CAPSTONE: DEFAULT OF CREDIT CARD CLIENTS

Abstract

Random Forest Optimized is the model selected to identify default in credit card clients, this helps the business to take decisions regarding new credit card products and develop effective and appropriate collection policies and strategies

Angela Santana

asantan4@my.centennialcollege.ca

Contents

Executive Summary

Financial crisis (the most recent one 2007-2009) have highlighted the importance of risk management in the banking industry. Credit Risk Management Practices and policies in the credit card product, are key to reduce exposure and reduce the potential number of accounts destined to default. Based on this data analysis, the Random Forest Optimized model is the best to predict possible default of credit card clients.

This analysis benefits institutions to manage credit card portfolios to increase value and cut credit lines on accounts likely to go on default.

Financial threats are showing a trend on the credit risk of commercial banks as the incredible improvement in the financial industry has emerged. Thus, one of the biggest threats commercial banks faces is predicting the risk of credit customers.
Default payments of credit card clients represents a significant problem for the Bank TID Taiwan and one of challenges is to identify a customer who can comply or defaulting in card obligation.

Recent studies are mainly focused on improving the performance of the classifier for credit card default prediction rather than an interpretable model. Through appropriate identification of possible default for credit card clients the Bank TID Taiwan would be able to design proper credit collection strategies and reduce the default rates. Building a classification model to predict default can support the bank approaches. The model documentation, and recommendation are provided in this report.

1. Executive Introduction

The Analytics objective is to create a model from a Default of Credit Card client's dataset, taking as base the demographic and behavioral information and find a classification model to predict the future default of credit card clients

2. Executive Objective

The main objective is to find a classification model to predict the TID Bank Taiwan’s Default of Credit Card clients. The credit card issuer wants to target consumers by using demographics and behavioral information. These factors are often based on credit card reports on how a person spends and repays obligations over time.

The present model was created, to predict and identify which clients will fail in their payments on the credit card obligations. This as part of the regular credit risk solutions (as stress testing, scores, and limits monitoring, etc.)

The model will be key to reduce loan losses and build effective collection policies within the bank

3. Executive Model Description

The developed models were Decision Tree, Random Forest, and Gradient Boosting (binary classification), using the selected variables (age, education, marital status, repayments status in different months, paid amounts...) to predict the default of credit card which variables are most significant related to the company. We will also use other techniques to verify our findings

4. Executive Recommendations

The data analysis generated by the Random Forest Optimized is the model selected to identify default in credit card clients, this to grant or deny new card products and develop effective and appropriate collection policies and strategies.

Introduction

The default of credit card client dataset is public, and contains information on default payments, demographic factors, credit data, history of payment, and bill statements of credit card clients in Taiwan from April 2005 to September 2005.
The present report is for academic purposes only.
The data exploration analysis and modeling were completed in Python.

5. Business background

The banking industry is part of the larger financial services industry, that are economic services, and includes insurance institutions, brokerages, mutual funds, individual asset managers, credit unions, trust companies, pension funds and similar organizations.

The banking industry, as part of the financial sector, includes systems of financial institutions that allow people to store and use their money with different purposes as saving or investments, also, those provide resources for individuals and organizations by distributing loans and giving credits cards for applicants, so they will be able to use those funds for various purposes such as acquiring property, education, and free investment.

Issuance of credit cards and processing of credit card transactions and billing is a type of financial service. A credit card is a payment card issued to cardholders to enable them to pay a merchant for goods and services.

Taiwan Banking industry has experienced changes as economy evolves and the main participation in the market is the following as The Top Banks in Taiwan (CFI, 2021):

- CTBC Bank

  Established in 1966, CTBC Bank is one of the largest private banks in Taiwan. Formerly known as China Securities and Investment Corporation, the bank became a subsidiary of CTBC Financial Holding Co. Ltd. In 2002.

- Hua Nan Com Bank

  Hua Nam Com Bank provides commercial banking products and services in Taiwan, as well as in other countries. Established in 1919, the bank oversees a network of 186 branches in the country, one offshore banking unit, and 12 overseas branches. It employs around 8,000 staff, operating out of Taipei City.

- Bank of Taiwan

  Founded in 1946, the Bank of Taiwan is the first of the government-owned banks in Taiwan. The bank now operates a subsidiary of Taiwan Financial Holdings. It manages a network of 163 branches in the country, 10 overseas branches, one sub-branch, two representative offices, one offshore banking unit, and three preparatory offices. With around 8,280 employees, it is based in Taipei City.

- Shanghai Commercial and Savings Bank

  Headquartered in Taipei City, Shanghai Commercial and Savings Bank offers commercial banking services to retail and corporate clients in Taiwan. Established in 1915, it currently employs 2,600 individuals and operates 72 branches. It also manages seven overseas branches in Shanghai, Shenzhen, Shanghai Pilot Free Trade Zone, London, New York, San Francisco, and Los Angeles, USA.

- Citibank Taiwan

  Established in 1964, Citibank Taiwan provides financial products and services to consumers, corporations, governments, and institutions. The bank operates as a subsidiary of Citibank Overseas Investment Corporation. With headquarters in Taipei City, its workforce comprises around 3,986 employees.

The share of market is local, also regional in the south Asian region. The main products offer by these institutions are offering savings, loans, mortgages, and related financial services to consumers and businesses.

Financial threats are showing a trend on the credit risk of commercial banks as the incredible improvement in the financial industry has emerged. Thus, one of the biggest threats commercial banks faces is predicting the risk of credit customers

6. Problem Statement

Financial threats are showing a trend on the credit risk of commercial banks as the incredible improvement in the financial industry has emerged. Thus, one of the biggest threats commercial banks faces is predicting the risk of credit customers.

When a cardholder does not pay on the due day their credit card obligations, the default can impact the business as the money given is not being recovered as planned and collection campaigns must be put in place.

7. Objectives & Measurement

Find a classification model to predict the Credit Default of clients of a bank in Taiwan. Credit card issuers as TID Bank Taiwan want to target consumers by using information about their behaviors and demographics. Behaviors are often based on credit card reports on how a person spends and repays obligations over time.

The project will focus on working with a default on credit card dataset, to demonstrate how a predictive model can foresee if a client will fail to make regular payments on their financial product or not. This will be a Binary Classification task, creating a Decision Tree model.

Measurements:

The Analytics objective is to create a model from a Default of Credit Card client's dataset, taking as base the demographic and behavioral information and find a classification model to predict the future default of credit card clients
The dataset has 30000 registers, with previous payments, amount of each Bill paid, and payment status by client as they were or not in default; the data has also socio demographic information as age, education, sex, marital status; with those variables a model will be identify to predict clients that will fail with their credit card payments and so design the appropriate and effective collection policies and grant new credit cards with highly repayment probability.

Decision criteria: FScore metric 1 balance between precision, identifying default of credit card clients and recovery or recall metric (How many false positives am I identifying?)

Clients who are going to be in default and avoid false positives (the model says if a client can fail on their payments). FScore controls both metrics. The FScore with higher rate will be selected. High precision in identifying default on credit card clients.

Success measures/metrics:
The key performance indicators or key metrics to meet the proposed objectives are the following cited:

- Credit card usage return rate.
- Collection rate effectiveness by creating collection strategies and policies based on the model.
- Reduction on the rate of default

8. Assumptions and Limitations

For the analysis and the model, we are not taking into consideration information as:

- Client's credit history.
- Client's payment behavior with other financial or banking entities.
- Macroeconomic environment of the country, or country's unemployment rates, inflation, among other factors.

Also, in classification problems, an unbalanced dataset is also crucial to improve model performance because most cases are in one class and only a few examples are in other categories.

Data Sources

Data Source: www.kaggle.com

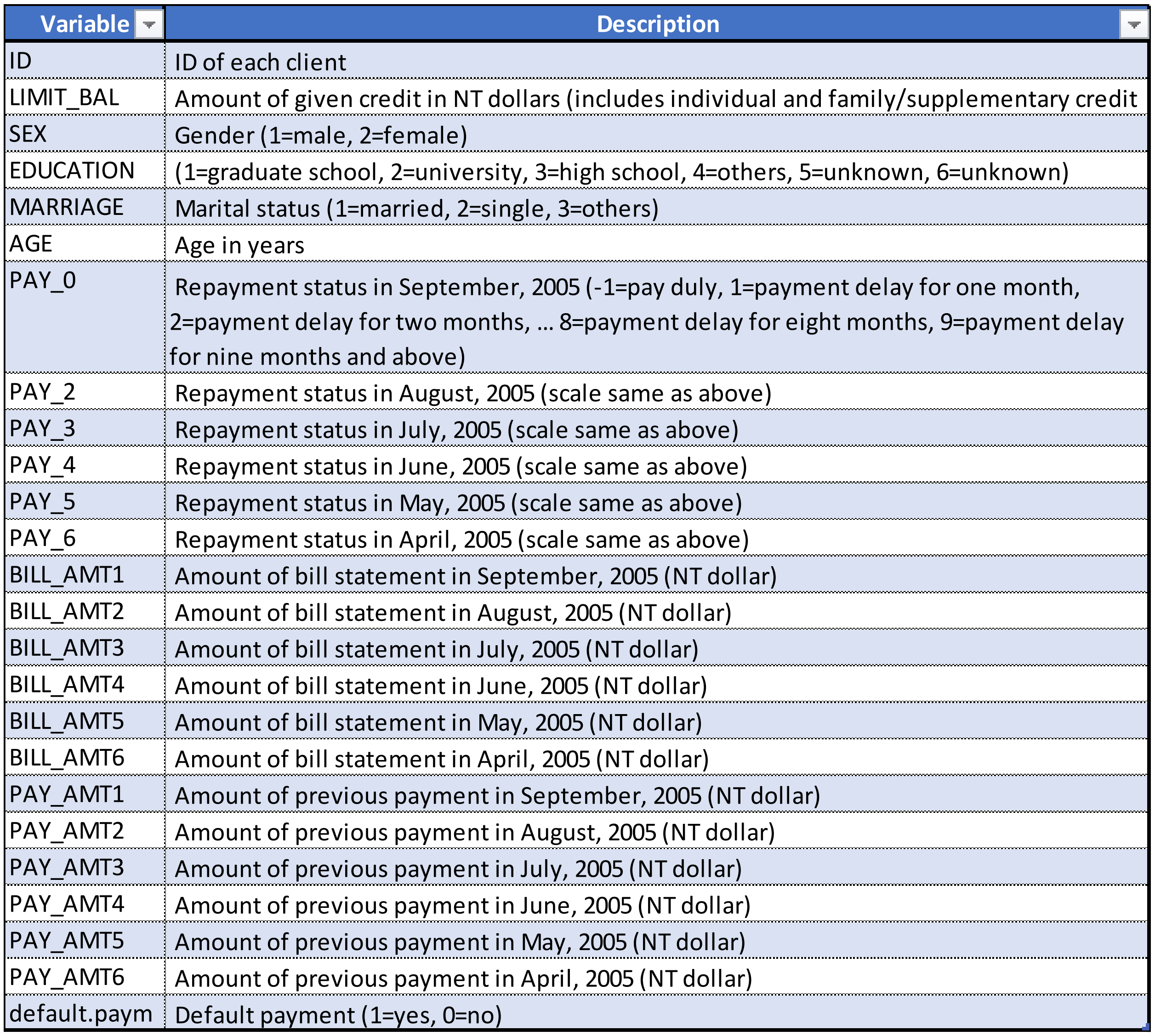
Dataset: https://www.kaggle.com/datasets/uciml/default-of-credit-card-clients-dataset

Dataset Information: This dataset contains information on default payments, demographic factors, credit data, history of payment, and bill statements of credit card clients in Taiwan from April 2005 to September 2005 (UCI Machine Learning Repository, 2016)

The target variable is Default.

The detailed information is shown below:

Figure 1



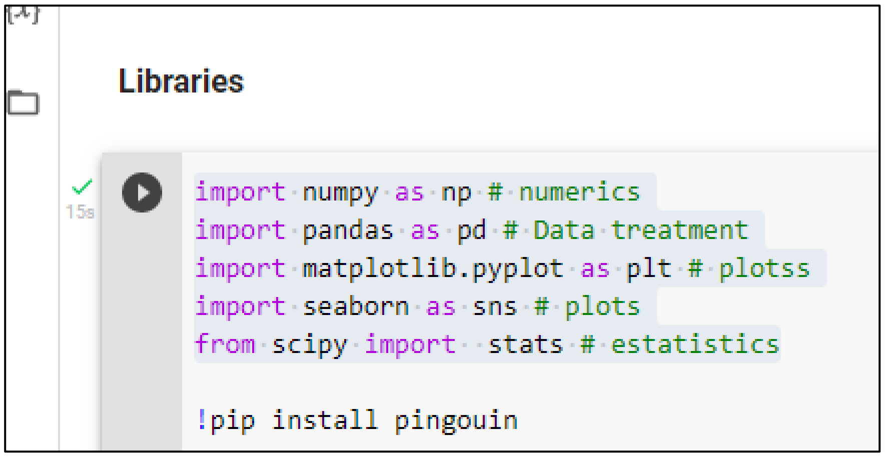
Variable	Description
ID	ID of each client
LIMIT_BAL	Amount of given credit in NT dollars (includes individual and family/supplementary credit
SEX	Gender (1=male, 2=female)
EDUCATION	(1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown)
MARRIAGE	Marital status (1=married, 2=single, 3=others)
AGE	Age in years
PAY_0	Repayment status in September, 2005 (-1=pay duly, 1=payment delay for one month, 2=payment delay for two months, … 8=payment delay for eight months, 9=payment delay for nine months and above)
PAY_2	Repayment status in August, 2005 (scale same as above)
PAY_3	Repayment status in July, 2005 (scale same as above)
PAY_4	Repayment status in June, 2005 (scale same as above)
PAY_5	Repayment status in May, 2005 (scale same as above)
PAY_6	Repayment status in April, 2005 (scale same as above)
BILL_AMT1	Amount of bill statement in September, 2005 (NT dollar)
BILL_AMT2	Amount of bill statement in August, 2005 (NT dollar)
BILL_AMT3	Amount of bill statement in July, 2005 (NT dollar)
BILL_AMT4	Amount of bill statement in June, 2005 (NT dollar)
BILL_AMT5	Amount of bill statement in May, 2005 (NT dollar)
BILL_AMT6	Amount of bill statement in April, 2005 (NT dollar)
PAY_AMT1	Amount of previous payment in September, 2005 (NT dollar)
PAY_AMT2	Amount of previous payment in August, 2005 (NT dollar)
PAY_AMT3	Amount of previous payment in July, 2005 (NT dollar)
PAY_AMT4	Amount of previous payment in June, 2005 (NT dollar)
PAY_AMT5	Amount of previous payment in May, 2005 (NT dollar)
PAY_AMT6	Amount of previous payment in April, 2005 (NT dollar)
default.paym	Default payment (1=yes, 0=no)

9. Data Set Introduction

Python is the programming language used to process and analyze the dataset. Python is a high-level, interpreted, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation.

The necessary libraries were imported:

Figure 2

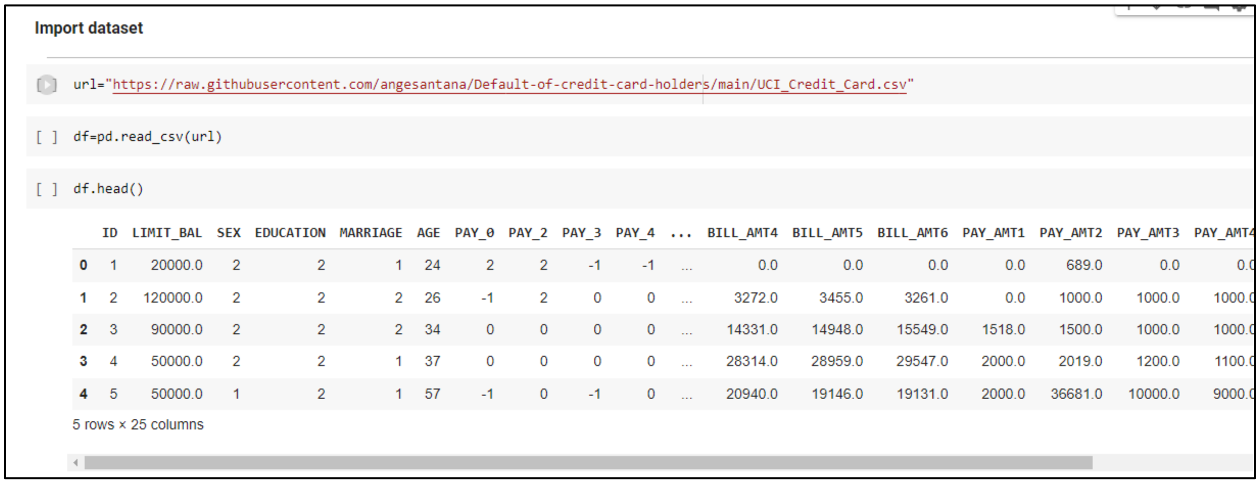


Libraries

```
import numpy as np # numerics
import pandas as pd # Data treatment
import matplotlib.pyplot as plt # plotss
import seaborn as sns # plots
from scipy import  stats # estatistics

!pip install pingouin
```

Dataset import and have a first view of the data:

Figure 3



Import dataset

```
url="https://raw.githubusercontent.com/angesantana/Default-of-credit-card-holders/main/UCI_Credit_Card.csv"
```

```
[ ] df=pd.read_csv(url)
```

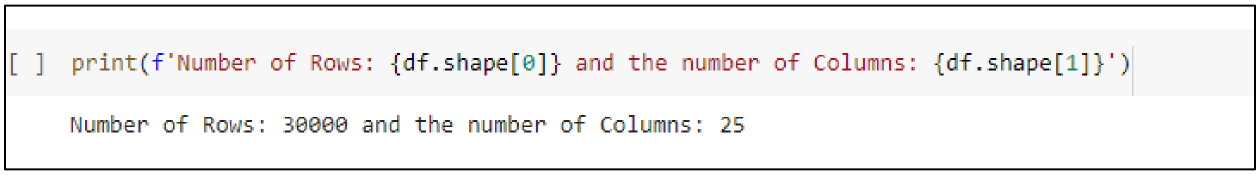
```
[ ] df.head()
```

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4
0	1	20000.0	2	2	1	24	2	2	-1	-1	...	0.0	0.0	0.0	0.0	689.0	0.0	0.
1	2	120000.0	2	2	2	26	-1	2	0	0	...	3272.0	3455.0	3261.0	0.0	1000.0	1000.0	1000.
2	3	90000.0	2	2	2	34	0	0	0	0	...	14331.0	14948.0	15549.0	1518.0	1500.0	1000.0	1000.
3	4	50000.0	2	2	1	37	0	0	0	0	...	28314.0	28959.0	29547.0	2000.0	2019.0	1200.0	1100.
4	5	50000.0	1	2	1	57	-1	0	-1	0	...	20940.0	19146.0	19131.0	2000.0	36681.0	10000.0	9000.

5 rows × 25 columns

Data dimension:

Figure 4



```
[ ] print(f'Number of Rows: {df.shape[0]} and the number of Columns: {df.shape[1]}')

    Number of Rows: 30000 and the number of Columns: 25
```

10. Exclusions

In the initial analysis there were not exclusions. Please see more detail in the Data Exploration section, as some variables were imputed or received different treatments.

