Machine Learning in Credit-Scoring

Isabella Arango, Sophia Giraldo, Valentina Yusty Mosquera

April 14, 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 model is a statistical method used in various fields of work such as medicine, epidemiology, finance, psychology, among others, used mainly for the purpose of predicting or classifying specific situations. This algorithm allows us to estimate the relationship of influence existing between a group of independent variables, known as co-variates or risk factors, and a binary or dichotomous categorical dependent variable, that is, those that can only take two possible values (yes-no, true-false, face-seal, male-female...) and thus establish how each one of the covariates affects the probability of occurrence of the event under study.

One of the procedures that this method employs is the use of the logit function, which allows you to estimate the probability that event belongs to one of the two classification groups.

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

Figure 1: Logit function

where p(x) indicates the probability of occurrence or success of an event and can be calculated by applying the inverse of the logit function as

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

Figure 2: Probability of occurrence or success of an event

Therefore, it turns out to be a linear model for which the beta parameters have to be estimated by implementing the maximum likelihood method [4]. 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.

This algorithm uses a procedure very similar to the linear regression, but reduces the number of assumptions that it assumes, mainly eliminates the assumption of normality, which turns out to be a great advance for the methods of prediction, since in reality it is very difficult to find data with this characteristic. On the other hand, this method is especially useful when data follow a sigmoid distribution. However, despite being the most widely used method at present, it has some disadvantages, for example, if the risk factors are dependent on each other, it is not a very useful tool, as it is not capable of identifying the necessary characteristics of the redundant ones. In general, it is necessary to process the data before applying the algorithm.

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 has two main properties:

- **Lazy learning:** algorithm generalizes the data after a query is made.
- **Non-parametric learning algorithm:** computationally slower, but makes fewer assumptions about the data.

KNN uses "feature similarity" to predict the values of new datapoints. The following steps describe how does KNN really works:

1. As every algorithm, load a dataset.
2. Choose the value of the nearest data points (K). K can be any integer.
3. For each point in the dataset do:
   (a) Calculate the distance between the query example and the current example from the data.
   (b) Add the distance and the index of the example to an ordered collection.
   (c) Sort the ordered collection of distances and indices in ascending order.
   (d) Pick the first K entries from the sorted collection.
   (e) Get the labels of the selected K entries.
   (f) If regression, return the mean of the K labels - If classification, return the mode of the K labels.

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.

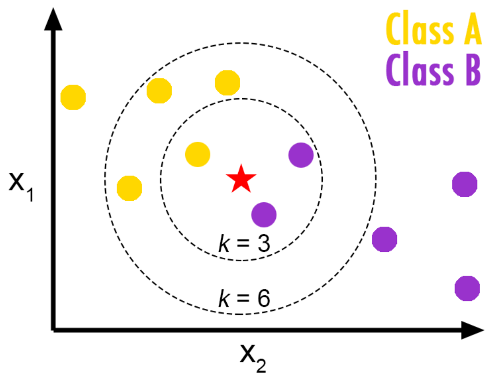


Figure 3: KNN training phase

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.

This method is applied to many areas, specially to bank systems and calculating credit ratings. [2]

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 $f : X \in \mathbb{R}^n \to \mathbb{R}$, 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).”

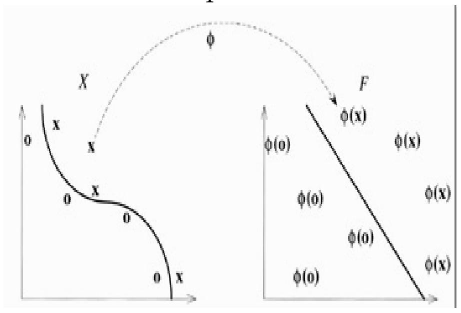
Ideally, a representation of the target function that matches the learning problem should be chosen. A common preprocessing method calls for altering the way in which data is represented in order to acquire a target function that can adequately distinguish between different classes. This is done by changing 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.

