1.5em 0pt

Machine Learning in Credit-Scoring

Isabella Arango, Sophia Giraldo, Valentina Yusty Mosquera

May 15, 2020

Abstract

Keywords: Credit scoring, withdrawal, machine learning, logistic regression, SVM, KNN

1 Introduction

As computers are required to solve problems of higher complexity, tasks arise where traditional programming approaches cannot be used. Such scenarios happen when the designer of the system cannot correctly declare a method that uses input data to compute a correct output. Therefore, as stated in (1), an approach in which a computer "attempts to learn the input/output functionality" is implemented to further understand and classify the data.As stated in (5), during the past decades, credit scoring has proved to be "one of the most successful applications of statistical and operations research modeling in finance and banking." Credit scoring practices have allowed financial institutions in the credit sector to witness an increase in their capital and prospect money investment when dealing with costumer credit. It is detrimental for financial institutions to have the ability to identify what types of credit requests fall under the category of clients that are likely to repay their debt and those who aren't. To do so, "the method produces a "score" that a bank can use to rank its loan applicants or borrowers in terms of risk (9)." As denoted in (9), the score produced by the method takes into account information such as "the applicant's monthly income, outstanding debt, financial assets, how long the applicant has been in the same job, whether the applicant has defaulted or was ever delinquent on a previous loan, among others." The higher the score, the lower the risk associated with the applicant and vice versa. Therefore, credit scoring serves as a tool to mitigate the risk that financial institution face when granting a credit to their clients.

2 Methods

2.1 Logistic Regression

The logistic regression method is a supervised Machine Learning algorithm, which can be used for regression problems or as a classification method and because it does not require too many computational resources, it turns out to be one of the most implemented algorithms nowadays in fields like economy, finance, epidemiology, among others. This allows us to estimate the relationship between a group of explanatory variables (Xi, i=1, ..., n) and a binary response variable Y in terms of the probability of occurrence of the two possible classification groups.

Actually, this method consists of two parts. Initially it can be understood as a linear method if we consider the logit of Y, that is the logarithm of the probabilities that Y is equal to one of the two categories.

$$\text{logit} \Rightarrow ln\left[\frac{p(x)}{1-p(x)}\right] = f(x) = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k$$

Figure 1: Logit function

For which the regression coefficients β must be estimated from the maximum likelihood method and with the p-values granted by the test, in an iterative way, those variables that present the highest of these will be eliminated until only those that have a p-value lower than 0.05 are obtained, since in this way the correlation between the explanatory variables would be eliminated and a better adjustment of the model would be achieved. Finally, we proceed to eliminate the linearity from the inverse of the logistic function and thus be able to obtain a behavior in the form of S, where the maximum value will be one and the minimum will be zero, because we are calculating the probability of permanence to one of the two groups, in which one can be considered as success and the other as failure.

$$p(\boldsymbol{x}) = \frac{e^{f(x)}}{1+e^{f(x)}} = \frac{e^{\beta_0+\beta_1 x_1+\cdots+\beta_k x_k}}{1 \quad + \quad e^{\beta_0+\beta_1 x_1+\cdots+\beta_k x_k}} = \frac{1}{1+e^{-\beta_0-\sum_{i=1}^{k}\beta_i x_i}}$$

Figure 2: Probability of occurrence or success of an event

Therefore, it turns out to be a linear model for which the β parameters have to be estimated by implementing the maximum likelihood method [5]. Now, the ratio of the probability of success to the probability of failure, known as odds, proves to be very useful, as it allows us to understand the relationship between individuals who depend or do not depend on a risk factor. In addition, the odds ratio provides the odds ratio (OR), which indicates the proportion in which the odds of being subject to a factor exceeds the odds of not being subject to it. Then, in order to classify the data in the two groups given by the binary or dichotomous variable, a cut-off point is established for the values obtained with the logit function and those that exceed this value will belong to one group and the others to the other, in most of the cases the cut−off point is 0.5.

Some of the advantages of this algorithm are:

1. It is an efficient and easy to implement algorithm because it does not require too many computer resources.
2. It eliminates assumptions made in the linear regression method such as the normality of the data under consideration.
3. It allows better adjustment of the data to be categorized according to a binary response variable.

However, it also has its disadvantages such as:

1. It cannot be applied to non-linear problems because it is based on linear regression.
2. It requires a good pre processing of the data in which those variables that are highly correlated with each other must be eliminated.

2.1.1 Example

Now, let′s look at a practical example to understand better the theory already explained [12]. Let's try to classify whether a basketball player scores a point or not depending on the distance from where he throws the ball. In this case, we are going to consider only one explanatory variable to facilitate the understanding in a graphic way and this will be the throwing distance X and as a response variable we consider a binary that indicates 1 in case of annotation and 0 in opposite case (Y). Then, if we plot X vs Y we get the following graph

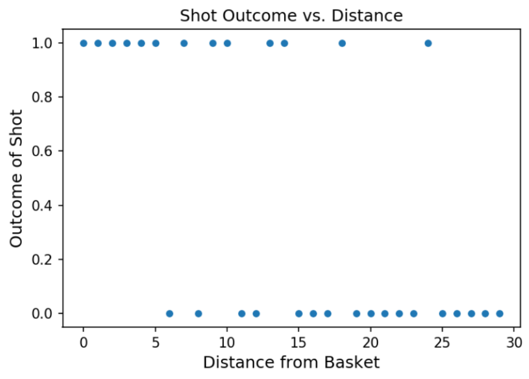


Figure 3: Shot Outcome vs. Distance from Basket

Where you can see that the further away you go, the more failures you have. As mentioned above, the logistic regression consists of two stages, the first of which is to consider it as a linear regression taking Y as the logarithm of the probability of occurrence of one of the two possible events over the probability of occurrence of the other, as expressed in Figure 1. Now, from the maximum likelihood method we obtain that B0 = 2.5 and B1 = −0.2, which means that for each unit that increases the launch distance, the probability of achieving the annotation will decrease by 0.2 and graphing the linear model obtained with these parameters we can see that it will not allow us to classify whether the data belong to the group of successful launch or failure

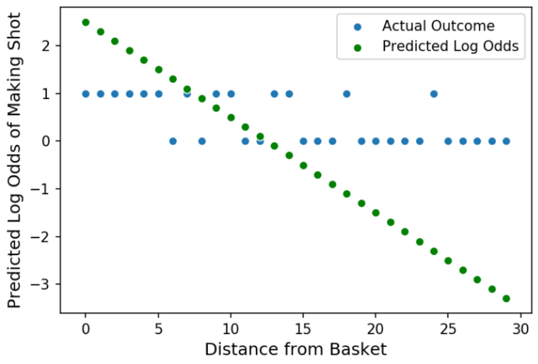


Figure 4: Linear Regression prediction

Therefore, we will apply the inverse of the logistic function as indicated in figure 2 to know the probability of belonging to one of the two groups and by graphing this result we obtain the following graph

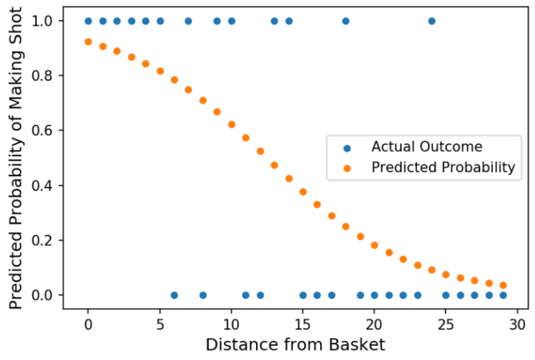


Figure 5: Logistic Regression prediction

This allows us to classify the data correctly, since those below the value of 0.5 on the Y axis belong to the group in which no annotations were made and those above this value will be part of the group in which a point was scored.

2.1.2 Pseudocode

Algorithm 1 Logistic Regression

```
data ← Load data
X ← explanatory variables
Y ← response variable
for all xi vector in X do
    i) Delete from data observations with missing information.
    ii) Calculate the regression coefficients using the maximum likelihood method.

    if xi p-value $< 0.05$ then
        sigVar(end+1)=Xi
    end

end

iii) Calculate the regression coefficients using the maximum likelihood method only with
sigVar.

iv) Apply the inverse logit function

end
```

2.2 K-Nearest Neighbor

K−Nearest Neighbor is a supervised Machine Learning algorithm which can be used for classification problems and as regression predictive problem. This method is very simple, nevertheless, its results may be highly accurate. It is a non-parametric method, has a lazy learning and it′s mostly used for recognition of patterns, data mining, anomaly detection, economics, bank systems and calculating credit ratings.

