APPLIED MACHINE LEARNING REPORT

Ignacio González

9/03/2019

THE UNIVERSITY OF GREENWICH, LONDON, UNITED KINGDOM

Content

Introduction

Machine learning is one of the branches of Artificial Intelligence (AI) in which different machine learning processes are studied with the main objective of emulating the human learning process.

The range of application of the tools provided by the ML field is widening as new discoveries emerge. That is why it could be used either in finances, medicine, artificial intelligence with different purposes, etc. In this paper, basic concepts of ML will be analyses with the purpose of classifying whether a loan will be granted or not.

In this case, the aim of this project is classifying the different answers depending on whether the person is going to get the loan or not. Thus, we can state that this is a classification problem in which we were provided with 2 different datasets. The first dataset (training dataset) has labels although the second dataset (testing dataset) does not. This is a problem since we will not be able to get the accuracy of the testing procedure, for this case we need the labels of each record. The final goal (besides understanding the basics of a machine leaning project) is to classify the records as ‘YES’ (they got the loan grant) or ‘NO’ through predictions.

As the reader goes through this paper, they will understand the problems and the suggested alternatives for getting the most optimized solution. The workflow will cover dataset uploading, cleaning process (also known as data pre-processing, defining and training the models, validating and testing the models and finally, presenting the results in a process known as visualization). As results, we will present some predictions and their level of accuracy with the purpose of contrasting the level of confidence of the described models.

In this section, related work and the motivation for this project will be explained. The ‘backbone’ information of this section was extracted from the book: (Hurwitz, J., & Kirsch, D., 2018) as the main concepts were quite good explained.

Once we understand what machine learning is, it necessary to understand what kind of problem we want to approach. We can state that there are 4 main kinds of approaches if we are talking about machine learning:

- Supervised learning
- Unsupervised learning
- Reinforcement learning
- Deep learning

In **Supervised Learning** we start with a set of data from which we already have some knowledge of how data is classified. The main objective is to find patterns within the data with the purpose of applying them for a bigger set of similar data. Here, each record has a label thus, we can affirm that each record has a name that we can use so as to classify the record itself. Now, it is important to differentiate when the label is a continuous number (regression approach) or is a set of categorical data (classification approach), we will talk about them later on. Mentioning the concept of overfitting is necessary in this field, we can use the metaphoric example of a student who studies the same 10 questions always during the course. He will pass the exam in case the professor asks them these particular 10 questions. However, if the professor changes the questions, the student will be lost. The same happens with the data when training the models, we you tune your own model for the data it is using for the training. The usage of unforeseen data when testing would help to assess the actual accuracy.

Unsupervised Learning deals with unlabeled data (clustering instead of classification), trying to group data depending on their attributes. We can state that in case a data scientist receives a huge set of unlabeled data, this would be the first step due to the fact that this very step adds labels to the extant records. Afterwards, once we have labeled data, we can conduct supervised learning procedures to analyze the data.

Reinforcement learning differs from the last two models in the fact that it nourishes itself with its own feedback. It uses this feedback to lead the model to the best solution. It does not use a training data set but instead, it performs ‘trial and error’ procedures thus, the answers are ‘reinforced’ consequently. Self-driving cars are a quite useful approach of using reinforcement learning solutions.

Neural networks and deep learning approaches covers the field that uses systems with a number of layers of neural networks so as to learn iteratively.

Deep learning refers to more complex neural networks which tries to emulate the human brain´s behavior. Nowadays, an average quality computer can beat the smartest human when talking about computing large scale calculations although every human can beat any computers in the field of defining abstract concepts as shown in **Figure 1**.

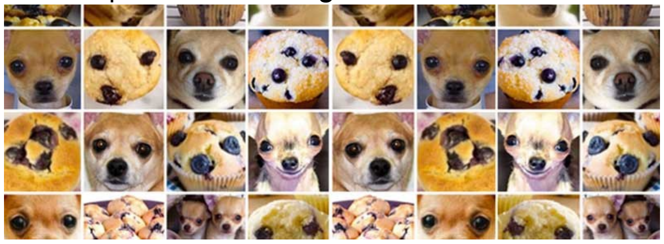


Figure 1 Dog or muffin abstract problem (Yao M, 2017)

Bifurcation: Classification or Regression?

The following sections will show how the provided datasets look like and how we dealt we them. However, it would be positive to talk about the problem we are trying to solve. Since we want to classify whether a person is getting the grating, we can state that we are going to tackle a supervised learning project that is based on classification because, we have a categorical range of options: ‘YES’ or ‘NO’. Thus, we will define a classifier that holds only 2 classes (labels). Once this was understood, we can start talking about models, which one is the one which best fits our requirements?

Models for classification problems

Microsoft provides us a really useful ‘cheat sheet’ that we can use as a template in order to differentiate which model would be the optimum for our problem. The link of the picture will be provided below in case the reader cannot read the image.

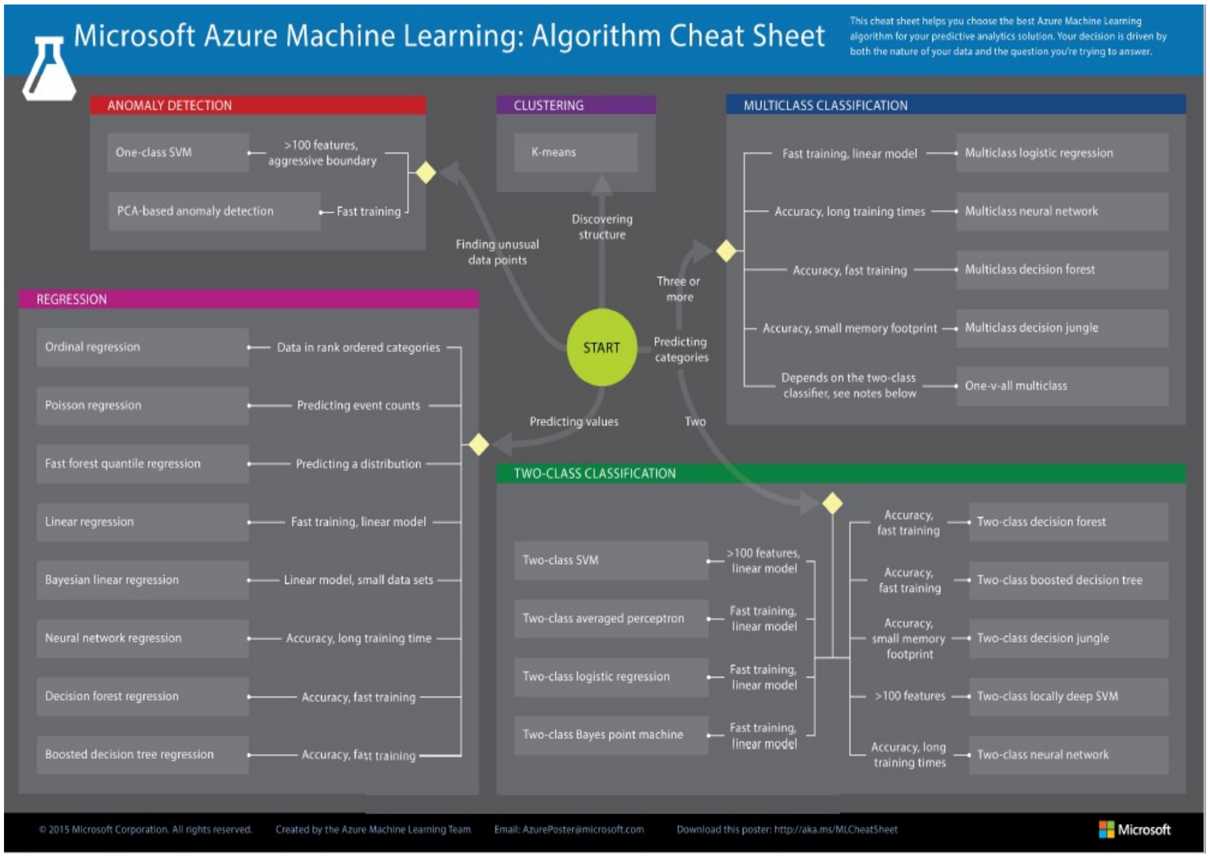

Microsoft Azure Machine Learning: Algorithm Cheat Sheet
This cheat sheet helps you choose the best Azure Machine Learning algorithm for your predictive analytics solution. Your decision is driven by both the nature of your data and the question you're trying to answer.
ANOMALY DETECTION
One-class SVM
>100 features, aggressive boundary
PCA-based anomaly detection
Fast training
CLUSTERING
K-means
Discovering structure
Finding unusual data points
MULTICLASS CLASSIFICATION
Fast training, linear model
Multiclass logistic regression
Accuracy, long training times
Multiclass neural network
Accuracy, fast training
Multiclass decision forest
Accuracy, small memory footprint
Multiclass decision jungle
Depends on the two-class classifier, see notes below
One-v-all multiclass
Three or more
Predicting categories
Two
START
Predicting values
REGRESSION
Ordinal regression
Data in rank ordered categories
Poisson regression
Predicting event counts
Fast forest quantile regression
Predicting a distribution
Linear regression
Fast training, linear model
Bayesian linear regression
Linear model, small data sets
Neural network regression
Accuracy, long training time
Decision forest regression
Accuracy, fast training
Boosted decision tree regression
Accuracy, fast training
TWO-CLASS CLASSIFICATION
Two-class SVM
>100 features, linear model
Two-class averaged perceptron
Fast training, linear model
Two-class logistic regression
Fast training, linear model
Two-class Bayes point machine
Fast training, linear model
Accuracy, fast training
Two-class decision forest
Accuracy, fast training
Two-class boosted decision tree
Accuracy, small memory footprint
Two-class decision jungle
>100 features
Two-class locally deep SVM
Accuracy, long training times
Two-class neural network
© 2015 Microsoft Corporation. All rights reserved.
Created by the Azure Machine Learning Team
Email: AzurePoster@microsoft.com
Download this poster: http://aka.ms/MLCheatSheet
Microsoft

Figure 2 Machine Learning cheat sheet (Microsoft Azure, 2019)

Therefore, following the workflow of our diagram we will go to the ‘Predicting categories’ as we have two categories (not numbers): ‘YES’ or ‘NO’. Then, we will jump into the ‘Two-Class Classification’. There, it suggests us trying different approaches depending on the attributes and records available.

1. SVM (Support Vector Machine): For linear models with more than 100 features.
2. Perceptron: For linear models that need fast training.
3. Logistic Regression: For linear models that need fast training.
4. Bayes point machine: For linear models that need fast training.
5. Decision Forest: Provides fast training with high accuracy.
6. Boosted Decision tree: Provides fast training with high accuracy.
7. Decision Jungle: Accuracy and small memory footprint.
8. Deep SVM: For datasets bigger than 100 features.
9. Neural Network: High accuracy but long training times.

These were the suggestions although we did not used some of them and in some cases, we decided to use different models for reasons that will be explained later on.

Therefore, we decided to use the following ones:

1. Gaussian Naïve Bayes

2. Linear Discriminant Analysis
3. Logistic Regression
4. Decision Trees
5. k-NN
6. SVC

Project´s workflow

Data Mining is a branch of artificial intelligence that is close to the Machine Learning process (some authors talk about them as the same thing). We can apply both concepts to the field of the Business Intelligence. However, Data Mining describes a process in which the scientist gets new useful insight that they can use for improving the state of the company by analyzing the data and constructing a dashboard. **Figure 3** shows how a Data Mining project should be structured. It takes into account the construction of an element called Data Warehouse which is the source of our data and engineering work is meant to update this data warehouse with new data periodically. Thus, it is possible for scientists to study the company´s projection during the time.