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

Figure 4: Feature map for a classification task



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

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:

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

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:

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

2.4 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 $\varphi : X \to Y$, 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:

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

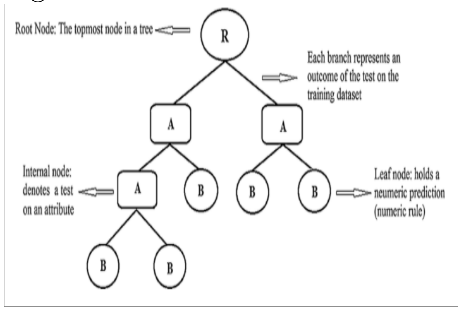
where C is the number of classes in the output domain and p(i) is the probability of picking label i at a node. The smaller the impurity measure of a given node, the purer the node and the more accurate predictions for all examples $(\boldsymbol{x_i}, y_i) \in S$ such that $\boldsymbol{x_i} \in X_t$.

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)".

Figure 5: 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

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:

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

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.

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). [9]

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.

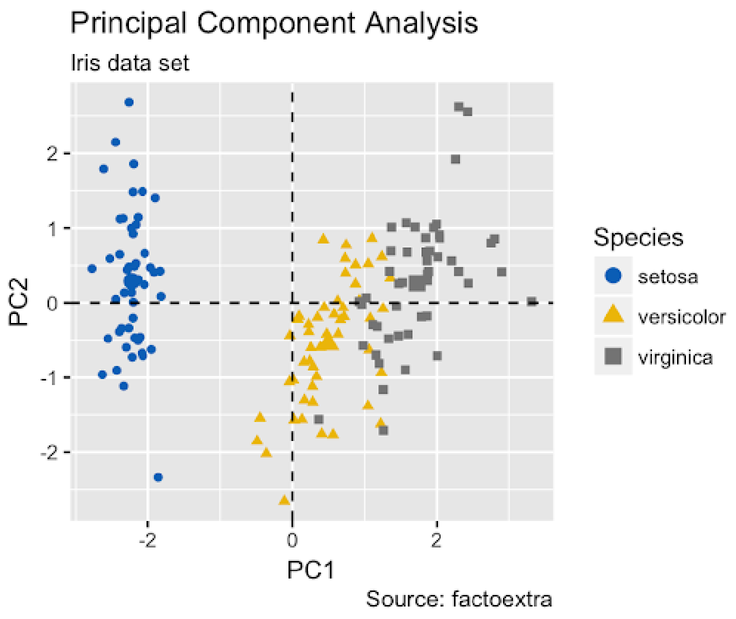


Figure 6: Principal Component Analysis example

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

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

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.

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

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.

The most common use for the Mahalanobis distance is to find multivariate outliers, which indicates unusual combinations of two or more variables. [8]

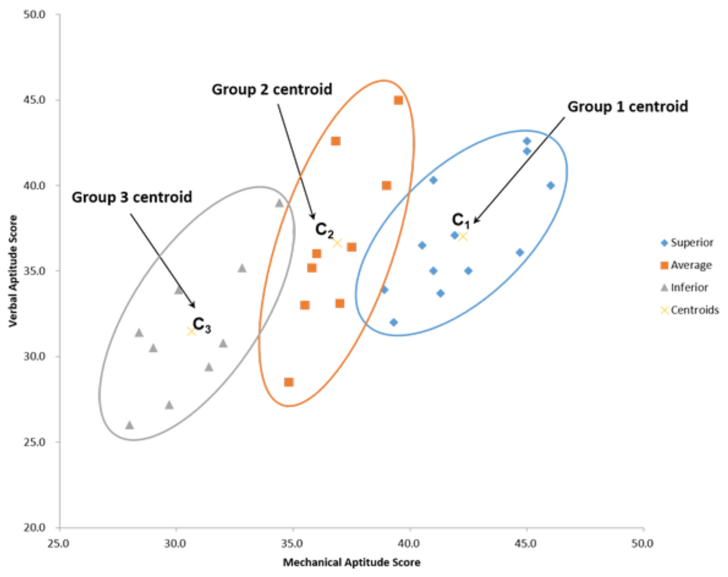
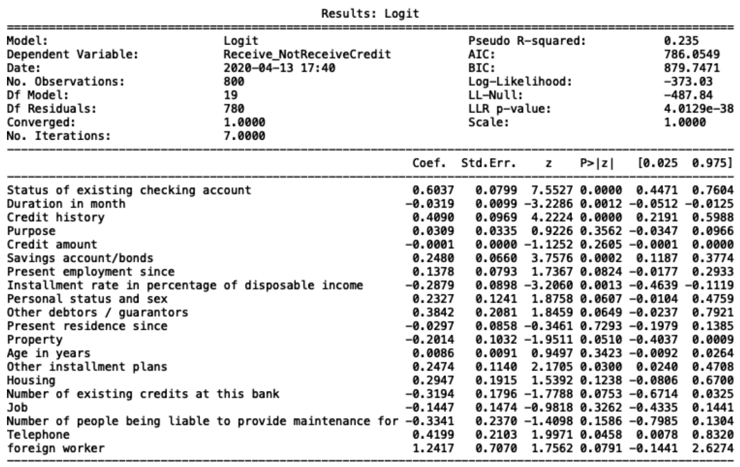