In Table 1, we compare 3 important aspects scoring them from 1 to 3, being 3 the best answer to each aspect. We can observe that KNN is mainly used for its easy interpretation and low calculation time. [8]

Table 1: Comparison of 3 principle aspects

	Logistic Regression	**Random Forest**	**KNN**
Ease to interpret output	2	1	3
Calculation time	3	1	3
Predictive power	2	3	2

Given a dataset, the K-NN algorithm consists in selecting a k value and in the moment of analysis the k nearest points to the desire class will be the solution. The most important phase of the method is the k value selection, it must be selected accordingly with the dataset we are working with.

For better understanding here is an example: lets say we have two classes, C1$-$red circles and C2$-$green squares. Now we want to classify the Bl$-$blue star as seen in Figure 1.

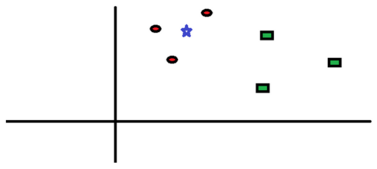


Figure 6: Scenario 1

Bl can be either from C1 or C2. Let's say that $k = 3$, now we take Bl as the centroid of a circle big enough that encloses k data points as seen in Figure 2.

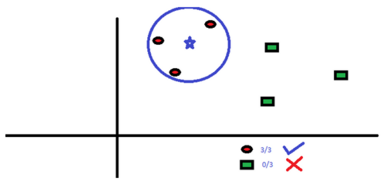


Figure 7: Scenario 2

The 3 closest points to Bl are C1, then we can say that it belongs to that class. Nevertheless, when the circle encloses more than one class points, it would classify Bl from the class that has more points in the circle. This is the reason why k is preferred to be a odd number. Nevertheless, as every algorithm, it has its advantages and disadvantages.

Advantages:

1. The algorithm is simple and easy to implement.
2. There's no need to build a model, tune several parameters, or make additional assumptions.
3. The algorithm is versatile. It can be used for classification, regression, and search.

Disadvantages:

1. The algorithm gets significantly slower as the number of examples and/or predictors/independent variables increase.

2.2.1 Pseudocode

Algorithm 2 K-nearest neighbor

```
$data \leftarrow Load\ data$
$k \leftarrow initialize\ value$
for all data point do
    i) Calculate distance between the query and the current example from the data.
    ii) Add the distance and the index of the example to an ordered collection.
    iii) Sort the ordered collection of distances and indices in ascending order.
    iv) Pick the first K entries from the sorted collection.
    v) Get the labels of the selected K entries.
    if regression then
        return the mean of the K labels
    else
        return the mode of the K labels.
end
```

2.3 Support Vector Machine

As defined by (1), Support Vector Machines "are learning systems that use a hypothesis space of linear functions in a high dimensional feature space, trained with a learning algorithm from optimization theory that implements a learning bias derived from statistical learning theory." In other words, the implementation of a support−vector machine revolves around the construction, in a high or infinite dimensional space, of a hyperplane or a set of hyperplanes. By definition, a hyperplane serves as a decision boundary in which each input vector from the input space is classified. This algorithm can be used for a variety of purposes: classification, regression and outlier detection. A successful implementation of the Support Vector Machine algorithm is that in which the distance to the nearest data point if the largest from the hyperplane, such distance is denoted as functional margin, which will be defined below. In theory, the larger the functional margin, the lower the error and the more successful the implementation of the Support Vector Machine Algorithm.

2.3.1 Linear Classification

A binary classification problem involves the use of a classification decision rule to classify elements into two distinct groups. For notation purposes let X, $X \in \mathbb{R}^n$, denote the input space and Y, $Y = \{-1, 1\}$, denote the output domain. Let an example from the data be represented as the pair $(\boldsymbol{x_i}, y_i)$, where $\boldsymbol{x_i}$ represents the input vector and y_i the respective output value. Lastly, let the training data be denoted as $S = ((\boldsymbol{x}_1, y_1)), ..., (\boldsymbol{x}_l, y_l))$ $\in (X, Y)^l$.

Binary classification for a testing point $(\boldsymbol{x}_t, y_t)$ is performed by using a function $f : X \in \mathbb{R}^n$ $\rightarrow \mathbb{R}$, denoted as:

$$f(\boldsymbol{x}_t) =< \boldsymbol{w}^T * \boldsymbol{x}_t >+b \tag{1}$$

where $(\boldsymbol{w}, b) \in \mathbb{R}^n * \mathbb{R}$ are parameters that control the decision rule and are learned from a S. Note that if the value returned after evaluating $\boldsymbol{x_t}$ in Eq.(1) is positive, then the testing point is assigned to the positive class and vice versa for the negative class.

Note that by substituting Eq.(17) into Eq.(1), the decision rule can also be evaluated as the inner product between a testing point $\boldsymbol{x}_t$ and the training points in S:

$$f(\boldsymbol{x}_t) = \left(\sum_{n=1}^{l} \alpha_i y_i \boldsymbol{x_i} \right)^T \boldsymbol{x}_t + b \tag{2}$$

$$f(\boldsymbol{x}_t) = \sum_{n=1}^{l} \alpha_i y_i \langle \boldsymbol{x_i}, \boldsymbol{x}_t \rangle + b \tag{3}$$

When dealing with linear classification, functional and geometric margins serve as a tool to determine the performance of the implementation. The functional margin of an example $(\boldsymbol{x_i}, y_i)$ with respect to the hyperplane $(\boldsymbol{w}, b)$ can be calculated as:

$$\hat{\gamma_i} = y_i(< \boldsymbol{w}^T * \boldsymbol{x_i} >+b) \tag{4}$$

Note that if the value $\hat{\gamma_i}$ is greater than 0, then the classification of $(\boldsymbol{x_i}, y_i)$ is correct. It is important to take into account that given S, we define the geometric margin of $(\boldsymbol{w}, b)$ with respect to S as:

$$\hat{\gamma} = \min_{i=1...l} \hat{\gamma_i} \tag{5}$$

The geometric margin, on the other hand, represents the Euclidean distance of an example with respect to the decision boundary and can be calculated as:

$$\gamma_i = y_i * \left(\left(\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} \right)^T \boldsymbol{x_i} + \left(\frac{b}{\|\boldsymbol{w}\|} \right) \right) \tag{6}$$

It is important to take into account that given S, we define the geometric margin of $(\boldsymbol{w}, b)$ with respect to S to be:

$$\gamma = \min_{i=1...l} \gamma_i \tag{7}$$

2.3.2 Kernel-Induced Feature Spaces

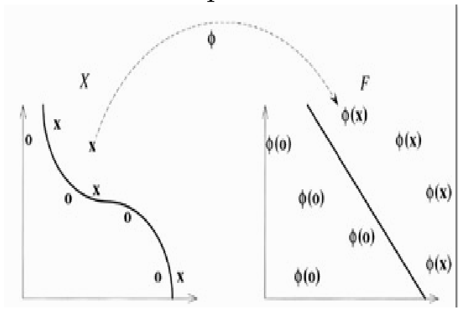
There are scenarios is which a simple linear combination of the attributes that are being studied cannot accurately express the target concept. In such scenarios, the idea of more abstract features of the data is appealing. Therefore, as mentioned in (1), an alternative to increase the computational power of the linear learning machines is through the use of kernel representation, which projects the data into a high dimensional feature space. The execution of the previous idea is handled by replacing the use of a "kernel" function to perform "a non-linear mapping to a high dimensional feature space without increasing the number of tunable parameters (1)."

Ideally, a representation of the target function that matches the learning problem should be chosen. A common preprocessing method calls for altering the way in which data is represented in order to acquire a target function that can adequately distinguish between different classes. This is done by changing the representation of all $\boldsymbol{x_i} \in X \ i = \{1, ..., n\}$:

$$\boldsymbol{x_i} = (x_1, ..., x_n) \rightarrow \phi(\boldsymbol{x_i}) = (\phi(x_1), ..., \phi(x_n)) \tag{8}$$

The procedure denoted in Eq.(5) is equivalent to mapping the input space X onto a new space represented by $F = \{\varphi(\boldsymbol{x_i})|\boldsymbol{x_i} \in X\}$, denoted as feature space, where $\varphi : X \rightarrow F$ is a non-linear map from the input space to some feature space. Figure 1 shows an scenario where data cannot be separated by a linear function in the input space but can be separated by a linear function in the feature space by utilizing feature mapping from a two dimensional input space to a two dimensional feature space.