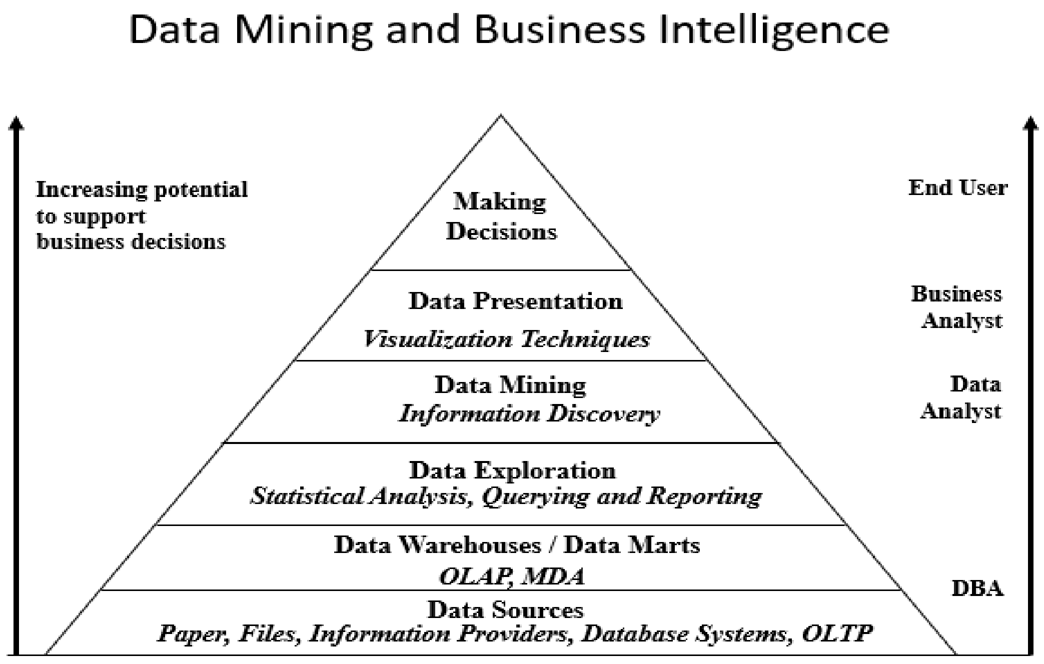

Data Mining and Business Intelligence
Increasing potential to support business decisions
Making Decisions
Data Presentation
Visualization Techniques
Data Mining
Information Discovery
Data Exploration
Statistical Analysis, Querying and Reporting
Data Warehouses / Data Marts
OLAP, MDA
Data Sources
Paper, Files, Information Providers, Database Systems, OLTP
End User
Business Analyst
Data Analyst
DBA

Figure 3 Structure of a Data Mining project (Hassouna M, 2019)

For our case, we will focus on the structure depicted in the **Figure 4.**

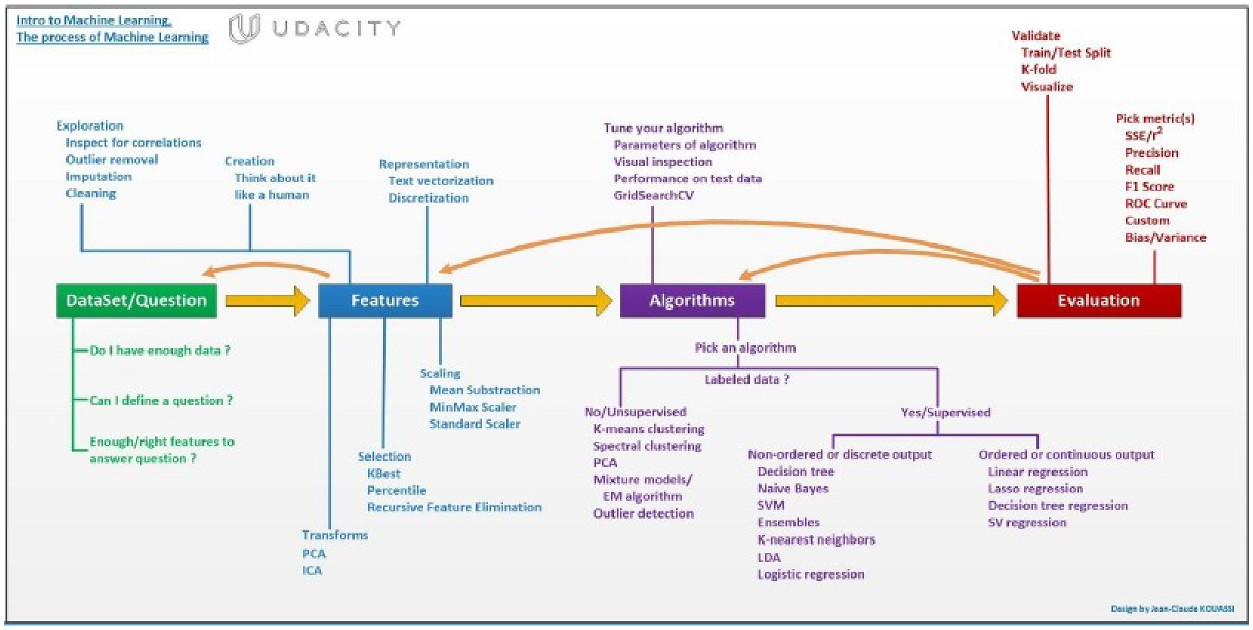


Figure 4 Project´s structure (Vuong T, 2019)

Therefore, we will divide our project in the following steps:

1. Data pre-analysis
2. Data pre-processing
3. Models
4. Results and Visualizations

Data pre-analysis

In this stage, we tried to get some information by just looking at the data. Understand the attributes is a must and some basic processes could be performed. In the case of this project, PyCharm was used as the programming platform thus, first steps were getting basic knowledge about the data and upload it.

Data-preprocessing

After the pre-analysis, the datasets might need to be cleaned and transformed into the same format. Processes like categorical substitution, ‘NaN’ values elimination, stratified balancing process and, attribute balancing procedures were performed.

Models development

This section covers deciding which models we are going to use, defining, fitting, training, validating and testing them.

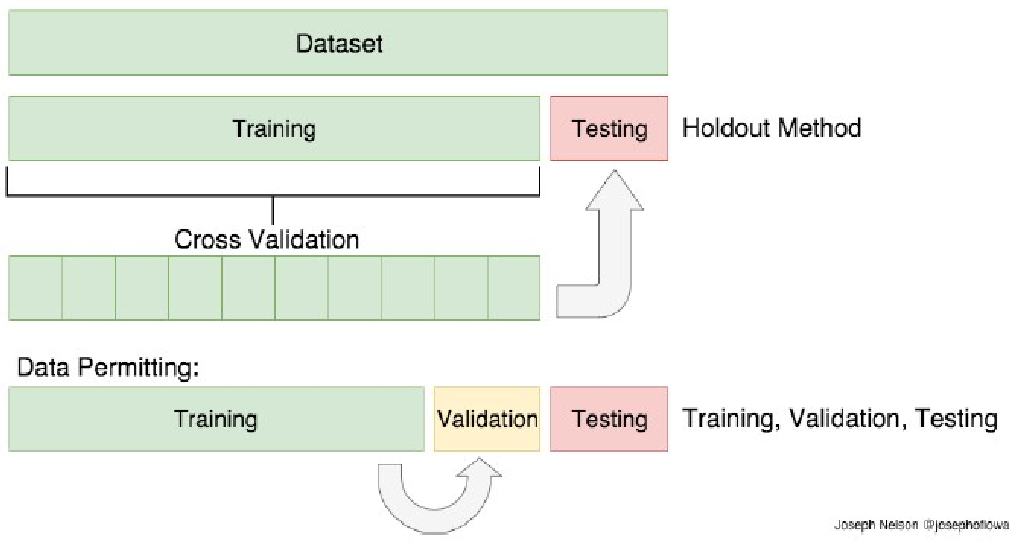


Figure 5 Training, validating and testing procedures (Vuong T, 2019)

Results and Visualizations

Once we obtained some predictions/results, it is recommendable to study the accuracy of these results in order to decide whether using this model or keep on improving it by optimizations (or even change the models).

Stages Achieved

There were 5 stages and all of them were approached.

<u>**Stage 1**</u>: Examine dataset and identify areas of potential insight for the client. Python was used as environment to develop the analysis. (**ACHIEVED**)

<u>**Stage 2**</u>: Data pre-processing. Data cleaning and numerical substitution processes were performed. (**ACHIEVED**)

<u>**Stage 3:**</u> Build Models. Models like k-NN, Decision Trees, Linear Discriminant Analysis were performed (among others). All of them were methods with the purpose of approaching the problem of classifying classes given particular attributes. In this case, using inputs like personal details from each subject we were able to predict results. (**ACHIEVED**)

<u>**Stage 4**</u>: Testing and Validating. We perform some training (from which we obtained an accuracy from the models using cross-validation) and after this step, we perform some testing once the models were fitted. We obtained real-tested accuracy from these steps. (**ACHIEVED**)

Note: It is necessary to clarify that the student had two datasets:

- **Training dataset or dataset 1**: Which has labels
- **Dataset 2**: Which does not have labels.

<u>Stage 5</u>: Visualization. All the results were explained and the models´ performance were showed. We used plots in Python for this purpose. (**ACHIEVED**)

Bugs and weaknesses

Even though a lot of attention was paid when writing the code, this part is always likely to be improved or made more efficient.

It would be really useful to have the classifier of the second dataset (test dataset), since it would be possible to measure the real accuracy of the predicted results. However, we did not have access to this information.

The fact that 'NaN' values were removed by removing the whole record, it would have increased the accuracy if any of the available approaches were tackled like substituting by the mean or the median.

Contributions

The project was not developed by pair programming. The student used information from the lectures during the first stages in order to understand how to proceed. Once everything was understood, the student added more functionalities and implementations.

Documentation

Documentation of each of the stages that were completed.

Stage 1- Data pre-analysis

The question that was approached was predicting whether the loan will be granted or not. Thus, we began with the pre-processing procedure and then, it was divided into two different parts (regarding the dataset 1). The training and testing part, so as to fit the models and find their accuracy. After that, we were provided with an additional dataset (dataset 2) which has records of all the attributes but not labels. We used the last dataset to predict some results. However, as indicated in the section about weaknesses, we could not get the real accuracy since we had nothing to compare the predictions with.

The attributes are:

- **Loan_ID**: ID
- **Gender**: Male/Female
- **Married**: Yes/No
- **Dependents**: 0/1/2/3+
- **Education**: Graduated/Not Graduated
- **Self_Employed**: Yes/No

- **ApplicantIncome**: Numeric
- **CoapplicantIncome**: Numeric
- **LoanAmount**: Numeric
- **Loan_Amount_Term**: Numeric
- **Credit_History**: Numeric
- **Property_Area**: Urban/Semiurban/Rural
- **Loan_Status**: Yes/No

The first dataset (training dataset) has 614 records while the test dataset has 367 records. The number of records will change when pre-processing the data as shown in **Figure 9**. This is because we got rid of the records with 'NaN' values.

Also, a new question appeared: '*Is there any correlation between the attributes?*'

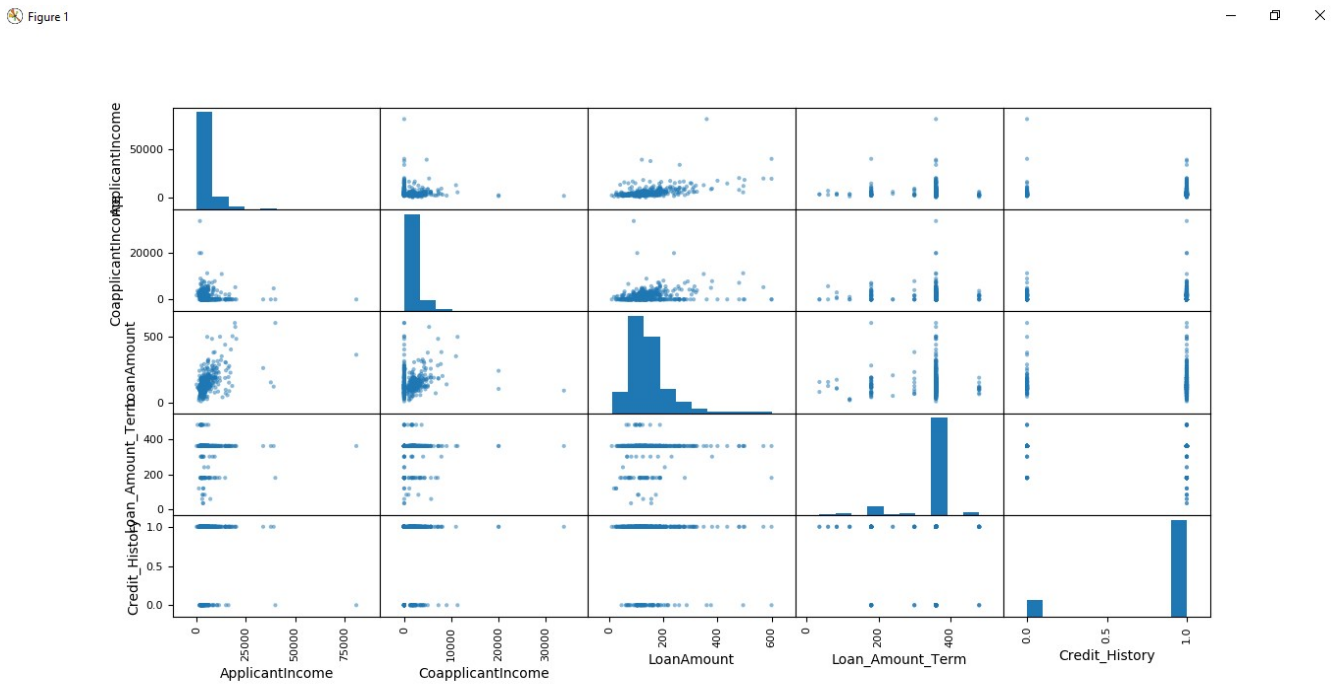


Figure 6 Scatter-matrix

At the first stages, we can state that there is a correlation between 'LoanAmount' and 'CoapplicationIncome' and again 'LoanAmount' and 'ApplicationIncome' as shown in **Figure 6**. In these cases it means, the bigger 'Loan Amount' gets, the bigger 'CoapplicationIncome' and 'LoanAmount' become. We can state that these variables have **a positive lineal correlation**.

