imp<-  om.results$sm.featuresImp[[sm]]
247         if (is.logical(om.sm_imp)) {
248           ds.models.corr.spearman.table[om,sm] <- as.numeric(0)
249           ds.models.corr.kendall.table[om,sm] <- as.numeric(0)
250           ds.models.corr.pearson.table[om,sm] <- as.numeric(0)
251         }
252         else{
253           om.sm.corr <- calc.corr(om_imp,om.sm_imp)
254           ds.models.corr.spearman.table[om,sm]  <- om.sm.corr$spearman$estimate
255           if(is.na(om.sm.corr$spearman$estimate))
256             ds.models.corr.spearman.table[om,sm]  <- 0
257           ds.models.corr.kendall.table[om,sm]   <- om.sm.corr$kendall$estimate
258           if(is.na(om.sm.corr$kendall$estimate))
259             ds.models.corr.kendall.table[om,sm]   <- 0
260           ds.models.corr.pearson.table[om,sm]   <- om.sm.corr$pearson$estimate
261           if(is.na(om.sm.corr$pearson$estimate))
262             ds.models.corr.pearson.table[om,sm]   <- 0
263         }
264         ds.models.kappa.table[om,sm]          <- om.results$sm.kappas[[sm]]
265       }
266
267       max_sm_index.spearman <- get.max.sm.corr(ds.models.corr.spearman.table,om,om.results$sm.kappas)
268       summary.table.sm.max.corr.spearman.name[ds,om] <- names(max_sm_index.spearman)
269       summary.table.sm.max.corr.spearman.value[ds,om] <- ds.models.corr.spearman.table[om,max_sm_index.spearman]
270
271       max_sm_index.kendall<-get.max.sm.corr(ds.models.corr.kendall.table,om,om.results$sm.kappas)
272       summary.table.sm.max.corr.kendall.name[ds,om] <- names(max_sm_index.kendall)
273       summary.table.sm.max.corr.kendall.value[ds,om] <- ds.models.corr.kendall.table[om,max_sm_index.kendall]
274
275       max_sm_index.pearson<-get.max.sm.corr(ds.models.corr.pearson.table,om,om.results$sm.kappas)
276       summary.table.sm.max.corr.pearson.name[ds,om] <- names(max_sm_index.pearson)
277       summary.table.sm.max.corr.pearson.value[ds,om] <- ds.models.corr.pearson.table[om,max_sm_index.kendall]
278
279       ############################
280
281
282
283       #summary.table.sm.max.corr.kendall.value[ds,om] <-  ds.models.corr.kendall.table[om,max_sm_index.kendall]
284       #summary.table.sm.max.corr.pearson.value[ds,om] <- ds.models.corr.pearson.table[om,max_sm_index.pearson]
285     }
286
287
288
289     # Generamos datasets surrogados para training y test
290     all.ds.sm.kappa.tables[[ds]] <- ds.models.kappa.table
291     all.ds.sm.corr.spearman.tables[[ds]] <- ds.models.corr.spearman.table
292     all.ds.sm.corr.pearson.tables[[ds]]<-ds.models.corr.pearson.table
293     all.ds.sm.corr.kendall.tables[[ds]]<-ds.models.corr.kendall.table
294     # summary.table.sm.max.corr.spearman.name[ds,om]<-all.ds.sm.corr.spearman.tables[which.max(all.ds.sm.corr.spearman.
            tables$),]
295
296
297
298
299     remove(ds.models.kappa.table)
300     remove(ds.models.corr.pearson.table )
301     remove(ds.models.corr.spearman.table )
302     remove(ds.models.corr.kendall.table  )
303   }
304
305   kappa.summary <- calc.kappa.summary(datasets_csv,models,all.ds.om.results)
306
307   list(
308     all.ds.sm.corr.spearman.tables=all.ds.sm.corr.spearman.tables  ,
309     all.ds.sm.corr.pearson.tables=all.ds.sm.corr.pearson.tables   ,
310     all.ds.sm.corr.kendall.tables=all.ds.sm.corr.kendall.tables   ,
311     all.ds.sm.kappa.tables = all.ds.sm.kappa.tables ,
312
313
314
315     summary.kappa.table = kappa.summary,
316     summary.corr.tables = list (
317       sm.max.corr.spearman.name=summary.table.sm.max.corr.spearman.name  ,
318       sm.max.corr.kendall.name=summary.table.sm.max.corr.kendall.name   ,
319       sm.max.corr.pearson.name=summary.table.sm.max.corr.pearson.name    ,
320       ### this for the   value
321       sm.max.corr.spearman.value=summary.table.sm.max.corr.spearman.value  ,
322       sm.max.corr.kendall.value=summary.table.sm.max.corr.kendall.value   ,
323       sm.max.corr.pearson.value=summary.table.sm.max.corr.pearson.value
```

```
    )
  )

}
####################### Constuct  Correlation tables ##########################