Figure 8: Feature map for a classification task



The main reason why feature mapping occurs is due to the necessity of machines to classify non-linear relationships. Ultimately, feature mapping allows non-linearly separable data to become linearly separable by rewriting the data into a new representation. In other words, as denoted by (1), the process "is equivalent to applying a fixed non-linear mapping of the data to a feature space, in which the linear machine can be used." Let the function below

$$f(\boldsymbol{x_i}) = \sum_{n=1}^{N} w_i \phi_i(\boldsymbol{x_i}) + b \tag{9}$$

be a linear machine used to classify the data that belongs to feature space F. Therefore, the previous theory implies that when dealing with non-linearly separable data, two steps have to be taken: "first a fixed non-linear mapping transforms the data into a feature space F, and then a linear machine is used to classify them in the feature space (1)."

Due to the fact that linear learning machines can also be expressed in dual representation, the decision rule can also be evaluated as the inner product between a testing point $\boldsymbol{x}_t$ and the training points in S in the feature space F as:

$$f(\boldsymbol{x}_t) = \sum_{n=1}^{l} \alpha_i y_i \langle \phi_i(\boldsymbol{x_i}) \phi_i(\boldsymbol{x}_t) \rangle + b \tag{10}$$

where the inner product between a test point and the training points is calculated. As defined in (1), a Kernel is a function of K such that for all $\boldsymbol{x}, \boldsymbol{z} \in X$.

$$K(\boldsymbol{x}, \boldsymbol{z}) = \langle \phi_i(\boldsymbol{x}) \phi_i(\boldsymbol{z}) \rangle \tag{11}$$

where ϕ is a mapping from the input space X to a feature space F. Therefore, Eq.(10) can be rewritten as:

$$f(\boldsymbol{x}_t) = \sum_{n=1}^{l} \alpha_i y_i K \langle \boldsymbol{x}_i, \boldsymbol{x}_t \rangle \tag{12}$$

2.3.3 Support Vector Classification: Maximal Margin Classifier

One of the most frequently used models of Support Vector Machine for linearly separable data in the feature space is denoted as maximal margin classifier. The main idea of the maximal margin classifier, as denoted in (1), is "to find the maximal margin hyperplane in an appropriately chosen kernel-induced feature space." Achieving the previous statement results in a classifier prone to classify in a correct and thorough manner the training examples that belong to the positive and negative classes.

As denoted in (2), a decision boundary that maximized the geometric margin, γ "would reflect very confident set of predictions on the training set and a good "fit" to the training set". Therefore, given a linearly separable training sample $S = ((\boldsymbol{x}_1, y_1)), ...(\boldsymbol{x}_n, y_l))$ $\in (X, Y)^l$, the following optimization problem is proposed:

$$
\begin{aligned}
\max_{\gamma,w,b} \quad & \gamma \\
\text{s.t.} \quad & y_i(< \boldsymbol{w}^T * \boldsymbol{x_i} >+b) \geq \gamma \qquad i = 1, ...l \\
& \|\boldsymbol{w}\| = 1
\end{aligned} \tag{13}
$$

The objective of Eq.(13) is to maximize the geometric margin γ. The first constraint specifies that each training example has to have functional margin of at least γ. The second constraint,a scale constraint, ensures that the functional margin is equal to the geometric margin. Note that Eq.(13), a non-convex optimization problem, and can be rewritten as a convex-optimization problem with a convex quadratic objective an a linear constraint:

$$
\begin{aligned}
\min_{\gamma,w,b} \quad & \frac{1}{2} \|\boldsymbol{w}\|^2 \\
\text{s.t.} \quad & y_i(< \boldsymbol{w}^T * \boldsymbol{x_i} >+b) \geq 1 \qquad i = 1, ...l
\end{aligned} \tag{14}
$$

Eq.(14),denoted as primal optimization problem for the maximum margin classifier, tries to maximize the distance between the hyperplanes $< \boldsymbol{w}^T * \boldsymbol{x_i} >+b = 1$ and $< \boldsymbol{w}^T * \boldsymbol{x_i} >+b = -1$, calculated as $\frac{2}{\|\boldsymbol{w}\|}$, and is subject to constraints that ensure the classes are separable.

In order to transform the primal optimization problem into its corresponding dual optimization problem for the maximum margin classifier, let Eq.(14) be rewritten as:

$$
\begin{aligned}
\min_{\gamma,w,b} \quad & \frac{1}{2} \|\boldsymbol{w}\|^2 \\
\text{s.t.} \quad & - y_i(< \boldsymbol{w}^T * \boldsymbol{x_i} >+b) + 1 \leq 0 \qquad i = 1, ...l
\end{aligned} \tag{15}
$$

Now, constructing the Lagrangian of Eq.(15) we obtain

$$
\mathcal{L}(\boldsymbol{w}, b, \alpha) = \frac{1}{2} \|\boldsymbol{w}\|^2 - \sum_{l=1}^{l} \alpha_i [y_i(< \boldsymbol{w}^T * \boldsymbol{x_i} >+b) - 1] \tag{16}
$$

where $\alpha_i \geq 0$ are the Lagrange multipliers. Note that only when the functional margin of a training example is equal to one, the value of α_i is greater that zero. The training points for which the value of $\alpha_i > 0$ are denoted as support vectors. As defined in (3), support vectors are "data points that are closer to the hyperplane and influence the position and orientation of the hyperplane." The corresponding dual of Eq.(15) is found by differentiating Eq.(16) in terms of $\boldsymbol{w}$ and b:

$$
\frac{\partial \mathcal{L}(\boldsymbol{w}, b, \alpha)}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i=1}^{l} \alpha_i y_i \boldsymbol{x_i} = 0 \Longrightarrow \boldsymbol{w} = \sum_{l=1}^{l} \alpha_i y_i \boldsymbol{x_i} \tag{17}
$$

$$\frac{\partial \mathcal{L}(\boldsymbol{w}, b, \alpha)}{\partial b} = \sum_{l=1}^{l} \alpha_i y_i = 0 \tag{18}$$

Substituting the results from Eq.(17) and Eq.(18) into the Eq.(16), the dual function

$$\mathcal{L}(\boldsymbol{w}, b, \alpha) = \sum_{l=1}^{l} \alpha_i - \frac{1}{2} * \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j < \boldsymbol{x}_i^T * \boldsymbol{x}_j > \tag{19}$$

Taking into account the previous constraints, given a linearly separable training sample $S = ((\boldsymbol{x}_1, y_1)), ..., (\boldsymbol{x}_n, y_l)) \in (X, Y)^l$, the following dual optimization problem is proposed:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{l=1}^{l} \alpha_i - \frac{1}{2} * \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j < \boldsymbol{x}_i^T * \boldsymbol{x}_j > \\ \text{s.t.} \quad & \alpha_i \geq 0, \quad i = 1, ...l \\ & \sum_{i=1}^{l} \alpha_i y_i = 0 \end{aligned} \tag{20}$$

2.3.4 Support Vector Classification: Soft Margin Optimization

In many real-world scenarios, were data is noisy, it is very likely that there will be non linear separation of the data in the feature space. Due to the previous statement, the idea of soft margin classifier is introduced. In such case, measures such as the margin distribution is considered. As denoted in (1), “such measures can tolerate noise and outliers, and take into consideration the positions of more training points than just those closest to the boundary.”

Taking into consideration Eq.(15) and noting that ”in order to optimise the margin slack vector we need to introduce slack variables to allow the margin constraints to be violated (1)” we obtain:

$$\begin{aligned} \min_{\gamma, w, b} \quad & \frac{1}{2} \|\boldsymbol{w}\|^2 + C \sum_{l=1}^{l} \xi_i \\ \text{s.t.} \quad & y_i(< \boldsymbol{w}^T * \boldsymbol{x_i} > + b) \geq 1 - \xi_i \qquad i = 1, ...l \\ & \xi_i \geq 0 \qquad i = 1, ...l \end{aligned} \tag{21}$$

The objective function of the optimization problems in Eq.(15) and Eq.(21) differentiate in the second term. In Eq.(21), C is a hyperparameter that decides between a trades-off. When the value of C is small, the maximizing of the margin is given more importance than the classification mistake. Alternatively, when the value of C is large, avoiding the classification mistake is giving more importance that maximizing the margin. Now, lets introduce the notion of the slack variable ξ_i for an example $(\boldsymbol{x_i}, y_i)$. If the example $(\boldsymbol{x_i}, y_i)$ is classified on the wrong side of the margin, the slack variable ξ_i corresponds to the distance of the example $(\boldsymbol{x_i}, y_i)$ from its corresponding class's margin, otherwise the value of ξ_i is zero. The previous statement implies that examples who are incorrectly classified and are the farthest from the margin would receive more penalty than those who are closer to the margin. Additionally, taking into account the definition of functional margin, where a confidence of greater or equal to one suggest that the classifier has classified the example correctly, a confidence score less than one would mean that the classifier has incorrectly classified the example and has incurred a linear penalty of ξ_i.