Figure 7: Mahalanobis Distance

5 Comparison of methods

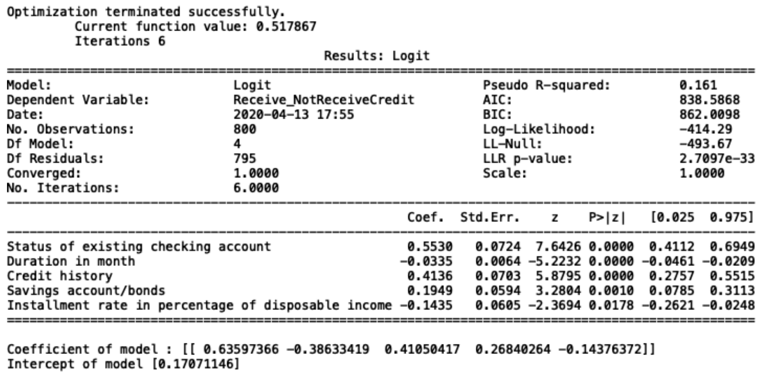
SIMILARITIES	DIFFERENCES
They are supervised machine learning algorithms that can be used to solve classification problems.	Random Forest is the only one that cannot be used to solve regression problems.
All of them look for a function that classifies the elements in the best way to have the least possible 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



```
                                          Results: Logit
==============================================================================================
Model:                   Logit                                Pseudo R-squared:      0.235
Dependent Variable:      Receive_NotReceiveCredit             AIC:                   786.0549
Date:                    2020-04-13 17:40                     BIC:                   879.7471
No. Observations:        800                                  Log-Likelihood:        -373.03
Df Model:                19                                   LL-Null:               -487.84
Df Residuals:            780                                  LLR p-value:           4.0129e-38
Converged:               1.0000                               Scale:                 1.0000
No. Iterations:          7.0000
----------------------------------------------------------------------------------------------
                                                          Coef.  Std.Err.    z    P>|z|  [0.025  0.975]
----------------------------------------------------------------------------------------------
Status of existing checking account                       0.6037  0.0799  7.5527 0.0000  0.4471  0.7604
Duration in month                                        -0.0319  0.0099 -3.2286 0.0012 -0.0512 -0.0125
Credit history                                            0.4090  0.0969  4.2224 0.0000  0.2191  0.5988
Purpose                                                   0.0309  0.0335  0.9226 0.3562 -0.0347  0.0966
Credit amount                                            -0.0001  0.0000 -1.1252 0.2605 -0.0001  0.0000
Savings account/bonds                                     0.2480  0.0660  3.7576 0.0002  0.1187  0.3774
Present employment since                                  0.1378  0.0793  1.7367 0.0824 -0.0177  0.2933
Installment rate in percentage of disposable income      -0.2879  0.0898 -3.2060 0.0013 -0.4639 -0.1119
Personal status and sex                                   0.2327  0.1241  1.8758 0.0607 -0.0104  0.4759
Other debtors / guarantors                                0.3842  0.2081  1.8459 0.0649 -0.0237  0.7921
Present residence since                                  -0.0297  0.0858 -0.3461 0.7293 -0.1979  0.1385
Property                                                 -0.2014  0.1032 -1.9511 0.0510 -0.4037  0.0009
Age in years                                              0.0086  0.0091  0.9497 0.3423 -0.0092  0.0264
Other installment plans                                   0.2474  0.1140  2.1705 0.0300  0.0240  0.4708
Housing                                                   0.2947  0.1915  1.5392 0.1238 -0.0806  0.6700
Number of existing credits at this bank                  -0.3194  0.1796 -1.7788 0.0753 -0.6714  0.0325
Job                                                      -0.1447  0.1474 -0.9818 0.3262 -0.4335  0.1441
Number of people being liable to provide maintenance for -0.3341  0.2370 -1.4098 0.1586 -0.7985  0.1304
Telephone                                                 0.4199  0.2103  1.9971 0.0458  0.0078  0.8320
foreign worker                                            1.2417  0.7070  1.7562 0.0791 -0.1441  2.6274
----------------------------------------------------------------------------------------------
```