################################### RUN methods ####################################
# Nombre del dataset, incluye el .csv
#ds <- "car.csv"
#ds<-"heart-c.csv"
# Nombre de los modelos. Debe corresponderse con los nombres
# del fichero methods_prob.R
models <- c("c5.0", "simpls_ncomp4","mlpML","glmnet", "multinom","gcvEarth_d3","knn_k5", "rf","rda" ,"svmRadialCost_C2_
    -5","NB_laplace")
models <- c("glmnet")
models <- c( "c5.0", "simpls_ncomp4","mlpML")


# this code to create the table that have the result of spearmancorrlation the orignial model and the surogate model
    which has the highest kappa with the dataset

datasets_csv <- c("car.csv","heart-c.csv","badges2.csv","balance-scale.csv","banknote-authentication.csv","credit-g.csv"
    ,"diabetes.csv","haberman.csv","iris.csv","labor.csv",
              "monks1.csv","phoneme.csv","tae.csv","vote.csv","PhishingWebsites.csv","wall-robot-navigation.csv","
                  waveform-5000.csv",
              "cmc.csv","heart-h.csv","credit-a.csv" )
# "blood-transfusion-service-center.csv"
datasets_csv <- c("cmc.csv","bupa.csv","blood-transfusion-service-center.csv","haberman.csv","heart-h.csv")
# "bupa.csv" did not work
datasets_csv <- c("blood-transfusion-service-center.csv","haberman.csv","heart-h.csv")

datasets_csv<-c("credit-a.csv")
datasets_csv <- c("bupa.csv","blood-transfusion-service-center.csv")

#############################################################################################
sink(type = c("output", "message"));
all.ds.om.results <- list()

## save to disk
# saveRDS(all.ds.om.results,file="all.ds.om.results.rds")
# all.ds.om.results <- readRDS("all.ds.om.results.rds")

####################
for (ds in datasets_csv){
  print.msg( "#########################################################################")
  print.msg( c("Working with Dataset : " , ds  )  )

  ds_path <- sprintf("datasets/extension_caepia/%s",ds)
  datos <- load_dataset(ds_path)
  Train<-datos
  if(any(is.na(Train))==TRUE)
    print(paste("dataset Contain NAs : ",ds))
  set.seed(3)
  GridTrain<-generateUnifTest(Train, 100 * (ncol(datos)-1))
  set.seed(4)
  GridTest<-generateUnifTest(Train, 100 * (ncol(datos)-1))

  ds.om.result <- list()
  for (om in models) {
    om.result <- calc.imp(models,om,ds, Train, GridTrain,GridTest)
    ds.om.result[[om]] <- om.result
  }
  all.ds.om.results[[ds]] <- ds.om.result
  print.msg(c( "############################## ",ds," ###################################"))

}
#############################################################################################

all.res.tables <- calc.all.corr(datasets_csv,models,all.ds.om.results)
################################### RUN methods ####################################
```