Now, constructing the Lagrangian of Eq.(21) we obtain:

$$\mathcal{L}(\boldsymbol{w}, b, \alpha, r) = \frac{1}{2}\left\|\boldsymbol{w}\right\|^2 C\sum_{l=1}^{l}\xi_i - \sum_{l=1}^{l}\alpha_i[y_i(<\boldsymbol{w}^T * \boldsymbol{x_i}>+b) - 1 + \xi_i] - \sum_{l=1}^{l} r_i\xi_i \tag{22}$$

where $\alpha_i \geq 0$ are the Lagrange multiplier. Analogously to the procedure of transforming the primal optimization problem into its corresponding dual optimization problem, the following dual optimization problem is introduced:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{l=1}^{l}\alpha_i - \frac{1}{2} * \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j < \boldsymbol{x}_i^T * \boldsymbol{x}_j > \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C, \quad i = 1, ...l \\ & \sum_{i=1}^{l}\alpha_i y_i = 0 \end{aligned} \tag{23}$$

2.3.5 Pseudocode

Algorithm 3 Support Vector Machine

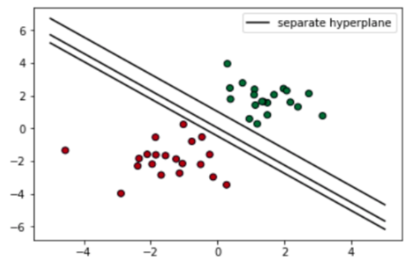
```
Training a SVM
Require: $\boldsymbol{X}$ and $\boldsymbol{y}$ loaded with training labeled data $\boldsymbol{\alpha} \leftarrow 0$.
for all $(\boldsymbol{x_i}, y_i)$, $(\boldsymbol{x_j}, y_j)$ do
    Optimize $\boldsymbol{\alpha_i}$ and $\boldsymbol{\alpha_j}$
end for

until no changes in $\boldsymbol{\alpha}$ or other constraint criteria met.
```

2.3.6 Example

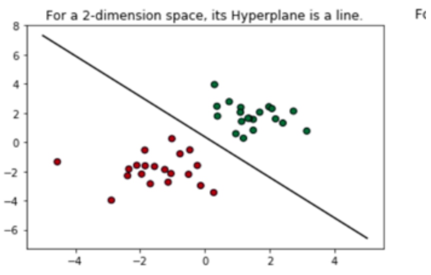
Now, let's consider an illustrative example to further understand the Support Vector Machine Algorithm. Let the green dots be the representation of a class, 1 , and let the red dots be a representation of an alternate class, -1. As seen in Figure (6), decision boundaries, the lines that separate the classes, are used to distinguish and further classify the two classes. Notice that all the red dots are below the decision boundaries, while all the green dots are placed above the decision boundaries.

Figure 9: Data Set



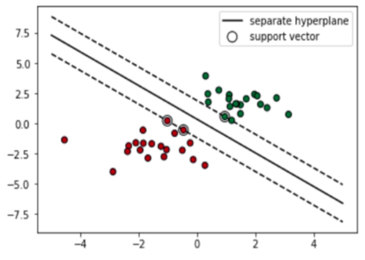
To find the decision boundary as shown in Figure (7), denoted as hyperplane, that best separates two classes, is the basic principle of Support Vector Machine.

Figure 10: Separating Hyperplanes



As shown in Figure (8), the support vectors are the data points who are closest to the hyperplane. The distance between a support vector and the decision boundary is denoted as hyperplane.

Figure 11: Support Vectors



When a new data point is tested, its classification will be based on whether it is above the decision boundary or below. In the case of this example, if the new data point is placed above the decision boundary, it is classified to the green class. On the other hand, if the new data point is placed below the decision boundary, it will be classified to the red class.

2.4 Random Forest

Random Forest, as defined in (5), is a classifier combination that utilizes L tree-structured classifiers, $h(X, \Theta_k), k = 1, ..., L$, where Θ_k are independent identically distributed random vectors and X denotes the input space. When dealing with testing data, each tree casts a unit vote for a class in which the testing example $(\boldsymbol{x_i}, y_i)$ is classified. The training example is classified to the class that has the greatest amount of unit votes. The data and the features from which the tree structured classifiers, denoted as decision trees, are trained is made by randomly selected training data from the input space X and randomly selected features for each decision split.

The main problem associated with decision trees is that they tend to produce overfitted models. Overfitted models perform poorly when being tested since the classification of such models correspond very closely to the data from which they were trained. The main benefit associated with Random Forest is that the classification of an example $(\boldsymbol{x_i}, y_i)$ is based on the classification obtained from multiple decision trees that are trained with different samples of the training set. Therefore, by implementing Random Forest there is a trade-off: a reduction of the variance at the expense of an increase in bias and a decrease of interpretability.

2.4.1 Decision Trees

As defined in (6), a Decision Tree is a tree-structured model that can be defined as a model $\varphi : X \rightarrow Y$, where X is represents the input space and Y the output domain. Most commonly, Decision Trees are represented as a binary tree, "a rooted tree where all internal nodes have exactly two children (6)." A Decision Tree is composed by branches (segments that connect nodes), nodes (the position where the branches divide), root node (the node from which the tree starts), leaf node (the node from which the tree ends). Branches are represented as segments that connect nodes while nodes are represented as circles. In a Decision Tree, each node t represents a feature test and a subspace $X_t \subset X$ and the root node t_o represents the input space X. Each branch represents the outcome of a test. To further expand, in the Decision Tree, the outcome of a set of feature questions Q determine the split s_t for every internal node t. Precisely, the split s_t for every internal node t divides the space X_t that the node represents into disjoint subspaces. Lastly, each leaf node, or terminal nodes, are labeled with a best guess value $y_t \in Y$.

As denoted in (6), "learning a decision tree ideally amounts to determine the tree structure producing the partition which is closest to the partition engendered by Y over X." To further expand, the main objective of the Decision Tree is to find a model that will be able to adequately distinguish classes of the training data S, $S = ((\boldsymbol{x}_1, y_1)), ..., (\boldsymbol{x}_n, y_l))$

$\in (X,Y)^l$. Take into account that there may be a variety of decision trees that explain S equally best. However, the best representation is that which makes the fewer assumptions and in turn the simplest solution that fits S.

In a Decision Tree, each node t has an impurity measure denoted as $i(t)$. As defined in (6), an impurity measure $i(t)$ "is a function that evaluates the goodness of any node t". One of the most frequently used ways to calculate purity at each node is by using the Gini impurity as:

$$G = \sum_{l=1}^{C} p(i) * (1 - p(i)) \tag{24}$$

where C is the number of classes in the output domain and p(i) is the probability of picking label i at a node. The smaller the impurity measure of a given node, the purer the node and the more accurate predictions for all examples $(\boldsymbol{x_i}, y_i) \in S$ such that $\boldsymbol{x_i} \in X_t$.

A Decision Tree initiates from a single node that represents the whole training set S and grows by iteratively by dividing nodes into purer ones. The previous statement indicates that the termination criteria for the Decision Tree is achieved once the splits are pure, meaning that each leaf node has a pure subset (all training examples in each leaf node have the same output value). Therefore, if all X_t at a given node t belong to the same class y_t , making the node a "pure" node, then the "pure" node will become a leaf node labeled as y_t. On the other hand, if X_t at a given node t contains examples that belong to more than one class, the node has to be divided by using a binary split. After a binary split, $s \in Q$, and node t being divided into a left node t_l and a right node t_R, the impurity decrease is calculated as:

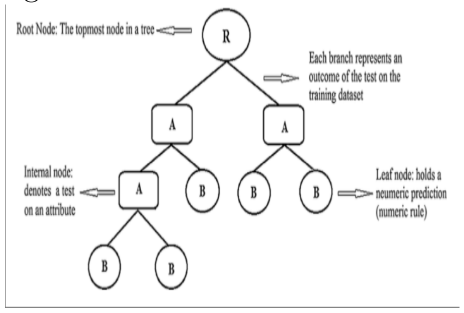
$$\triangle i(s,t) = i(t) - p_L i(t_L) - p_L i(t_R) \tag{25}$$

where $p_R = \frac{N_{tR}}{N_t}$ and $p_L = \frac{N_{tL}}{N_t}$ are the proportion of training examples from X_t, with size N_t ,at node t that go to t_R and t_L. It is important to note that when dealing with Decision Trees not all features are going to be utilized, only those that best divide the examples into their respective classes are used. This termination criteria makes the Decision Trees unstable since "small changes in the data can lead to a large change in the structure of the optimal decision tree (5)".