11. Data Dictionary

ID: ID of each client

LIMIT_BAL: Amount of given credit in NT dollars (includes individual and family/supplementary credit

SEX: Gender (1=male, 2=female)

EDUCATION: (1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown)

MARRIAGE: Marital status (1=married, 2=single, 3=others)

AGE: Age in years

PAY_0: Repayment status in September 2005 (-1=pay duly, 1=payment delay for one month, 2=payment delay for two months, … 8=payment delay for eight months, 9=payment delay for nine months and above)

PAY_2: Repayment status in August 2005 (scale same as above)

PAY_3: Repayment status in July 2005 (scale same as above)

PAY_4: Repayment status in June 2005 (scale same as above)

PAY_5: Repayment status in May 2005 (scale same as above)

PAY_6: Repayment status in April 2005 (scale same as above)

BILL_AMT1: Amount of bill statement in September 2005 (NT dollar)

BILL_AMT2: Amount of bill statement in August 2005 (NT dollar)

BILL_AMT3: Amount of bill statement in July 2005 (NT dollar)

BILL_AMT4: Amount of bill statement in June 2005 (NT dollar)

BILL_AMT5: Amount of bill statement in May 2005 (NT dollar)

BILL_AMT6: Amount of bill statement in April 2005 (NT dollar)

PAY_AMT1: Amount of previous payment in September 2005 (NT dollar)

PAY_AMT2: Amount of previous payment in August 2005 (NT dollar)

PAY_AMT3: Amount of previous payment in July 2005 (NT dollar)

PAY_AMT4: Amount of previous payment in June 2005 (NT dollar)

PAY_AMT5: Amount of previous payment in May 2005 (NT dollar)

PAY_AMT6: Amount of previous payment in April 2005 (NT dollar)

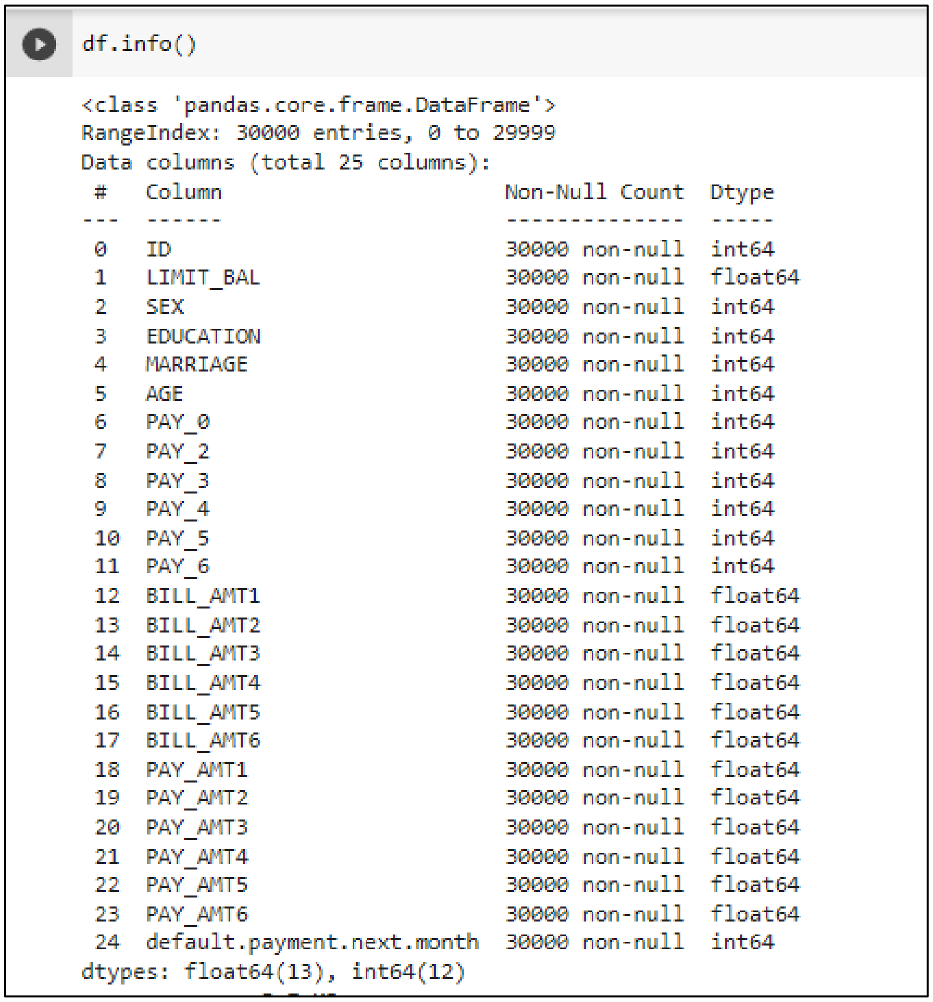
default.payment.next.month: Default payment (1=yes, 0=no)

Data Exploration

12. Data Exploration Techniques

12.1. Review of the Type of Variables

Figure 5

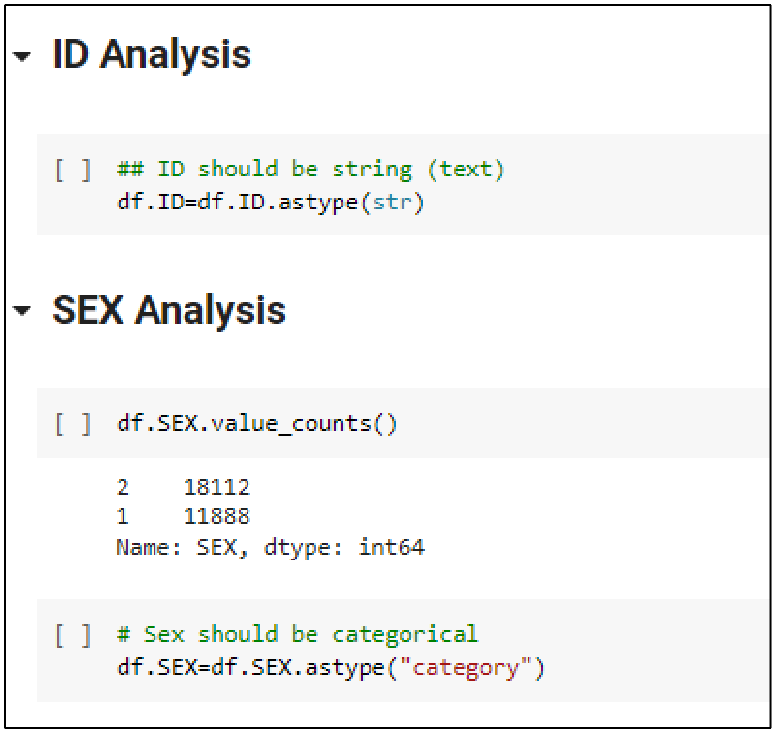


```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 25 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   ID                          30000 non-null  int64  
 1   LIMIT_BAL                   30000 non-null  float64
 2   SEX                         30000 non-null  int64  
 3   EDUCATION                   30000 non-null  int64  
 4   MARRIAGE                    30000 non-null  int64  
 5   AGE                         30000 non-null  int64  
 6   PAY_0                       30000 non-null  int64  
 7   PAY_2                       30000 non-null  int64  
 8   PAY_3                       30000 non-null  int64  
 9   PAY_4                       30000 non-null  int64  
 10  PAY_5                       30000 non-null  int64  
 11  PAY_6                       30000 non-null  int64  
 12  BILL_AMT1                   30000 non-null  float64
 13  BILL_AMT2                   30000 non-null  float64
 14  BILL_AMT3                   30000 non-null  float64
 15  BILL_AMT4                   30000 non-null  float64
 16  BILL_AMT5                   30000 non-null  float64
 17  BILL_AMT6                   30000 non-null  float64
 18  PAY_AMT1                    30000 non-null  float64
 19  PAY_AMT2                    30000 non-null  float64
 20  PAY_AMT3                    30000 non-null  float64
 21  PAY_AMT4                    30000 non-null  float64
 22  PAY_AMT5                    30000 non-null  float64
 23  PAY_AMT6                    30000 non-null  float64
 24  default.payment.next.month  30000 non-null  int64  
dtypes: float64(13), int64(12)
```

The type of variables for ID, SEX, EDUCATION, MARRIAGE, PAYMENT STATUS and default.payment.next.month, was modified from int into categorical as it can be seen below:

Figure 6



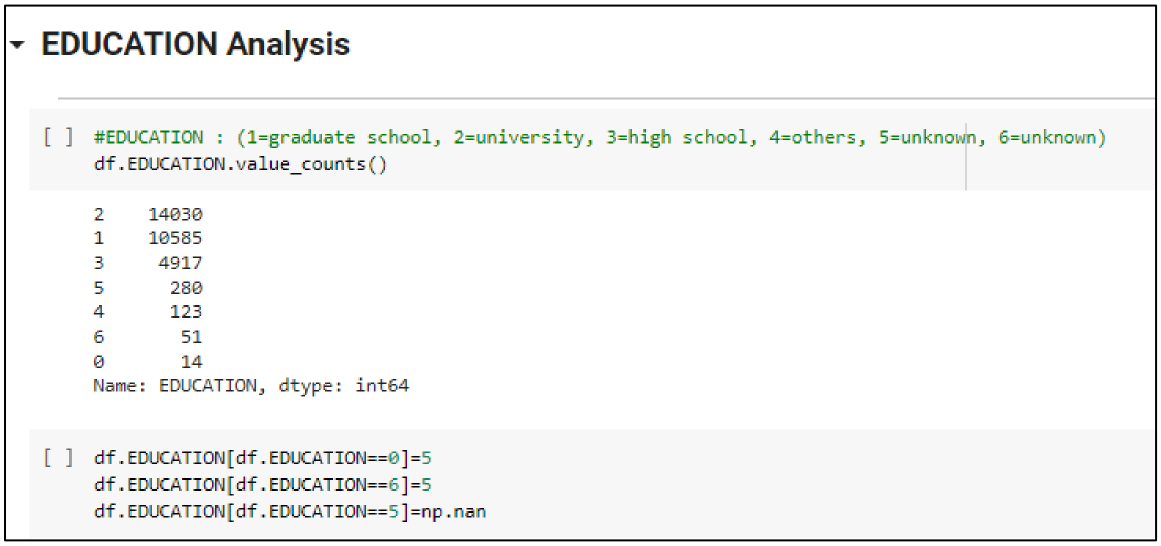
ID Analysis

```
[ ]  ## ID should be string (text)
     df.ID=df.ID.astype(str)
```

SEX Analysis

```
[ ]  df.SEX.value_counts()

     2    18112
     1    11888
     Name: SEX, dtype: int64

[ ]  # Sex should be categorical
     df.SEX=df.SEX.astype("category")
```

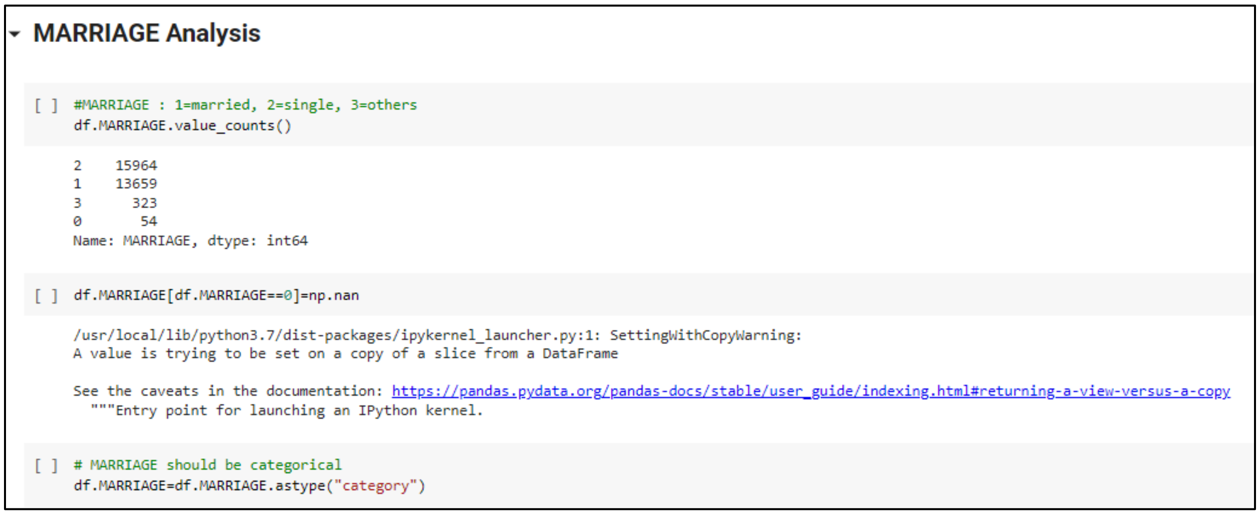
Figure 7



EDUCATION Analysis

```
[ ]  #EDUCATION : (1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown)
     df.EDUCATION.value_counts()

     2    14030
     1    10585
     3     4917
     5      280
     4      123
     6       51
     0       14
     Name: EDUCATION, dtype: int64

[ ]  df.EDUCATION[df.EDUCATION==0]=5
     df.EDUCATION[df.EDUCATION==6]=5
     df.EDUCATION[df.EDUCATION==5]=np.nan
```

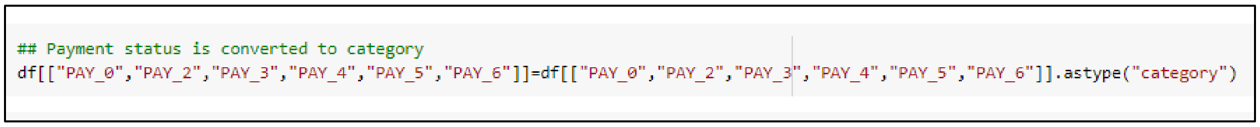
Figure 8



MARRIAGE Analysis

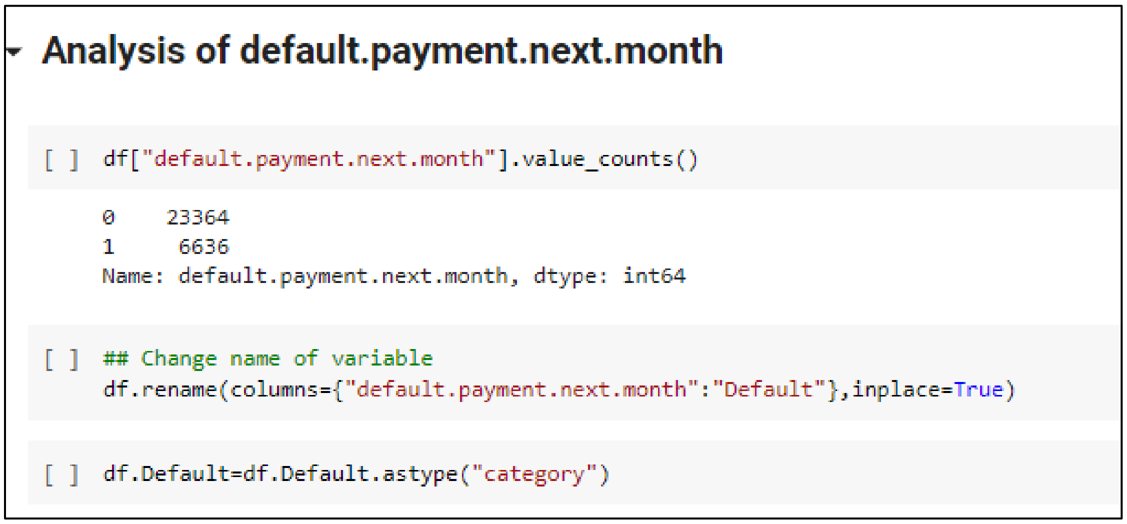
```
[ ] #MARRIAGE : 1=married, 2=single, 3=others
    df.MARRIAGE.value_counts()

    2    15964
    1    13659
    3      323
    0       54
    Name: MARRIAGE, dtype: int64

[ ] df.MARRIAGE[df.MARRIAGE==0]=np.nan

    /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
    A value is trying to be set on a copy of a slice from a DataFrame

    See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
      """Entry point for launching an IPython kernel.

[ ] # MARRIAGE should be categorical
    df.MARRIAGE=df.MARRIAGE.astype("category")
```

Figure 9



```
## Payment status is converted to category
df[["PAY_0","PAY_2","PAY_3","PAY_4","PAY_5","PAY_6"]]=df[["PAY_0","PAY_2","PAY_3","PAY_4","PAY_5","PAY_6"]].astype("category")
```

Figure 10

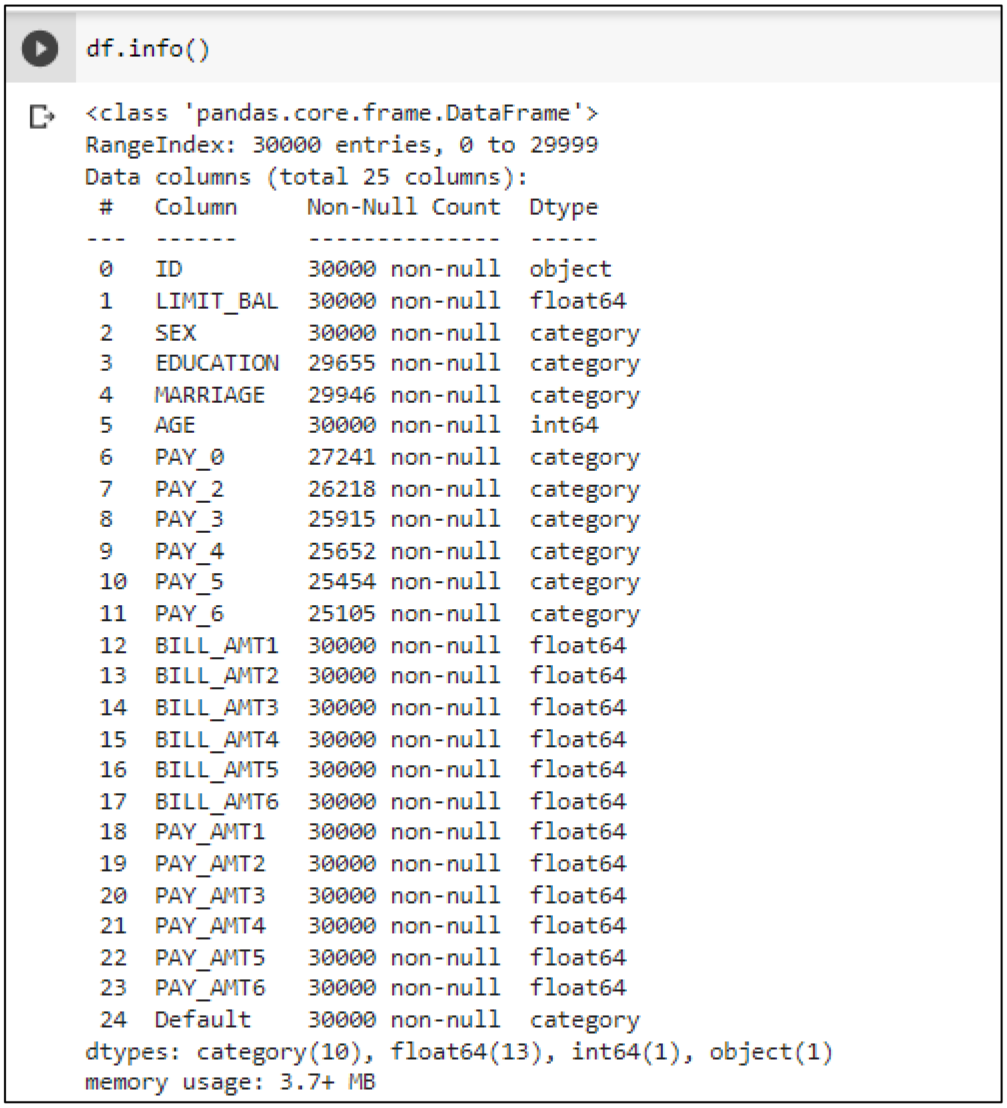


Analysis of default.payment.next.month

```
[ ] df["default.payment.next.month"].value_counts()

    0    23364
    1     6636
    Name: default.payment.next.month, dtype: int64

[ ] ## Change name of variable
    df.rename(columns={"default.payment.next.month":"Default"},inplace=True)

[ ] df.Default=df.Default.astype("category")
```

Final review of variable types, after modifications is the following:

Figure 11



```
df.info()
```

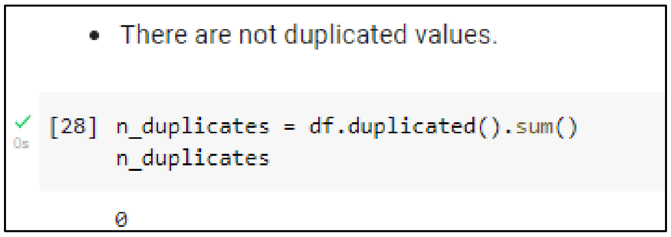
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 25 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   ID         30000 non-null  object
 1   LIMIT_BAL  30000 non-null  float64
 2   SEX        30000 non-null  category
 3   EDUCATION  29655 non-null  category
 4   MARRIAGE   29946 non-null  category
 5   AGE        30000 non-null  int64
 6   PAY_0      27241 non-null  category
 7   PAY_2      26218 non-null  category
 8   PAY_3      25915 non-null  category
 9   PAY_4      25652 non-null  category
 10  PAY_5      25454 non-null  category
 11  PAY_6      25105 non-null  category
 12  BILL_AMT1  30000 non-null  float64
 13  BILL_AMT2  30000 non-null  float64
 14  BILL_AMT3  30000 non-null  float64
 15  BILL_AMT4  30000 non-null  float64
 16  BILL_AMT5  30000 non-null  float64
 17  BILL_AMT6  30000 non-null  float64
 18  PAY_AMT1   30000 non-null  float64
 19  PAY_AMT2   30000 non-null  float64
 20  PAY_AMT3   30000 non-null  float64
 21  PAY_AMT4   30000 non-null  float64
 22  PAY_AMT5   30000 non-null  float64
 23  PAY_AMT6   30000 non-null  float64
 24  Default    30000 non-null  category
dtypes: category(10), float64(13), int64(1), object(1)
memory usage: 3.7+ MB
```

12.2. Data Cleansing

12.2.1. Review of duplicates

There were not duplicated values in the data set. Below the code to validate the duplicates data is shown:

Figure 12

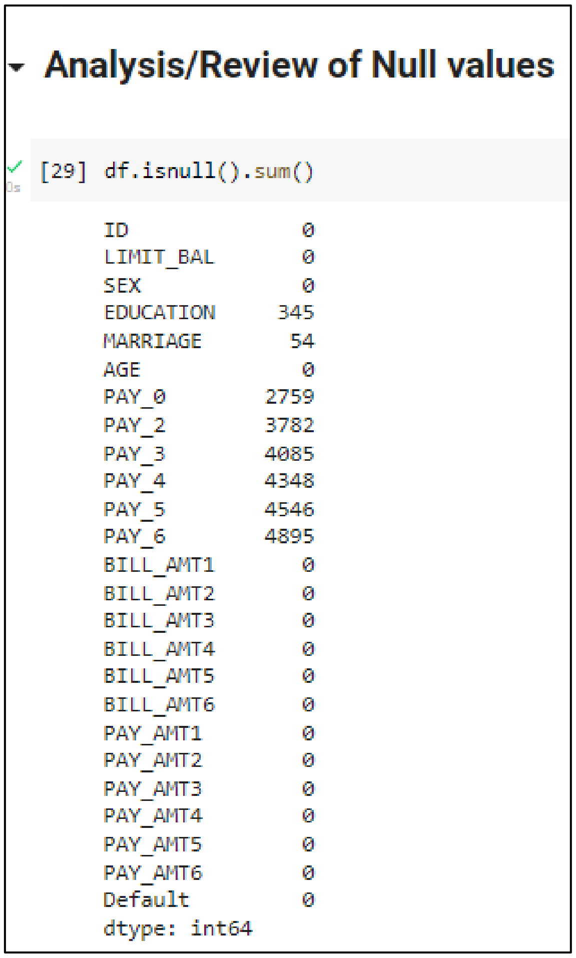


- There are not duplicated values.

```
[28] n_duplicates = df.duplicated().sum()
     n_duplicates

     0
```

12.2.2. Null Values

The Null values can be addressed by imputation or elimination. The null values were reviewed and treated, as it is shown below. There were in total 8 variables with null values; 8 of the variables with null values contained demographic information. MARRIAGE and EDUCATION variables null values were imputed with the MODE; and PAYMENT STATUS null values were removed.



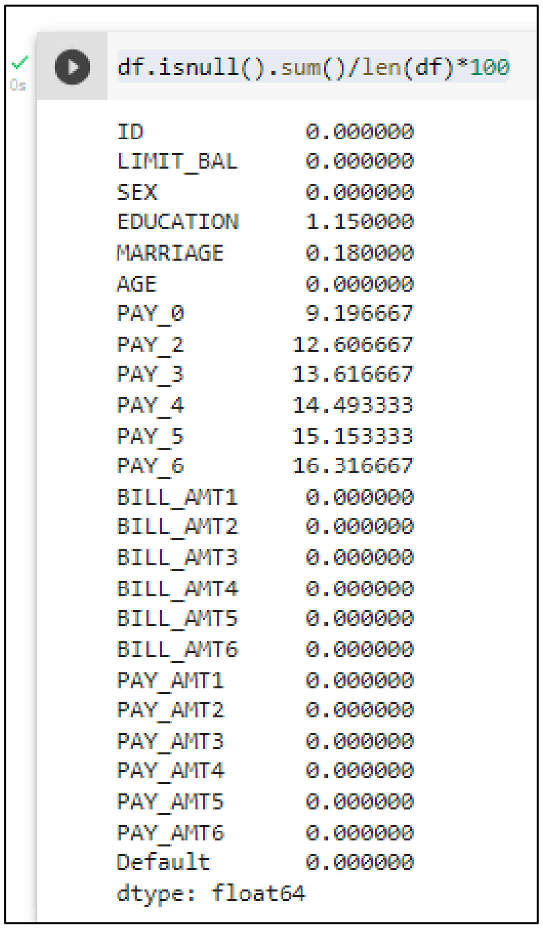
Analysis/Review of Null values

```
[29] df.isnull().sum()

     ID              0
     LIMIT_BAL       0
     SEX             0
     EDUCATION     345
     MARRIAGE       54
     AGE             0
     PAY_0        2759
     PAY_2        3782
     PAY_3        4085
     PAY_4        4348
     PAY_5        4546
     PAY_6        4895
     BILL_AMT1       0
     BILL_AMT2       0
     BILL_AMT3       0
     BILL_AMT4       0
     BILL_AMT5       0
     BILL_AMT6       0
     PAY_AMT1        0
     PAY_AMT2        0
     PAY_AMT3        0
     PAY_AMT4        0
     PAY_AMT5        0
     PAY_AMT6        0
     Default         0
     dtype: int64
```

Figure 13

The percentage of missing values was calculated by variable, giving the following results:

Figure 14

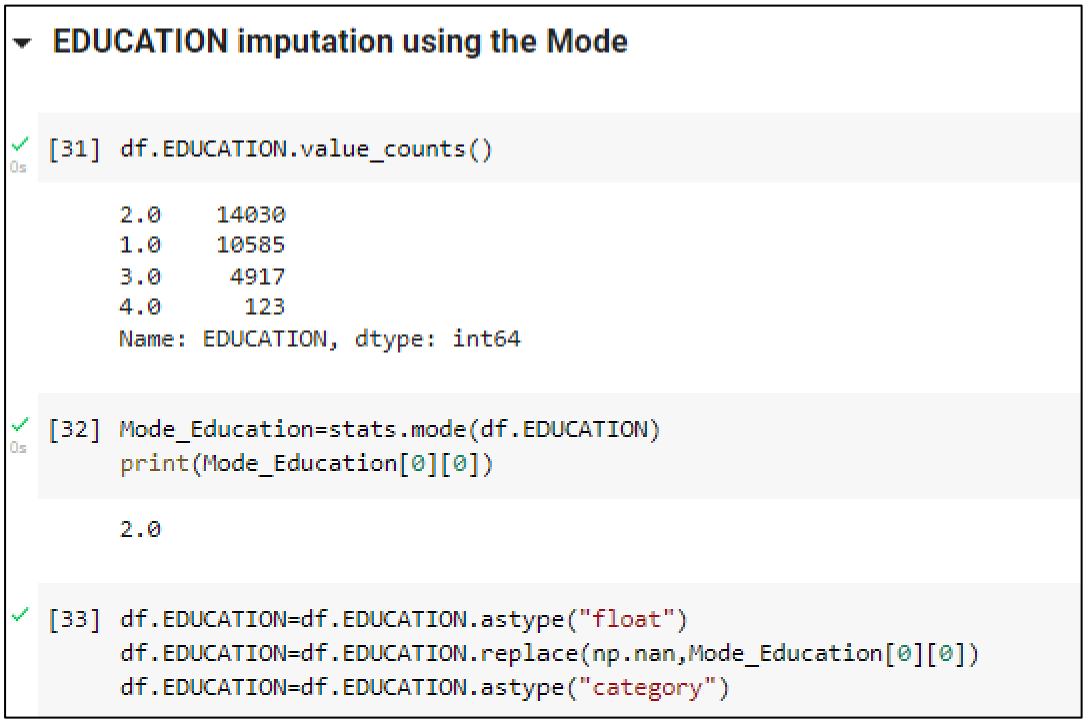


```
df.isnull().sum()/len(df)*100
```

```
ID            0.000000
LIMIT_BAL     0.000000
SEX           0.000000
EDUCATION     1.150000
MARRIAGE      0.180000
AGE           0.000000
PAY_0         9.196667
PAY_2        12.606667
PAY_3        13.616667
PAY_4        14.493333
PAY_5        15.153333
PAY_6        16.316667
BILL_AMT1     0.000000
BILL_AMT2     0.000000
BILL_AMT3     0.000000
BILL_AMT4     0.000000
BILL_AMT5     0.000000
BILL_AMT6     0.000000
PAY_AMT1      0.000000
PAY_AMT2      0.000000
PAY_AMT3      0.000000
PAY_AMT4      0.000000
PAY_AMT5      0.000000
PAY_AMT6      0.000000
Default       0.000000
dtype: float64
```

In this case, for the variables EDUCATION and MARRIAGE the imputation was completed using the MODE

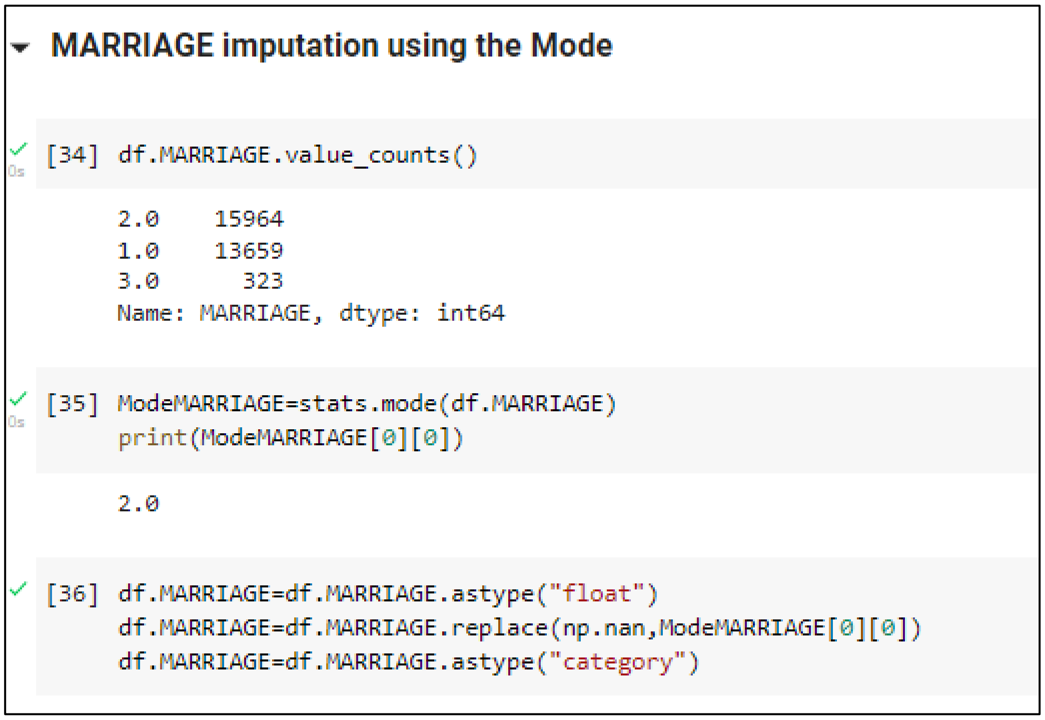
Figure 15



EDUCATION imputation using the Mode

```
[31] df.EDUCATION.value_counts()

     2.0    14030
     1.0    10585
     3.0     4917
     4.0      123
     Name: EDUCATION, dtype: int64

[32] Mode_Education=stats.mode(df.EDUCATION)
     print(Mode_Education[0][0])

     2.0

[33] df.EDUCATION=df.EDUCATION.astype("float")
     df.EDUCATION=df.EDUCATION.replace(np.nan,Mode_Education[0][0])
     df.EDUCATION=df.EDUCATION.astype("category")
```

Figure 16

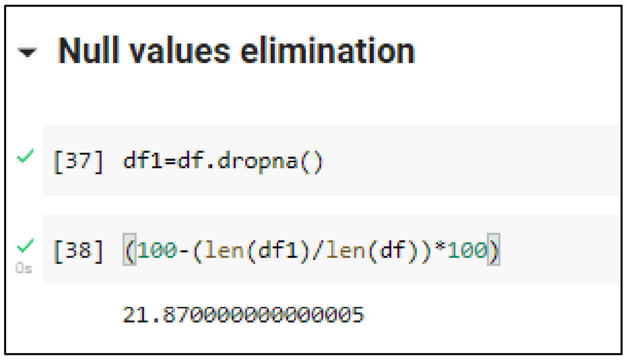


MARRIAGE imputation using the Mode

```
[34] df.MARRIAGE.value_counts()

     2.0    15964
     1.0    13659
     3.0      323
     Name: MARRIAGE, dtype: int64

[35] ModeMARRIAGE=stats.mode(df.MARRIAGE)
     print(ModeMARRIAGE[0][0])

     2.0

[36] df.MARRIAGE=df.MARRIAGE.astype("float")
     df.MARRIAGE=df.MARRIAGE.replace(np.nan,ModeMARRIAGE[0][0])
     df.MARRIAGE=df.MARRIAGE.astype("category")
```

The NULL values, corresponding to the 21% of the data, in other variables, were eliminated as it is shown:

Figure 17



Null values elimination

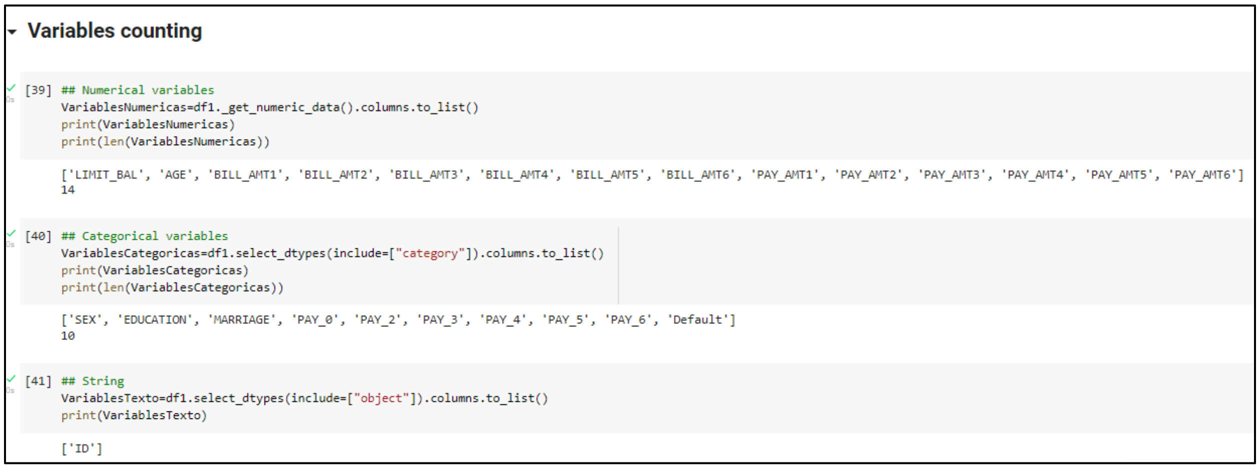
```
[37] df1=df.dropna()
```

```
[38] (100-(len(df1)/len(df))*100)

     21.870000000000005
```

13. Summary

As part of the data exploratory analysis, it was done the Variables counting:

Figure 18



Variables counting

```
[39] ## Numerical variables
     VariablesNumericas=df1._get_numeric_data().columns.to_list()
     print(VariablesNumericas)
     print(len(VariablesNumericas))

     ['LIMIT_BAL', 'AGE', 'BILL_AMT1', 'BILL_AMT2', 'BILL_AMT3', 'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6', 'PAY_AMT1', 'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6']
     14
```

```
[40] ## Categorical variables
     VariablesCategoricas=df1.select_dtypes(include=["category"]).columns.to_list()
     print(VariablesCategoricas)
     print(len(VariablesCategoricas))

     ['SEX', 'EDUCATION', 'MARRIAGE', 'PAY_0', 'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6', 'Default']
     10
```

```
[41] ## String
     VariablesTexto=df1.select_dtypes(include=["object"]).columns.to_list()
     print(VariablesTexto)

     ['ID']
```

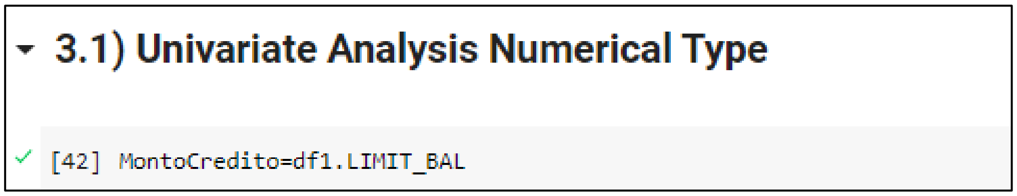
Data Preparation and Feature Engineering

14. Data Preparation Needs

Univariate analysis Numerical Type: explores variables (attributes) one by one. Variables could be either categorical or numerical. There are different statistical and visualization techniques of investigation for each type of variable.

The variable LIMIT_BAL was changed to Credit Amount

Figure 19

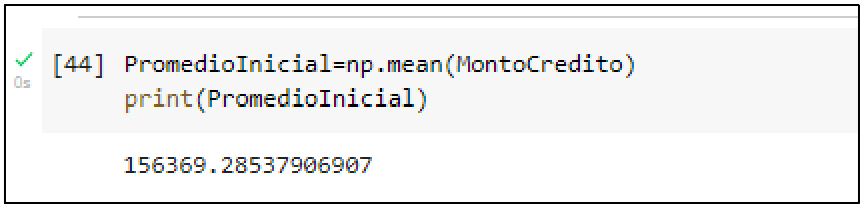


3.1) Univariate Analysis Numerical Type