Stage 2- Data pre-processing

Nevertheless, there are more useful insight within the data that are accessible from the first stages of the data analysis like:

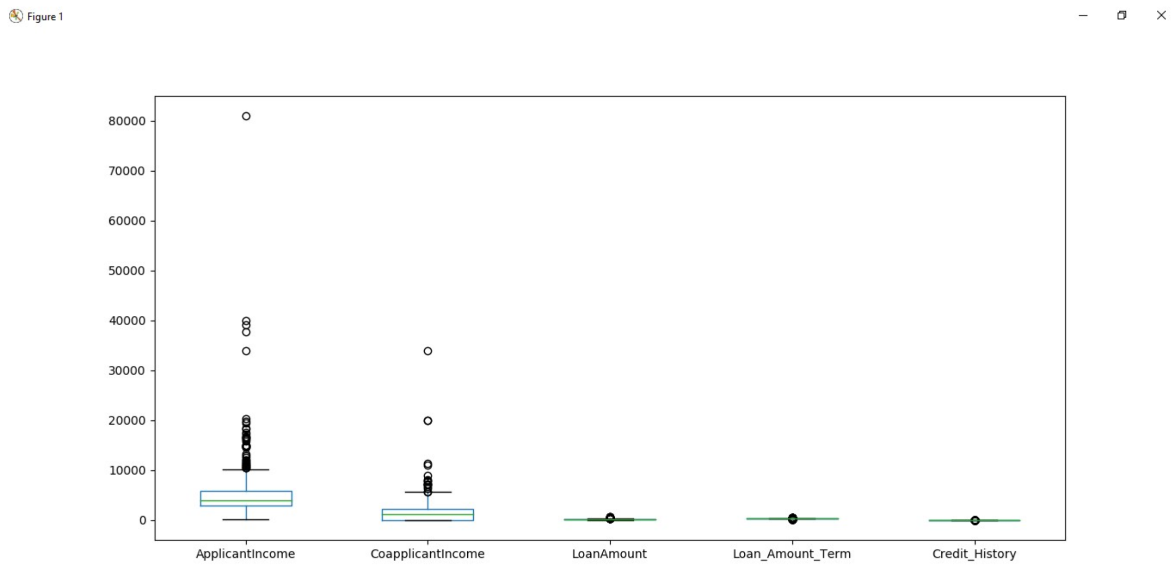


Figure 7 Whisker-plot

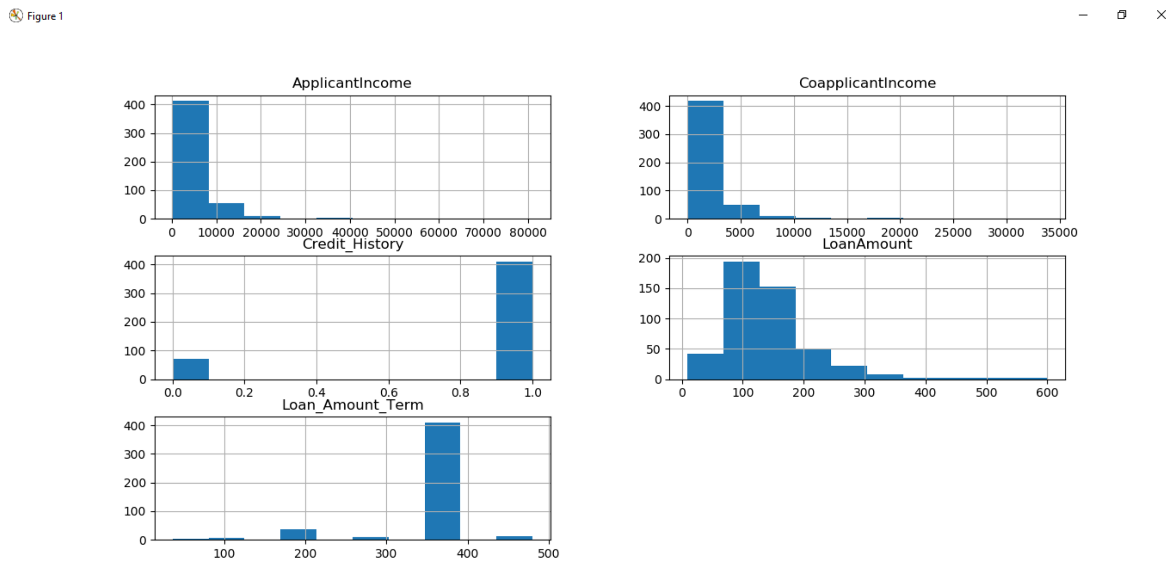
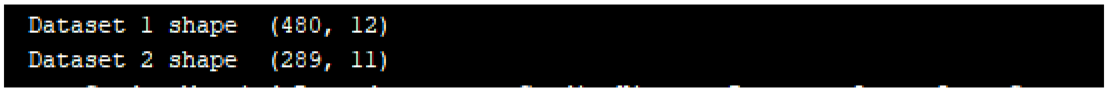


Figure 8 Histogram

Figures 7 and 8 show information about the numerical data available in the training dataset that could be useful to build our dashboard. It could be interesting to talk about the number of outliers of the ‘ApplicationIncome’ as it can be seen in the **Figure 8** knowing that the vast majority of records are between 0 and 10000 pounds. The same could be stated in the ‘CoapplicationIncome’ but in this particular case, the majority of values go from 0 to 5000 pounds.

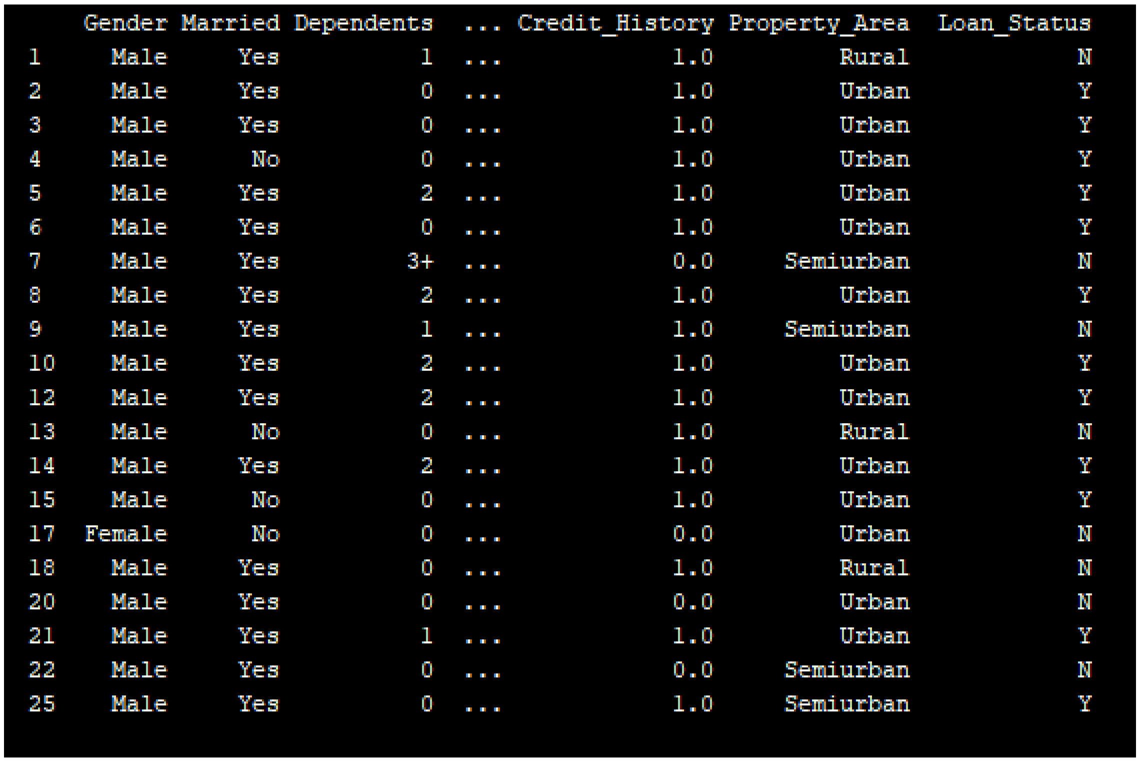
Next figures show another point of view of the same data, the more information we have, the better we are informed:



```
Dataset 1 shape  (480, 12)
Dataset 2 shape  (289, 11)
```

Figure 9 Size of the datasets after removing 'NaN' values(Dataset 1=Training dataset)

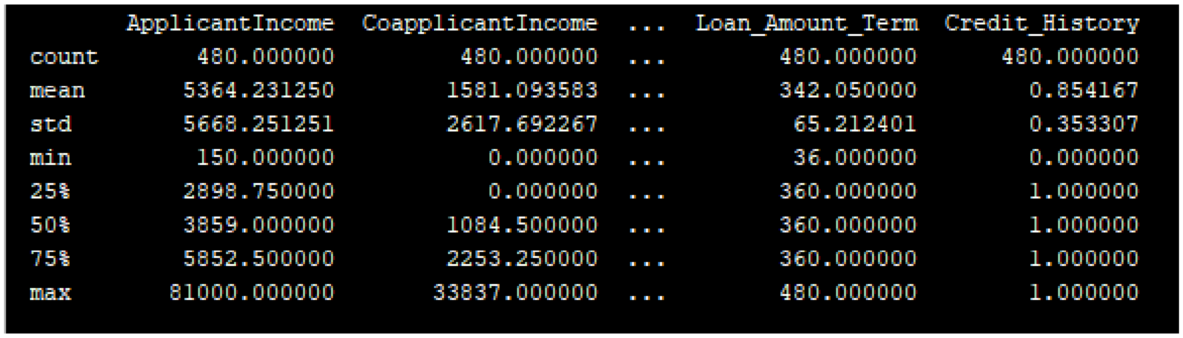
A useful way of make sure the data is correctly uploaded is loading the first records to check them, **Figure 10**.



```
    Gender Married Dependents  ... Credit_History Property_Area Loan_Status
1     Male     Yes          1  ...            1.0         Rural           N
2     Male     Yes          0  ...            1.0         Urban           Y
3     Male     Yes          0  ...            1.0         Urban           Y
4     Male      No          0  ...            1.0         Urban           Y
5     Male     Yes          2  ...            1.0         Urban           Y
6     Male     Yes          0  ...            1.0         Urban           Y
7     Male     Yes         3+  ...            0.0     Semiurban           N
8     Male     Yes          2  ...            1.0         Urban           Y
9     Male     Yes          1  ...            1.0     Semiurban           N
10    Male     Yes          2  ...            1.0         Urban           Y
12    Male     Yes          2  ...            1.0         Urban           Y
13    Male      No          0  ...            1.0         Rural           N
14    Male     Yes          2  ...            1.0         Urban           Y
15    Male      No          0  ...            1.0         Urban           Y
17  Female      No          0  ...            0.0         Urban           N
18    Male     Yes          0  ...            1.0         Rural           N
20    Male     Yes          0  ...            0.0         Urban           N
21    Male     Yes          1  ...            1.0         Urban           Y
22    Male     Yes          0  ...            0.0     Semiurban           N
25    Male     Yes          0  ...            1.0     Semiurban           Y
```

Figure 10 First 20 records of the training dataset

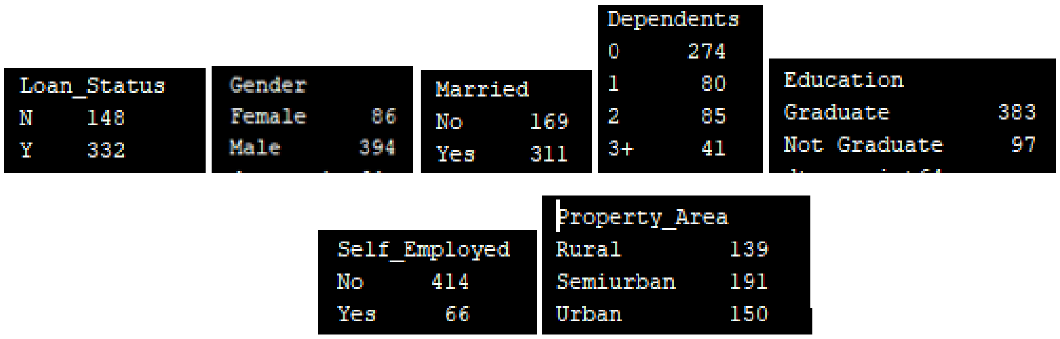
As we can see in **Figure 11**, we have now access to information regarding mean and standard deviation of the numerical data. We can extract, using this information, values like outliers by just comparing the 'max' and 'min' with the 'mean'.



```
       ApplicantIncome  CoapplicantIncome  ...  Loan_Amount_Term  Credit_History
count       480.000000         480.000000  ...        480.000000      480.000000
mean       5364.231250        1581.093583  ...        342.050000        0.854167
std        5668.251251        2617.692267  ...         65.212401        0.353307
min         150.000000           0.000000  ...         36.000000        0.000000
25%        2898.750000           0.000000  ...        360.000000        1.000000
50%        3859.000000        1084.500000  ...        360.000000        1.000000
75%        5852.500000        2253.250000  ...        360.000000        1.000000
max       81000.000000       33837.000000  ...        480.000000        1.000000
```

Figure 11 Useful information about the available numerical data

Last but not least, relevant information about the extant categorical data is depicted in the following figures:



```
Loan_Status
N     148
Y     332
```

```
Gender
Female      86
Male       394
```

```
Married
No      169
Yes     311
```

```
Dependents
0      274
1       80
2       85
3+      41
```

```
Education
Graduate         383
Not Graduate      97
```