The generalization error is used to calculate the accuracy of a model. Given a testing set , S , of the form $S' = ((\boldsymbol{x}_1, y_1)), ..., (\boldsymbol{x}_k, y_k)) \in (X,Y)^k$, the generalization error of the model is calculated as:

$$Err(\varphi_L) = P(Y \neq \varphi_L(X)) \tag{26}$$

Figure 12: Decision Tree Structure


Root Node: The topmost node in a tree
R
Each branch represents an outcome of the test on the training dataset
A
A
Internal node: denotes a test on an attribute
A
B
B
B
Leaf node: holds a neumeric prediction (numeric rule)
B
B

where P denotes the theoretical probability. However, since the theoretical probability is unknown, Eq.(26) is rewritten as:

$$\widehat{Err}(\varphi_L) = \frac{1}{N} \sum_{(\boldsymbol{x},y)\in S'} 1(y \neq \varphi((\boldsymbol{x})) \tag{27}$$

where the N denotes the number of examples in S' and 1 denotes the unit condition:

$$1 = \begin{cases} 1 & \text{if condition is true} \\ 0 & \text{if condition is false} \end{cases}$$

Eq.(27) measures the probability of misclassification of φ_L, the Decision Tree. Note that the goal of a Decision Tree is not to make the most accurate predictions of the training set, but rather of the testing set since it will prove that the model is reliable. For the generalization error of the model φ_L to be the smallest, the generalization error of each node t has to be minimized as:

$$y_t = \arg\min_{y\in c} P(Y \neq \varphi_L(X)) \tag{28}$$

where P denotes the theoretical probability. As denoted in (6), "the generalization error of t is minimized by predicting the class which is the most likely for the samples in the subspace of t." However, since the theoretical probability is unknown, Eq.(28) is rewritten as:

$$\hat{y}_t = \arg\min_{y\in c} 1 - p(Y \neq c|X \in X_t) \tag{29}$$

$$\hat{y}_t = \arg\max_{y\in c} p(Y = c|X \in X_t) \tag{30}$$

where $p(Y = c|X \in X_t) = \frac{N_{ct}}{N_t}$ denotes the proportion of the number of of class c in S'_t over the number of examples in S'_t .

2.4.2 Random Forest Algorithm

As defined in (6), "Random Forest form a family that consist in building an ensemble (or forest) of decision trees grown from a randomized variant." Because of random perturbations in the induction procedure, each decision tree is made up from an arbitrary selection of examples from the training data and arbitrarily chosen features. One of the benefits of using Decision Trees when working with ensemble methods is that the result obtain from the averaging process of the results is likely to be accurate since each decision tree provides high variance and low bias.

One of the main benefits associated with the ensemble of decision trees is that, in general, the expected generalization error of the ensemble of decision trees is smaller than that of every individual decision tree. As denoted in (6), "the core principle of ensemble methods based on randomization is to introduce random perturbations into the learning procedure in order to produce several different models from a single learning set and then to combine the predictions of those models to form the prediction of the ensemble."

Let M denote a set of randomized models $\{\varphi_{S,\theta_m} | m = 1, .., M\}$ that where build from different random seeds θ_m and where learned from different random samples of training data S. An ensemble model is built by combining the predictions of models made from ensemble methods. When evaluating the model given a testing set , S , of the form $S' = ((\boldsymbol{x}_1, y_1)), ..., (\boldsymbol{x}_k, y_k)) \in (X, Y)^k$, for each $(\boldsymbol{x_i}, y_i) \in S'$ testing set, the result returned is computed by:

$$\Psi_{S,\theta_1,...,\theta_n} = \arg\max_{y \in c} \sum_{l=1}^{C} 1(\varphi_{S,\theta_m}(\boldsymbol{x_i}) = c) \tag{31}$$

where 1 denotes the unit condition:

$$1 = \begin{cases} 1 & \text{if condition is true} \\ 0 & \text{if condition is false} \end{cases}$$

In Eq.(31), for every $(\boldsymbol{x_i}, y_i) \in S'$, the result returned will be the label prediction. For every example, this result is obtained by considering the models in the ensemble as voters that each provide a unit vote for a label. Therefore, the testing example $x_i \in X^K$ will be classified in the label that has the majority of unit votes from the models in the ensemble to form a final prediction.

2.4.3 Pseudocode

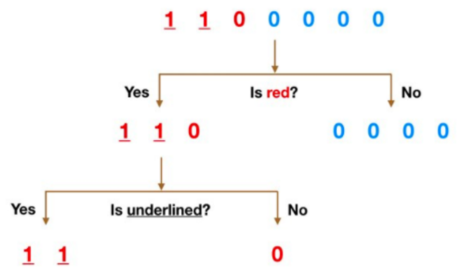
```
Algorithm 4 Random Forest
  To generate c classifiers
  for all i=1 to c do
      Randomly sample the training data D with replacement Di.
      Create a root node, Ni , containing Di.
      Call BuildTree Ni.
  end for

  BuildTree(N)
  if N contains of only one class then
      return
  else
      Randomly select a percentage of the possible splitting features in N
      Select the feature F with the highest information gain to split on
      Create f child nodes of N, N1,...,Nf, where F has the f possible values (F1,..., Ff).
      for i=1 to f do
          Set the contents of Ni to Di, where Di is all instances in N that match Fi.
          Call BuildTree(Ni).
      end for
  end if
```

2.4.4 Example

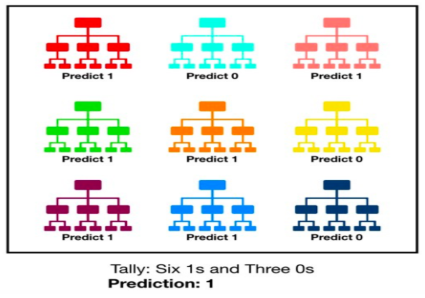
To illustrate this algorithm lets take into consideration the fact that a Random Forest is a recompilation of Decision Trees. Figure (10), is an illustration of a Decision Tree that separates two classes: a class of ones and a class of zeros. The separation between classes is done by using variables. Also, note that the termination criteria for the Decision Tree is achieved once the splits are pure, meaning that each leaf node has a pure subset (all training examples in each leaf node have the same output value), in this case 0 and 1.

Figure 13: Decision Trees Separation of Classes



When testing an example from the testing data, since a Random Forest is a recompilation of Decision Trees, each Decision Tree has to cast a unit vote for a class in which the example is be classified. The class that achieves the most unit votes among the Decision Trees is that in which the example will be classified. As shown in Figure (11), given an example of the testing data, given a Random Forest made up by nine Decision Trees, it can be seen that six of them predicted the class one, while three of them predicted the class zero. This indicated that the example from the testing data will be classified on class one. This process is done iteratively until all the testing examples are classified.