```
[42] MontoCredito=df1.LIMIT_BAL
```

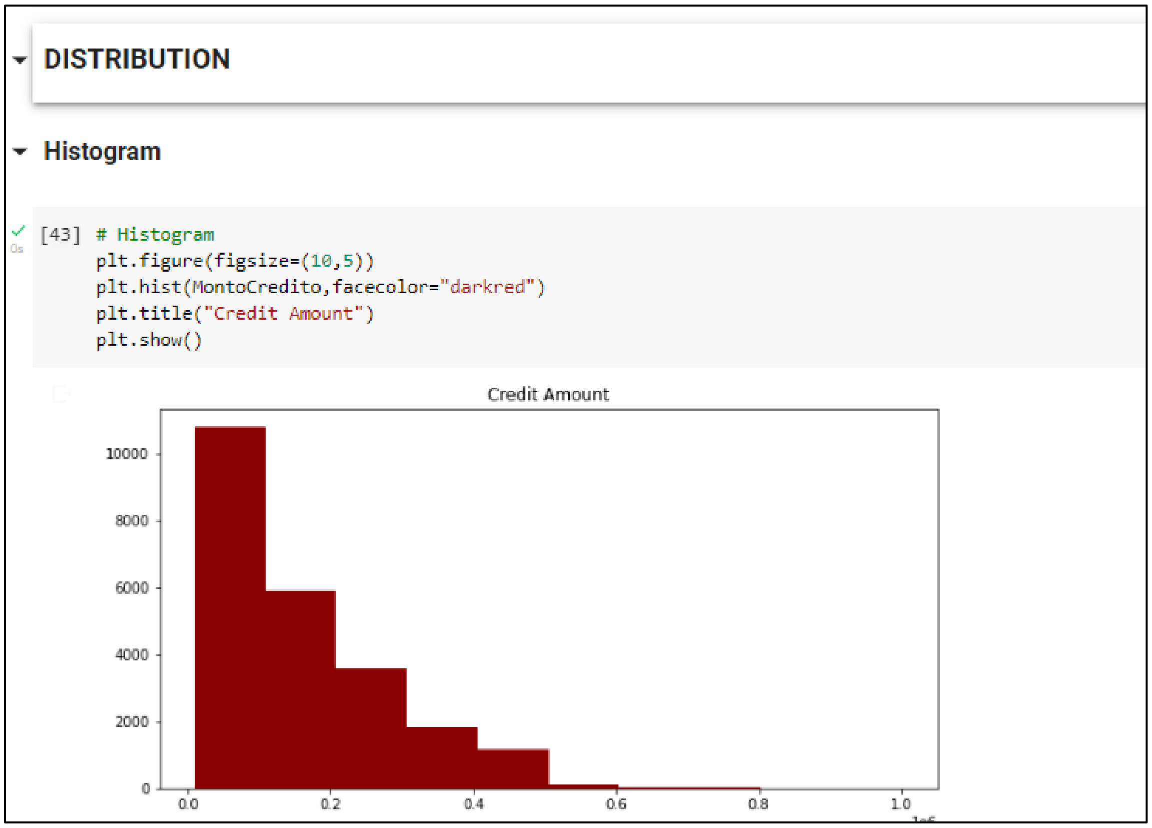
The distribution was reviewed by histogram and the Mean was calculated:

Figure 20



```
[44] PromedioInicial=np.mean(MontoCredito)
     print(PromedioInicial)

     156369.28537906907
```

Figure 21



DISTRIBUTION

Histogram

```
[43] # Histogram
     plt.figure(figsize=(10,5))
     plt.hist(MontoCredito,facecolor="darkred")
     plt.title("Credit Amount")
     plt.show()
```

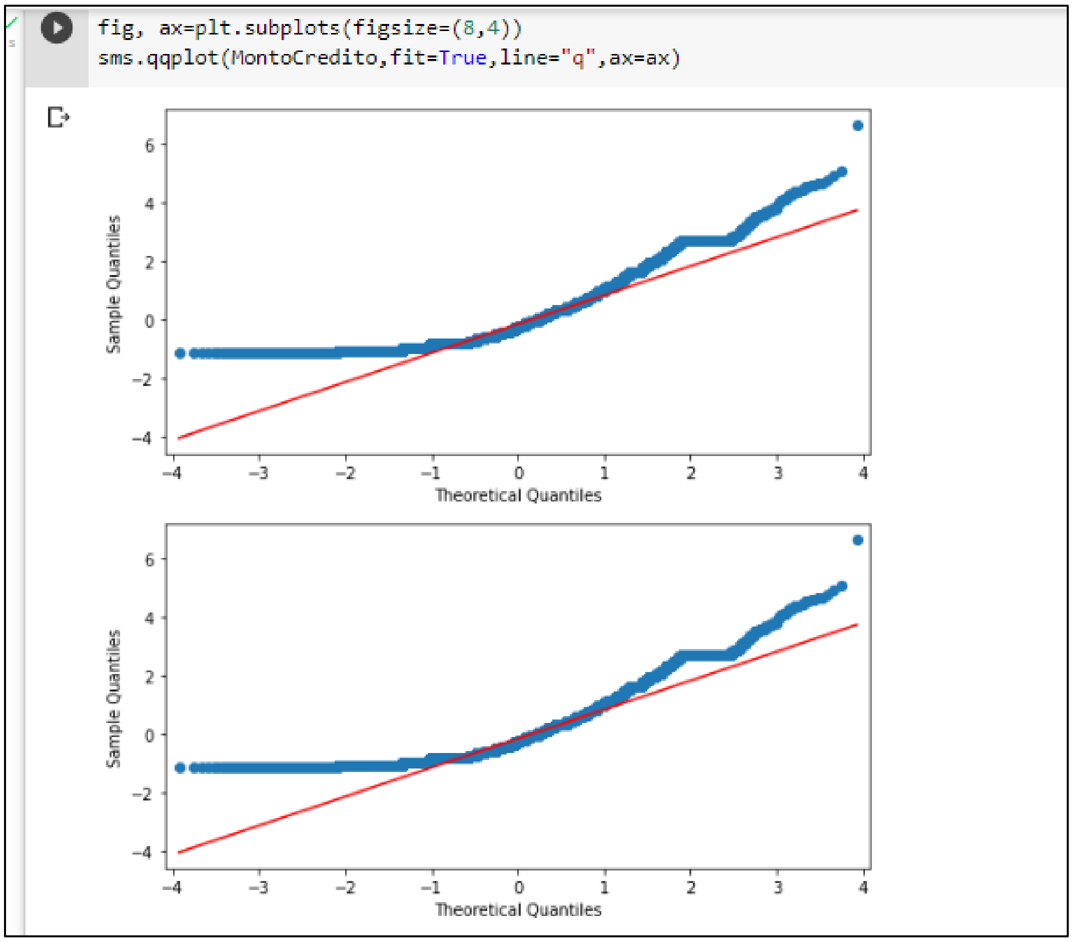
Q-Q Plot The theoretical quantile plots (Q-Q Plots)

The variable has a skewed distribution, and the Normality test was run with the Plot Q-Q. These plots compare the quantiles of the observed distribution to the theoretical quantiles of a normal distribution with the same mean and standard deviation as the data. The closer the data is to a normal, the more aligned the points are around the line.

The statsmodel library was imported

statsmodels is a Python module that provides classes and functions for the estimation of many different statistical models, as well as for conducting statistical tests, and statistical data exploration. An extensive list of result statistics is available for each estimator.

Figure 22



```
fig, ax=plt.subplots(figsize=(8,4))
sms.qqplot(MontoCredito,fit=True,line="q",ax=ax)
```

14.1. Normality Hypothesis Test

A hypothesis test is a rule that specifies whether a statement about a population can be accepted or rejected depending on the evidence provided by a sample of data.

An important decision point when working with a sample of data is whether to use parametric or nonparametric statistical methods. Parametric statistical methods assume that the data have a known and specific distribution, often a Gaussian distribution. If a data sample is not Gaussian, then the assumptions of parametric statistical tests are violated, and nonparametric statistical methods must be used.

Interpretation of a test

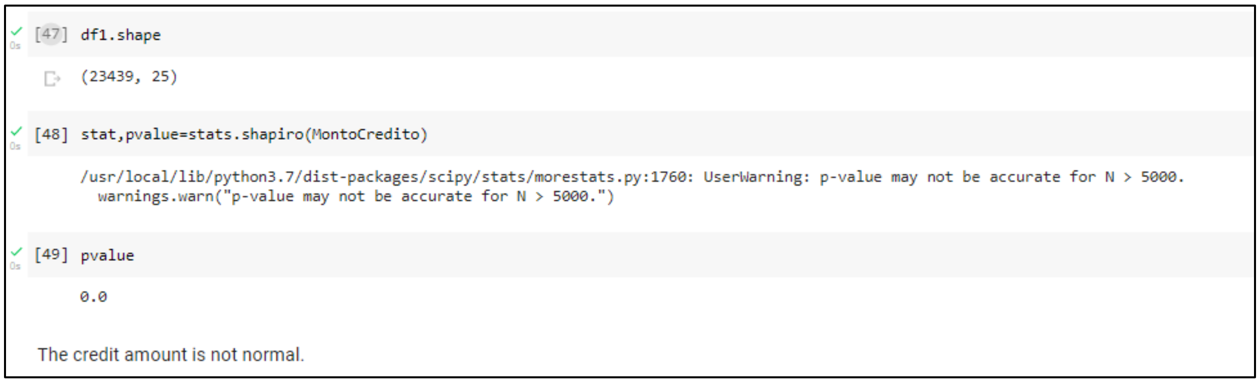
H0: The credit amount comes from a normal distribution
Ha: The credit amount does not come from a normal distribution
p value;

p <= alpha : reject H0, it is not normal.
p> alpha : Do not reject H0, It does not mean that it is certain normality, It means that it is very likely that it is Normal

Shapiro-Wilk test Shapiro-Wilk evaluates a sample of data and quantifies the probability that the data will be drawn from a Gaussian distribution, named for Samuel Shapiro and Martin Wilk.

In practice, the Shapiro-Wilk test is believed to be a reliable test of normality, although it is suggested that the test may be suitable for smaller data samples, for example thousands of observations or fewer.

Figure 23



```
[47] df1.shape

(23439, 25)

[48] stat,pvalue=stats.shapiro(MontoCredito)

/usr/local/lib/python3.7/dist-packages/scipy/stats/morestats.py:1760: UserWarning: p-value may not be accurate for N > 5000.
  warnings.warn("p-value may not be accurate for N > 5000.")

[49] pvalue

0.0
```

The credit amount is not normal.

Normality tests

It could be stablished that the CREDIT AMOUNT VARIABLE was not normal. Normality can be affected by atypical data

Consequences of the lack of normality

The fact of not being able to assume normality mainly influences the parametric hypothesis tests (t-test, ANOVA,...) and the regression models. The main consequences of the lack of normality are:

Least-squares estimators are not efficient (of least variance).

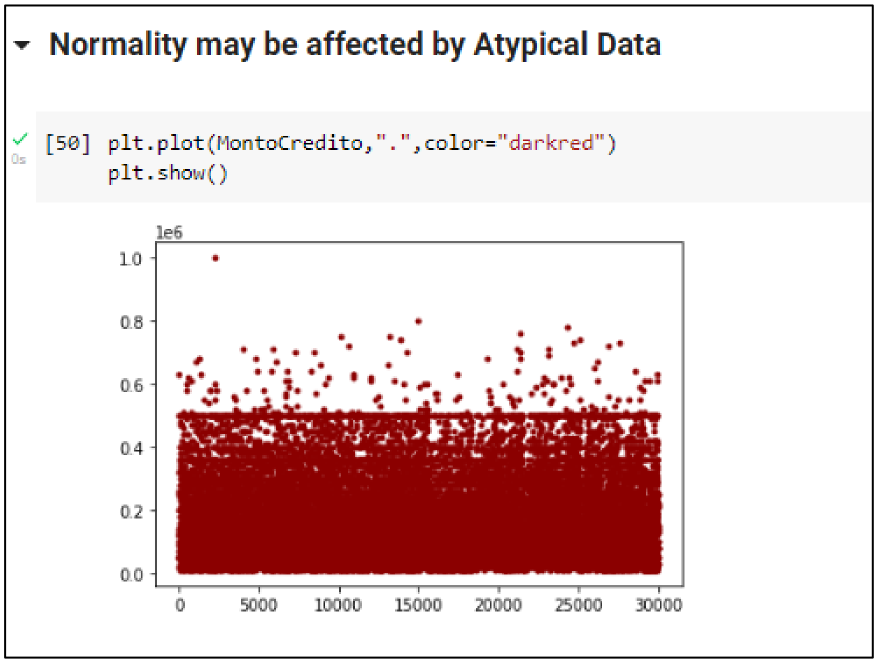
The confidence intervals of the model parameters and the significance tests are only approximate and not exact.

The exposed statistical tests require that the population from which the sample comes have a normal distribution, not the sample itself. If the sample is normally distributed, it can be accepted that the original population does so. In the event that the sample is not normally distributed but it is certain that the original population is, then it may be justified to accept the results obtained by the parametric tests.

Central Limit Theorem

The central limit theorem is a fundamental theorem of probability and statistics. The theorem describes the distribution of the mean of a random sample from a population with finite variance. When the sample size is large enough, the distribution of means approximately follows a normal distribution.

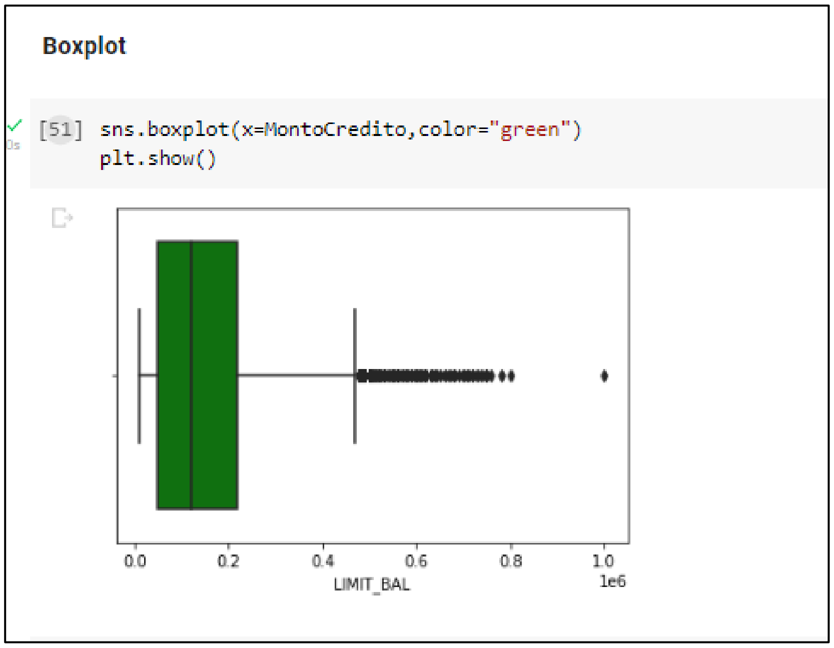
Figure 24



Normality may be affected by Atypical Data

```
[50] plt.plot(MontoCredito,".",color="darkred")
     plt.show()
```

Figure 25

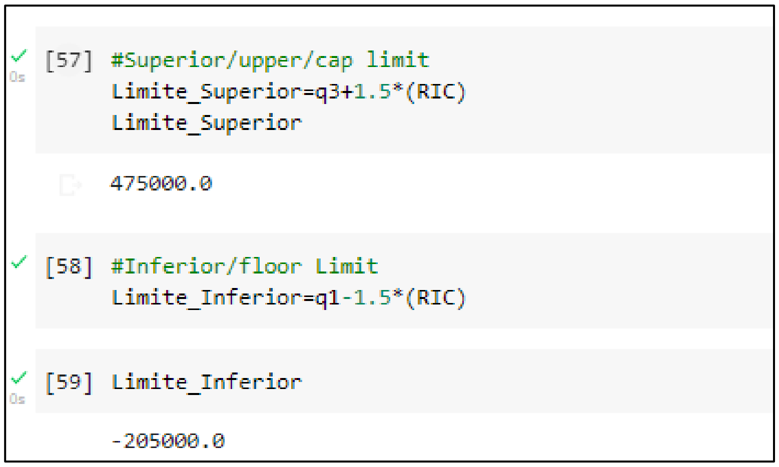


Boxplot

```
[51] sns.boxplot(x=MontoCredito,color="green")
     plt.show()
```

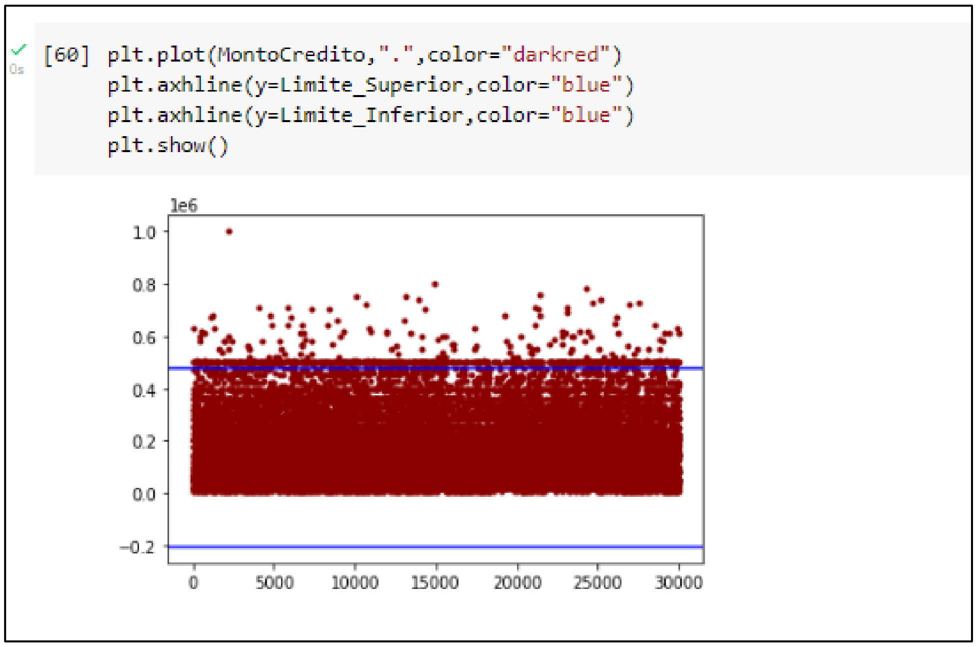
14.2. Cap & Floor Limit

Figure 26



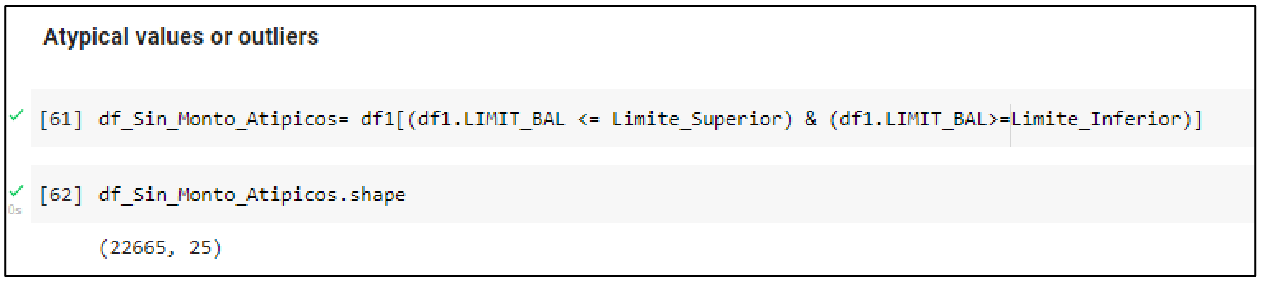
```
[57] #Superior/upper/cap limit
     Limite_Superior=q3+1.5*(RIC)
     Limite_Superior

     475000.0

[58] #Inferior/floor Limit
     Limite_Inferior=q1-1.5*(RIC)

