Machine Learning in Credit-Scoring

Isabella Arango, Sophia Giraldo, Valentina Yusty Mosquera

June 3, 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. 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 (11). 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

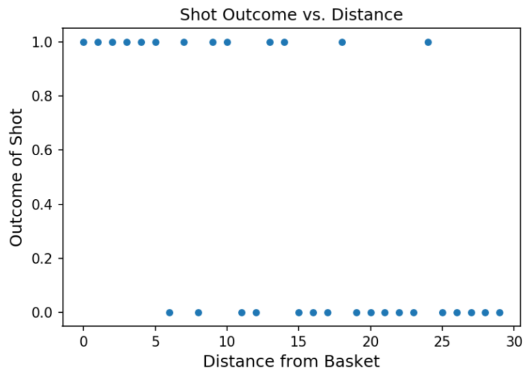


Figure 3: Shot Outcome vs. Distance from Basket

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 $\beta_0 = 2.5$ and $\beta_1 = -0.2$, 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

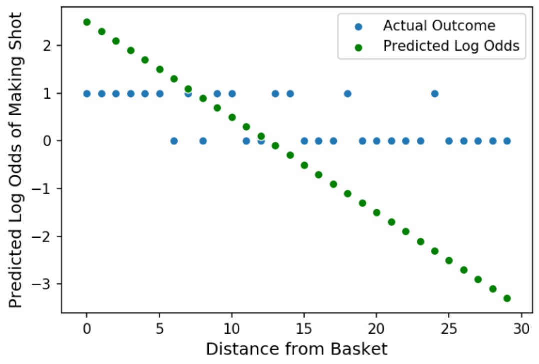


Figure 4: Linear Regression prediction

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

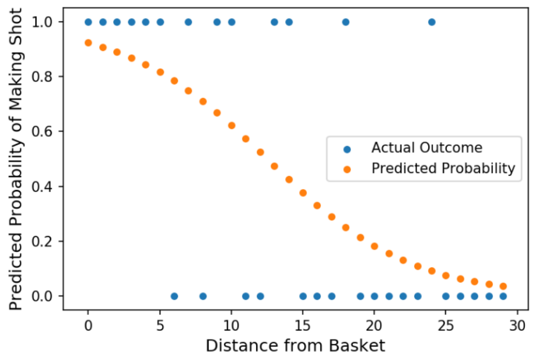


Figure 5: Logistic Regression prediction

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

```
$data \leftarrow Load\ data$
$X \leftarrow explanatory\ variables$
$Y \leftarrow response\ variable$
$ypred \leftarrow initialize\ vector$
$pv \leftarrow\ true$
while pv is true

for all xi vector in X do
    i) Calculate the regression coefficients using the maximum likelihood method.
end

ii) Find the maximum p-value

if maximum p-value $> 0.05$ then
    iii) Delete variable xi from X
else pv = false
end

end

iv) Apply the inverse logit function that will give you a probability p for each observation.

for all p do
    if  p >= 0.5 then
        ypred = 1
    else ypred = 0
    end
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. Srivastava 2018

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.

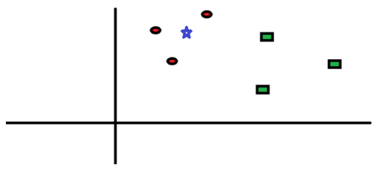


Figure 6: Scenario 1

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.

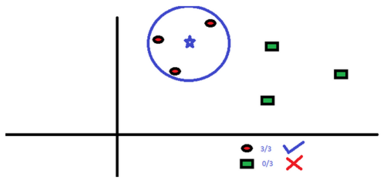


Figure 7: Scenario 2

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 ← Load data
k ← 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

According to Cristianini and Shawe-Taylor (2000), Support Vector Machines is a discriminative classifier whose classification is based on the construction, in a high or infinite dimensional space, of a hyperplane or a set of hyperplanes. By definition, the hyperplane is used 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.

2.3.1 Linear Classification

According to Cristianini and Shawe-Taylor (2000), a binary classification problem involves the use of a classification decision rule to classify elements into two distinct groups. For notation purposes let $X \in \mathbb{R}^n$, denote the input space and $Y = \{-1, 1\}$ denote the output domain. Let an example from the data set be represented as the pair $(\boldsymbol{x}_i, y_i)$, where $\boldsymbol{x}_i = (x_1, .., x_n)$ represents the input vector and y_i the respective output value. Lastly, let the training data be denoted as $S = ((\boldsymbol{x}_1, y_1)), ..., (\boldsymbol{x}_l, y_l)) \in (X, Y)^l$ and let the testing data be denoted as $S' = ((\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_k, y_k)) \in (X, Y)^k$.

Given a testing example $(\boldsymbol{x}_i, y_i) \in S'$, binary classification is performed by using a function $f : X \in \mathbb{R}^n \to \mathbb{R}$, denoted as

$$f(\boldsymbol{x}_i) = \langle \boldsymbol{w}^T \cdot \boldsymbol{x}_i \rangle + b, \tag{1}$$

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 $\boldsymbol{x}_i$ in (1) is positive, then the testing example is assigned to the positive class and vice versa for the negative class.

Note that by substituting (12) into (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, denoted as

$$f(\boldsymbol{x}_i) = \sum_{j=1}^{l} \alpha_j y_j \langle \boldsymbol{x}_j \cdot \boldsymbol{x}_i \rangle + b, \tag{2}$$

where $\alpha_j \geq 0$ are the Lagrange multipliers. When dealing with linear classification, functional and geometric margins can be used as a tool to evaluate the performance of the algorithm. The functional margin of an example $(\boldsymbol{x_i}, y_i)$ with respect to the hyperplane $(\boldsymbol{w}, b)$ can be calculated as

$$\hat{\gamma}_i = y_i(\langle \boldsymbol{w}^T \cdot \boldsymbol{x}_i \rangle + b).$$

Note that if the value $\hat{\gamma}_i$ is greater than zero, 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