Figure 14: Unit Vote by Decision Trees



3 Variables description

The following description takes into account the dataset taken from Professor Dr. Hans Hofmann [3] named German Credit data. It contains categorical/symbolic attributes, more precisely, 7 numerical and 13 categorical making 20 variables in total. The attributes are the following:

1. Status of existing checking account:
    - A11 : $x < 0$ DM
    - A12 : $0 \leq x < 200$ DM
    - A13 : $x \geq 200$ DM
    - A14 : no checking account
2. Duration in month
3. Credit history
    - A30 : no credits taken / all credits paid back duly
    - A31 : all credits at this bank paid back duly
    - A32 : existing credits paid back duly till now
    - A33 : delay in paying off in the past

- A34 : critical account / other credits existing (not at this bank)

4. Purpose
    - A40 : car (new)
    - A41 : car (used)
    - A42 : furniture/equipment
    - A43 : radio/television
    - A44 : domestic appliances
    - A45 : repairs
    - A46 : education
    - A47 : (vacation - does not exist?)
    - A48 : retraining
    - A49 : business
    - A410 : others
5. Credit amount
6. Savings account / bonds
    - A61 : $x < 100$ DM
    - A62 : $100 \leq x < 500$ DM
    - A63 : $500 \leq x < 1000$ DM
    - A64 : $x \geq 1000$ DM
    - A65 : unknown / no savings account
7. Present employment since
    - A71 : unemployed
    - A72 : $x < 1$year
    - A73 : $1 \leq x < 4$years
    - A74 : $4 \leq x < 7$years
    - A75 : $x \geq 7$years
8. Installment rate in percentage of disposable income
9. Personal status and sex
    - A91 : male : divorced/separated
    - A92 : female : divorced/separated/married

- A93 : male : single
- A94 : male : married/widowed
- A95 : female : single

10. Other debtors / guarantors
    - A101 : none
    - A102 : co-applicant
    - A103 : guarantor
11. Present residence since
12. Property
    - A121 : real estate
    - A122 : if not A121 : building society savings agreement/ life insurance
    - A123 : if not A121/ A122 : car or other, not in attribute 6
    - A124 : unknown / no property
13. Age in years
14. Other installment plans
    - A141 : bank
    - A142 : stores
    - A143 : none
15. Housing
    - A151 : rent
    - A152 : own
    - A153 : for free
16. Number of existing credits at this bank
17. Job
    - A171 : unemployed/ unskilled - non-resident
    - A172 : unskilled - resident
    - A173 : skilled employee / official
    - A174 : management/ self-employed/ highly qualified employee/ officer
18. Number of people being liable to provide maintenance for

19. Telephone
    - A191 : none
    - A192 : yes, registered under the customers name
20. Foreign worker
    - A201 : yes
    - A202 : no

4 Principle Components Analysis and Mahalanobis Distance

PCA is a mathematical procedure that transforms a number of possible correlated variables into a smaller number of uncorrelated variables called principal components. These components are chosen by its variability in the data; the bigger the variability, the better it qualifies as principal component.

Traditionally, principal component analysis is performed on a square symmetric matrix. It can be a SSCP matrix (pure sums of squares and cross products), Covariance matrix (scaled sums of squares and cross products), or Correlation matrix (sums of squares and cross products from standardized data). [11]

Principal objectives of PCA:

- To reduce attribute space from a larger number of variables to a smaller number of factors.
- To reduce dimension and there is no guarantee that the dimensions are interpretable.
- To select a subset of variables from a larger set, based on which original variables have the highest correlations with the principal component.

In the other hand, there is the Mahalanobis distance (MD) which is the distance between two points in multivariate space, even for correlated points.

$$d(MD) = \sqrt{(x_b - x_a)^T * C^{-1} * (x_b - x_a)} \quad (32)$$

where x_b and x_a are two points and C is the covariance matrix. Nevertheless, there is another form which uses the central mean.

$$d(MD) = \sqrt{(x_i - \bar{x})^T * C^{-1} * (x_i - \bar{x})} \quad (33)$$

The Mahalanobis distance measures distance relative to the centroid. The centroid is a point in multivariate space where all means from all variables intersect. The larger the MD, the further away from the centroid the data point is.

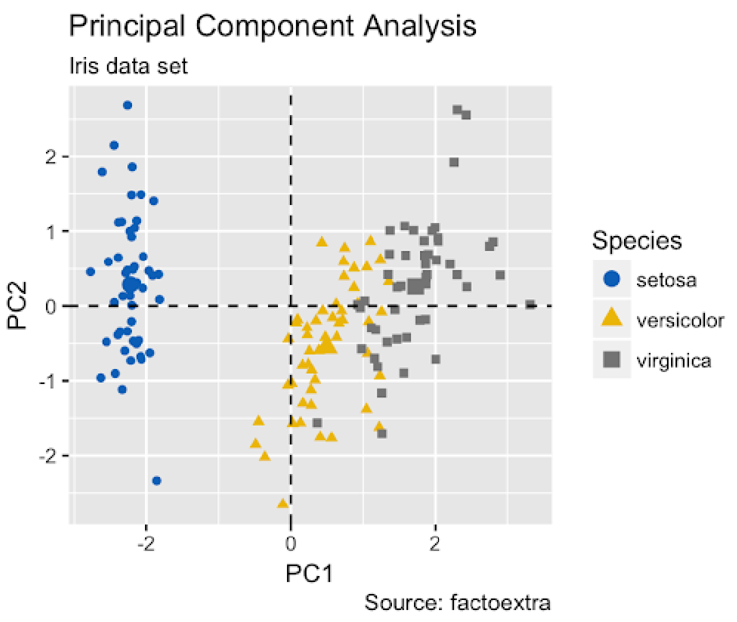


Figure 15: Principal Component Analysis example

The most common use for the Mahalanobis distance is to find multivariate outliers, which indicates unusual combinations of two or more variables. [10]

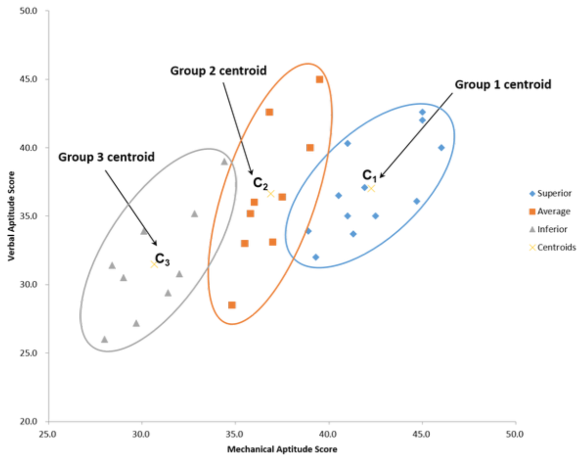


Figure 16: Mahalanobis Distance

5 Comparison of methods

SIMILARITIES	DIFFERENCES
They are supervised machine learning algorithms that can be used to solve classification problems.	Random Forest is the only one that cannot be used to solve regression problems.
All of them look for a function that classifies the elements in the best way to have the least possible error.	The logistic regression method usually presents lower performance indices compared to the other methods due to the large number of assumptions it makes.
	They present different execution time, being the Random Forest and SVM the fastest and most efficient classification methods, then it goes KNN and in the last place it is logistic regression method.

6 Results

After a proper review of the theory, each algorithm was implemented and the following are the results we obtained. For each method besides the accuracy in training and test, there are also three other statistical measures that are presented. For better understanding lets define them:

- Precision: In machine learning, precision represents the percentage of correctly classified examples of a class over the number of examples classified in that class.
- Recall: the recall metric of a class expresses how well the model can predict it. This is a value between 0 and 1, and the closer it is to one, the more the model is a good classifier. In few words it indicates the percentage of correctly classified examples of a given class.
- F1-score: is used as a statistical measure to rate performance. It is a balanced mean between precision and recall. This measure is mostly used when for example you get a higher precision but a lower recall.
- Accuracy: indicates the total number of correct predictions divided by the total number of predictions made.

For a better understanding of these metrics lets consider the image in figure 17

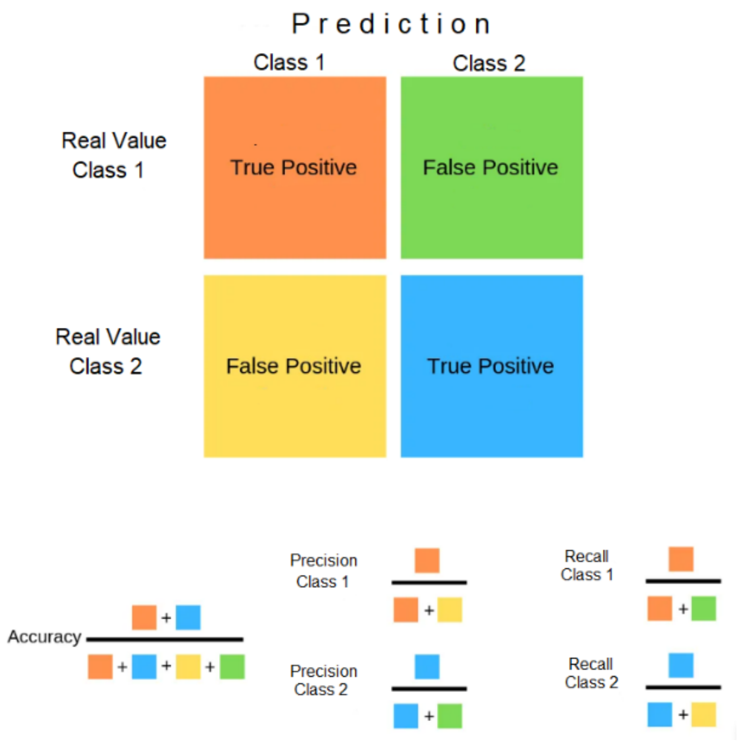


Figure 17: Statistical measures explanation

[1]
Therefore, one of these possible cases can be given for each class.