[59] Limite_Inferior

     -205000.0
```

Figure 27



```
[60] plt.plot(MontoCredito,".",color="darkred")
     plt.axhline(y=Limite_Superior,color="blue")
     plt.axhline(y=Limite_Inferior,color="blue")
     plt.show()
```

Figure 28



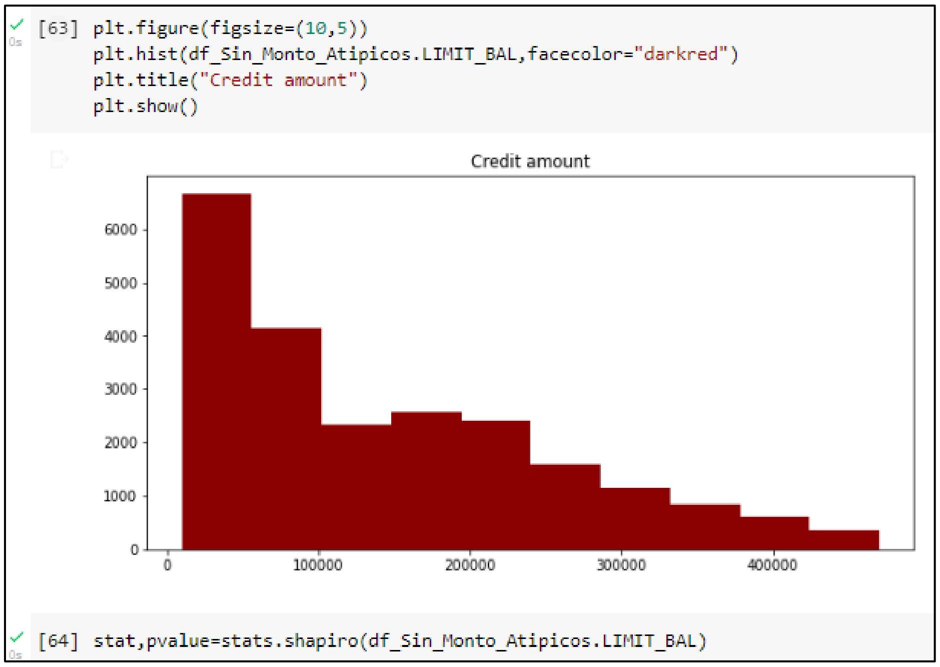
Atypical values or outliers

```
[61] df_Sin_Monto_Atipicos= df1[(df1.LIMIT_BAL <= Limite_Superior) & (df1.LIMIT_BAL>=Limite_Inferior)]

[62] df_Sin_Monto_Atipicos.shape

     (22665, 25)
```

14.3. Normality test for atypical values

Removing Atypical Values there is not normality:

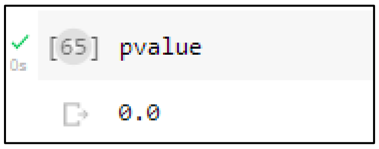
Figure 29



```
[63] plt.figure(figsize=(10,5))
     plt.hist(df_Sin_Monto_Atipicos.LIMIT_BAL,facecolor="darkred")
     plt.title("Credit amount")
     plt.show()
```

```
[64] stat,pvalue=stats.shapiro(df_Sin_Monto_Atipicos.LIMIT_BAL)
```

Figure 30

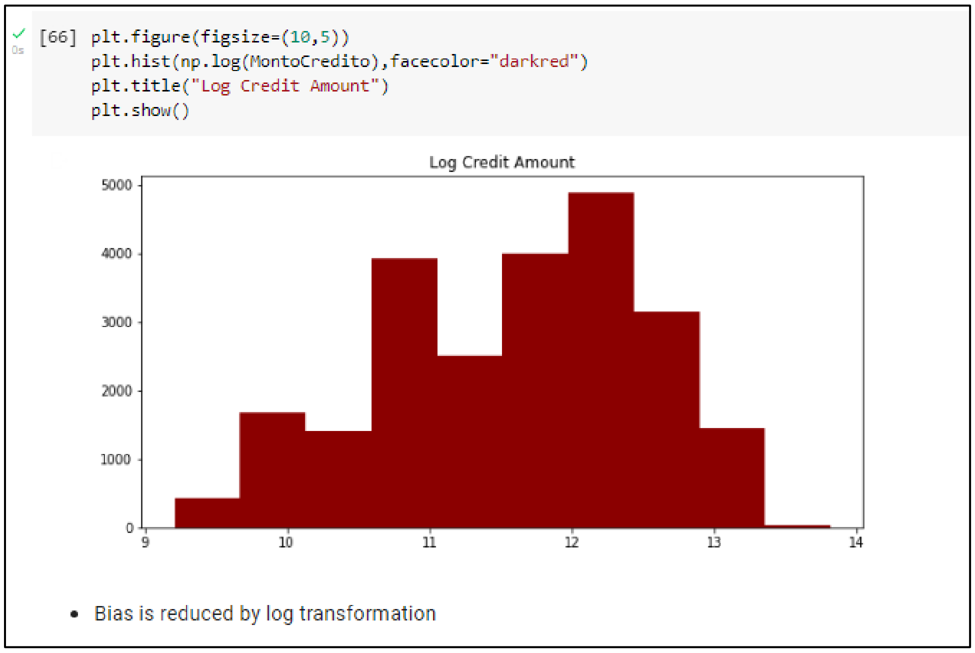


```
[65] pvalue

0.0
```

14.4. Transformation

Transformation log was applied to reduce BIAS and find Normality in the variable LIMIT_BAL (CREDIT AMOUNT):

Figure 31



```
[66] plt.figure(figsize=(10,5))
     plt.hist(np.log(MontoCredito),facecolor="darkred")
     plt.title("Log Credit Amount")
     plt.show()
```

- Bias is reduced by log transformation

Bivariate Analysis

Bivariate Analysis is a statistical technique applied to a pair of variables of data to determine the empirical relationship between them.

Bivariate analysis is a kind of statistical analysis when two variables are observed against each other. One of the variables will be dependent and the other is independent. The variables are denoted by X and Y. The changes are analyzed between the two variables to understand to what extent the change has occurred.

14.5. Numeric vs Numeric

Pearson correlation Pearson's correlation measures the linear dependence between two variables X and Y.

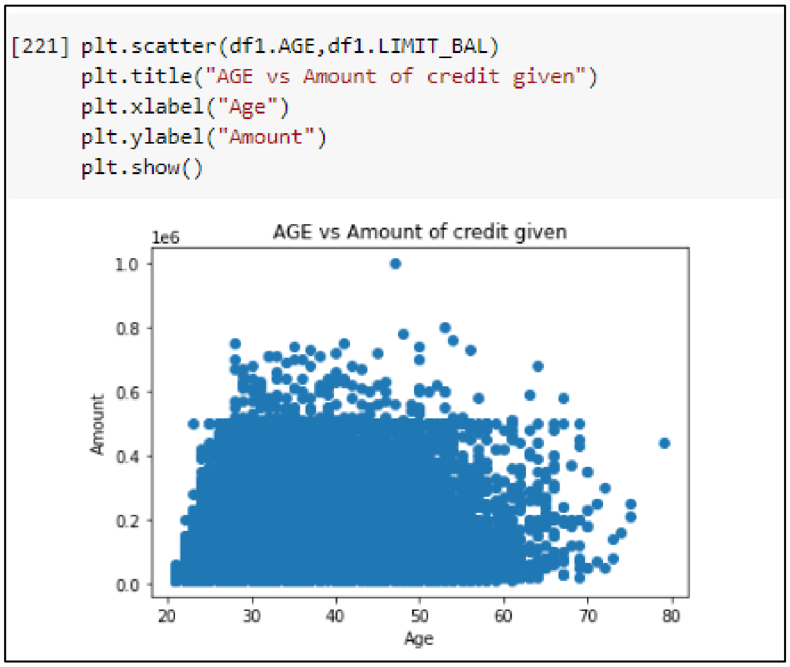
The resulting coefficient is a value between -1 and 1 inclusive, where:

1: total positive linear correlation.

0: No linear correlation, the two variables probably do not affect each other.

-1: total negative linear correlation

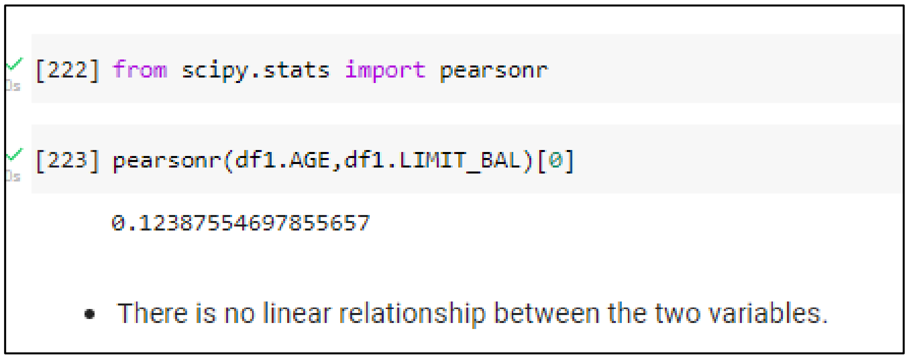
Figure 32



```
[221] plt.scatter(df1.AGE,df1.LIMIT_BAL)
      plt.title("AGE vs Amount of credit given")
      plt.xlabel("Age")
      plt.ylabel("Amount")
      plt.show()
```

Import spicy.stats

Figure 33



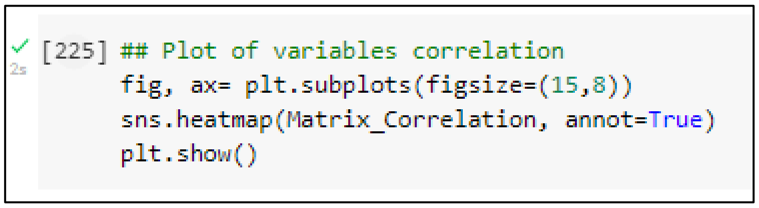
```
[222] from scipy.stats import pearsonr
```

```
[223] pearsonr(df1.AGE,df1.LIMIT_BAL)[0]

      0.12387554697855657
```

- There is no linear relationship between the two variables.

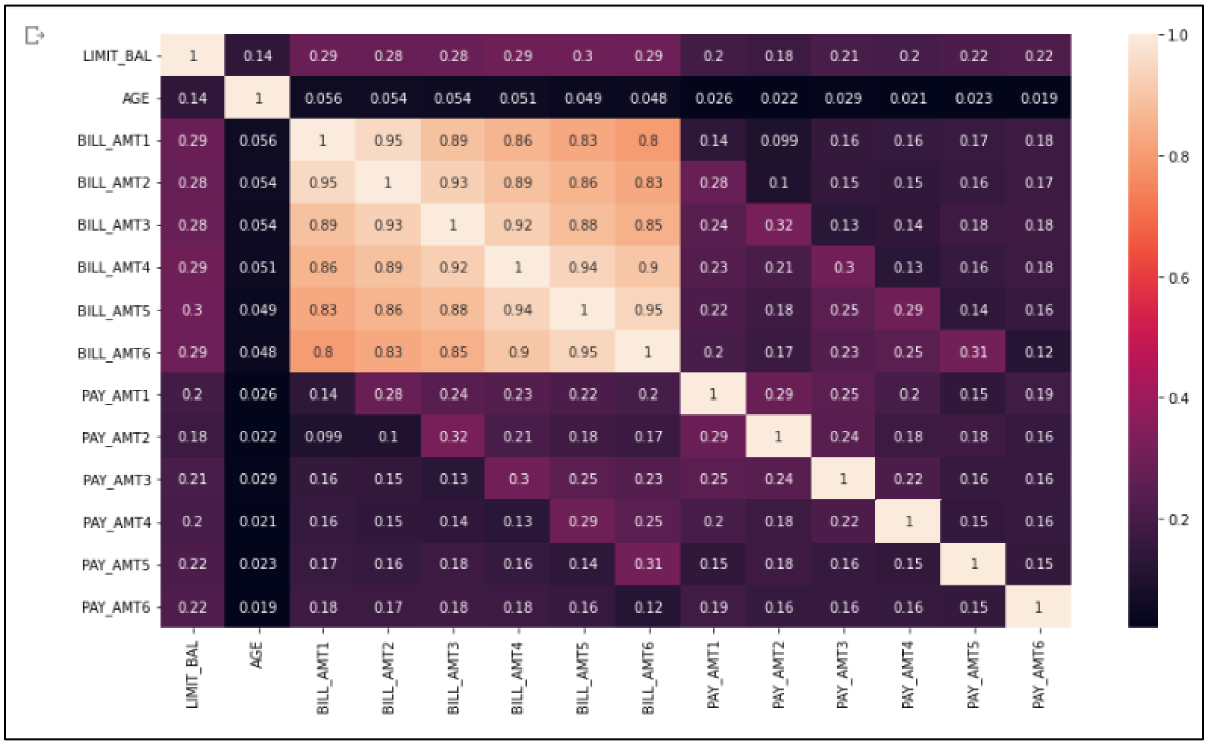
Correlation MATRIX for the two variables:

Figure 34



```
[225] ## Plot of variables correlation
      fig, ax= plt.subplots(figsize=(15,8))
      sns.heatmap(Matrix_Correlation, annot=True)
      plt.show()
```

Figure 35

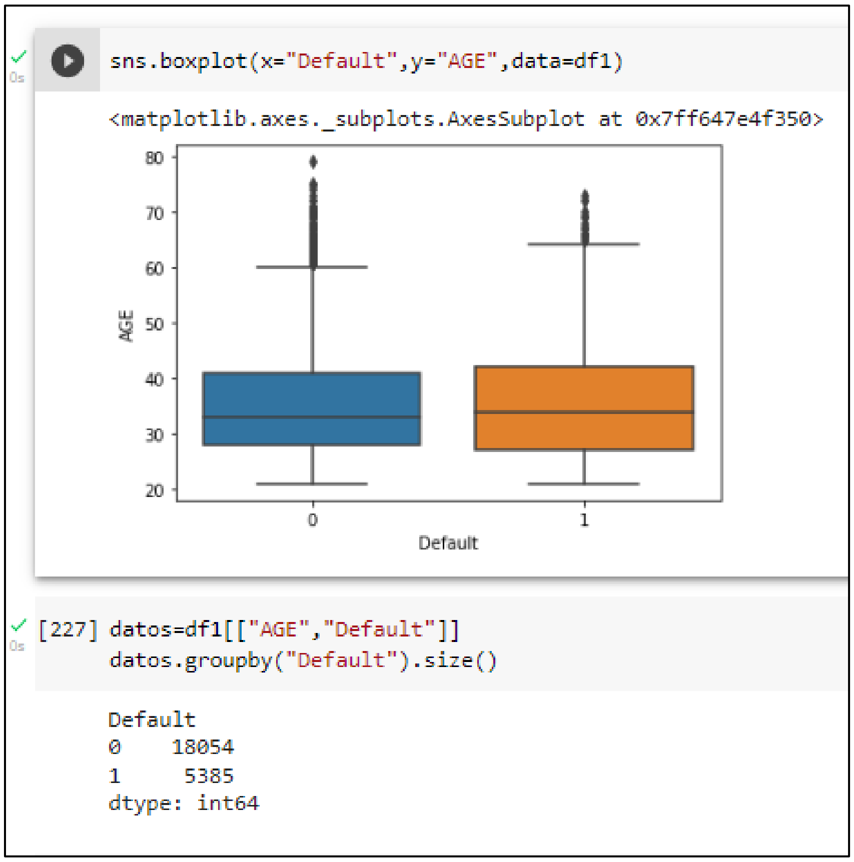


	LIMIT_BAL	AGE	BILL_AMT1	BILL_AMT2	BILL_AMT3	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6
LIMIT_BAL	1	0.14	0.29	0.28	0.28	0.29	0.3	0.29	0.2	0.18	0.21	0.2	0.22	0.22
AGE	0.14	1	0.056	0.054	0.054	0.051	0.049	0.048	0.026	0.022	0.029	0.021	0.023	0.019
BILL_AMT1	0.29	0.056	1	0.95	0.89	0.86	0.83	0.8	0.14	0.099	0.16	0.16	0.17	0.18
BILL_AMT2	0.28	0.054	0.95	1	0.93	0.89	0.86	0.83	0.28	0.1	0.15	0.15	0.16	0.17
BILL_AMT3	0.28	0.054	0.89	0.93	1	0.92	0.88	0.85	0.24	0.32	0.13	0.14	0.18	0.18
BILL_AMT4	0.29	0.051	0.86	0.89	0.92	1	0.94	0.9	0.23	0.21	0.3	0.13	0.16	0.18
BILL_AMT5	0.3	0.049	0.83	0.86	0.88	0.94	1	0.95	0.22	0.18	0.25	0.29	0.14	0.16
BILL_AMT6	0.29	0.048	0.8	0.83	0.85	0.9	0.95	1	0.2	0.17	0.23	0.25	0.31	0.12
PAY_AMT1	0.2	0.026	0.14	0.28	0.24	0.23	0.22	0.2	1	0.29	0.25	0.2	0.15	0.19
PAY_AMT2	0.18	0.022	0.099	0.1	0.32	0.21	0.18	0.17	0.29	1	0.24	0.18	0.18	0.16
PAY_AMT3	0.21	0.029	0.16	0.15	0.13	0.3	0.25	0.23	0.25	0.24	1	0.22	0.16	0.16
PAY_AMT4	0.2	0.021	0.16	0.15	0.14	0.13	0.29	0.25	0.2	0.18	0.22	1	0.15	0.16
PAY_AMT5	0.22	0.023	0.17	0.16	0.18	0.16	0.14	0.31	0.15	0.18	0.16	0.15	1	0.15
PAY_AMT6	0.22	0.019	0.18	0.17	0.18	0.18	0.16	0.12	0.19	0.16	0.16	0.16	0.15	1

14.6. Categorical vs Numerical

ANOVA Analysis of Variance (ANOVA) is a statistical method used to assess whether there are significant differences between the means of two or more groups. ANOVA returns two parameters.

Figure 36



```
sns.boxplot(x="Default",y="AGE",data=df1)
```

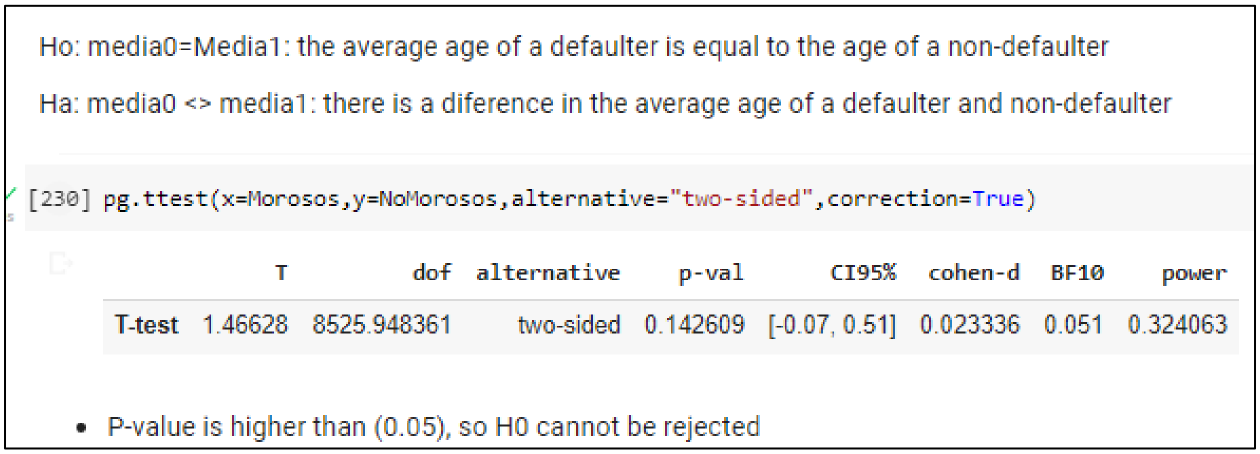
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff647e4f350>
```

```
[227] datos=df1[["AGE","Default"]]
      datos.groupby("Default").size()
```

```
Default
0    18054
1     5385
dtype: int64
```

Hypothesis:

Figure 37



Ho: media0=Media1: the average age of a defaulter is equal to the age of a non-defaulter

Ha: media0 <> media1: there is a diference in the average age of a defaulter and non-defaulter

```
[230] pg.ttest(x=Morosos,y=NoMorosos,alternative="two-sided",correction=True)
```

	T	dof	alternative	p-val	CI95%	cohen-d	BF10	power
T-test	1.46628	8525.948361	two-sided	0.142609	[-0.07, 0.51]	0.023336	0.051	0.324063