$$\hat{\gamma} = \min_{i=1 \ldots l} \hat{\gamma}_i.$$

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

$$\gamma_i = y_i \left(\left(\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} \right)^T \cdot \boldsymbol{x}_i + \left(\frac{b}{\|\boldsymbol{w}\|} \right) \right).$$

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

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

2.3.2 Kernel-Induced Feature Spaces

In some cases, the linear combination of independent variables cannot effectively classify the response variable. In this case, the idea of more abstract features of the data is considered. According to Cristianini and Shawe-Taylor (2000), by using the kernel representation of the data, linear classification learning machines can better classify data on a high dimensional feature space.

The purpose of kernel representation is to change the way data is represented in order to obtain an objective function that can fully distinguish different classes. The preprocessing method consist on changing the representation of all $\boldsymbol{x}_i \in X \ i = \{1, ..., n\}$:

$$\boldsymbol{x}_i = (x_1, ..., x_n) \to \boldsymbol{\phi}(\boldsymbol{x}_i) = (\phi_1(\boldsymbol{x}_i), ..., \phi_d(\boldsymbol{x}_i)) \quad d \leq n. \tag{3}$$

The process represented in (3) is equivalent to mapping the input space X into a feature space represented by $F = \{\boldsymbol{\phi}(\boldsymbol{x}_i) | \boldsymbol{x}_i \in X\}$. Let $\boldsymbol{\phi} : X \to F$ represent a non-linear map from the input space to a feature space.

A reason why feature mapping occurs is due to the need of machines to classify non-linear relationships. Feature mapping allows non-linearly separable data to become linearly separable by rewriting the data into a new representation. According to Cristianini and Shawe-Taylor (2000), this process consists on using a linear machine to classify the data on the feature space, which was initially applied a non-linear mapping. Given a testing example $(\boldsymbol{x}_i, y_i) \in S'$, the linear machine used to classify the testing example on feature space F is denoted as

$$f(\boldsymbol{x}_i) = \langle \boldsymbol{w}^T \cdot \boldsymbol{\phi}(\boldsymbol{x}_i) \rangle + b, \tag{4}$$

where $(\boldsymbol{w}, b) \in \mathbb{R}^d * \mathbb{R}$ are parameters that control the decision rule in the feature space and are learned from a S.

Due to the fact that linear learning machines can also be expressed in dual representation, (4) can be rewritten as the inner product between the input vector from a testing example $(\boldsymbol{x}_i, y_i) \in S'$ and the input vector from the training points in S in the feature space F:

$$f(\boldsymbol{x}_i) = \sum_{j=1}^{l} \alpha_j y_j \langle \boldsymbol{\phi}(\boldsymbol{x}_j) \cdot \boldsymbol{\phi}(\boldsymbol{x}_i) \rangle + b. \tag{5}$$

According to Cristianini and Shawe-Taylor (2000), for all pairs of examples in the input space, a kernel function, K, can be represented as:

$$K(\boldsymbol{x}, \boldsymbol{z}) = \langle \boldsymbol{\phi}(\boldsymbol{x}) \cdot \boldsymbol{\phi}(\boldsymbol{z}) \rangle. \tag{6}$$

Hence, by substituting (6) into (5), we obtain

$$f(\boldsymbol{x}_i) = \sum_{j=1}^{l} \alpha_j y_j K\langle \boldsymbol{x}_j, \boldsymbol{x}_i \rangle. \tag{7}$$

2.3.3 Support Vector Classification: Maximal Margin Classifier

A frequently used model of Support Vector classification for linearly separable data on the input space is denoted as the maximal margin classifier. According to Cristianini and Shawe-Taylor (2000), the main idea associated to the maximal margin classifier is to correctly classify data by using a maximal margin hyperplane.

Given a linearly separable training sample S, the optimization problem associated to the maximal margin classifier is

$$\begin{aligned} \max_{\gamma,w,b} \quad & \gamma \\ \text{s.t.} \quad & y_i(\langle \boldsymbol{w}^T \cdot \boldsymbol{x_i} \rangle + b) \geq \gamma \qquad i = 1, ...l \\ & \|\boldsymbol{w}\| = 1. \end{aligned} \tag{8}$$