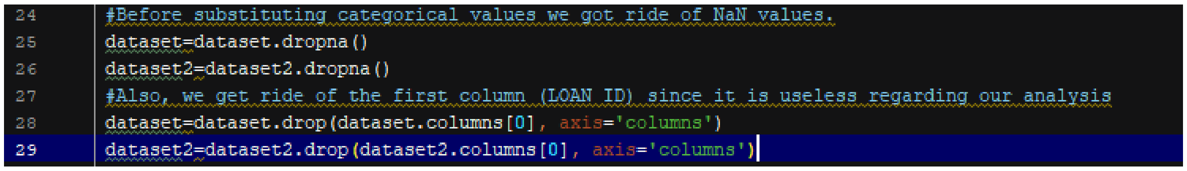
```
Self_Employed
No       414
Yes       66
```

```
Property_Area
Rural         139
Semiurban     191
Urban         150
```

Figure 12 Figure composed of different screenshots of the numbers of records with their attributes

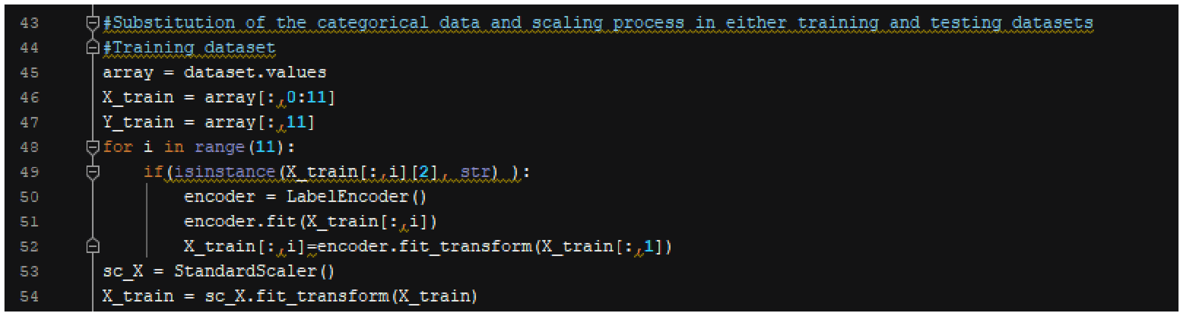
After this stage, it was necessary to decide which methods we were going to use. Bearing in mind that our question was '*Are we able to predict the results?*' what we needed to approach was a classification problem. Given certain number of attributes, predict the result ('YES' or 'NO') of the loan granting decision. **Classification** is a technique widely used in Machine Learning for different purposes like handwriting recognition, speech recognition, biometric identification, etc. There are different kinds of classifiers: Binary Classifiers (two possibilities) and Multi-Class Classifiers (a range of possibilities). It could be stated that classification is a supervised learning approach in which the system tries to learn from different inputs and each record has been labelled.

Before start talking more deeply about the methods, it would be interesting to say that all the data have to be pre-processed (in this particular case because it was necessary) with the purpose of eliminating 'NaN' (missing values) values or substituting categorical data by numerical data, so we can compute an answer based on all the attributes. A scaling procedure was also performed with the purpose of eliminating differences between high values ('ApplicationIncome') and smaller ones ('LoanAmount') so we avoid the results to be biased. The following figures show these processes:



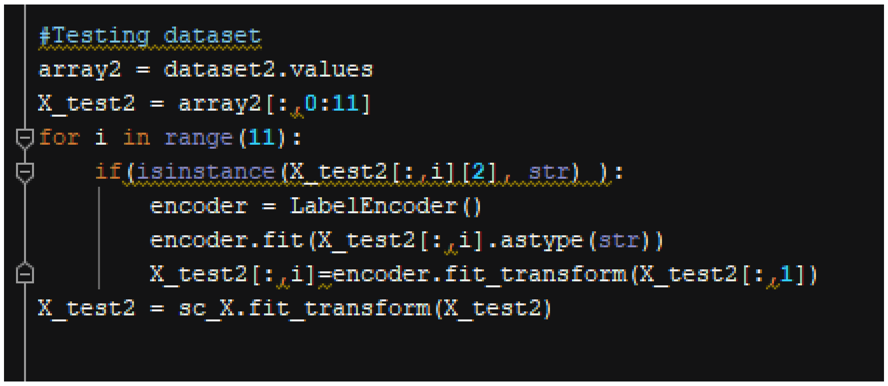
```
#Before substituting categorical values we got ride of NaN values.
dataset=dataset.dropna()
dataset2=dataset2.dropna()
#Also, we get ride of the first column (LOAN ID) since it is useless regarding our analysis
dataset=dataset.drop(dataset.columns[0], axis='columns')
dataset2=dataset2.drop(dataset2.columns[0], axis='columns')
```

Figure 13 'NaN' values elimination



```
#Substitution of the categorical data and scaling process in either training and testing datasets
#Training dataset
array = dataset.values
X_train = array[:,0:11]
Y_train = array[:,11]
for i in range(11):
    if(isinstance(X_train[:,i][2], str) ):
        encoder = LabelEncoder()
        encoder.fit(X_train[:,i])
        X_train[:,i]=encoder.fit_transform(X_train[:,1])
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
```

Figure 14 Numerical substitution and scaling procedure of the training dataset



```
#Testing dataset
array2 = dataset2.values
X_test2 = array2[:,0:11]
for i in range(11):
    if(isinstance(X_test2[:,i][2], str) ):
        encoder = LabelEncoder()
        encoder.fit(X_test2[:,i].astype(str))
        X_test2[:,i]=encoder.fit_transform(X_test2[:,1])
X_test2 = sc_X.fit_transform(X_test2)
```

Figure 15 Numerical substitution and scaling procedure of the testing dataset

Therefore, we can conclude that what we need is a solution for classifying labels with the purpose of differentiating whether a person is going to get the loan or not. Next section tackles the methods we approached to accomplish the results.

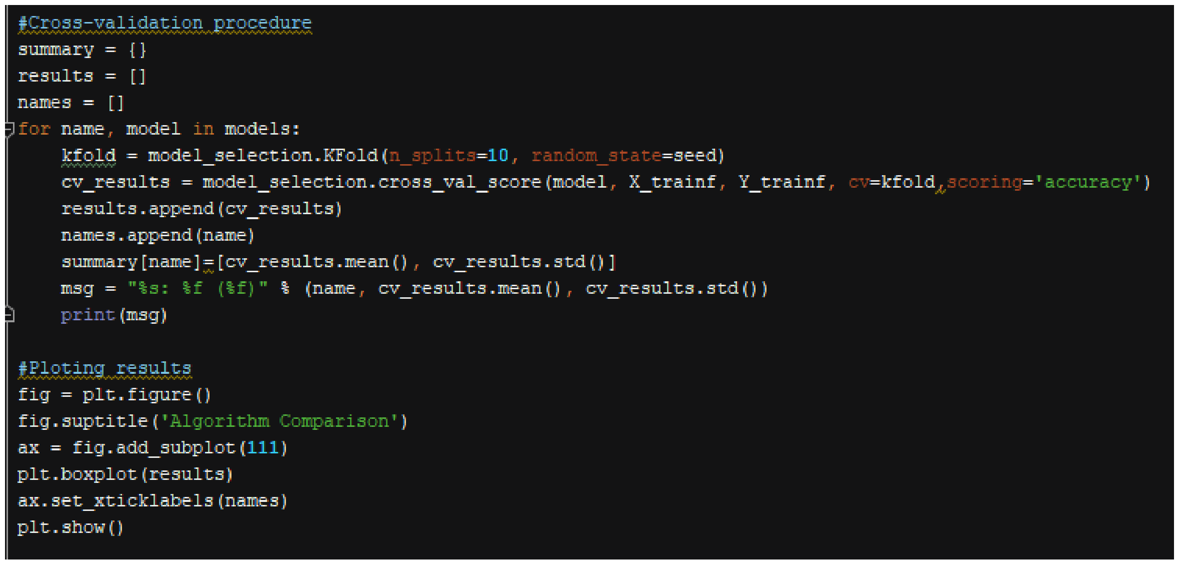
Stage 3-Models development

Once the datasets are already cleaned (training dataset and dataset 2), we can start analyzing the models/methods. Notice in the code (accessible in the last section of the report) that stratified sampling was performed in order to avoid label distribution imbalance.

The selected methods were (further research will be conducted in the previously mentioned paper):

- **Logistic Regression**: Statistical approach in which there are a range of independent variables that decide the result.
- **K-Neighbors Classifier**: k-NN classifies results depending on neighbors. The result is the most common value among its k neighbors (the nearest ones).
- **Decision trees**: Classifies data based on rules. Once a differentiator is selected, the method considers the different answers and chooses the most accurate one.
- **Linear Discriminant Analysis**: It is a generalization of Fisher´s linear discriminant, in which using statistical analysis, it finds a linear combination of the attributes (it uses independent variables).
- **Gaussian Naïve Bayes**: It assumes the independence between variables (that is why it is naïve). The best part of this method is that a small number of variables is required to predict new values.
- **C-Support Vector Classification**: Different implementation of the same algorithm. Another popular method is SVM (Support Vector Machine). It analyzes different points in space and divide them by a gap as wide as possible. Results are stated depending on the side of the plane of the point we are analyzing.