```
402
403
404 resultFolder="D:/Master/FinalProject TFM/result/"
405
406
407 ## write correlation tables for each dataset
408 for (ds in datasets_csv){
409   write.csv(  all.res.tables$all.ds.sm.kappa.tables[[ds]],file=paste("D:/Master/FinalProject TFM/result/ds.models.kappa.
        table",ds))
410   write.csv( all.res.tables$all.ds.sm.corr.spearman.tables[[ds]] ,file=paste("D:/Master/FinalProject TFM/result/spearman
        .table",ds))
411   write.csv(all.res.tables$all.ds.sm.corr.pearson.tables[[ds]],file=paste("D:/Master/FinalProject TFM/result/pearson.
        table",ds))
412   write.csv( all.res.tables$all.ds.sm.corr.kendall.tables[[ds]],file=paste("D:/Master/FinalProject TFM/result/kendall.
        table",ds))
413 }
414 ### Write kappa summary tables
415 write.csv(all.res.tables$summary.kappa.table$sm.kappa.max.name,file="D:/Master/FinalProject TFM/result/table.maxkappa.
      name.csv")
416 write.csv(all.res.tables$summary.kappa.table$sm.kappa.max.corr.spearman,file="D:/Master/FinalProject TFM/result/table.
      spearman.maxkappa.csv")
417 write.csv(all.res.tables$summary.kappa.table$sm.kappa.max.corr.kendall,file="D:/Master/FinalProject TFM/result/table.
      kendall.maxkappa.csv")
418 write.csv(all.res.tables$summary.kappa.table$sm.kappa.max.corr.pearson,file="D:/Master/FinalProject TFM/result/table.
      pearson.maxkappa.csv")
419 ### Write correlation summary tables
420
421 write.csv(all.res.tables$summary.corr.tables$sm.max.corr.spearman.name,file="D:/Master/FinalProject TFM/result/table.
      spearman.maxcorr.csv")
422 write.csv(all.res.tables$summary.corr.tables$sm.max.corr.kendall.name,file="D:/Master/FinalProject TFM/result/table.
      kendall.maxcorr.csv")
423 write.csv(all.res.tables$summary.corr.tables$sm.max.corr.pearson.name,file="D:/Master/FinalProject TFM/result/table.
      pearson.maxcorr.csv")
424 write.csv(all.res.tables$summary.corr.tables$sm.max.corr.spearman.value,file="D:/Master/FinalProject TFM/result/table.
      spearman.maxcorrval.csv")
425 write.csv(all.res.tables$summary.corr.tables$sm.max.corr.kendall.value,file="D:/Master/FinalProject TFM/result/table.
      kendall.maxcorrval.csv")
426 write.csv(all.res.tables$summary.corr.tables$sm.max.corr.pearson.value,file="D:/Master/FinalProject TFM/result/table.
      pearson.maxcorrval.csv")
427 ### Write summary tables
428
429 ##############################
430 compare_result <- function(res.table1,res.table2) {
431   result_table <- matrix( as.integer(res.table1 == res.table2),ncol=ncol(res.table1) , nrow =nrow(res.table1) )
432   colnames(result_table)<-colnames(res.table1)
433   rownames(result_table)<-rownames(res.table1)
434   result_table
435 }
436 # compare_result_name <- function(res.table1,res.table2) {
437 #   result_table <- matrix( res.table1[res.table1 == res.table2],ncol=ncol(res.table1) , nrow =nrow(res.table1) )
438 #   colnames(result_table)<-colnames(res.table1)
439 #   rownames(result_table)<-rownames(res.table1)
440 #   result_table
441 # }
442
443 count.all <- function(models,res) {
444   counter <- list()
445   for(om in models) {
446     counter[[om]] <- 0
447   }
448   for(model in res) {
449     counter[[model]] <- counter[[model]] + 1
450   }
451   counter
452 }
453
454
455 # res.table1 = all.res.tables$summary.corr.tables$sm.max.corr.spearman.name
456 # res.table2 = all.res.tables$summary.kappa.table$sm.kappa.max.name
457 count.ds.sm <- function(res.table1,res.table2) {
458   result_table <-  compare_result(res.table1,res.table2 )
459
460   datasets <- rownames(result_table)
461   models <- colnames(result_table)
462   ds.counter <- init.table(datasets,models,0)
463   for(ds in datasets) {
464     rawData <-  res.table1[ds,as.logical(result_table[ds,])  ]
465     counters <- count.all(models,rawData)
466     for(model in models)
467       ds.counter[ds,model] = counters[[model]]
468   }
469   ds.counter
```

```
}



#result_table.spearman <-  compare_result(all.res.tables$summary.corr.tables$sm.max.corr.spearman.name ,  all.res.tables$summary.kappa.table$sm.kappa.max.name )


#result_table.kendall <-  compare_result(all.res.tables$summary.corr.tables$sm.max.corr.kendall.name ,  all.res.tables$summary.kappa.table$sm.kappa.max.name )
#result_table.pearson <-  compare_result(all.res.tables$summary.corr.tables$sm.max.corr.pearson.name ,  all.res.tables$summary.kappa.table$sm.kappa.max.name )

accuracy.ourhypothis<-colMeans(result_table.spearman)
avg.sumofavg.dataset.numoracle<-mean(accuracy.ourhypothis)




random.average.ds <- 1/length(models)
random.average.all <- 1/(length(models)*length(datasets_csv))


ds.counter.spearman <-  count.ds.sm(all.res.tables$summary.corr.tables$sm.max.corr.spearman.name ,  all.res.tables$summary.kappa.table$sm.kappa.max.name )
ds.counter.kendall  <-  count.ds.sm(all.res.tables$summary.corr.tables$sm.max.corr.kendall.name  ,  all.res.tables$summary.kappa.table$sm.kappa.max.name )
ds.counter.pearson  <-  count.ds.sm(all.res.tables$summary.corr.tables$sm.max.corr.pearson.name  ,  all.res.tables$summary.kappa.table$sm.kappa.max.name )

write.counter.table <-  function( ds.counter.xyz, corr.name ) {
  ds.counter.xyz.average.ds <- ds.counter.xyz/length(models)
  ds.counter.xyz.models  <-colSums(ds.counter.xyz)

  ds.counter.xyz.average.all <- ds.counter.xyz.models/(length(models)*length(datasets_csv))

  fileName <- paste0(resultFolder, paste0( corr.name , "ds.counter.average.ds.csv") )

  write.csv(ds.counter.xyz.average.ds,file=fileName)
  fileName <- paste0(resultFolder, paste0( corr.name , "ds.counter.xyz.average.all.csv") )
  write.csv(ds.counter.xyz.average.all,file=fileName)

}
write.counter.table(ds.counter.spearman,"spearman")
write.counter.table(ds.counter.kendall,"kendall")
write.counter.table(ds.counter.pearson,"pearson")

####
```