The objective function of (8) is to maximize the geometric margin γ. The first constraint indicates that each training example must have functional margin of at least γ. The second constraint, a scale constraint, ensures that the functional margin is equal to the geometric margin. Since (8) is a non-convex optimization problem, it can be written as a convex-optimization problem with a convex quadratic objective and a linear constraint:

$$\begin{aligned} \min_{\gamma,w,b} \quad & \frac{1}{2} \|\boldsymbol{w}\|^2 \\ \text{s.t.} \quad & y_i(\langle \boldsymbol{w}^T \cdot \boldsymbol{x_i} \rangle + b) \geq 1 \qquad i = 1, ...l. \end{aligned} \tag{9}$$

(9), denoted as primal optimization problem for the maximal 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.

To convert the primal optimization for the maximal margin classifier into a dual optimization problem, let (9) be rewritten as

$$\begin{aligned} \min_{\gamma,w,b} \quad & \frac{1}{2} \|\boldsymbol{w}\|^2 \\ \text{s.t.} \quad & -y_i(\langle \boldsymbol{w}^T \cdot \boldsymbol{x_i} \rangle + b) + 1 \leq 0 \qquad i = 1, ...l. \end{aligned} \tag{10}$$

Constructing the Lagrangian of (10) we obtain

$$\mathcal{L}(\boldsymbol{w}, b, \alpha) = \frac{1}{2} \|\boldsymbol{w}\|^2 - \sum_{l=1}^{l} \alpha_i [y_i(\langle \boldsymbol{w}^T \cdot \boldsymbol{x_i} \rangle + b) - 1]. \tag{11}$$

The Karush-Kush-Tucker conditions associated with (10) are:

$$\frac{\partial \mathcal{L}(\boldsymbol{w}, b, \alpha)}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i=1}^{l} \alpha_i y_i \boldsymbol{x_i} = 0 \Longrightarrow \boldsymbol{w} = \sum_{l=1}^{l} \alpha_i y_i \boldsymbol{x_i}, \tag{12}$$

$$\frac{\partial \mathcal{L}(\boldsymbol{w}, b, \alpha)}{\partial b} = \sum_{l=1}^{l} \alpha_i y_i = 0, \tag{13}$$

$$\alpha_i[-y_i(\langle \boldsymbol{w}^T \cdot \boldsymbol{x_i} \rangle + b) + 1] = 0 \quad i = 1, ...l, \tag{14}$$

$$\alpha_i \geq 0, \quad i = 1, ...l.$$

Note that the training examples for which the value of $\alpha_i \neq 0$ are denoted as support vectors. According to Tax and Duin (1999), support vectors are training examples which have the smallest geometrical margin to the hyperplane and greatly affect the direction of the hyperplane.

By substituting (12) into (11) and considering the result obtained from (13) the objective function for the dual function can be obtained:

$$\mathcal{L}(\boldsymbol{w}, b, \alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j \langle \boldsymbol{x}_i^T \cdot \boldsymbol{x}_j \rangle. \tag{15}$$

Taking into account (15) and the previous constraints, the following dual optimization problem for the maximal margin classifier is proposed:

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

2.3.4 Support Vector Classification: Soft Margin Optimization

When the data cannot be linearly separated in the input space, Support Vector classification methods, such as soft margin optimization, are used. As mentioned in Cristianini and Shawe-Taylor (2000), unlike the maximal margin classifier, soft margin optimization can tolerate the presence of outliers and noise in the data

Taking into consideration (9) and allowing margin constraints to be violated by adding a slack variable we obtain

$$\begin{aligned} \min_{\gamma,w,b} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{l=1}^{l}\xi_i \\ \text{s.t.} \quad & y_i(\langle \boldsymbol{w}^T \cdot \boldsymbol{x_i}\rangle + b) \geq 1 - \xi_i \qquad i = 1, ...l \\ & \xi_i \geq 0 \qquad\qquad\qquad\qquad i = 1, ...l, \end{aligned} \tag{16}$$

where C hyperparameter and the slack variable, ξ, measures the distance of an incorrectly classified training point from its corresponding class's margin. Note that if the value of C is small, maximizing the margin is given more importance than the classification error, and vice versa, if the value of C is large. It is important to note that the objective function of the optimization problems (9) and (17) differentiate in the second term.

Similar to the process shown for the maximal margin classifier, the dual optimization problem for soft margin optimization can be written as

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

2.3.5 Pseudocode

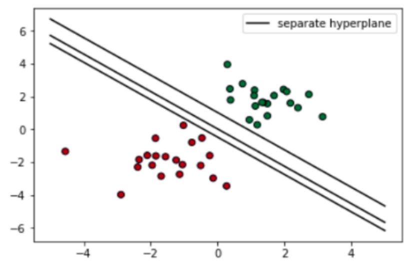
Algorithm 3 Support Vector Machine