Cross-validation was performed with the purpose of assessing the results of the statistical analysis. It created 10 different/independent groups (training and test data groups) with the purpose of validating the models and calculate the mean of the accuracy and the standard deviation (we will call them '*t_accuracy*' and '*t_std*' in this report).



```
#Cross-validation procedure
summary = {}
results = []
names = []
for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state=seed)
    cv_results = model_selection.cross_val_score(model, X_trainf, Y_trainf, cv=kfold,scoring='accuracy')
    results.append(cv_results)
    names.append(name)
    summary[name]=[cv_results.mean(), cv_results.std()]
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)

#Ploting results
fig = plt.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```

Figure 16 Part of the code that deals with the cross-validation procedure

Next figures depicts the obtained results of the models analysis. **Figure 17 and 18** show a different way of evaluate the t_accuracy of the models. In both we can find out which one is worst model regarding this experiment (decision tree) and using the whiskers plot we can compare which ones were closest of being the best (LR, SVC and Linear Discriminant Analysis), because the outliers.

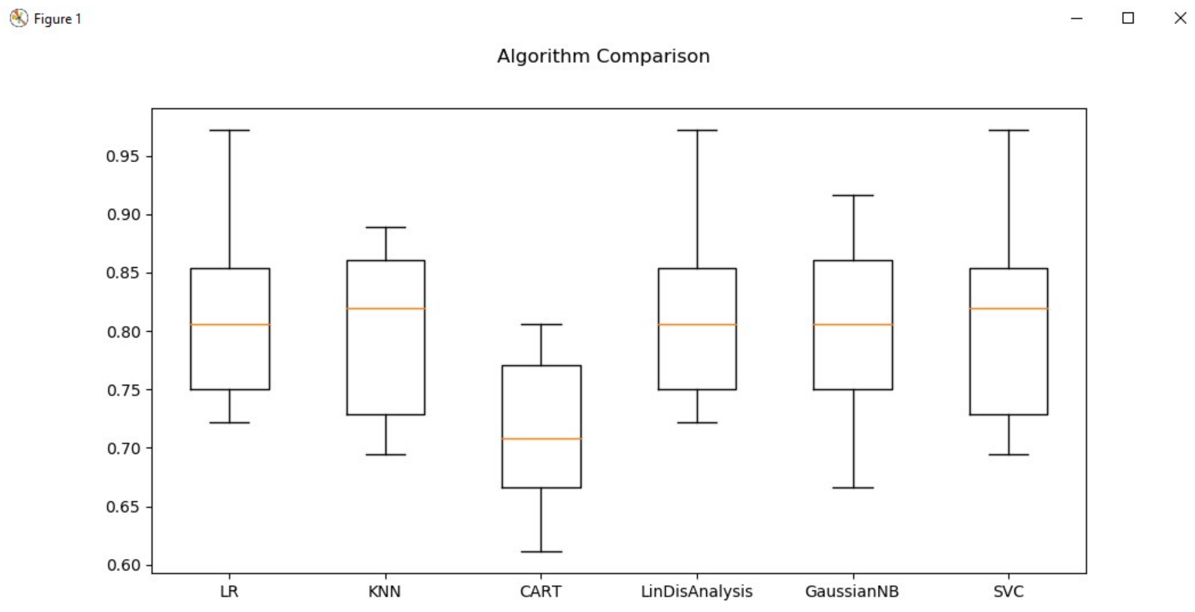


Figure 17 Algorithm Comparison

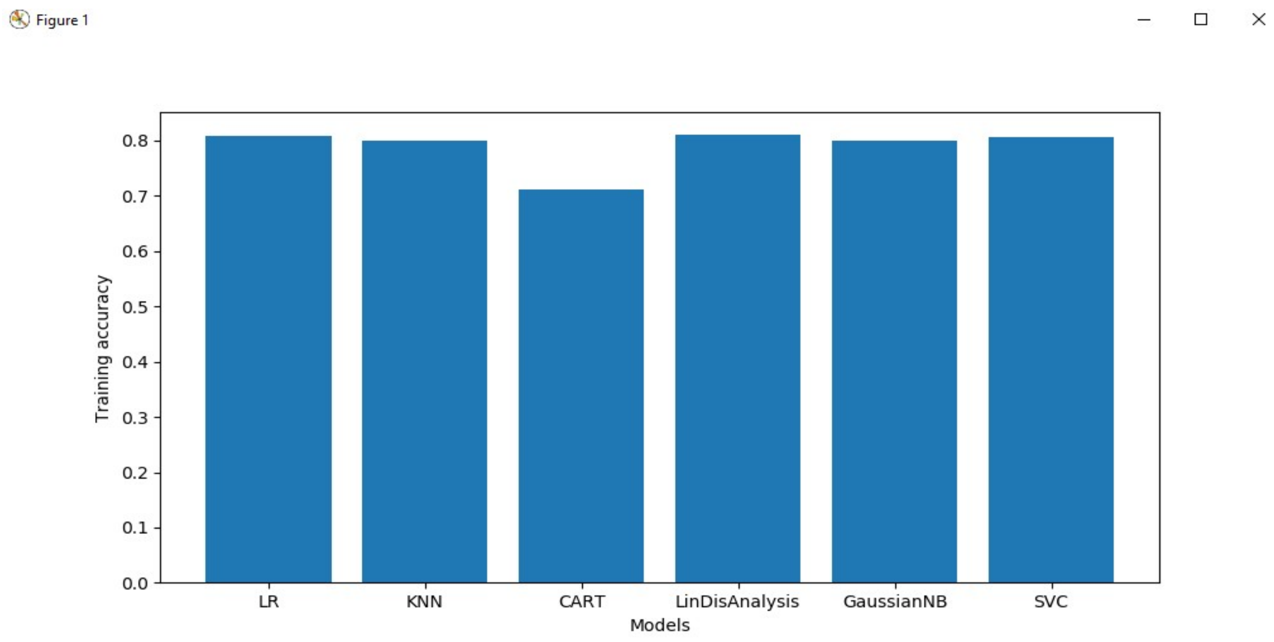


Figure 18 t_accuracy of the models

Figure 19 shows a graph in which '*t_std*' is presented, and we can state that 'CART' has the lowest std and 'SVC' the highest one. That means, during the process of cross-validation, 'CART' model was the one with smallest standard deviation, the results of the accuracy were more similar for the same model even though it was the worst one for the case under study.

Tested accuracy represents the accuracy obtained when testing the models once they were fitted (still dataset 1). The graph in the **Figure 20** concludes that the best accuracy comes from the Logistic Regression although in the 't_accuracy' graph it was not clear.

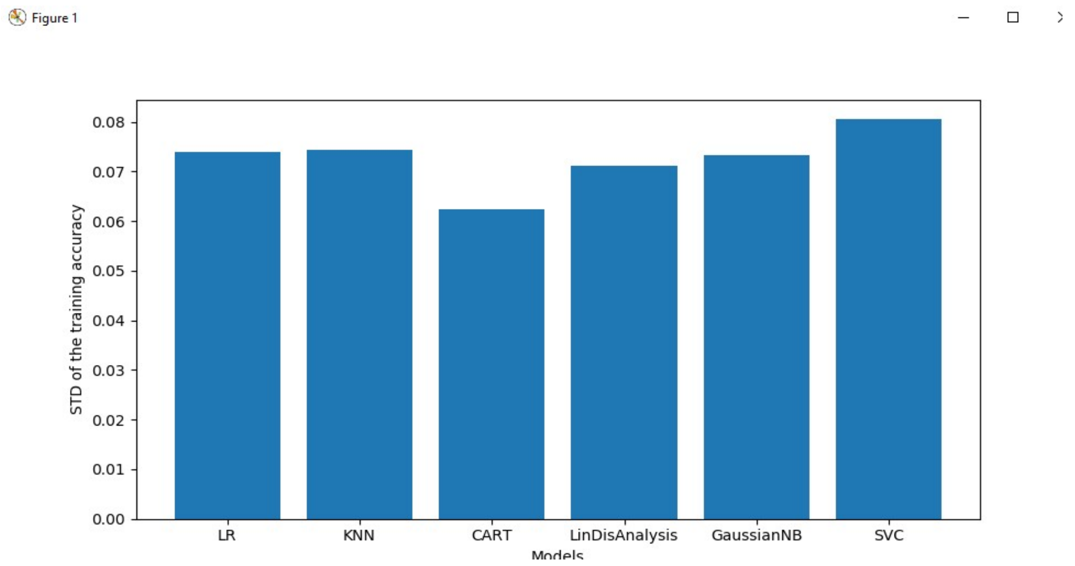


Figure 19 t_std of the models

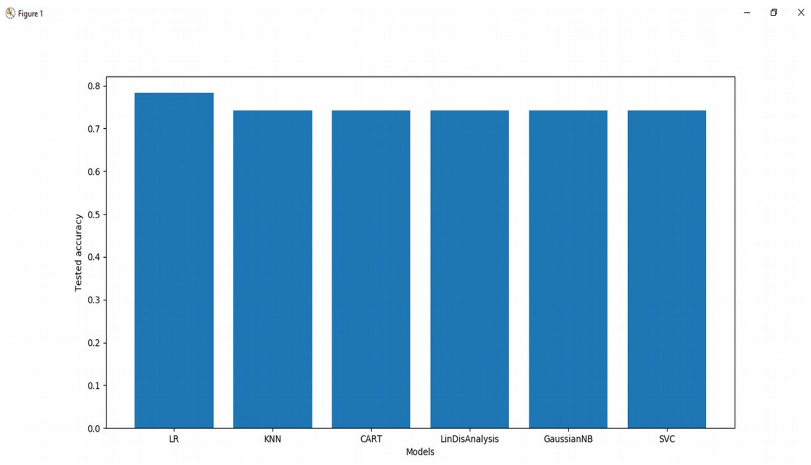
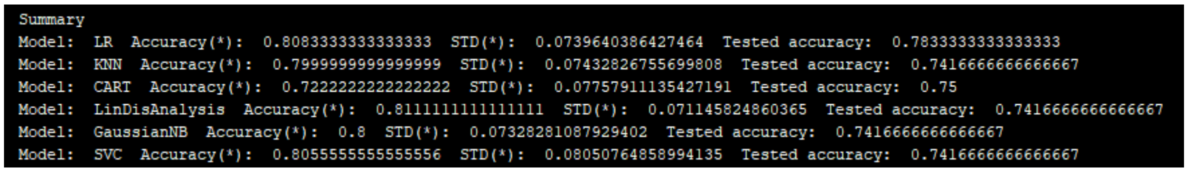


Figure 20 Tested accuracy

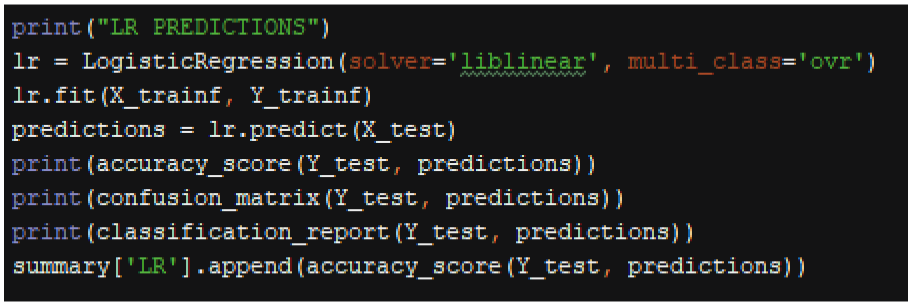


```
Summary
Model:  LR  Accuracy(*):  0.8083333333333333  STD(*):  0.0739640386427464  Tested accuracy:  0.7833333333333333
Model:  KNN  Accuracy(*):  0.7999999999999999  STD(*):  0.07432826755699808  Tested accuracy:  0.7416666666666667
Model:  CART  Accuracy(*):  0.7222222222222222  STD(*):  0.07757911135427191  Tested accuracy:  0.75
Model:  LinDisAnalysis  Accuracy(*):  0.8111111111111111  STD(*):  0.07114582486O365  Tested accuracy:  0.7416666666666667
Model:  GaussianNB  Accuracy(*):  0.8  STD(*):  0.07328281087929402  Tested accuracy:  0.7416666666666667
Model:  SVC  Accuracy(*):  0.8055555555555556  STD(*):  0.08050764858994135  Tested accuracy:  0.7416666666666667
```

Figure 21 Models summary: ‘Accuracy(*)’ as ‘t_accuracy’, ‘STD(*)’ as ‘t_std’

Stage 4-Results and Visualizations

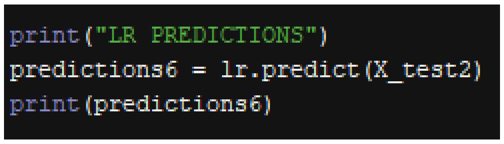
In this section we are going to focus on the results that we have obtained. However, it is necessary to talk about the fitting process of the models, snipes of code as an example of how ‘LR’ was fitted could be found in **Figure 22**. We are going to focus on this particular model since it was the best for the dataset according to its accuracy although all the models follow a similar way of fitting process. The code for the rest of the models could be found in the last section of the report.



```
print("LR PREDICTIONS")
lr = LogisticRegression(solver='liblinear', multi_class='ovr')
lr.fit(X_trainf, Y_trainf)
predictions = lr.predict(X_test)
print(accuracy_score(Y_test, predictions))
print(confusion_matrix(Y_test, predictions))
print(classification_report(Y_test, predictions))
summary['LR'].append(accuracy_score(Y_test, predictions))
```

Figure 22 Fitting and predicting process for Logistic Regression model

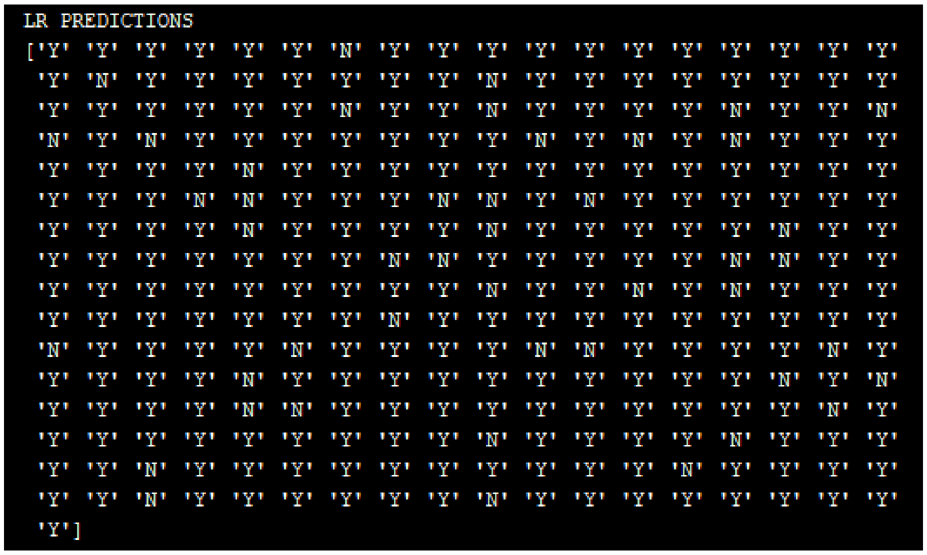
In **Figure 22** we can find how we defined the classifier of the model ‘lr’, how we fitted the classifier and how we obtained the predictions (the results) based on the ‘Y_test’. ‘Y_test’ is part of the main dataset (the training dataset) and we used it so as to measure the accuracy on a real test. Nevertheless, we were provided with another dataset (testing dataset) which does not have label. Thus, we can predict the values from this dataset, but we cannot compare them in order to get the accuracy of these predictions.



```
print("LR PREDICTIONS")
predictions6 = lr.predict(X_test2)
print(predictions6)
```

Figure 23 Predictions of the testing dataset

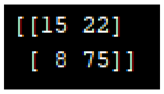
Figure 23 depicts the piece of code that deals with the predictions of the second dataset (training dataset) and the results are shown in **Figure 24**. Unfortunately, we were not provided with the labels of the records therefore, we cannot be sure of the accuracy of this procedure.