- High precision and high recall: this will indicate that the model handles that class perfectly, or at least, does it very well.
- High precision and low recall: The model does not predict class very well, but when it does it is highly reliable.
- Low precision and high recall: The model predicts the class well, but also includes samples from other classes by mistake, which makes it unreliable.
- Low precision and low recall: The model fails to classify correctly.

6.1 Logistic Regression

After having processed the data properly, it was possible to reduce the size of 20 explanatory variables to six, as there was a correlation between all the others, which turned out to be the only significant ones for the model. Thus, in table 2 we can see the coefficients obtained for each of them, which will indicate how much the probability of an individual belonging to one group or the other is increased or decreased.

Table 2: Coefficientes of the model

Variable name	**Coefficient β**
Status of existing checking account	0.64421019
Duration in month	-0.4150148
Credit history	0.40488193
Savings account/bonds	0.28306031
Installment rate in % of disposable income	-0.15594317

Subsequently having defined the variables, we proceeded to make the training and the data testing and with this last one we made a confusion matrix that is presented in table 3, where it can be observed that it seems to be classifying well due to the high values in the main diagonal, however, the values outside this one indicate that mistakes were made at the moment of classifying in the different groups.

Table 3: Confusion matrix and accuracy for Logistic Regression

	Actual Reject	**Actual Accept**
Predicted Reject	48	19
Predicted Accept	36	97
Train-accuracy	0.33125	
Test-accuracy	0.725	

If we look at the metrics registered in table 4 we can see that from the prediction made, 57% was done correctly for the reject data, but with the recall metric we see that these only represent 72% of the real data, however, we see that the F1-score, that is, the combination of both measures, for the reject class is very low, which would indicate me that there are failures in the classifier. Now if we consider the other class, it is more accurate, it means that 84% of the prediction is correct and represents 73% of the real data within the group considered and by having a combination of both, we see that 78%is obtained, which tells me that the model is classifying one class better than another.

Table 4: Effectiveness metrics for Logistic Regression

	Precision	**Recall**	**F1-Score**
Reject	0.57	0.72	0.64
Accept	0.84	0.73	0.78

Now, focusing on the accuracy obtained for the training and testing sets, we see that overall 33% and 72.5% are well predicted, respectively, which is somewhat suspicious since it is expected that in training will have a higher adjustment and in testing a slightly lower value, so let's see the possible reasons why this is happening:

- There is test data in the training set.
- The training data are unbalanced and do not behave in the same way as the testing data, so it could generate a bias in the ranking.
- The division into testing and training groups was not done randomly.
- An underfitting problem is occurring, which would indicate that it is not a good model because it is not capable of training correctly, which would generate errors in the testing set.

Finally, let's see figure 18 in which the ROC curve is presented as an indicator of how much the model is capable of distinguishing between classes. In this case, it has a value of 71%, which means that it can distinguish between classes even though it makes mistakes.

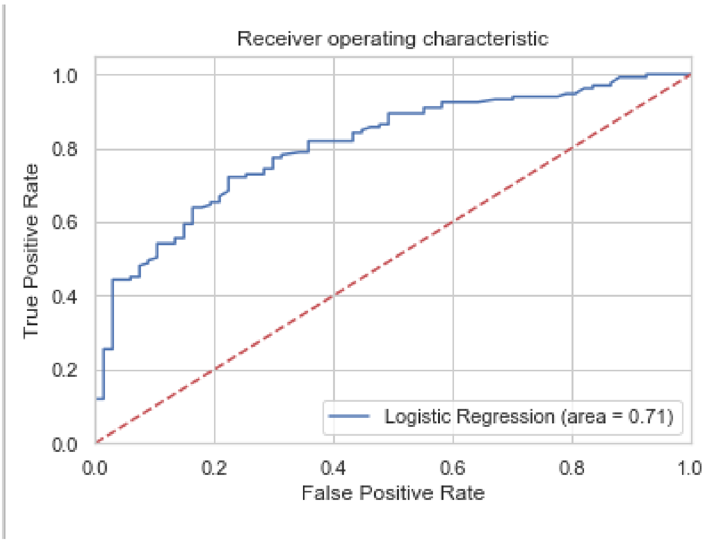


Figure 18: Logistic Regression

[4]

6.2 KNN

We can observe in Table – that for both, reject and accept, precision and recall give values with a significant difference, therefore we ought to take into account the F1-score; this measure gives 0.43 for reject and 0.66 for accept, in which we can conclude that when $k = 2$, this method doesn't classify correctly enough each class.

In the other hand, in Table – we observe the confusion matrix and its accuracy, in which we observe a good performance in training but it doesn't generalizes good enough when it is tested. It shows that the method is 58% accurate, not ideal for classifying credit scoring.

Table 5: KNN with k = 2

	Precision	**Recall**	**F1-Score**
Reject	0.33	0.60	0.43
Accept	0.80	0.56	0.66

Table 6: Confusion matrix and accuracy for KNN with k = 2

	Actual Reject	**Actual Accept**
Predicted Reject	32	21
Predicted Accept	64	83
Train-accuracy	0.79	
Test-accuracy	0.58	

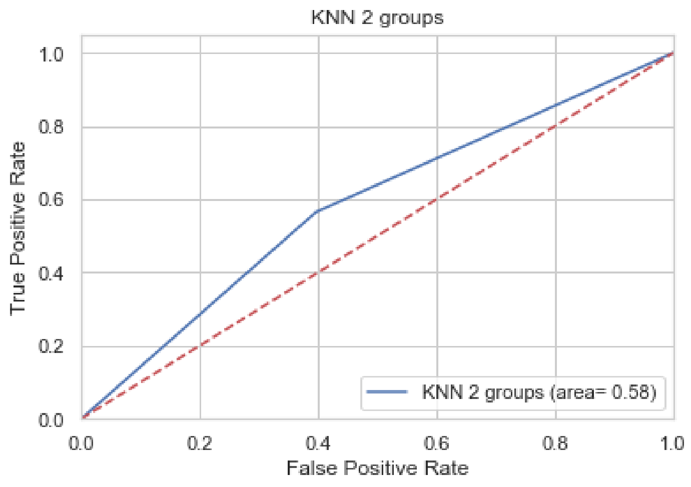


Figure 19: KNN 2 groups ROC Curve

Now, changing $k = 4$ we can observe in Table – that for reject class, precision and recall gives a low rate, that means that it doesn't classify in a proper way this class. Nevertheless, for accept class, precision and recall gives a high rate, therefore it classifies it properly.

In addition, in Table – we observe the confusion matrix and its accuracy, in which we observe a good performance in training but it stills doesn't generalizes good enough when it is tested. It shows that the method is 58% accurate, not ideal for classifying credit scoring.

Table 7: KNN with k = 4

	Precision	**Recall**	**F1-Score**
Reject	0.33	0.39	0.36
Accept	0.70	0.65	0.68

Table 8: Confusion matrix and accuracy for KNN with k = 4

	Actual Reject	**Actual Accept**
Predicted Reject	24	38
Predicted Accept	48	90
Train-accuracy	0.77	
Test-accuracy	0.58	

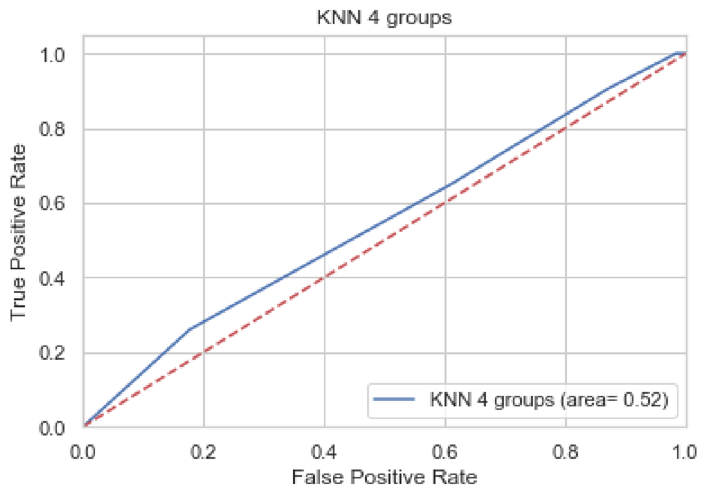


Figure 20: KNN 4 ROC Curve

To finalize, lets observe both ROC curves, for $k = 2$ the blue line is significantly far from the red one, but for $k = 4$ the blue line is getting closer to the red one, in which we can conclude that this model is the best one of these two.