```
  *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

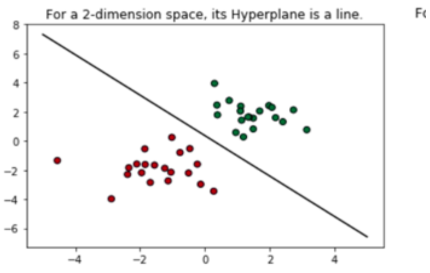
Now, let's consider an illustrative example to further understand the Support Vector Machine Algorithm. Let the green dots be the representation of a class, 1 , and let the red dots be a representation of an alternate class, -1. As seen in Figure (6), decision boundaries, the lines that separate the 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 8: Data Set



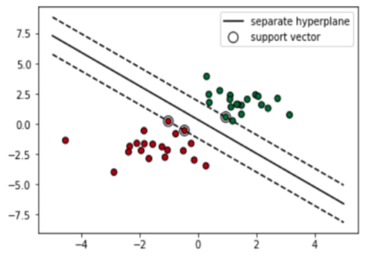
To find the decision boundary as shown in Figure (7), denoted as hyperplane, that best separates two classes, is the basic principle of Support Vector Machine.

Figure 9: 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 10: 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

As defined in Ali et al. (2012), Random Forest is a collection of decision tree classifiers. When testing S', each decision tree casts a unit vote for a class in which the testing example is classified based on its own criteria. Hence, the testing example will be classified to the class that has the greatest number of unit votes. It is important to note that each decision tree varies in structure and variable selection: the data and the features from which the decision trees are trained is made by randomly selected training data from the input space and randomly selected variables.

The main problem with the decision trees classifier is that they tend to produce overfitted models. According to Ali et al. (2012), overfitted models perform poorly when being tested since the classification of such models correspond very closely to the data from which they were trained. Therefore, a benefit of Random Forest is that the classification of a testing example is based on the classification obtained from multiple decision trees. Therefore, the use of Random Forest has 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 Louppe (2014), a Decision Tree is a tree-structured model: $\varphi : X \rightarrow Y$, where X is the input space and $Y = \{0, 1\}$ the output domain. A Decision Trees is represented as a binary tree composed by branches (segments that connect nodes), nodes (the position where the branches divide), leaf nodes (the node from which the tree ends) and a root node (the node from which the tree starts). Branches are represented as segments that connect nodes while nodes are represented as circles. In a Decision Tree, each node represents a subspace X_t and has a feature test $s(t)$. 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 into disjoint subspaces.

Each leaf node, or terminal nodes, are labeled with a guess value $y_t \in Y$. Lastly, the main objective of the Decision Tree is to find a tree-structured classifier that is able to distinguish between classes.

In a Decision Tree, each node has an impurity measure, denoted as $i(t)$. According to Louppe (2014), an impurity measure evaluates the goodness of the classification at a node. One of the most frequently used ways to calculate impurity at a node is by using the Gini impurity calculated as

$$i_G(t) = \sum_{i=1}^{C} p(c_i|t)(1 - p(c_i|t)),$$

where C is the number of classes in the output domain and $p(c_k|t)$ is a conditional probability representing the fractions of the examples labeled with class i at node t. It is important to note that the smaller the impurity measure on a given node, the purer the node.

The creation of a Decision Tree begins from a root node (representing the input space) that iteratively grows by dividing nodes into purer ones. Hence, if all X_t at a given node t belong to the same class, the pure node t will become a leaf node. On the other hand, if the X_t at a given node t contain examples that belong to more than one class, the node is divided into a binary split using a feature test. After a binary split, the node t is divided into a left node t_L and a right node t_R. In such cases, the impurity decrease at node t is calculated as:

$$\triangle i(s,t) = i(t) - p_L i(t_L) - p_R i(t_R),$$

where $p_R = \frac{N_{tR}}{N_t}$ and $p_L = \frac{N_{tL}}{N_t}$ are the fraction of training examples from X_t, with size N_t, that go to t_R and t_L, respectively. The previous statement indicates that the termination criteria for the Decision Tree is achieved once each leaf node is pure. As indicated in Ali et al. (2012), this termination criteria is what precisely makes the Decision Trees prone to overfitting: comprehensive changes in the structure of the decision tree can be caused by small changes in the data. 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 training examples into their respective classes are used.

The generalization error is used to calculate the accuracy of a Decision Tree. Given a testing data S', the generalization error of the model φ_L is calculated as:

$$Err(\varphi_L) = P(Y \neq \varphi_L(X)), \tag{17}$$

where P denotes the theoretical probability. However, since the theoretical probability is unknown, (17) is rewritten as:

$$\widehat{Err}(\varphi_L) = \frac{1}{N} \sum_{(\boldsymbol{x},y)\in S'} 1(y \neq \varphi((\boldsymbol{x})), \tag{18}$$

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

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

(18) measures the probability of misclassification of φ_L, the Decision Tree. Note that the purpose of the Decision Tree is not to make the most accurate prediction on the training set, but on the testing set because it will prove that the model is reliable.

For the generalization error of the model φ_L to be the smallest, the generalization error at each leaf node t has to be minimized as:

$$y_t = \arg\max_{c\in Y} P(Y = c|X \in X_t). \tag{19}$$

However, since the theoretical probability is unknown, (19) is rewritten as:

$$\hat{y}_t = \arg\max_{c\in Y} p(c|t).$$

2.4.2 Random Forest Algorithm

One of the benefits associated with a ensemble of Decision Trees is that, generally speaking, the expected generalization error of a set of Decision Trees is less than the generalization error of each individual Decision Tree. According to Louppe (2014), the Random Forest algorithm uses an ensemble method based on randomization. The idea behind this method is to generate different models from the training data by introducing random perturbations in order to make a prediction based on the set of predictions.

Let M denote a set of randomized models $\{\varphi_{S,\theta_m}|m = 1, .., M\}$ that were built from different random seeds θ_m and where learned from different random samples of training data S. When evaluating the Random Forest algorithm with S', the result obtained is computed by:

$$\Psi_{S,\theta_1,\dots,\theta_n} = \arg\max_{y\in c} \sum_{l=1}^{C} 1(\varphi_{S,\theta_m}(\boldsymbol{x_i}) = c). \tag{20}$$

As shown in (20), the label prediction for each $(\boldsymbol{x_i}, y_i) \in S'$ is obtained by considering the classification from the ensemble of Decision Trees. Given that each Decision Tree provides a unit vote for a label prediction, the testing example will be classified on the label prediction that has the majority of votes.