```
LR PREDICTIONS
['Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'N'
 'N' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'N' 'Y' 'N' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'N' 'Y' 'N' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'Y'
 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'N'
 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y'
 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y']
```

Figure 24 Predictions for Logistic Regression once we introduce the second dataset (testing dataset)

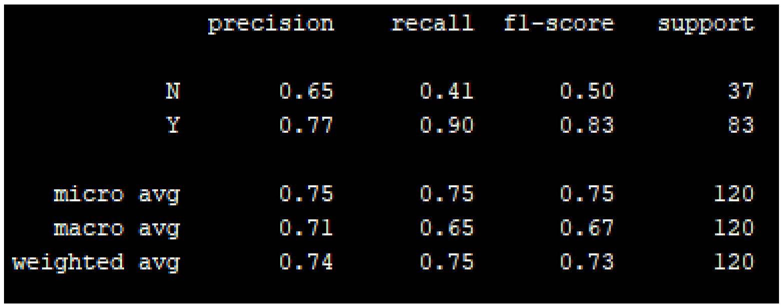
It could be interesting to compare the confusion matrix and the final report of either ‘CART’ and ‘LR’ models since they were the worst and the best one:



```
[[15 22]
 [ 8 75]]
```

Figure 25 'CART' confusion matrix

A confusion matrix is used as a tool to visualize the performance of a supervised-learning model. Each column represents the number of prediction of each class (Yes/No) while each row represents the real instances. $C_{0,0}$ represents true negatives, $C_{1,0}$ represents false negatives, $C_{1,1}$ deals with true positives and finally, $C_{0,1}$ is the false positive group.

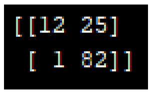


```
              precision    recall  f1-score   support

           N       0.65      0.41      0.50        37
           Y       0.77      0.90      0.83        83

   micro avg       0.75      0.75      0.75       120
   macro avg       0.71      0.65      0.67       120
weighted avg       0.74      0.75      0.73       120
```

Figure 26 'CART' classification report

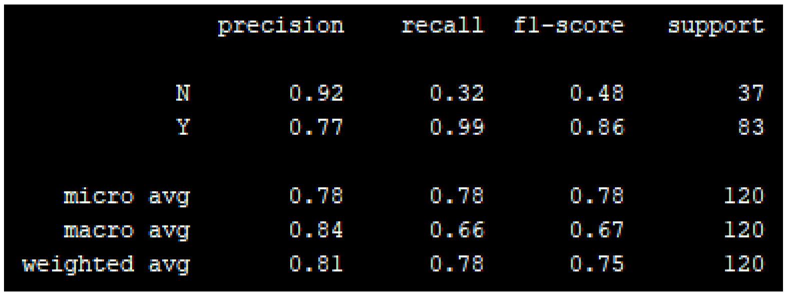
‘What can we extract from the classification report?’

- **Micro average**: averaging the total true positives, false negatives and false positives
- **Macro average**: averaging the unweighted mean per label
- **Weighted average**: averaging the support-weighted mean per label
- **Precision**= (True positives)/(True positives+ false positives)
- **Recall**= (True positives)/(True positives+ false negatives)
- **F1**= 2x ((precision*recall)/(precision+recall))



```
[[12 25]
 [ 1 82]]
```

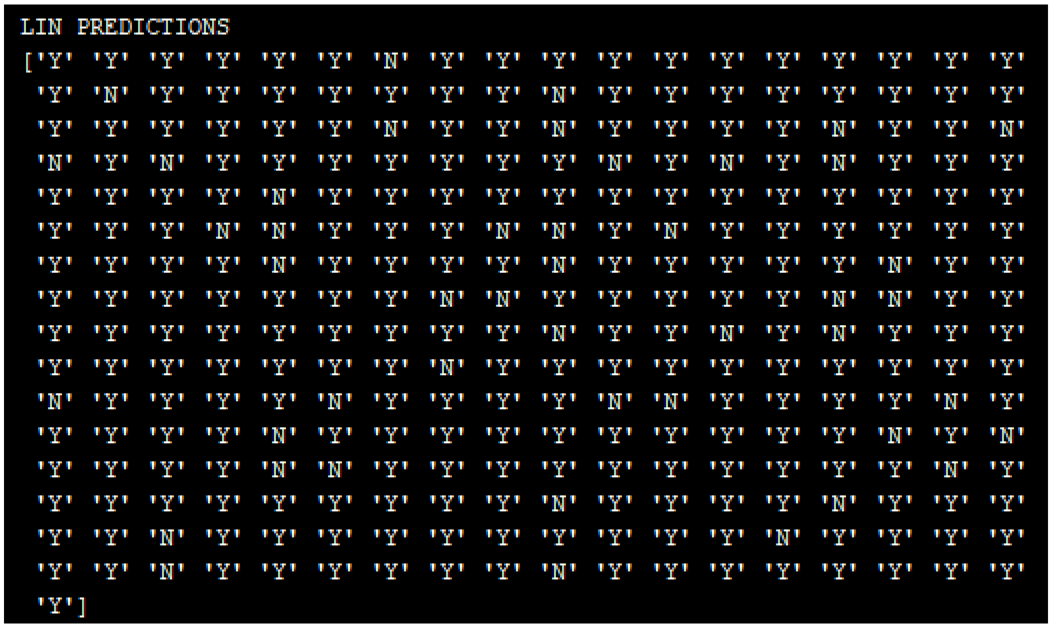
Figure 27 'LR' confusion matrix



```
              precision    recall  f1-score   support

           N       0.92      0.32      0.48        37
           Y       0.77      0.99      0.86        83

   micro avg       0.78      0.78      0.78       120
   macro avg       0.84      0.66      0.67       120
weighted avg       0.81      0.78      0.75       120
```

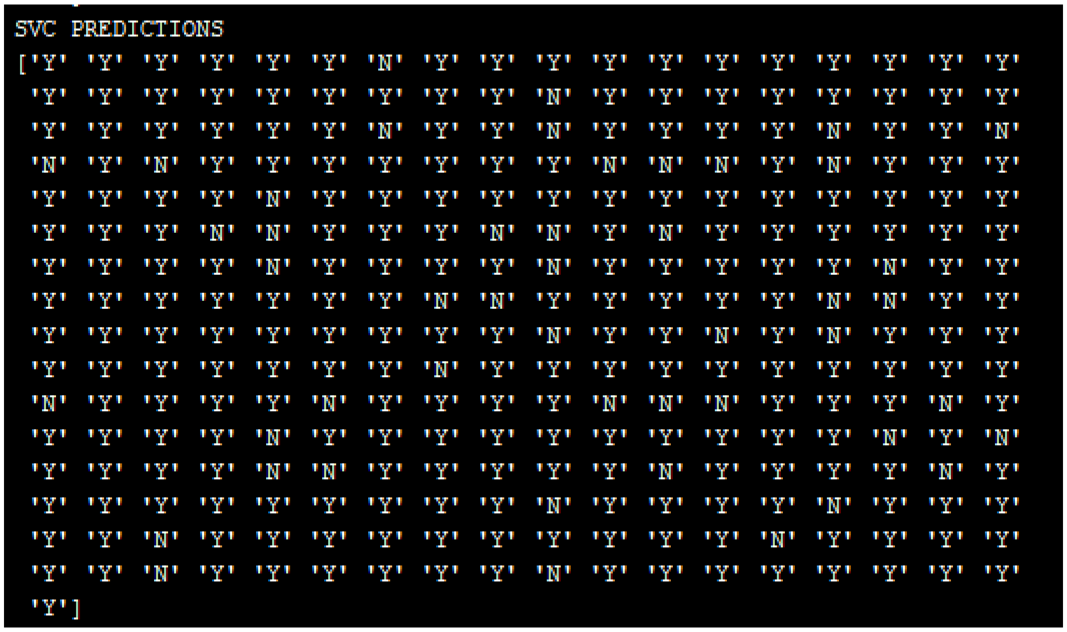
Figure 28 'LR' classification report

In case the reader wants to access to the rest of the predictions they will be provided in the pictures below:



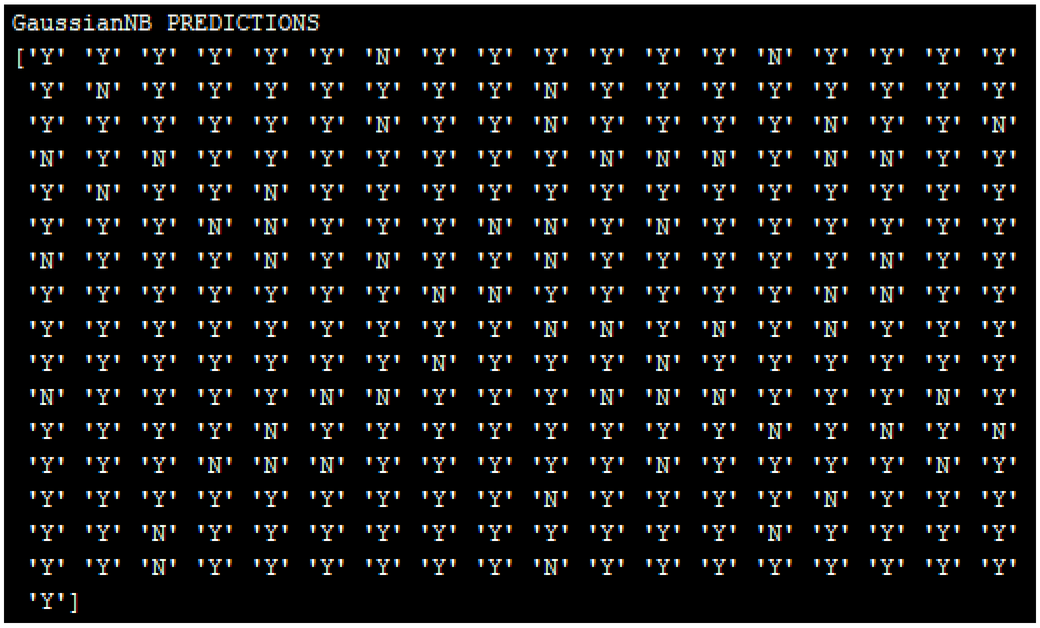
```
LIN PREDICTIONS
['Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'N'
 'N' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'N' 'Y' 'N' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'N' 'Y' 'N' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'Y'
 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'N'
 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y'
 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y']
```

Figure 29 'LIN' predictions



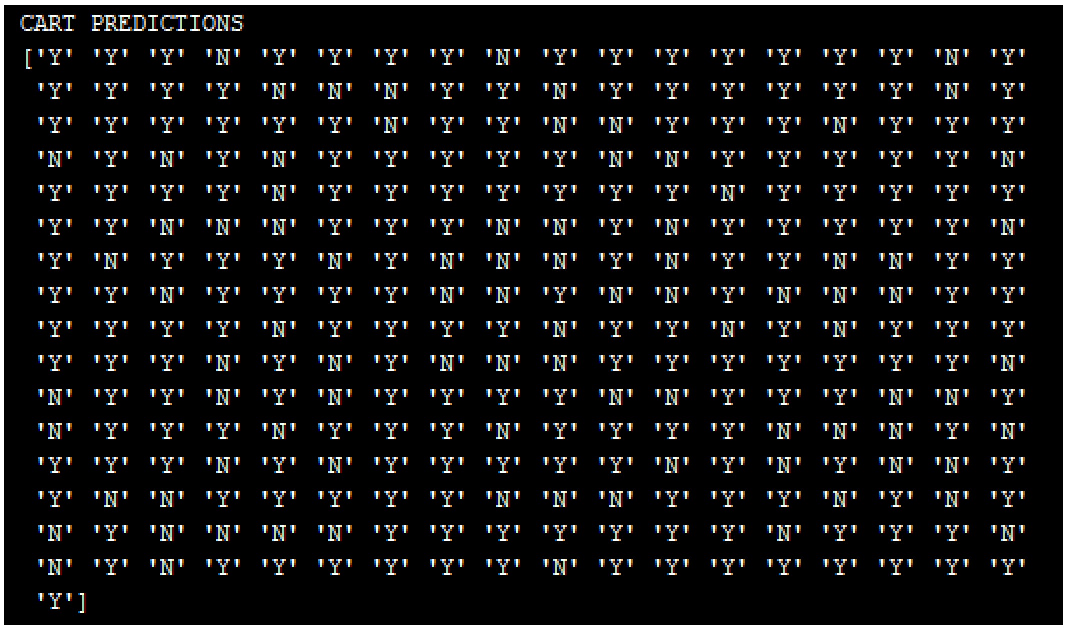
```
SVC PREDICTIONS
['Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'N'
 'N' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'N' 'Y' 'N' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'N' 'Y' 'N' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'N' 'Y' 'Y' 'Y' 'N' 'Y'
 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'N'
 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y'
 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y']
```

Figure 30 'SVC' predictions



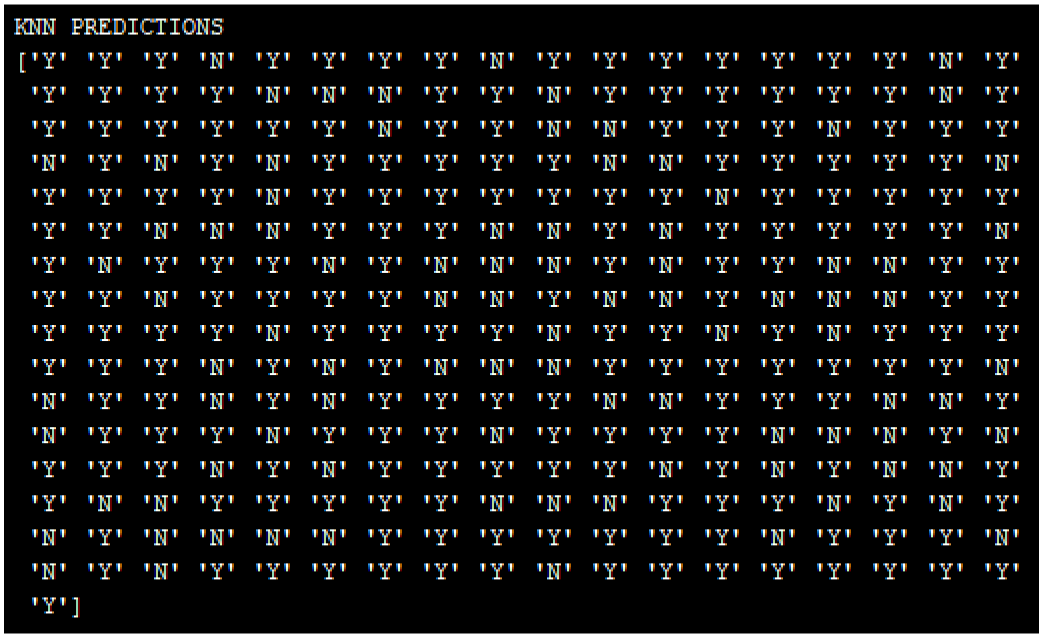
```
GaussianNB PREDICTIONS
['Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y'
 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'N'
 'N' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'N' 'Y' 'N' 'N' 'Y' 'Y'
 'Y' 'N' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'N' 'Y' 'Y' 'Y' 'N' 'Y' 'N' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'N' 'Y' 'N' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'Y' 'Y' 'N' 'N' 'N' 'Y' 'Y' 'Y' 'N' 'Y'
 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'N' 'Y' 'N'
 'Y' 'Y' 'Y' 'N' 'N' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y'
 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y']
```

Figure 31 'GaussianNB' predictions