- P-value is higher than (0.05), so H0 cannot be rejected

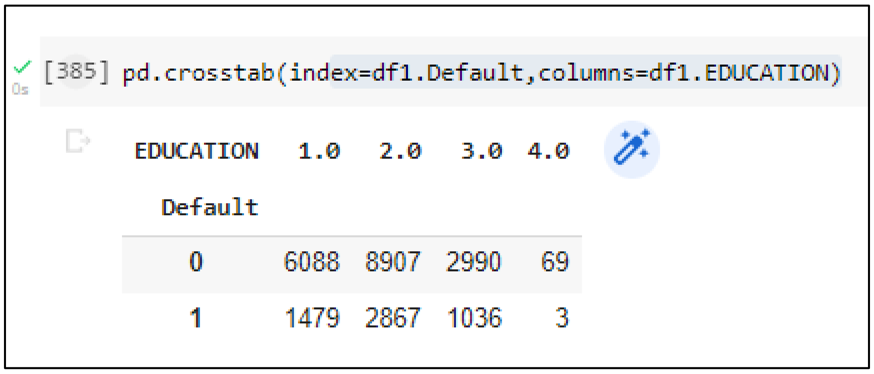
14.7. Categorical vs Categorical

Chi-square test of independence

- H0: The variables are Independent
- Ha: The variables are not Independent

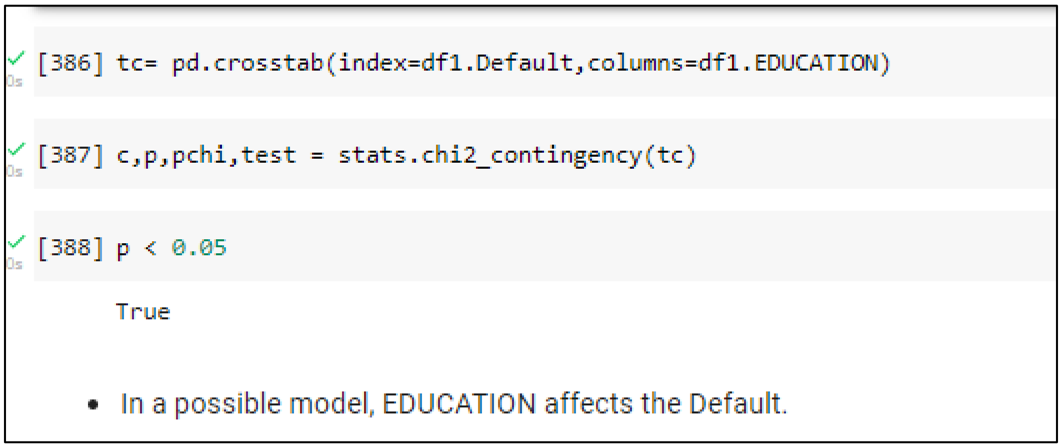
If p value is lower than0,05, the variables are statistically correlated and in a possible model, the variable should be included

Figure 38



```
[385] pd.crosstab(index=df1.Default,columns=df1.EDUCATION)

EDUCATION   1.0   2.0   3.0  4.0
 Default
     0     6088  8907  2990   69
     1     1479  2867  1036    3
```

Figure 39



```
[386] tc= pd.crosstab(index=df1.Default,columns=df1.EDUCATION)

[387] c,p,pchi,test = stats.chi2_contingency(tc)

[388] p < 0.05

      True
```

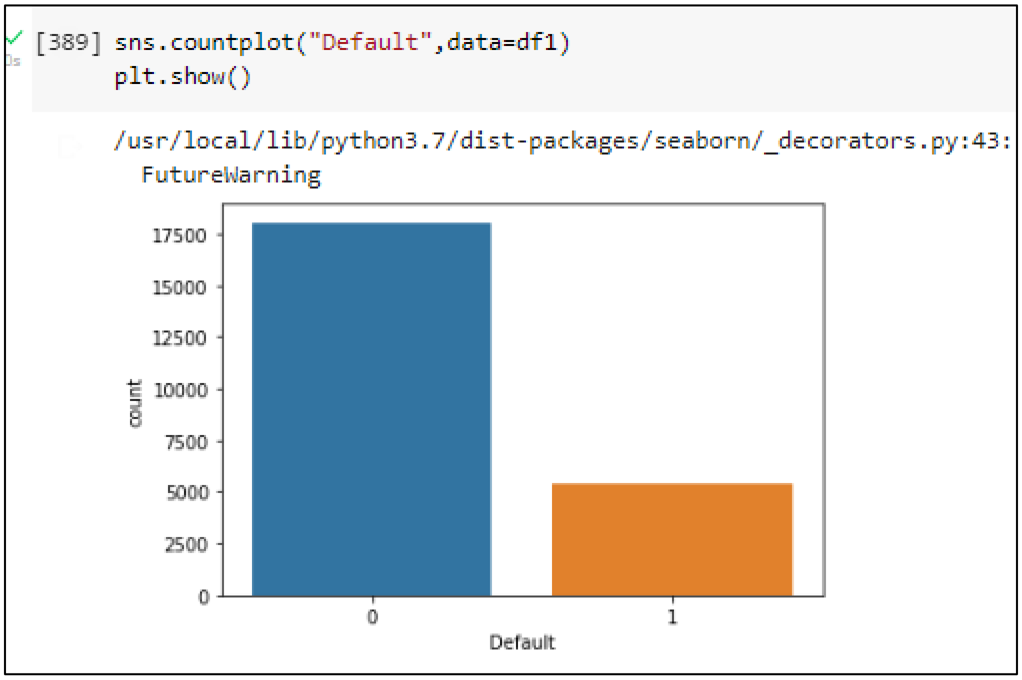
- In a possible model, EDUCATION affects the Default.

14.8. SMOTE

Before start modeling, one of the main variables presented imbalance, as it is shown below, and the SMOTE, technique was used for the imbalance classification of the DEFAULT variable.

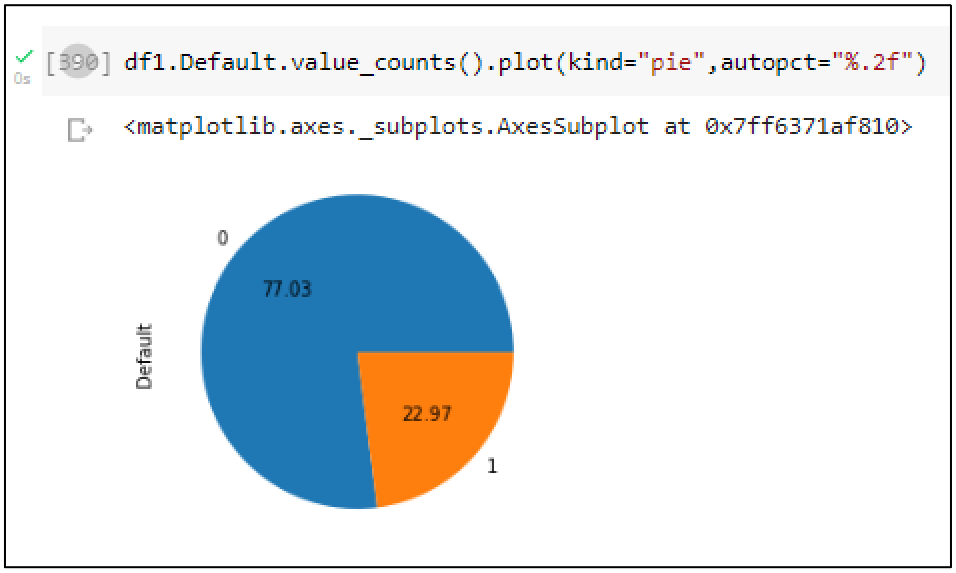
SMOTE (synthetic minority oversampling technique) is one of the most commonly used oversampling methods to solve the imbalance problem. It aims to balance class distribution by randomly increasing minority class examples by replicating them. SMOTE synthesizes new minority instances between existing minority instances

Figure 40



```
[389] sns.countplot("Default",data=df1)
      plt.show()

/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
  FutureWarning
```

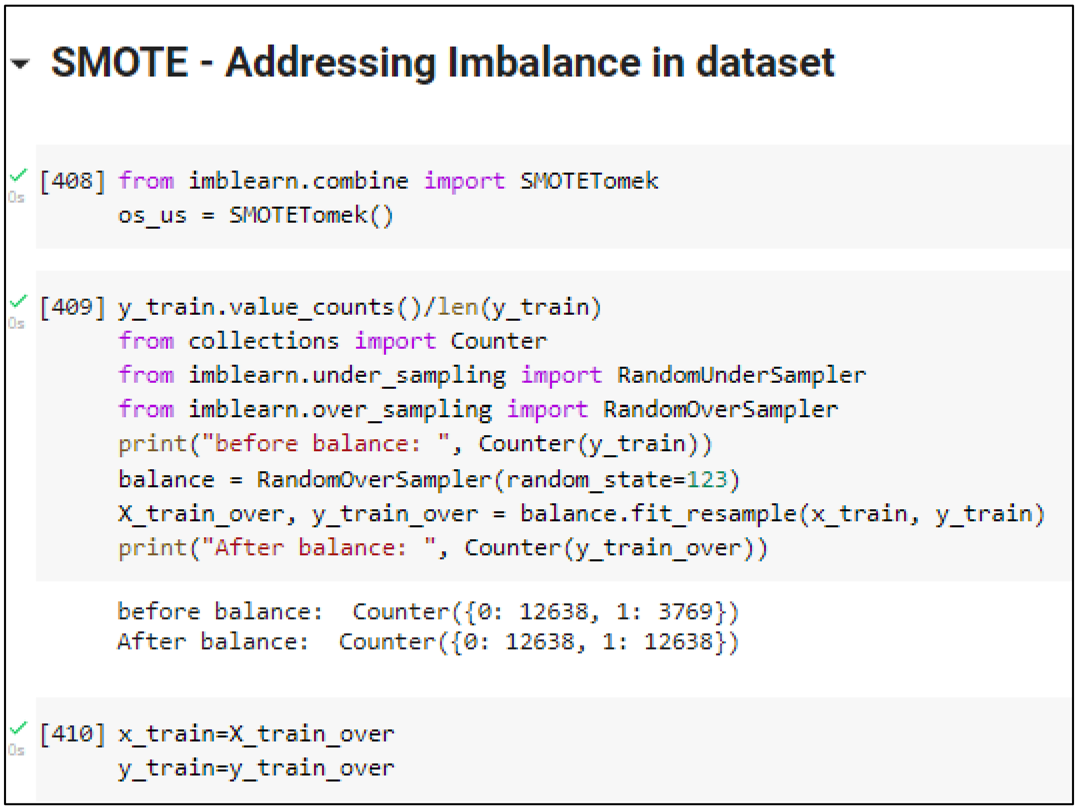
Figure 41



```
[390] df1.Default.value_counts().plot(kind="pie",autopct="%.2f")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff6371af810>
```

Figure 42



SMOTE - Addressing Imbalance in dataset

```
[408] from imblearn.combine import SMOTETomek
      os_us = SMOTETomek()
```

```
[409] y_train.value_counts()/len(y_train)
      from collections import Counter
      from imblearn.under_sampling import RandomUnderSampler
      from imblearn.over_sampling import RandomOverSampler
      print("before balance: ", Counter(y_train))
      balance = RandomOverSampler(random_state=123)
      X_train_over, y_train_over = balance.fit_resample(x_train, y_train)
      print("After balance: ", Counter(y_train_over))
```

```
before balance:  Counter({0: 12638, 1: 3769})
After balance:  Counter({0: 12638, 1: 12638})
```

```
[410] x_train=X_train_over
      y_train=y_train_over
```

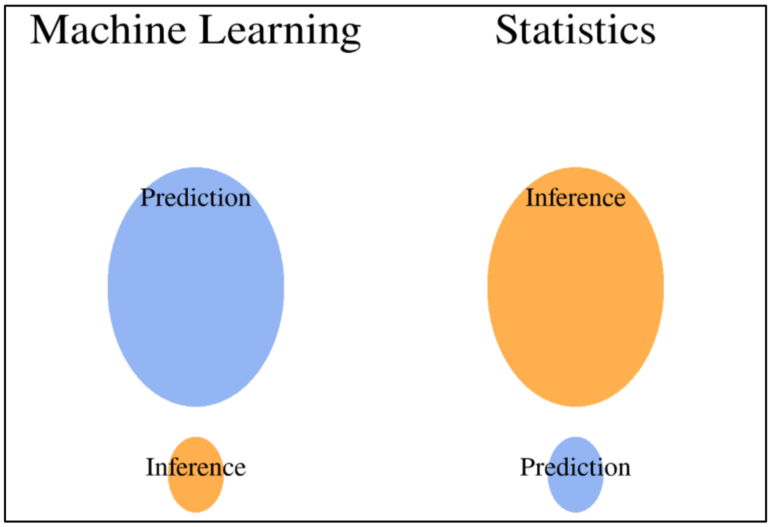
Model Exploration

The terms inference and prediction describe tasks in which we learn from the data in a supervised way to find a model that describes the relationship between the independent variables and the outcome. However, inference and prediction diverge when it comes to the use of the resulting model:

Inference: Use the model to learn about the process of generating data.

Prediction: use the model to predict the results of new data points

Figure 43



Workflow for inference and prediction

The basic workflows for inference and prediction are described in the following sections.

Inference

- Modeling: Reason about the data generation process and choose the stochastic model that best approximates the data generation process.
- Model Validation: Assess the validity of the stochastic model using residual analysis or goodness-of-fit tests.

- Inference: Use the stochastic model to understand the process of generating data.

Prediction

- Modeling: Consider several different models and different parameter settings.
- Model selection: identify the model with the highest predictive performance using validation / test sets; select the model with the highest performance on the test computer.
- Prediction: Apply the selected model on new data with the expectation that the selected model generalizes to unseen data as well.

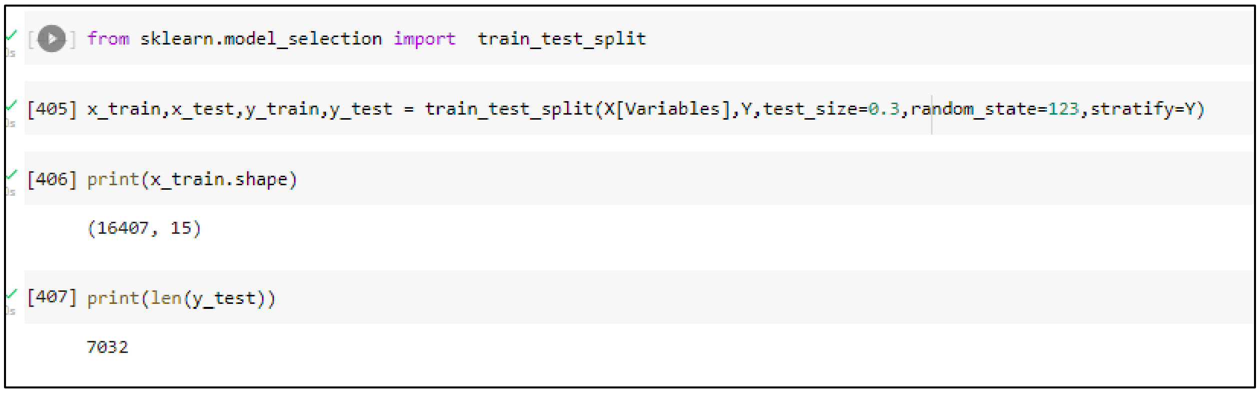
Train y Test

Assessing the predictive capacity of a model consists of checking its predictions to approximate the true values of the response variable. In order to be able to quantify it correctly, it is necessary to have a set of observations, for which the response variable is known, but that the model has not "seen", that is, that they have not participated in its adjustment. For this purpose, the available data is divided into a training set and a test set.

It is important to verify that the distribution of the response variable is similar in the training set and in the test set. To ensure that this is true, the scikit-learn function train_test_split() allows, in classification problems, to identify with the stratify argument the variable based on which to make the distribution.

This type of stratified distribution ensures that the training set and the test set are similar in terms of the response variable, however, it does not guarantee that the same thing happens with the predictors.

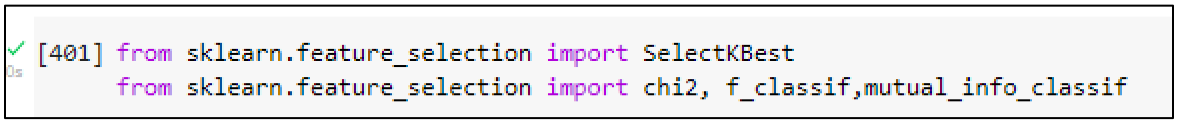
Figure 44



```
[ ] from sklearn.model_selection import  train_test_split

[405] x_train,x_test,y_train,y_test = train_test_split(X[Variables],Y,test_size=0.3,random_state=123,stratify=Y)

[406] print(x_train.shape)

      (16407, 15)

[407] print(len(y_test))

      7032
```

15. Variables selection

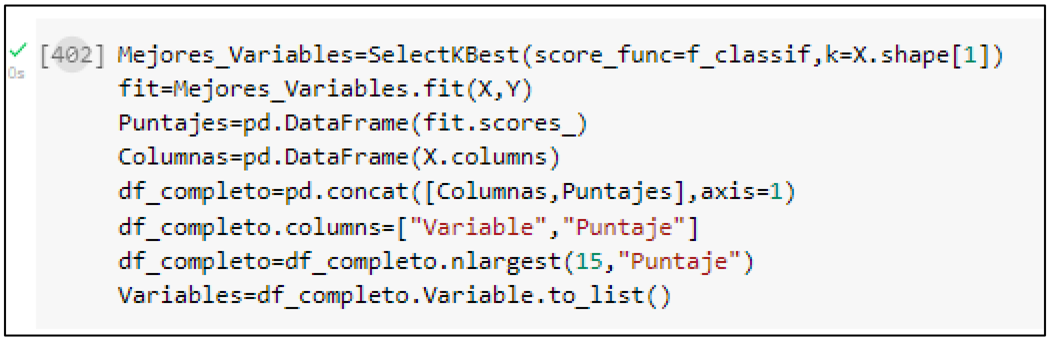
Figure 45



```
[401] from sklearn.feature_selection import SelectKBest
      from sklearn.feature_selection import chi2, f_classif,mutual_info_classif
```

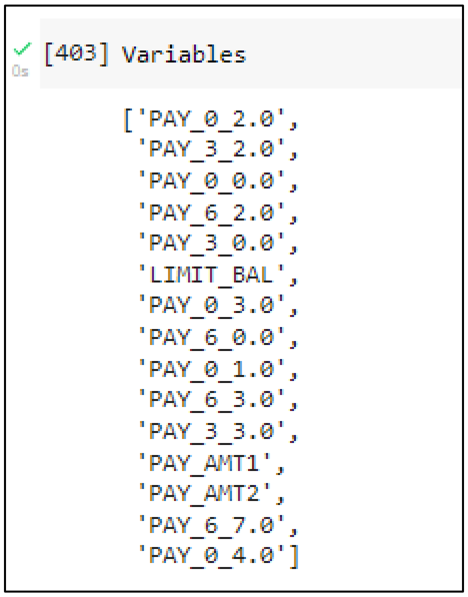
The best variables selected are detailed as the result of the code above; Status payments were the most important variables, next to LIMIT_BAL.