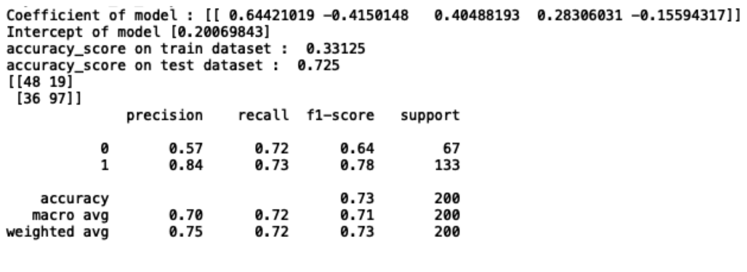
Figure 8: Logistic Regression



```
Optimization terminated successfully.
         Current function value: 0.517867
         Iterations 6
                                          Results: Logit
=================================================================================================
Model:                   Logit                                Pseudo R-squared:      0.161
Dependent Variable:      Receive_NotReceiveCredit             AIC:                   838.5868
Date:                    2020-04-13 17:55                     BIC:                   862.0098
No. Observations:        800                                  Log-Likelihood:        -414.29
Df Model:                4                                    LL-Null:               -493.67
Df Residuals:            795                                  LLR p-value:           2.7097e-33
Converged:               1.0000                               Scale:                 1.0000
No. Iterations:          6.0000
-------------------------------------------------------------------------------------------------
                                                     Coef.  Std.Err.    z    P>|z|  [0.025  0.975]
-------------------------------------------------------------------------------------------------
Status of existing checking account                  0.5530  0.0724  7.6426 0.0000  0.4112  0.6949
Duration in month                                   -0.0335  0.0064 -5.2232 0.0000 -0.0461 -0.0209
Credit history                                       0.4136  0.0703  5.8795 0.0000  0.2757  0.5515
Savings account/bonds                                0.1949  0.0594  3.2804 0.0010  0.0785  0.3113
Installment rate in percentage of disposable income -0.1435  0.0605 -2.3694 0.0178 -0.2621 -0.0248
=================================================================================================

Coefficient of model : [[ 0.63597366 -0.38633419  0.41050417  0.26840264 -0.14376372]]
Intercept of model [0.17071146]
```

Figure 9: Logistic Regression



```
Coefficient of model : [[ 0.64421019 -0.4150148   0.40488193  0.28306031 -0.15594317]]
Intercept of model [0.20069843]
accuracy_score on train dataset :  0.33125
accuracy_score on test dataset :  0.725
[[48 19]
 [36 97]]
              precision    recall  f1-score   support

           0       0.57      0.72      0.64        67
           1       0.84      0.73      0.78       133

    accuracy                           0.73       200
   macro avg       0.70      0.72      0.71       200
weighted avg       0.75      0.72      0.73       200
```