```
CART PREDICTIONS
['Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y'
 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'N' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'N' 'N' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y'
 'N' 'Y' 'N' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'N'
 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'N' 'N' 'N' 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'N'
 'Y' 'N' 'Y' 'Y' 'Y' 'N' 'Y' 'N' 'N' 'N' 'Y' 'N' 'Y' 'Y' 'N' 'N' 'Y' 'Y'
 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'N' 'N' 'Y' 'N' 'N' 'N' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'N' 'Y' 'N' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'N' 'Y' 'N' 'Y' 'N' 'N' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N'
 'N' 'Y' 'Y' 'N' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'Y' 'Y' 'N' 'N' 'Y'
 'N' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'N' 'Y' 'N'
 'Y' 'Y' 'Y' 'N' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'N' 'Y' 'N' 'N' 'Y'
 'Y' 'N' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'N' 'Y' 'Y' 'Y' 'N' 'Y' 'N' 'Y'
 'N' 'Y' 'N' 'N' 'N' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'N'
 'N' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y']
```

Figure 32 'CART' predictions



```
KNN PREDICTIONS
['Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y'
 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'N' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'N' 'N' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y'
 'N' 'Y' 'N' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'N'
 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'N' 'N' 'N' 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'N'
 'Y' 'N' 'Y' 'Y' 'Y' 'N' 'Y' 'N' 'N' 'N' 'Y' 'N' 'Y' 'Y' 'N' 'N' 'Y' 'Y'
 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'N' 'N' 'Y' 'N' 'N' 'N' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'N' 'Y' 'N' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'N' 'Y' 'N' 'Y' 'N' 'N' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N'
 'N' 'Y' 'Y' 'N' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'Y' 'Y' 'N' 'N' 'Y'
 'N' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'N' 'Y' 'N'
 'Y' 'Y' 'Y' 'N' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'N' 'Y' 'N' 'N' 'Y'
 'Y' 'N' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'N' 'Y' 'Y' 'Y' 'N' 'Y' 'N' 'Y'
 'N' 'Y' 'N' 'N' 'N' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'N'
 'N' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y']
```

Figure 33 'KNN' predictions

Excel as an extra visualization tool

Excel was used to allow us to access to some of the insights within the data, pivot tables and side-graphs were used providing to the report a 'Business Intelligence view' since we can use this data to take decisions. The following graphs depict the statistics behind the records that had 'YES' as a 'loan_status':

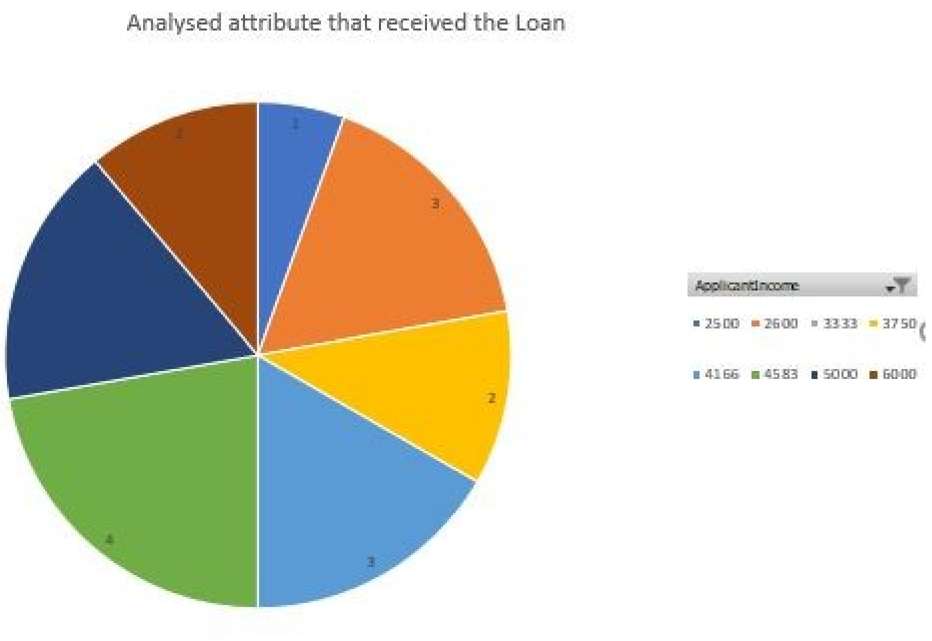


Figure 34 5 Most likely 'ApplicationIncomes'

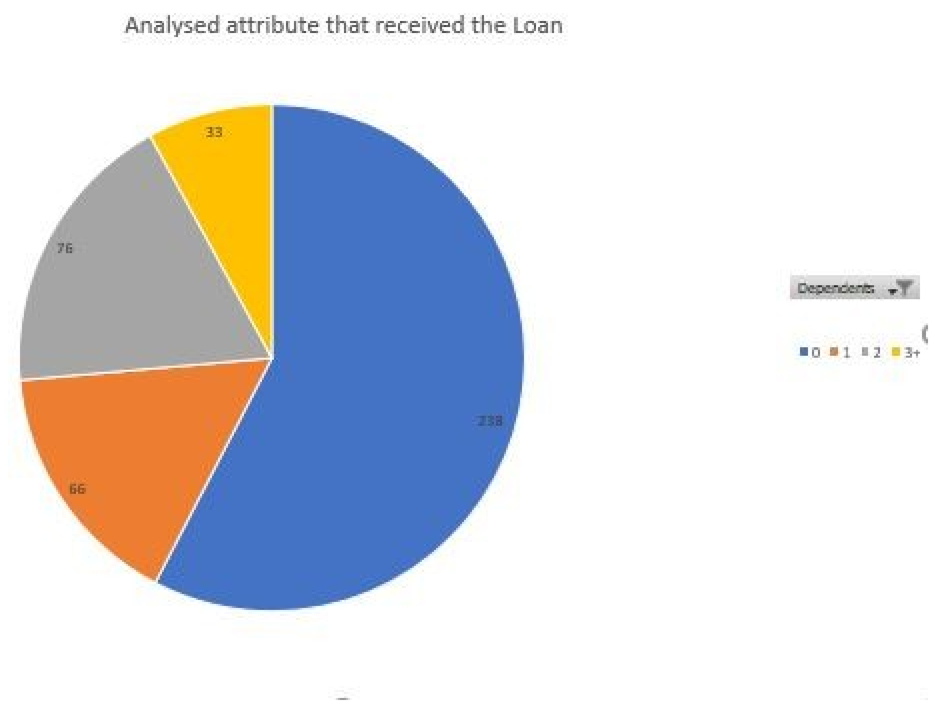


Figure 35 Dependents

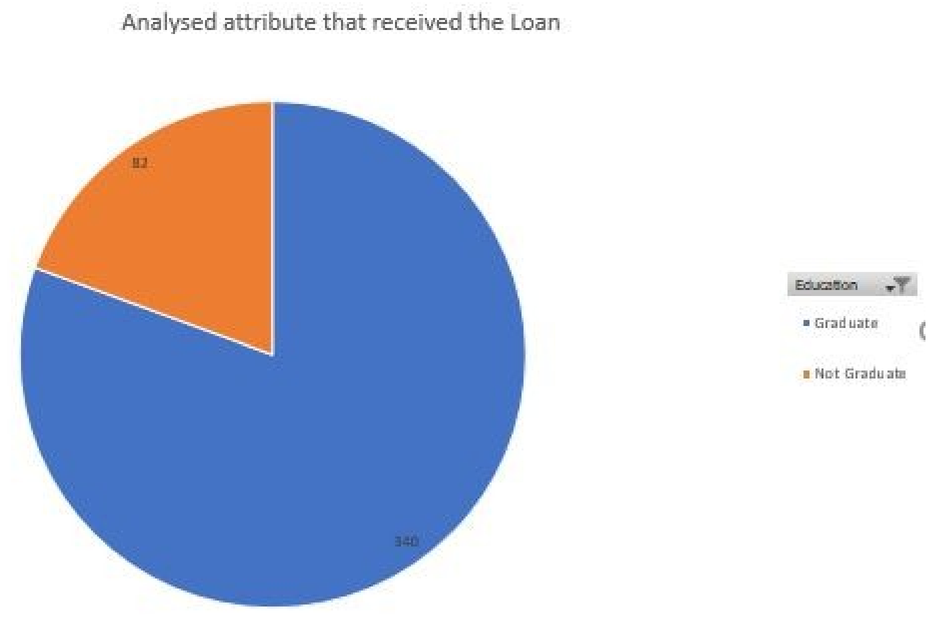


Figure 36 Education

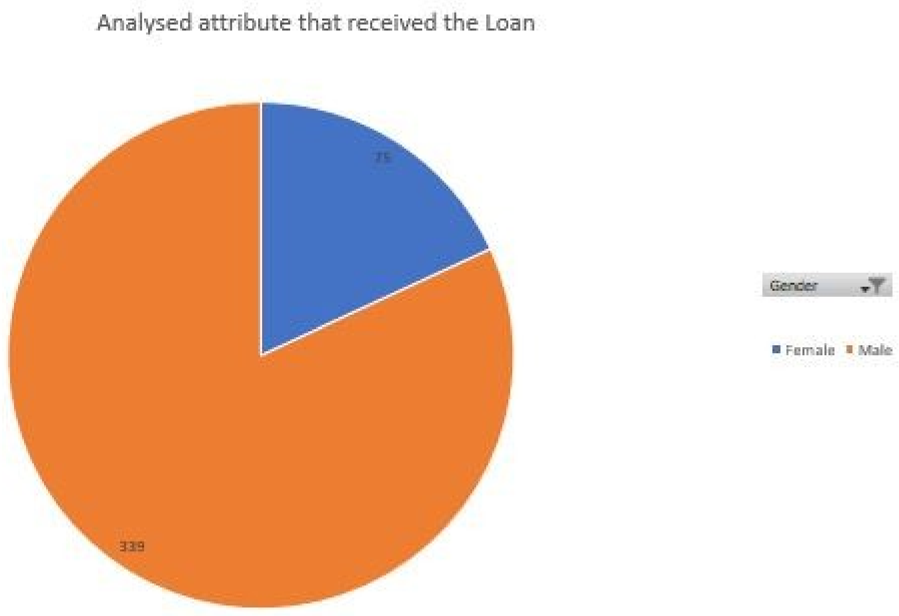


Figure 37 Gender

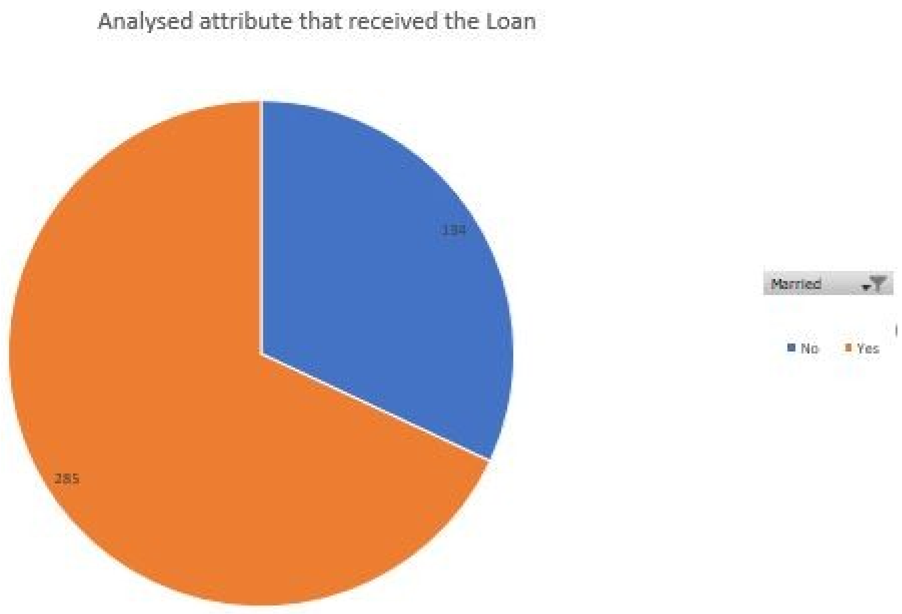


Figure 38 Married

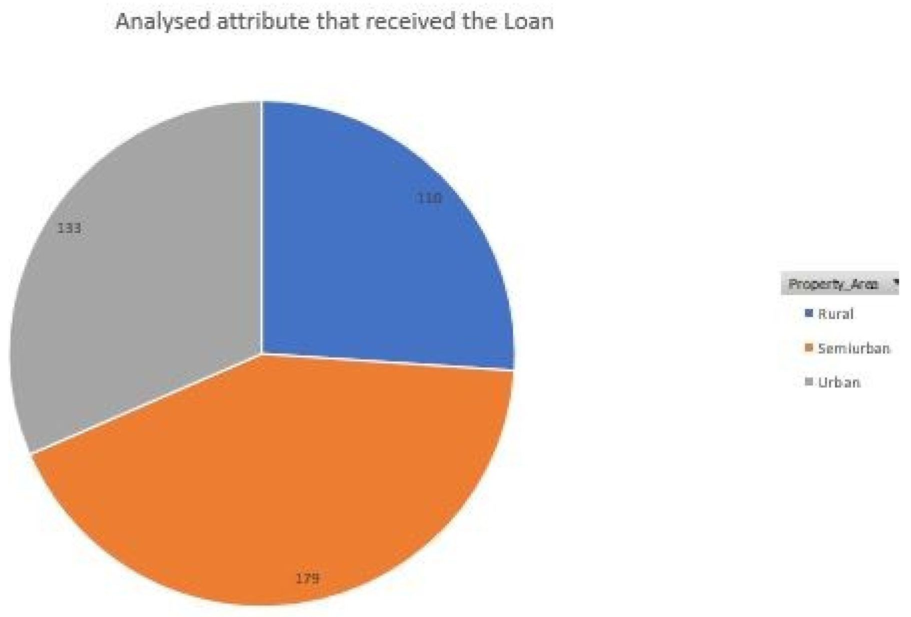


Figure 39 Property

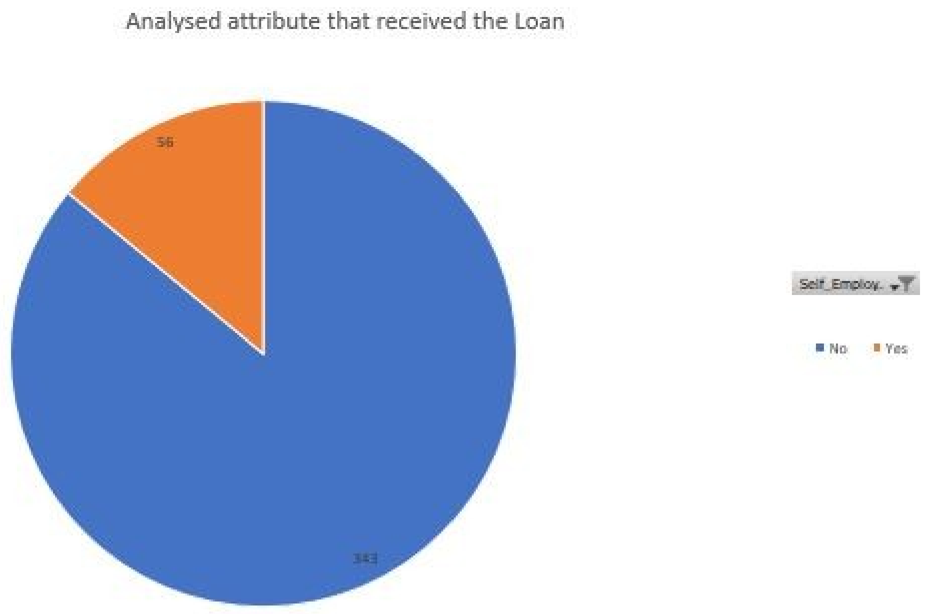


Figure 40 Self-employed

Future work

The purpose of this sub-section is critically assessing the part of the project that can be improved. All the tasks of the project were achieved, and, in some cases, the student went beyond the requirements (assessing more than 3 models in this case).

As a future work, it could be interesting to improve the accuracy by code optimizations. Also, the testing part could have been better if we had access to the labels of the records since we could have measured the accuracy. For future work, richer (bigger) datasets would be really useful.

Also, the 'NaN' elimination procedure. The student deleted all the rows with 'NaN' values, but the optimal solution would be methods like mean/median substitution either for categorical or numerical data.

To sum up, the final objective of this kind of projects is accurately predict data given different records, for this purpose optimized models must be used. In our case, some models were presented although the obtained accuracy was a little bit low in all of them. Some alternatives and solutions for this problem were explained in this report but, this information is accompanied with a paper in which further research was done and where the reader will find more answers. The aim of this report is making the reader understand the basics behind a small machine learning project.

Conclusion

To sum up, the final objective was to predict accurately the results, and for this purpose optimized models must be used. In our case, some models were analysed although the obtained accuracy was a little bit low in all of them. The whole process and code were presented and explained.

Screenshots of the code

mainCode.py