2.4.3 Pseudocode

Algorithm 4 Random Forest

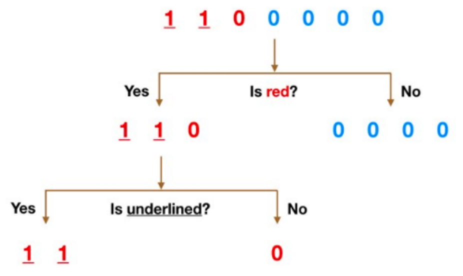
```
To generate c classifiers
for all i=1 to c do
    Randomly sample the training data D with replacement Di.
    Create a root node, Ni , containing Di.
    Call BuildTree Ni.
end for

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.

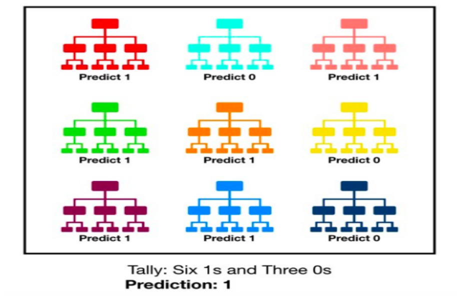
When testing an example from the testing data, since a Random Forest is a recompilation of Decision Trees, each Decision Tree has to cast a unit vote for a class in which the example is be classified. The class that achieves the most unit votes among the Decision Trees is that in which the example will 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

Figure 11: Decision Trees Separation of Classes



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 12: Unit Vote by Decision Trees



3 Variables description

The following description takes into account the dataset taken from Professor Dr. Hans Hofmann Hofmann 2000 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 : $x <$ 1year
   - A73 : $1 \leq x <$ 4years
   - A74 : $4 \leq x <$ 7years
   - A75 : $x \geq$ 7years

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). Wallisch 2014

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.

$$d(MD) = \sqrt{(x_b - x_a)^T * C^{-1} * (x_b - x_a)} \tag{21}$$

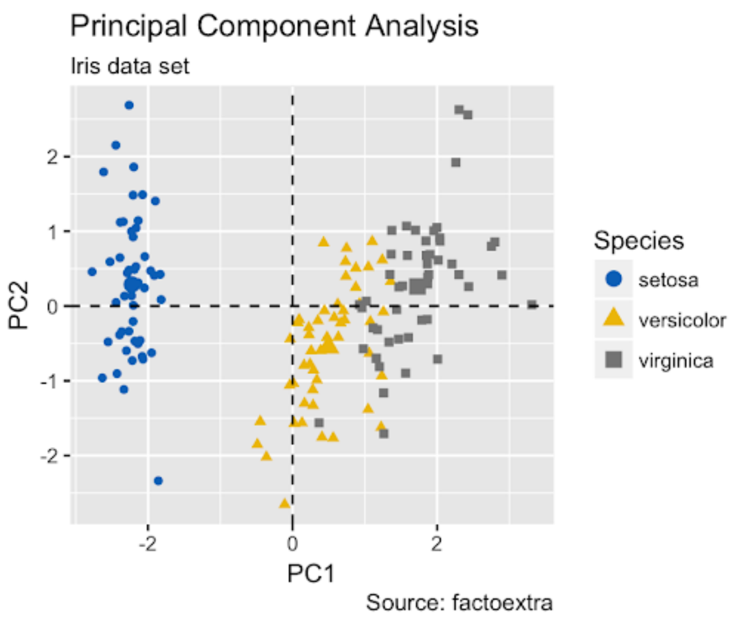


Figure 13: Principal Component Analysis example

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.

$$d(MD) = \sqrt{(x_i - \bar{x})^T * C^{-1} * (x_i - \bar{x})} \tag{22}$$

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.

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

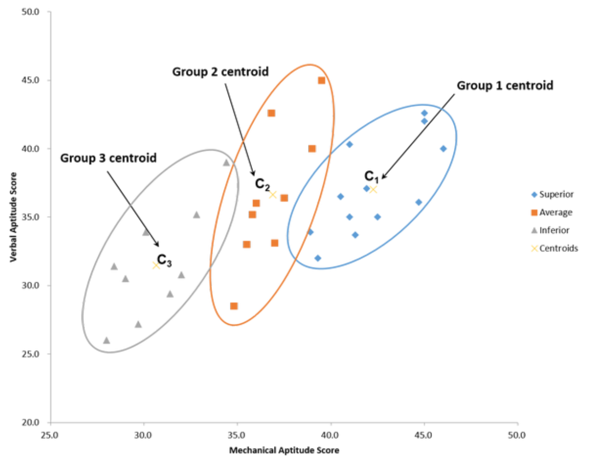


Figure 14: Mahalanobis Distance

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 Python 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

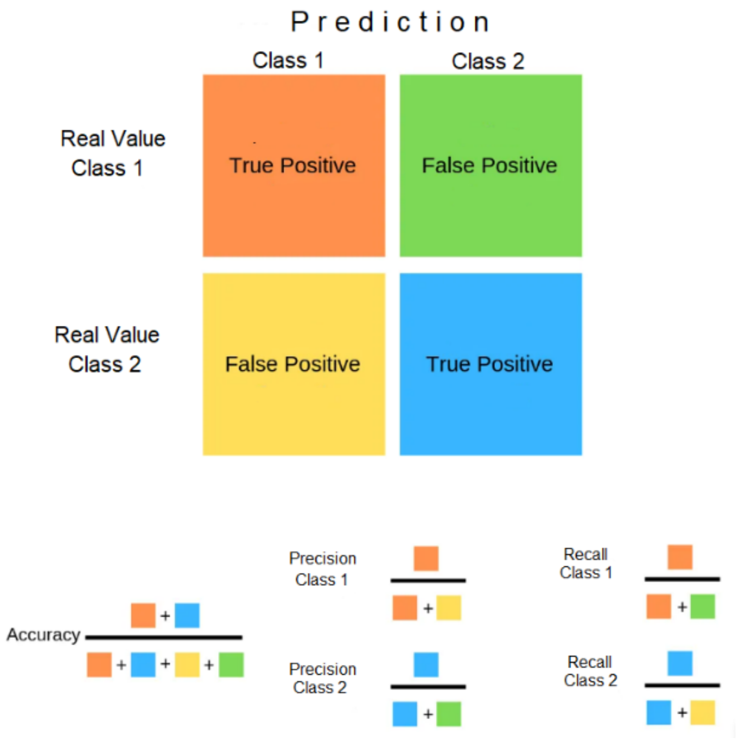


Figure 15: Statistical measures explanation (2)

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**
Intercept	0.20069843
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

So the model can be written as

$$p(x) = \frac{1}{1 + \exp^{-0.2-0.64x_1+0.41x_2-0.40x_3-0.28x_4+0.16x_5}}$$

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

	Predicted Reject	**Predicted Accept**
Actual Reject	48	19
Actual 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.