Figure 46



```
[402] Mejores_Variables=SelectKBest(score_func=f_classif,k=X.shape[1])
      fit=Mejores_Variables.fit(X,Y)
      Puntajes=pd.DataFrame(fit.scores_)
      Columnas=pd.DataFrame(X.columns)
      df_completo=pd.concat([Columnas,Puntajes],axis=1)
      df_completo.columns=["Variable","Puntaje"]
      df_completo=df_completo.nlargest(15,"Puntaje")
      Variables=df_completo.Variable.to_list()
```

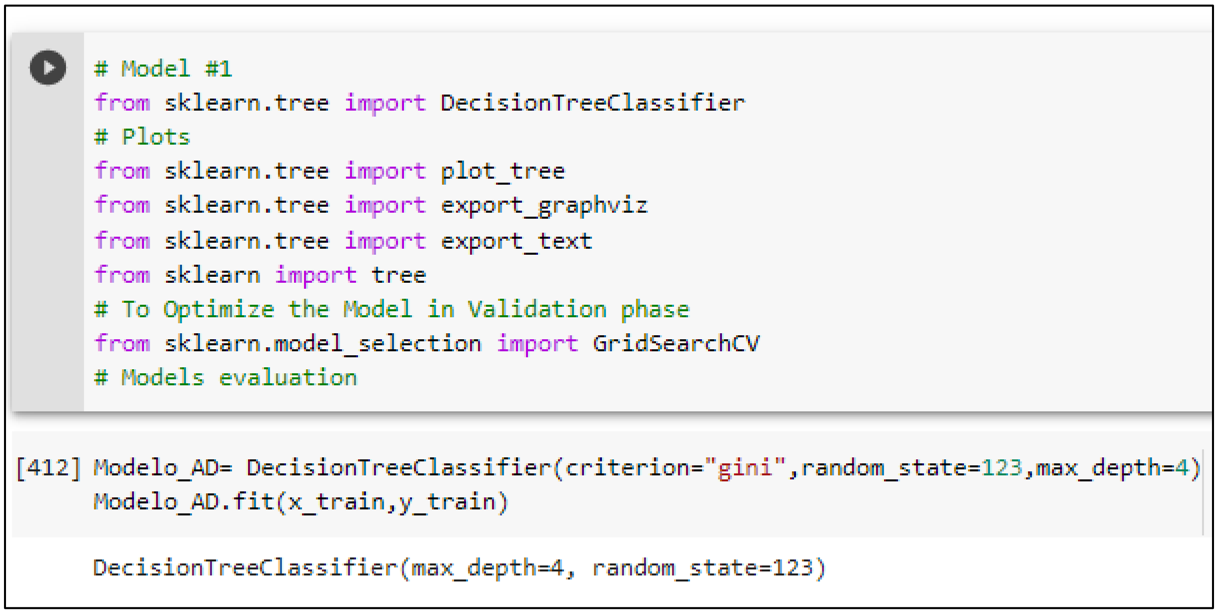
Figure 47



```
[403] Variables

      ['PAY_0_2.0',
       'PAY_3_2.0',
       'PAY_0_0.0',
       'PAY_6_2.0',
       'PAY_3_0.0',
       'LIMIT_BAL',
       'PAY_0_3.0',
       'PAY_6_0.0',
       'PAY_0_1.0',
       'PAY_6_3.0',
       'PAY_3_3.0',
       'PAY_AMT1',
       'PAY_AMT2',
       'PAY_6_7.0',
       'PAY_0_4.0']
```

16. Model Technique #1: Decision trees: Classifier

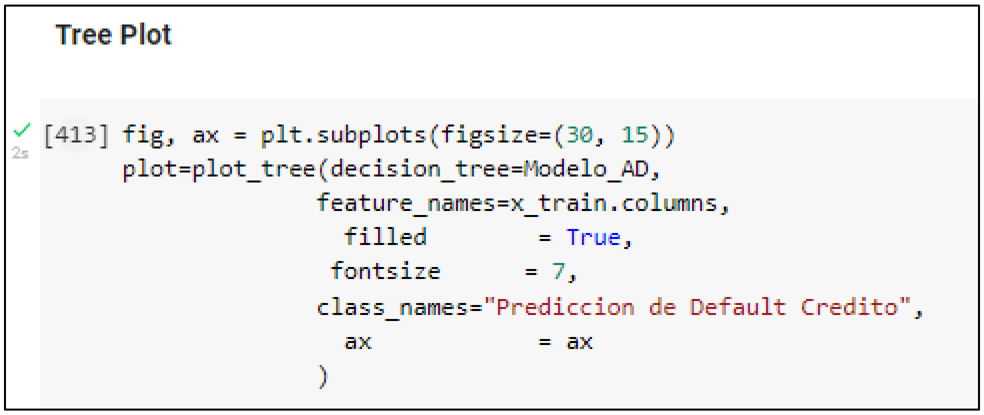
Figure 48



```
# Model #1
from sklearn.tree import DecisionTreeClassifier
# Plots
from sklearn.tree import plot_tree
from sklearn.tree import export_graphviz
from sklearn.tree import export_text
from sklearn import tree
# To Optimize the Model in Validation phase
from sklearn.model_selection import GridSearchCV
# Models evaluation
```

```
[412] Modelo_AD= DecisionTreeClassifier(criterion="gini",random_state=123,max_depth=4)
      Modelo_AD.fit(x_train,y_train)

      DecisionTreeClassifier(max_depth=4, random_state=123)
```

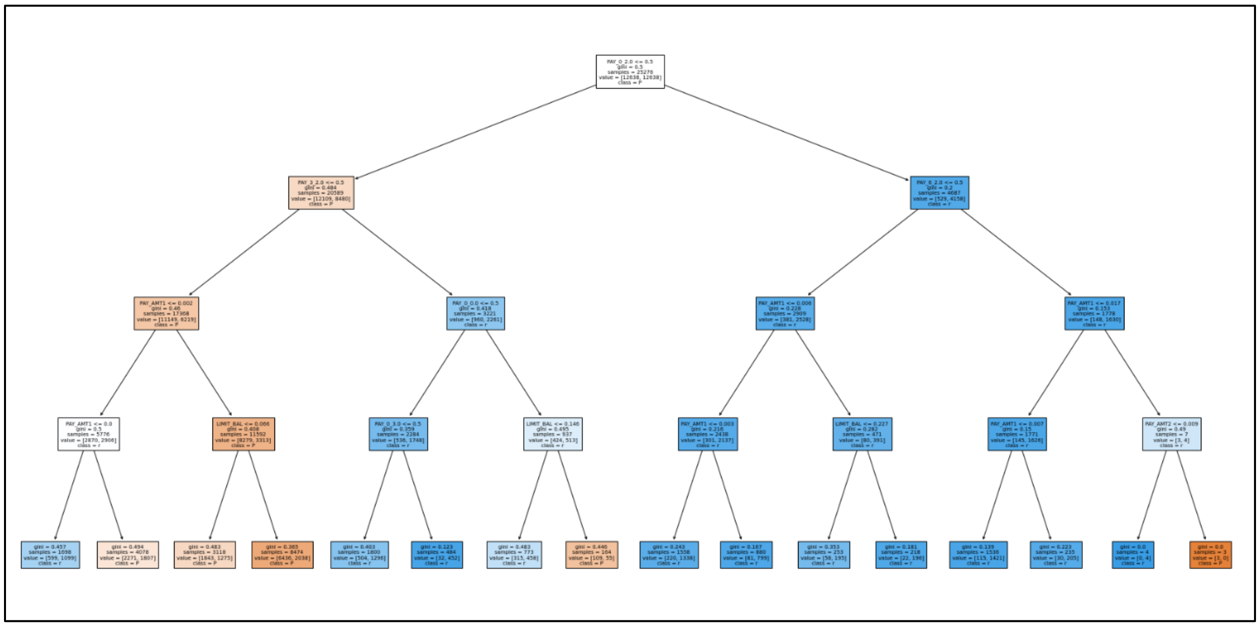
Figure 49



Tree Plot

```
[413] fig, ax = plt.subplots(figsize=(30, 15))
      plot=plot_tree(decision_tree=Modelo_AD,
                     feature_names=x_train.columns,
                       filled        = True,
                      fontsize      = 7,
                     class_names="Prediccion de Default Credito",
                       ax            = ax
                     )
```

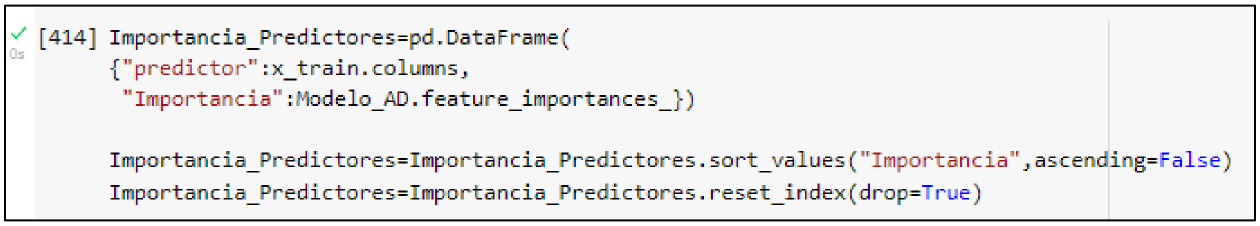
Figure 50



16.1. Importance of variables

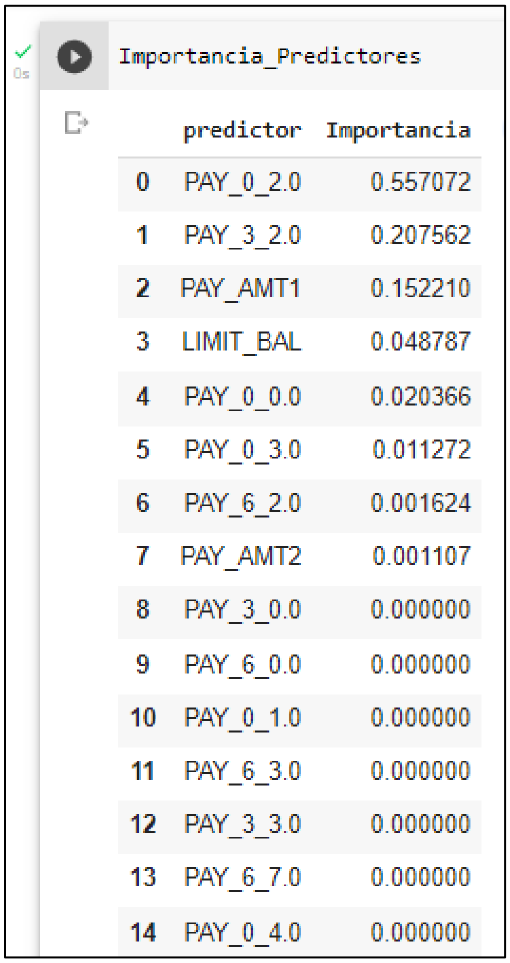
The Variables represent numeric values, characters, character strings, or memory addresses; The most used data type of variables are integer, string, float, double and Boolean. Variables represent the data. The most important predictors in this model were PAY0 to PAY6 for each month, and the credit amount.

Figure 51



```
[414] Importancia_Predictores=pd.DataFrame(
      {"predictor":x_train.columns,
       "Importancia":Modelo_AD.feature_importances_})

      Importancia_Predictores=Importancia_Predictores.sort_values("Importancia",ascending=False)
      Importancia_Predictores=Importancia_Predictores.reset_index(drop=True)
```

Figure 52



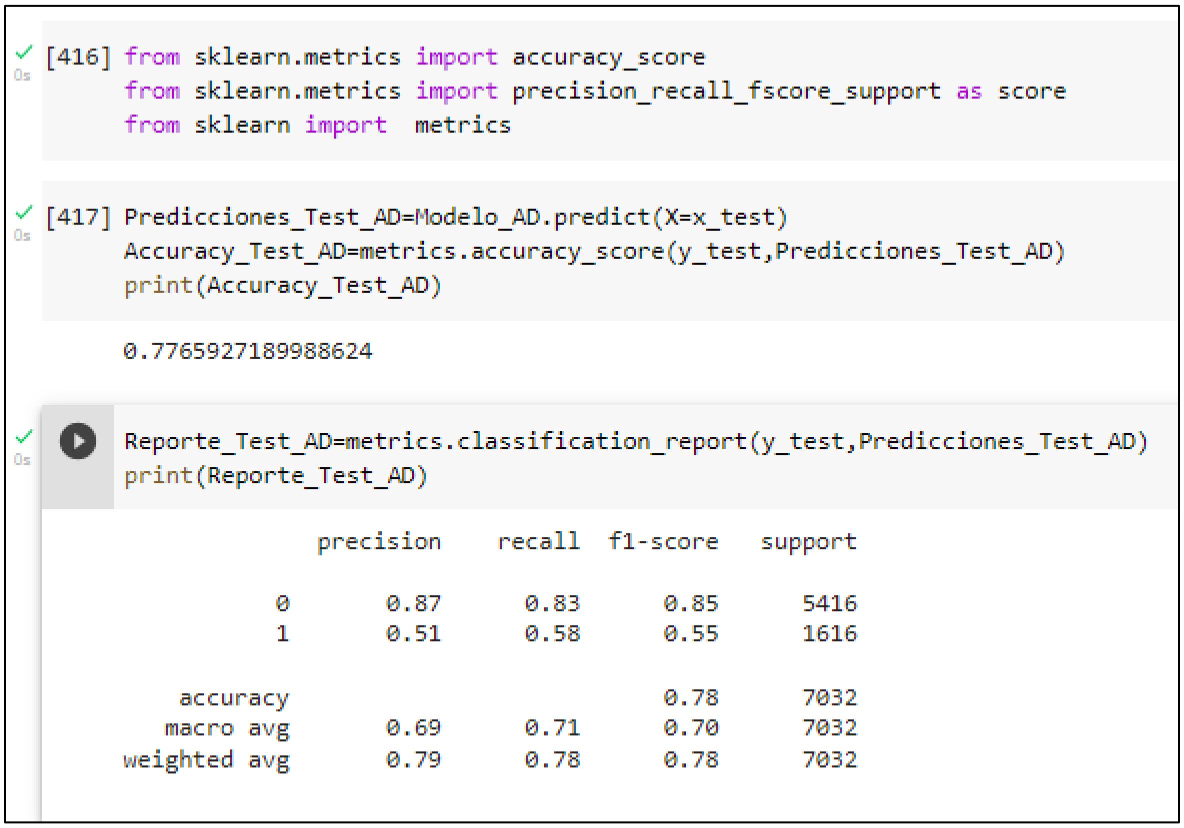
```
Importancia_Predictores
```

	predictor	Importancia
0	PAY_0_2.0	0.557072
1	PAY_3_2.0	0.207562
2	PAY_AMT1	0.152210
3	LIMIT_BAL	0.048787
4	PAY_0_0.0	0.020366
5	PAY_0_3.0	0.011272
6	PAY_6_2.0	0.001624
7	PAY_AMT2	0.001107
8	PAY_3_0.0	0.000000
9	PAY_6_0.0	0.000000
10	PAY_0_1.0	0.000000
11	PAY_6_3.0	0.000000
12	PAY_3_3.0	0.000000
13	PAY_6_7.0	0.000000
14	PAY_0_4.0	0.000000

Prediction error

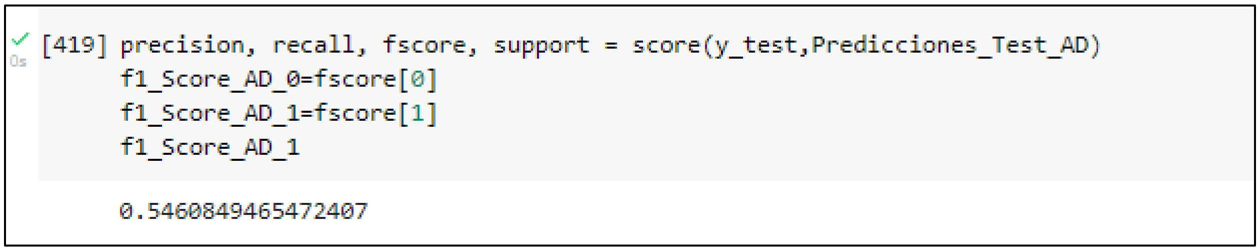
One conventional approach to determine performance of binary classification models is to calculate precision and recall. In this model, precision is defined as the number of correctly predicted clients who will fail to pay their obligation divided by the predicted number of default clients, while recall is defined as the number of correctly predicted clients who will fail to pay their obligations divided by the actual number of default credit card clients.

Figure 53



```
[416] from sklearn.metrics import accuracy_score
      from sklearn.metrics import precision_recall_fscore_support as score
      from sklearn import  metrics

[417] Predicciones_Test_AD=Modelo_AD.predict(X=x_test)
      Accuracy_Test_AD=metrics.accuracy_score(y_test,Predicciones_Test_AD)
      print(Accuracy_Test_AD)

      0.7765927189988624

      Reporte_Test_AD=metrics.classification_report(y_test,Predicciones_Test_AD)
      print(Reporte_Test_AD)

                    precision    recall  f1-score   support

                 0       0.87      0.83      0.85      5416
                 1       0.51      0.58      0.55      1616

          accuracy                           0.78      7032
         macro avg       0.69      0.71      0.70      7032
      weighted avg       0.79      0.78      0.78      7032
```

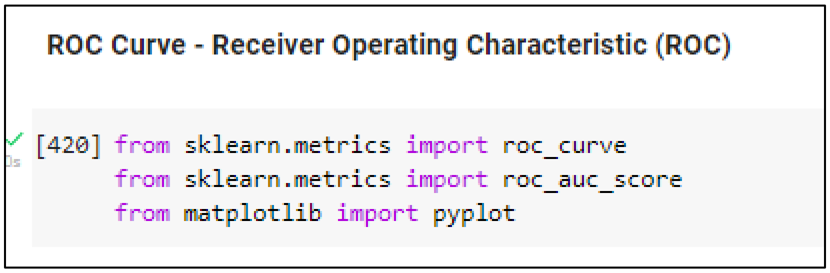
Figure 54



```
[419] precision, recall, fscore, support = score(y_test,Predicciones_Test_AD)
      f1_Score_AD_0=fscore[0]
      f1_Score_AD_1=fscore[1]
      f1_Score_AD_1

      0.5460849465472407
```

The ROC CURVE: An ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters: True Positive Rate. False Positive Rate. The ROC CURVE for the decision tree model is 0.76.