```python
#Imports
import pandas
from sklearn.preprocessing import StandardScaler
from pandas.plotting import scatter_matrix
import matplotlib.pyplot as plt
from sklearn import model_selection
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

#Loading datasets, both training and testing datasets.
#Training dataset will be splitted into training/testing again because we need to fit the
```

```
model
# and calculate the level of accuracy. This will be possible
#as this dataset has 1 more attribute (the class) that allows us to know the results of
each
# record (whether the loan was adjudicated or not)
dataset = pandas.read_csv("train_ORIGINAL.csv", sep=",")
dataset2 = pandas.read_csv("test_ORIGINAL.csv", sep=",")

#Before substituting categorical values we get rid of NaN values.
dataset=dataset.dropna()
dataset2=dataset2.dropna()
#Also, we get rid of the first column (LOAN ID) since it is useless regarding our analysis
dataset=dataset.drop(dataset.columns[0], axis='columns')
dataset2=dataset2.drop(dataset2.columns[0], axis='columns')
#Here, we perform an analysis related to the data that we already have.
print("Dataset 1 shape ",dataset.shape)
print("Dataset 2 shape ",dataset2.shape)
print(dataset.head(20))
print(dataset.describe())
print(dataset.groupby('Loan_Status').size())
dataset.plot(kind='box', subplots=False, layout=(2,2), sharex=False, sharey=False)
plt.show()
dataset.hist()
plt.show()
scatter_matrix(dataset, range_padding= 0.3, figsize = (8,8)) # figsize in inches
plt.show()

#Substitution of the categorical data and scaling process in either training and testing
datasets
#Training dataset
array = dataset.values
X_train = array[:,0:11]
Y_train = array[:,11]
for i in range(11):
    if(isinstance(X_train[:,i][2], str) ):
        encoder = LabelEncoder()
        encoder.fit(X_train[:,i])
        X_train[:,i]=encoder.fit_transform(X_train[:,1])
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)


#Testing dataset
array2 = dataset2.values
X_test2 = array2[:,0:11]
for i in range(11):
    if(isinstance(X_test2[:,i][2], str) ):
        encoder = LabelEncoder()
        encoder.fit(X_test2[:,i].astype(str))
        X_test2[:,i]=encoder.fit_transform(X_test2[:,1])
X_test2 = sc_X.fit_transform(X_test2)
```

```
#After some trials we choose a test size of the 25% of our data. It gave us good results
during
# the trials.
test_size = 0.25
seed = 7
#Now, we split the training dataset into training_2 dataset and testing_1 dataset. These
records
# will fit the model and measure the accuracy later on
X_trainf, X_test, Y_trainf, Y_test = model_selection.train_test_split(X_train,
Y_train ,stratify=Y_train, test_size=test_size, random_state=seed)
scoring = 'accuracy'

#Model definitions
models = []
models.append(('LR', LogisticRegression(solver='liblinear', multi_class='ovr')))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('LinDisAnalysis',LinearDiscriminantAnalysis()))
models.append(('GaussianNB',GaussianNB()))
models.append(('SVC',SVC(gamma='auto')))

#Cross-validation procedure
summary = {}
results = []
names = []
for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state=seed)
    cv_results = model_selection.cross_val_score(model, X_trainf, Y_trainf,
cv=kfold,scoring='accuracy')
    results.append(cv_results)
    names.append(name)
    summary[name]=[cv_results.mean(), cv_results.std()]
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)

#Ploting results
fig = plt.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()

print("CART PREDICTIONS")
#Estimator definition
cart = DecisionTreeClassifier()
#Estimator fitting process
cart.fit(X_trainf, Y_trainf)
#Testing and result
predictions = cart.predict(X_test)
#Results analysis
print(accuracy_score(Y_test, predictions))
print(confusion_matrix(Y_test, predictions))
print(classification_report(Y_test, predictions))
summary['CART'].append(accuracy_score(Y_test, predictions))

print("LR PREDICTIONS")
lr = LogisticRegression(solver='liblinear', multi_class='ovr')
lr.fit(X_trainf, Y_trainf)
predictions = lr.predict(X_test)
print(accuracy_score(Y_test, predictions))
print(confusion_matrix(Y_test, predictions))
print(classification_report(Y_test, predictions))
summary['LR'].append(accuracy_score(Y_test, predictions))

print("KNN PREDICTIONS")
knn = KNeighborsClassifier()
knn.fit(X_trainf, Y_trainf)
predictions = knn.predict(X_test)
print(accuracy_score(Y_test, predictions))
print(confusion_matrix(Y_test, predictions))
print(classification_report(Y_test, predictions))
summary['KNN'].append(accuracy_score(Y_test, predictions))

print("LinDisAnalysis PREDICTIONS (LinearDiscriminantAnalysis)")
LinDisAnalysis=LinearDiscriminantAnalysis()
LinDisAnalysis.fit(X_trainf, Y_trainf)
predictions2 = LinDisAnalysis.predict(X_test)
print(accuracy_score(Y_test, predictions2))
print(confusion_matrix(Y_test, predictions2))
print(classification_report(Y_test, predictions2))
summary['LinDisAnalysis'].append(accuracy_score(Y_test, predictions))


print("GaussianNB PREDICTIONS")
GaussianNB=GaussianNB()
GaussianNB.fit(X_trainf, Y_trainf)
predictions3 = GaussianNB.predict(X_test)
print(accuracy_score(Y_test, predictions3))
print(confusion_matrix(Y_test, predictions3))
print(classification_report(Y_test, predictions3))
summary['GaussianNB'].append(accuracy_score(Y_test, predictions))

print("SVC PREDICTIONS")
SVC=SVC(gamma='auto')
SVC.fit(X_trainf, Y_trainf)
predictions4 = SVC.predict(X_test)
print(accuracy_score(Y_test, predictions4))
print(confusion_matrix(Y_test, predictions4))
print(classification_report(Y_test, predictions4))
summary['SVC'].append(accuracy_score(Y_test, predictions))


print("Summary ")
for i in summary:
    print("Model: ",i, " Accuracy(*): ",float(summary[i][0])," STD(*): ",float(summary[i]
[1]),
          " Tested accuracy: ",float(summary[i][2]))

x = [1,2,3,4,5,6]
plt.bar(x, height=[summary['LR'][0],summary['KNN'][0],summary['CART']
[0],summary['LinDisAnalysis'][0],
                   summary['GaussianNB'][0],summary['SVC'][0]])
plt.xticks(x, list(summary.keys()))
plt.ylabel('Training accuracy')
plt.xlabel('Models')
plt.show()

plt.bar(x, height=[summary['LR'][1],summary['KNN'][1],summary['CART']
[1],summary['LinDisAnalysis'][1],
                   summary['GaussianNB'][1],summary['SVC'][1]])
plt.xticks(x, list(summary.keys()))
plt.ylabel('STD of the training accuracy')
plt.xlabel('Models')
plt.show()

plt.bar(x, height=[summary['LR'][2],summary['KNN'][2],summary['CART']
[2],summary['LinDisAnalysis'][2],
                   summary['GaussianNB'][2],summary['SVC'][2]])
plt.xticks(x, list(summary.keys()))
plt.ylabel('Tested accuracy')
plt.xlabel('Models')
plt.show()

#Dataset 2 predictions using all models we had analized
print("LIN PREDICTIONS")
predictions5 = LinDisAnalysis.predict(X_test2)
print(predictions5)

print("LR PREDICTIONS")
predictions6 = lr.predict(X_test2)
print(predictions6)

print("SVC PREDICTIONS")
predictions7 = SVC.predict(X_test2)
print(predictions7)

print("GaussianNB PREDICTIONS")
predictions8 = GaussianNB.predict(X_test2)
print(predictions8)

print("CART PREDICTIONS")
predictions9 = cart.predict(X_test2)
print(predictions9)

print("KNN PREDICTIONS")
predictions10 = cart.predict(X_test2)
```

References

Hurwitz, J., & Kirsch, D. (2018). Machine learning for dummies. [Online] Available at: https://mscdss.ds.unipi.gr/wp-content/uploads/2018/02/Untitled-attachment-00056-2-1.pdf [Accessed 21/03/2019]

Yao M, 2017. 'Chihuahua or muffin? My search for the best computer vision API', Medium [Online] Available at: https://medium.freecodecamp.org/chihuahua-or-muffin-my-search-for-the-best-computer-vision-api-cbda4d6b425d [Accessed 21/03/2019]

Microsoft Azure, 2019, 'Machine learning algorithm cheat sheet for Azure Machine Learning Studio' [Online] Available at: https://docs.microsoft.com/en-us/azure/machine-learning/studio/algorithm-cheat-sheet [Accessed: 22/03/2019]

Hassouna M, 2019, Seminar 7: Data Mining, Business Intelligence and Data Mining course, The University of Greenwich. Delivered 26/2/2019

Vuong T, 2019. Seminar 4: Machine Learning Landscape, Applied Machine Learning, The University of Greenwich. Delivered 7/2/2019

Vuong T, 2019. Seminar 5: Machine Learning Processes, Applied Machine Learning, The University of Greenwich. Delivered 14/2/2019