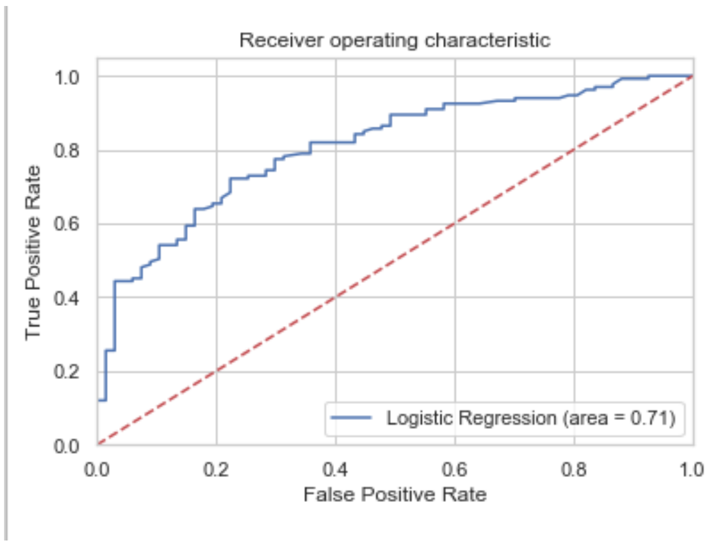


Figure 16: Logistic Regression

Huang et al. 2007

6.2 KNN

We can observe in Table 5, 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. We can conclude that when $k = 3$, it can tell if the credid is granted with 74% accurate.

In the other hand, in Table 6 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 62% accurate, not ideal for classifying credit scoring.

Table 5: KNN with k = 3

	Precision	**Recall**	**F1-Score**
Reject	0.32	0.29	0.30
Accept	0.73	0.75	0.74

Table 6: Accuracy for KNN with k = 3

Train-accuracy
0.811
Test-accuracy
0.62

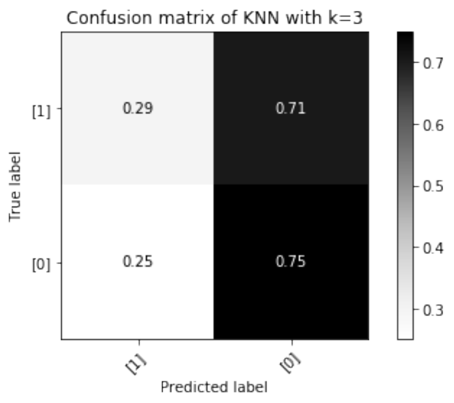


Figure 17: KNN Confusion matrix with k = 3

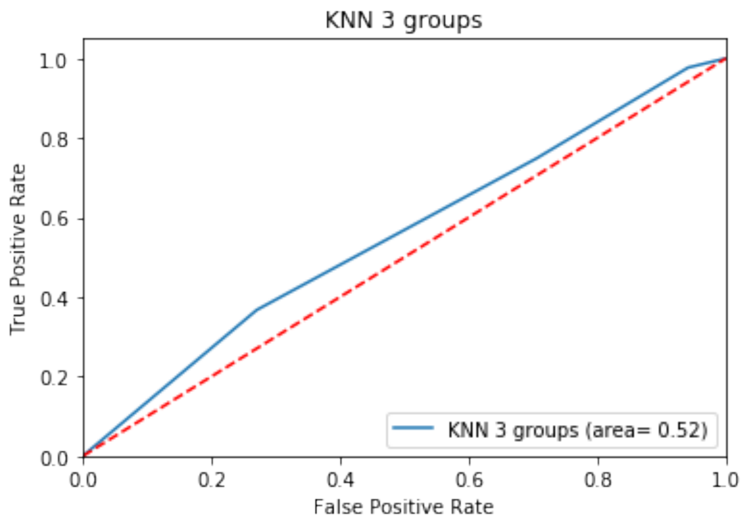


Figure 18: KNN 3 groups ROC Curve

Now, changing $k = 9$ we can observe in Table 7 that for reject class, precision and recall gives a low rate, that means that it doesn't classifies 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 8 we observe the confusion matrix and its accuracy, in which we observe a good performance in training and a better generalization of the model because the test accurate is higher and close to the training one. It shows that the method is 67% accurate, not ideal for classifying credit scoring but it improved.

Table 7: KNN with k = 9

	Precision	**Recall**	**F1-Score**
Reject	0.36	0.21	0.27
Accept	0.73	0.85	0.79

Table 8: Accuracy for KNN with k = 9

Train-accuracy
0.72
Test-accuracy
0.67

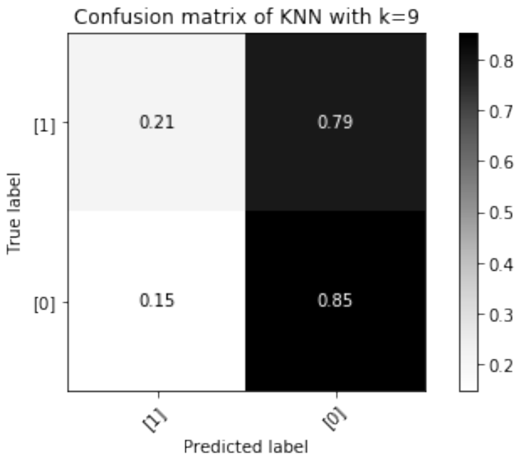


Figure 19: KNN 9 Confusion Matrix

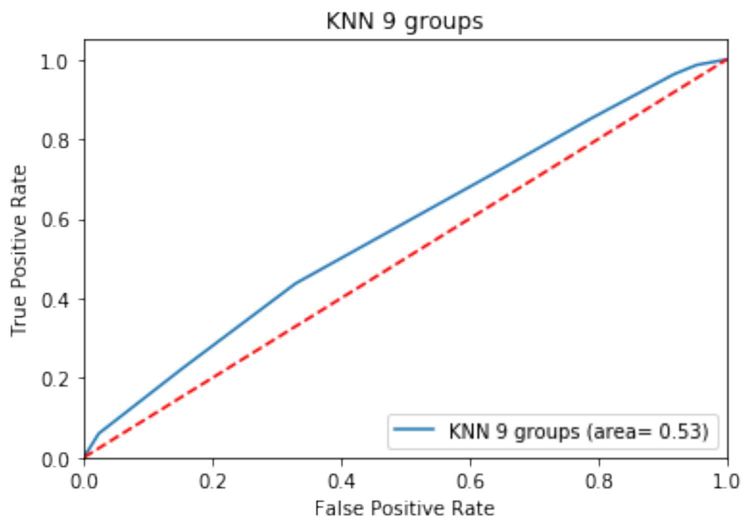


Figure 20: KNN 9 ROC Curve

To finalize, lets observe both ROC curves, in both graphs the blue line it is not close enough to the red one, so for choosing one of these models, we take into account the accuracy, then is better when $k = 9$.

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.

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

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.

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

	Precision	**Recall**	**F1-Score**
Reject	0.50	0.32	0.34
Accept	0.52	0.43	0.44

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

	Predicted Reject	**Predicted Accept**
Actual Reject	34	56
Actual Accept	42	68
Train-accuracy	1.00	
Test-accuracy	0.51	

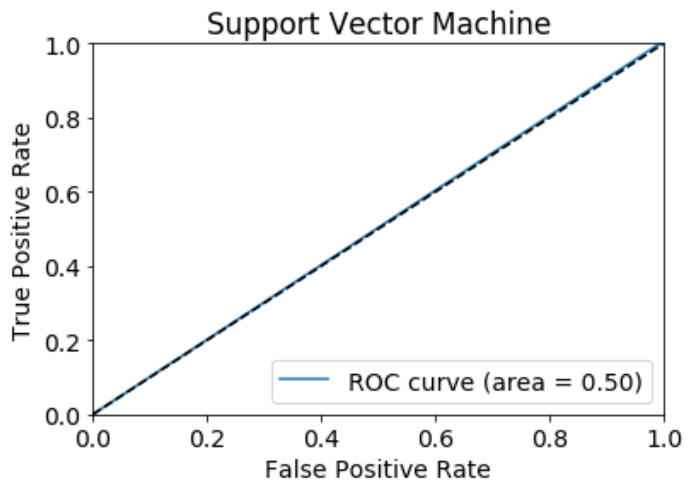