6.3 Support Vector Machine

Initially, lets consider which parameter changes will indicate the highest accuracy when altering the parameters of the Support Vector Machine algorithm, as shown in Table 9.

Now, lets consider the implementation of the Support Vector Machine whose accuracy score it the highest (C value of 100, Gamma value of 0.0001 and rbf Kernel). It can be concluded, according to the performance measurements scores (precision, recall and F1-Score), that this algorithm are is very efficient when classifying individuals whose credit offering is accepted but is not successful when classifying individuals whose credit request is rejected. As what can be seen in the confusion matrix, all of the instances of the accepted credit offerings are correctly classified and all of the instances of the rejected credit offerings are incorrectly classified.

Accuracy	***C***	***Gamma***	***Kernel***
0.687	1.00	0.001	rbf
0.701	1.00	1.00	rbf
0.669	10	0.001	rbf
0.678	10	0.0001	rbf
0.686	100	0.001	rbf
0.709	100	0.0001	rbf
0.703	1000	0.001	rbf
0.712	100	0.0001	rbf

Table 9: Parameter Changes

Table 10: Support Vector Machine with C=100, Gamma= 0.0001 and Kernel Type: rbf

	Precision	**Recall**	**F1-Score**
Reject	0.00	0.00	0.00
Accept	0.72	1.00	0.84

Table 11: Confusion matrix and accuracy for Support Vector Machine with C=100, Gamma= 0.0001 and Kernel Type: rbf

	Actual Reject	**Actual Accept**
Predicted Reject	0	56
Predicted Accept	0	144
Train-accuracy	1.00	
Test-accuracy	0.72	

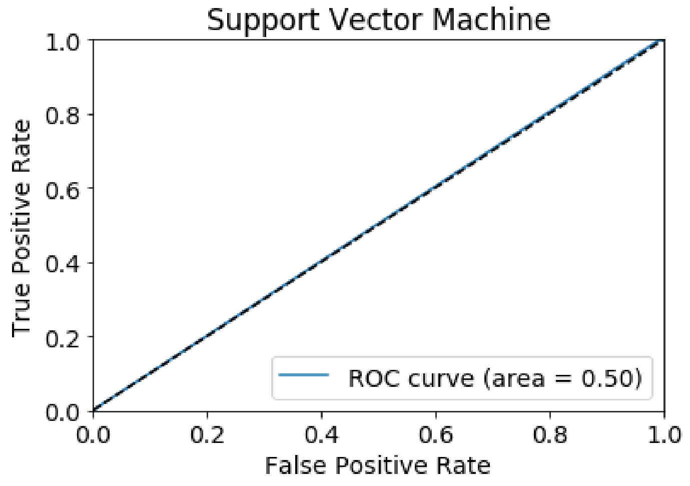


Figure 21: Support Vector Machine

6.4 Random Forest

Now, lets consider the implementation of the Random Forest algorithm for the continuous variables. It can be concluded, according to the performance measurements scores (precision, recall and F1-Score), that this algorithm are is very efficient when classifying individuals whose credit offering is accepted but is not as successful when classifying individuals whose credit request is rejected. Additionally, by taking into account the testing accuracy of the algorithm, which was 0.69, we can conclude that its performance is better than other algorithms such as the Support Vector Machine and the KNN.

Table 12: Random Forest for Continuous Variables

	Precision	**Recall**	**F1-Score**
Reject	0.48	0.43	0.46
Accept	0.77	0.80	0.78

Table 13: Confusion matrix and accuracy for Random Forest with all Variables

	Actual Reject	**Actual Accept**
Predicted Reject	26	34
Predicted Accept	28	112
Train-accuracy	0.89	
Test-accuracy	0.69	

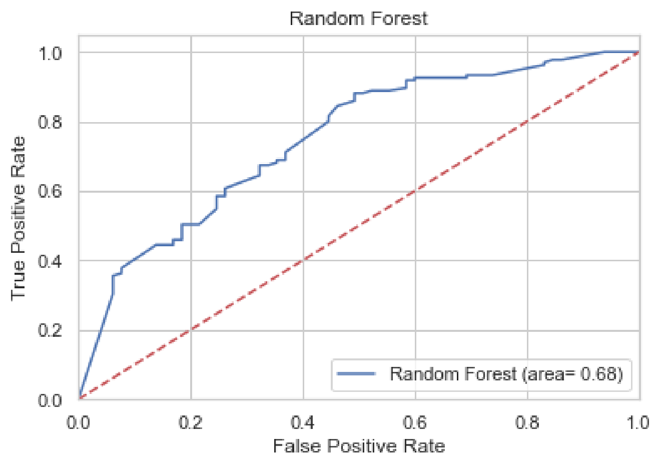


Figure 22: Random Forest ROC Curve for Continuous Variables

Now, lets consider the implementation of the Random Forest algorithm considering all the variables available on the data set. It can be concluded, according to the performance measurements scores (precision, recall and F1-Score), that this algorithm are is extremely efficient when classifying individuals whose credit offering is accepted and whose credit request is rejected. Additionally, by taking into account the testing accuracy of the algorithm, which was 1.00, we can conclude that its performance is much better than that of any other algorithm presented in this paper.

Table 14: Random Forest for all Variables

	Precision	**Recall**	**F1-Score**
Reject	1.00	1.00	1.00
Accept	1.00	1.00	1.00

Table 15: Confusion matrix and accuracy for Random Forest with all Variables

	Actual Reject	**Actual Accept**
Predicted Reject	66	0
Predicted Accept	0	134
Train-accuracy	1.00	
Test-accuracy	1.00	

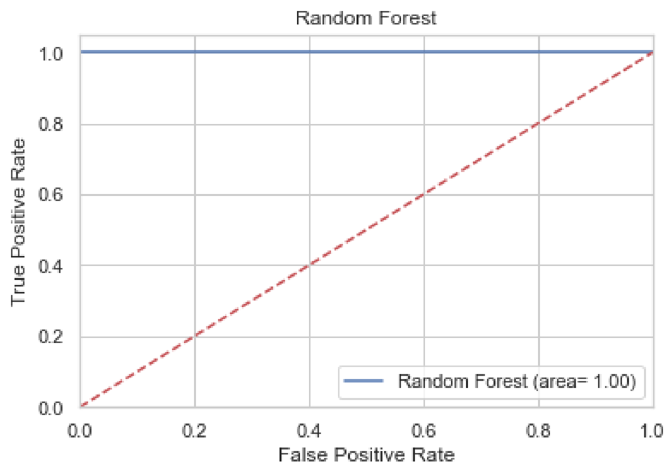


Figure 23: Random Forest ROC Curve for all Variables

7 Conclusion

References

[1] Juan Ignacio Bagnato. Clasificación con datos desbalanceados. url-https://bit.ly/3dMj6UC, 2019.

[2] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[3] Professor Dr. Hans Hofmann. German credit data. *Institut fur Statistik und Okonometrie Universitat Hamburg*, 2000.

[4] Jih-Jeng Huang, Gwo-Hshiung Tzeng, and Chorng-Shyong Ong. Marketing segmentation using support vector clustering. *Expert systems with applications*, 32(2):313–317, 2007.

[5] C.A. López Miranda. Modelo predictivo de riesgo de morosidad para créditos bancarios usando datos simulados. Junio 2013.

[6] Yanjun Qi. Random forest for bioinformatics. In *Ensemble machine learning*, pages 307–323. Springer, 2012.

[7] Sebastian Raschka and Vahid Mirjalili. *Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2*. Packt Publishing Ltd, 2019.

[8] Tavish Srivastava. *Introduction to k-Nearest Neighbors: A powerful Machine Learning Algorithm (with implementation in Python R)*, 2018.

[9] Lyn C Thomas, David B Edelman, and Jonathan N Crook. *Credit scoring and its applications*. SIAM, 2002.

[10] P Varmuza, K. Filzmoser. Introduction to multivariate statistical analysis in chemometrics. *CRC Press*.

[11] Pascal Wallisch. Principal component analysis. *Science Direct, Comprehensive Chemometrics*, 2014.

[12] Tony Yiu. Understanding logistic regression. urlhttps://bit.ly/2X0CJ4v, 2019.

[2] [6] [9] [7]