Figure 10: Logistic Regression

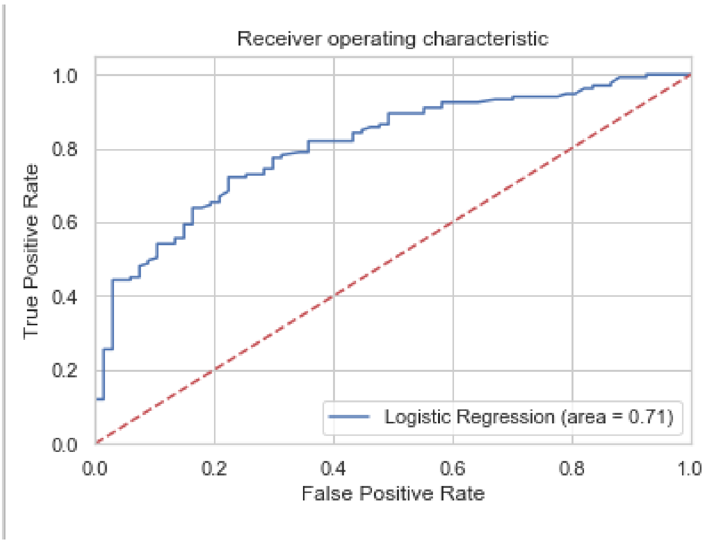
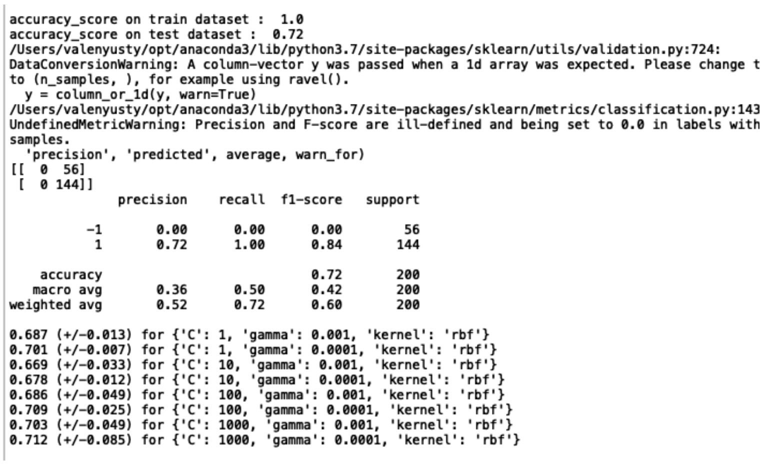


Figure 11: Logistic Regression



```
accuracy_score on train dataset :  1.0
accuracy_score on test dataset :  0.72
/Users/valenyusty/opt/anaconda3/lib/python3.7/site-packages/sklearn/utils/validation.py:724:
DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change t
to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
/Users/valenyusty/opt/anaconda3/lib/python3.7/site-packages/sklearn/metrics/classification.py:143
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
samples.
  'precision', 'predicted', average, warn_for)
[[  0  56]
 [  0 144]]
              precision    recall  f1-score   support

          -1       0.00      0.00      0.00        56
           1       0.72      1.00      0.84       144

    accuracy                           0.72       200
   macro avg       0.36      0.50      0.42       200
weighted avg       0.52      0.72      0.60       200

0.687 (+/-0.013) for {'C': 1, 'gamma': 0.001, 'kernel': 'rbf'}
0.701 (+/-0.007) for {'C': 1, 'gamma': 0.0001, 'kernel': 'rbf'}
0.669 (+/-0.033) for {'C': 10, 'gamma': 0.001, 'kernel': 'rbf'}
0.678 (+/-0.012) for {'C': 10, 'gamma': 0.0001, 'kernel': 'rbf'}
0.686 (+/-0.049) for {'C': 100, 'gamma': 0.001, 'kernel': 'rbf'}
0.709 (+/-0.025) for {'C': 100, 'gamma': 0.0001, 'kernel': 'rbf'}
0.703 (+/-0.049) for {'C': 1000, 'gamma': 0.001, 'kernel': 'rbf'}
0.712 (+/-0.085) for {'C': 1000, 'gamma': 0.0001, 'kernel': 'rbf'}
```

Figure 12: Support Vector Machine

7 Conclusion

References

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

[2] Onel Harrison. *Machine Learning Basics with the K-Nearest Neighbors Algorithm.* Towards Data Science, 2018.

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

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

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

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

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

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

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