Figure 21: Support Vector Machine

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

	Predicted Reject	**Predicted Accept**
Actual Reject	41	19
Actual Accept	28	112
Train-accuracy	0.89	
Test-accuracy	0.77	

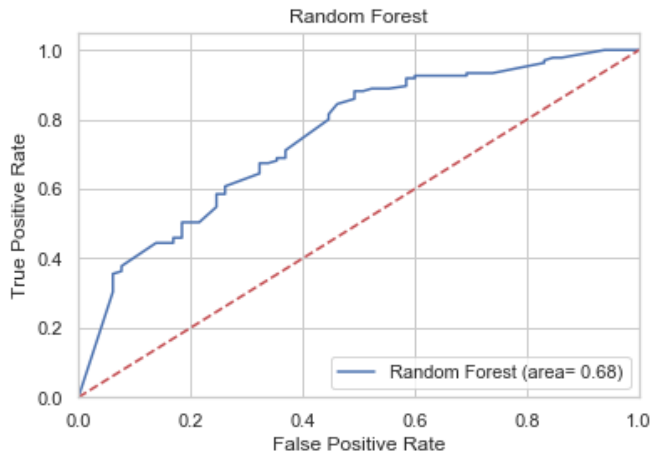


Figure 22: Random Forest ROC Curve for Continuous Variables

7 Matlab Results

7.1 Logistic Regression

Table 14: Coefficientes of the model

Variable name	**Coefficient β**
Intercept	-0.038
Status of existing checking account	-0.6028
Duration in month	0.0295
Credit history	-0.3458
Savings account/bonds	-0.2214
Present employment since	-0.2378
Installment rate in % of disposable income	0.2262
Other debtors/guarantors	-0.3974
Property	0.2213

So the model can be written as

$$p(x) = \frac{1}{1 + \exp^{0.04+0.60x_1-0.03x_2+0.35x_3+0.22x_4+0.24x_5-0.23x_6+0.40x_7-0.22x_8}}$$

Table 15: Confusion matrix and accuracy for Logistic Regression

	Predicted Reject	**Predicted Accept**
Actual Reject	122	16
Actual Accept	31	31
Train-accuracy	0.7712	
Test-accuracy	0.7650	

Table 16: Effectiveness metrics for Logistic Regression

	Precision	**Recall**	**F1-Score**
Reject	0.7974	0.8841	0.8385
Accept	0.6596	0.5000	0.5688

As what can be evidenced, the MATLAB result is slightly more accurate than the Python implementation and it present a good behavior of test and train accuracy. Therefore, specifically for this method, we suggest the use of MATLAB.

7.2 KNN

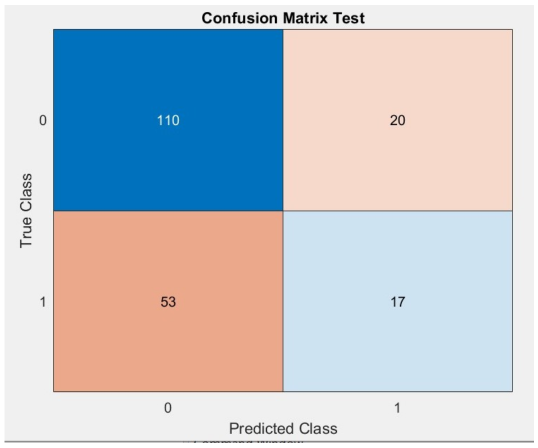


Figure 23: KNN confusion matrix for k=3

Table 17: KNN with k = 3

	Precision	**Recall**	**F1-Score**
Reject	0.67	0.84	0.75
Accept	0.46	0.25	0.32

Table 18: Accuracy for KNN with k = 3

Train-accuracy
0.80
Test-accuracy
0.63

As what can be evidenced, the MATLAB result is slightly more accurate than the Python implementation. Therefore, specifically for this method with k-=3, we suggest the use of MATLAB.

7.3 Support Vector Machine

Table 19: Support Vector Machine with C=100, Gamma= 0.0001 and Kernel Type: rbf

	Precision	**Recall**	**F1-Score**
Reject	0.76	0.80	0.78
Accept	0.40	0.36	0.38

Table 20: Confusion matrix and accuracy for Support Vector Machine with C=100, Gamma= 0.0001 and Kernel Type: rbf

	Predicted Reject	**Predicted Accept**
Actual Reject	115	29
Actual Accept	36	20
Train-accuracy	0.69	
Test-accuracy	0.67	

As what can be evidenced, the MATLAB result is slightly less accurate than the Python implementation. Therefore, specifically for this method, we suggest the use of Python.

8 Conclusion

As what can be evidenced, the Random Forest Algorithm is that which brigs the best results for tackling this specific project related to the acceptance and the rejection from a financial institution to a credit applicant. As stated in the literature, factors that may influence the decision of an entity to reject or accept a credit application are related to income, expenses, age and housing stratum. This findings are very important since practices such as credit scoring allow institutions to suffer from financial catastrophes and, inclusively, economic catastrophes. We would like to note that algorithms such as the Logistic Regression algorithms such a very decent performance when evaluating the testing class, and therefore can also be considered as successful.Lastly, the results obtained from the K Nearest Neighbor and the Support Vector Machine algorithms were that which performed the poorest, so we don't recommend this for further deployment.

References

Ali, Jehad et al. (2012). "Random forests and decision trees". In: *International Journal of Computer Science Issues (IJCSI)* 9.5, p. 272.

Cristianini, Nello and John Shawe-Taylor (2000). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods.* Cambridge University Press. ISBN: 9780511801389.

Hofmann, Professor Dr. Hans (2000). "German Credit data". In: *Institut fur Statistik und Okonometrie Universitat Hamburg.*

Huang, Jih-Jeng, Gwo-Hshiung Tzeng, and Chorng-Shyong Ong (2007). "Marketing segmentation using support vector clustering". In: *Expert systems with applications* 32.2, pp. 313–317.

Louppe, Gilles (2014). "Understanding random forests: From theory to practice". In: *arXiv preprint arXiv:1407.7502.*

Srivastava, Tavish (2018). *Introduction to k-Nearest Neighbors: A powerful Machine Learning Algorithm (with implementation in Python R)*. URL: `https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/`.

Tax, David MJ and Robert PW Duin (1999). "Data domain description using support vectors." In: *ESANN*. Vol. 99, pp. 251–256.

Varmuza K. Filzmoser, P (n.d.). "Introduction to Multivariate Statistical Analysis in Chemometrics". In: *CRC Press* ().

Wallisch, Pascal (2014). "Principal Component Analysis". In: *Science Direct, Comprehensive Chemometrics*.