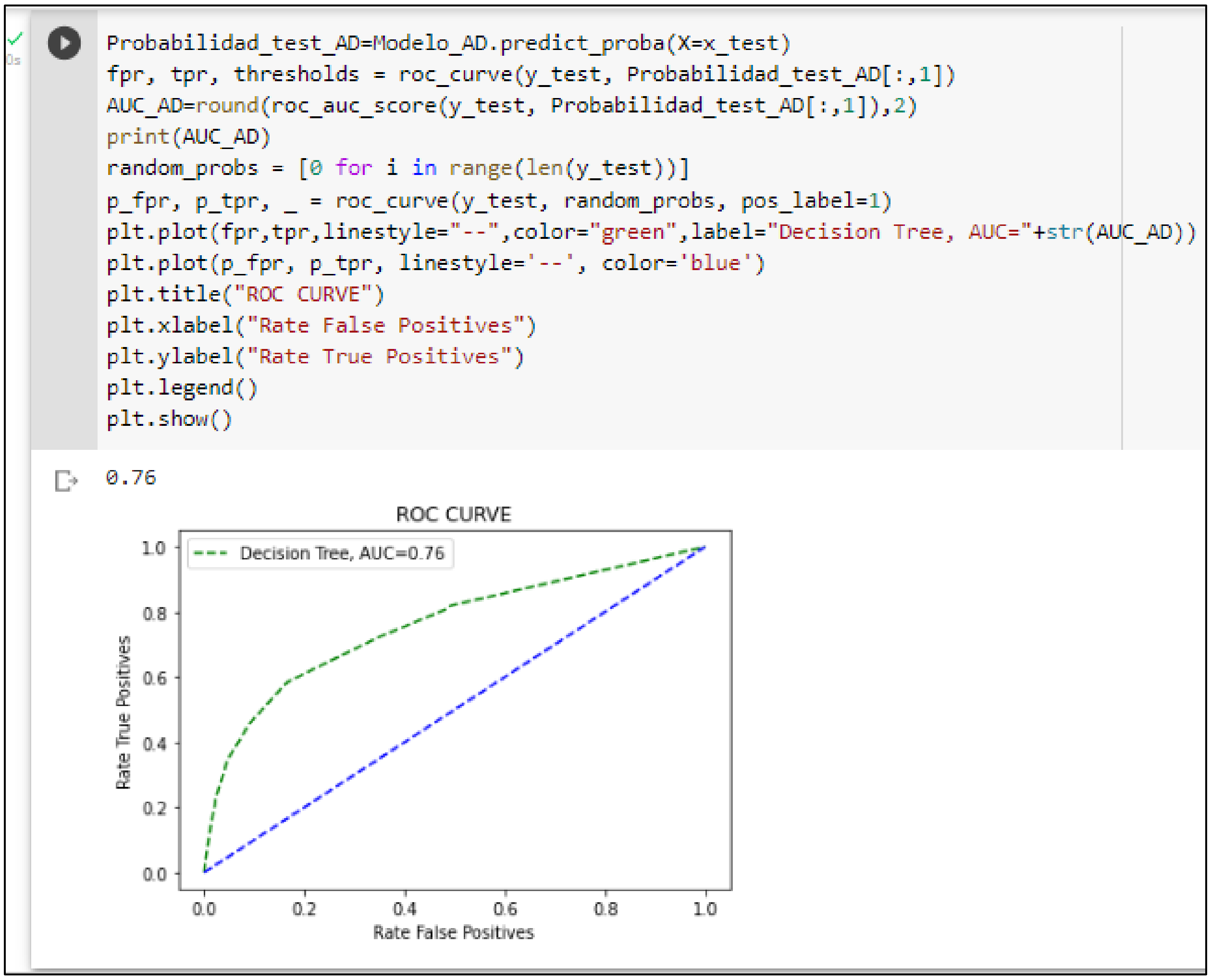
Figure 55



ROC Curve - Receiver Operating Characteristic (ROC)

```
[420] from sklearn.metrics import roc_curve
      from sklearn.metrics import roc_auc_score
      from matplotlib import pyplot
```

Figure 56



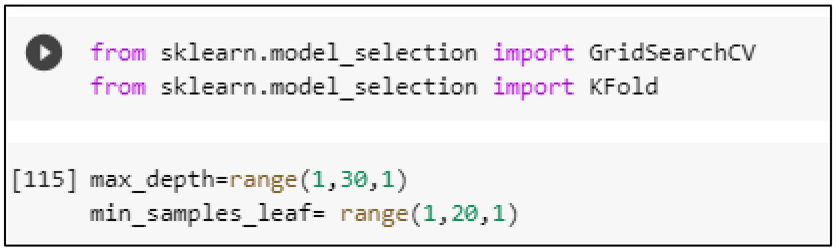
```
Probabilidad_test_AD=Modelo_AD.predict_proba(X=x_test)
fpr, tpr, thresholds = roc_curve(y_test, Probabilidad_test_AD[:,1])
AUC_AD=round(roc_auc_score(y_test, Probabilidad_test_AD[:,1]),2)
print(AUC_AD)
random_probs = [0 for i in range(len(y_test))]
p_fpr, p_tpr, _ = roc_curve(y_test, random_probs, pos_label=1)
plt.plot(fpr,tpr,linestyle="--",color="green",label="Decision Tree, AUC="+str(AUC_AD))
plt.plot(p_fpr, p_tpr, linestyle='--', color='blue')
plt.title("ROC CURVE")
plt.xlabel("Rate False Positives")
plt.ylabel("Rate True Positives")
plt.legend()
plt.show()
```

```
0.76
```

17. Optimization Decision Tree

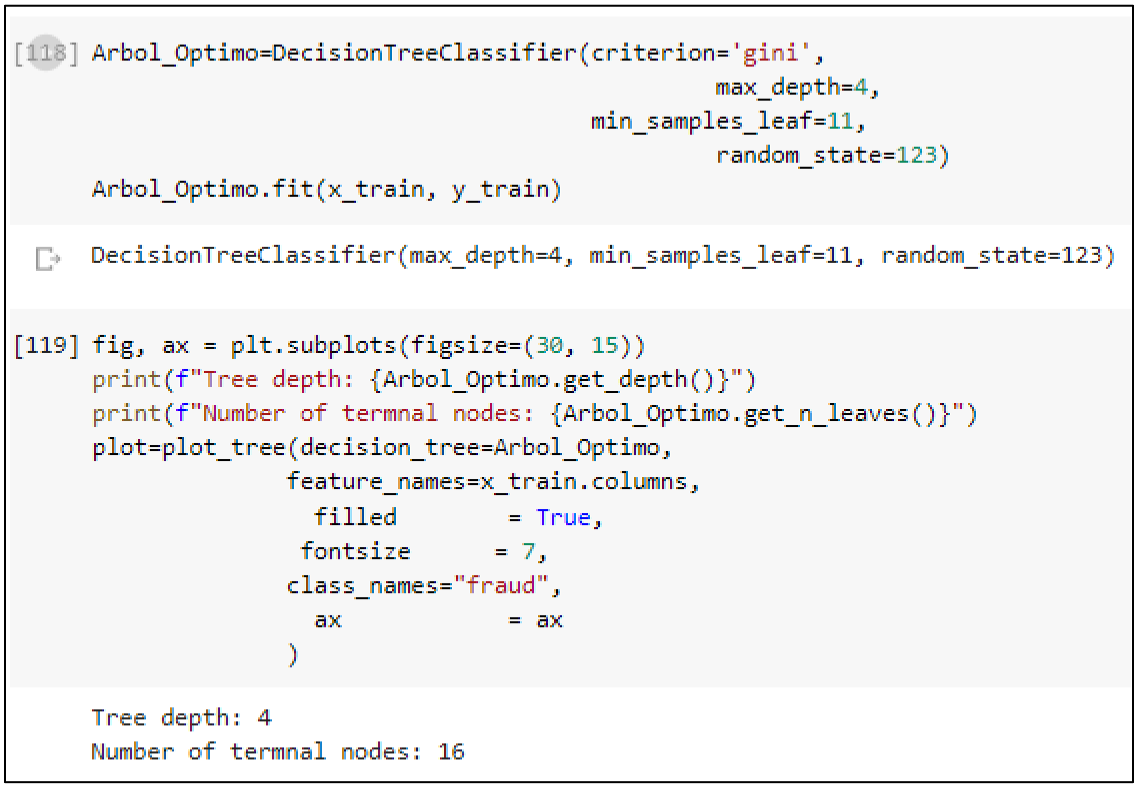
The decision tree was optimized, and the precision was calculated, the ROC CURVE was 0.76

Figure 57



```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import KFold

[115] max_depth=range(1,30,1)
      min_samples_leaf= range(1,20,1)
```

Figure 58



```
[118] Arbol_Optimo=DecisionTreeClassifier(criterion='gini',
                                             max_depth=4,
                                      min_samples_leaf=11,
                                             random_state=123)
      Arbol_Optimo.fit(x_train, y_train)

      DecisionTreeClassifier(max_depth=4, min_samples_leaf=11, random_state=123)

[119] fig, ax = plt.subplots(figsize=(30, 15))
      print(f"Tree depth: {Arbol_Optimo.get_depth()}")
      print(f"Number of termnal nodes: {Arbol_Optimo.get_n_leaves()}")
      plot=plot_tree(decision_tree=Arbol_Optimo,
                    feature_names=x_train.columns,
                      filled        = True,
                     fontsize      = 7,
                    class_names="fraud",
                      ax            = ax
                    )

      Tree depth: 4
      Number of termnal nodes: 16
```

Figure 59

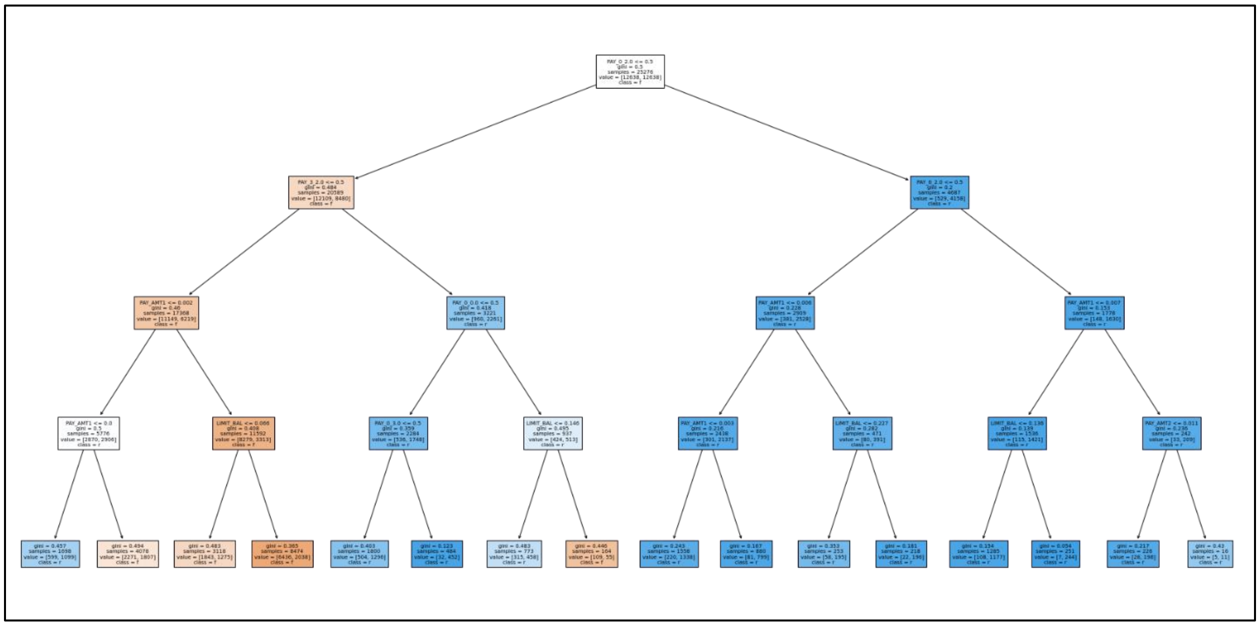
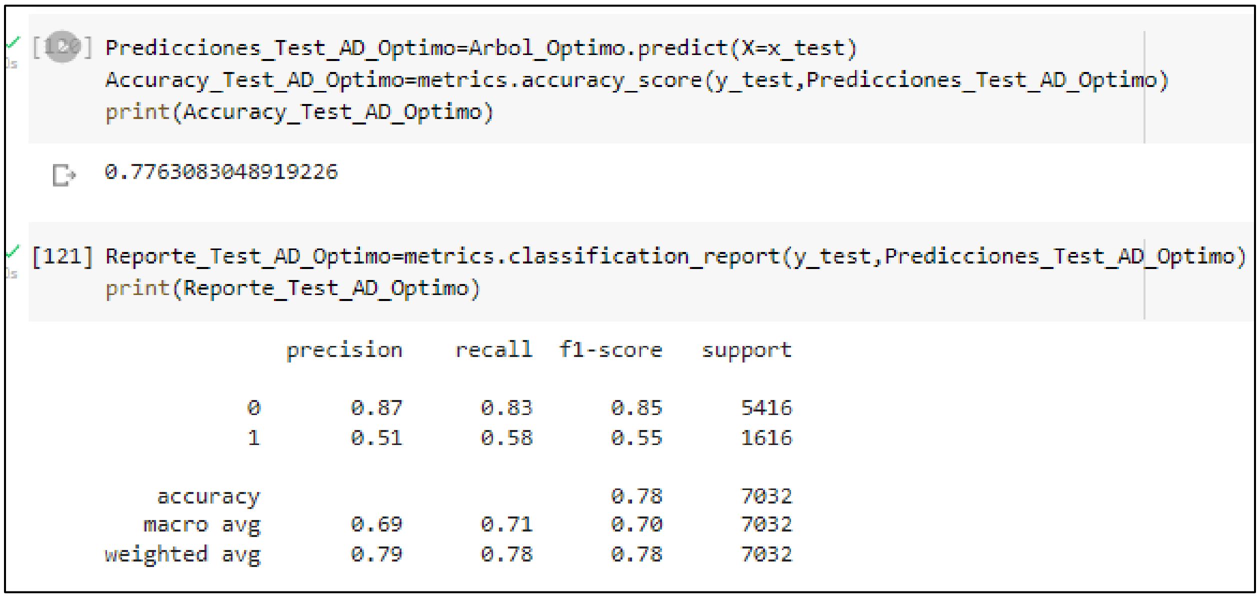
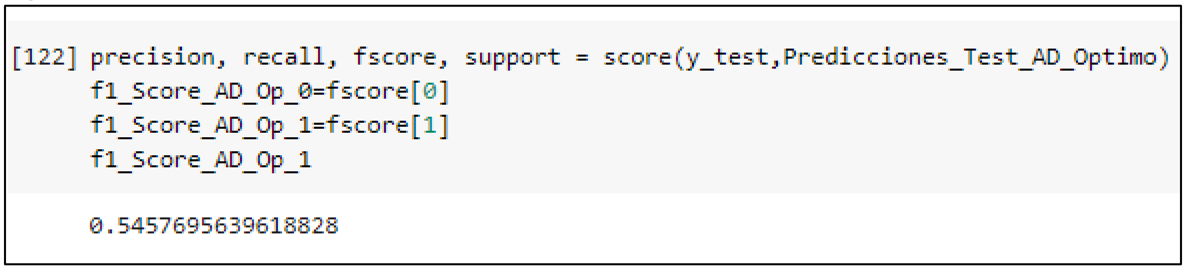


Figure 60



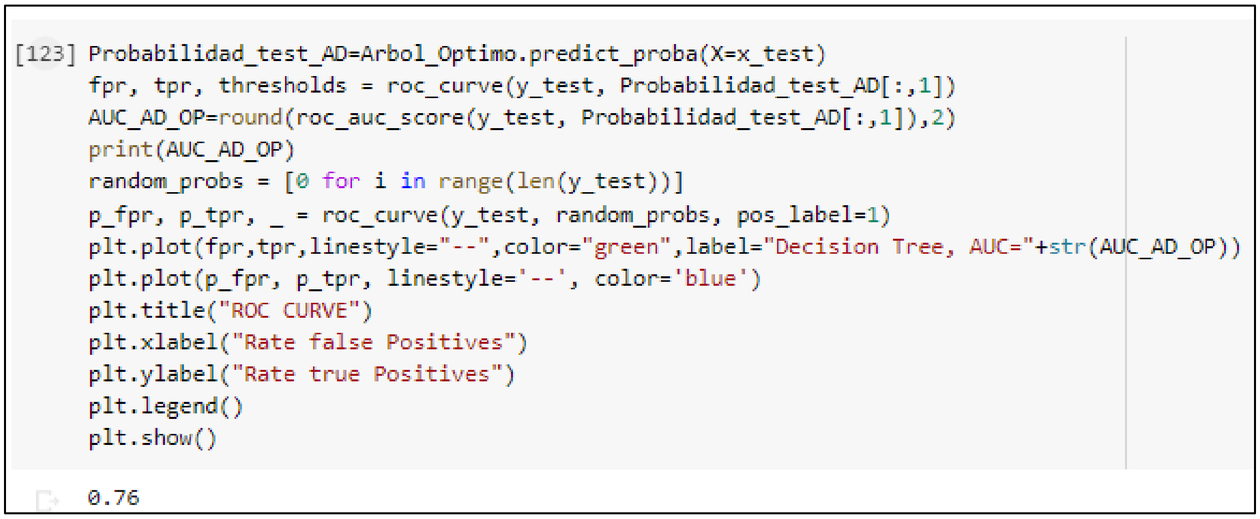
```
[120] Predicciones_Test_AD_Optimo=Arbol_Optimo.predict(X=x_test)
      Accuracy_Test_AD_Optimo=metrics.accuracy_score(y_test,Predicciones_Test_AD_Optimo)
      print(Accuracy_Test_AD_Optimo)

      0.7763083048919226

[121] Reporte_Test_AD_Optimo=metrics.classification_report(y_test,Predicciones_Test_AD_Optimo)
      print(Reporte_Test_AD_Optimo)

                   precision    recall  f1-score   support

               0       0.87      0.83      0.85      5416
               1       0.51      0.58      0.55      1616

        accuracy                           0.78      7032
       macro avg       0.69      0.71      0.70      7032
    weighted avg       0.79      0.78      0.78      7032
```

Figure 61



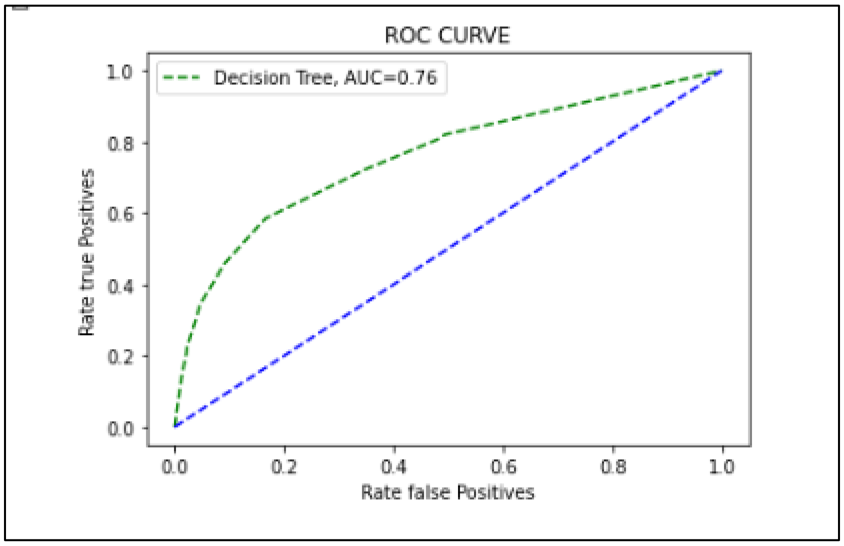
```
[122] precision, recall, fscore, support = score(y_test,Predicciones_Test_AD_Optimo)
      f1_Score_AD_Op_0=fscore[0]
      f1_Score_AD_Op_1=fscore[1]
      f1_Score_AD_Op_1

      0.5457695639618828
```

Figure 62



```
[123] Probabilidad_test_AD=Arbol_Optimo.predict_proba(X=x_test)
      fpr, tpr, thresholds = roc_curve(y_test, Probabilidad_test_AD[:,1])
      AUC_AD_OP=round(roc_auc_score(y_test, Probabilidad_test_AD[:,1]),2)
      print(AUC_AD_OP)
      random_probs = [0 for i in range(len(y_test))]
      p_fpr, p_tpr, _ = roc_curve(y_test, random_probs, pos_label=1)
      plt.plot(fpr,tpr,linestyle="--",color="green",label="Decision Tree, AUC="+str(AUC_AD_OP))
      plt.plot(p_fpr, p_tpr, linestyle='--', color='blue')
      plt.title("ROC CURVE")
      plt.xlabel("Rate false Positives")
      plt.ylabel("Rate true Positives")
      plt.legend()
      plt.show()

      0.76
```

Figure 63



18. Model Technique #2: Random Forest

The term assembler means group. Assembler-type methods are made up of a group of predictive models that allow to achieve a better precision and stability of the model. These provide a significant improvement to decision tree models.

Why do tree assemblers arise?

- Like all models, a decision tree also suffers from bias and variance problems. That is, 'how much on average are the predicted values different from the actual values' (bias) and 'how different would the predictions of a model be at the same point if different samples were taken from the same population' (variance).
- Building a small tree will result in a model with low variance and high bias. Typically, with increasing model complexity, you will see a reduction in prediction error due to a lower bias in the model. At some point the model will be very complex and an over-fitting of the model will occur which will start to suffer from high variance.
- The optimal model should maintain a balance between these two types of errors. This is known as a “trade-off” between bias and variance errors. The use of assemblers is one way to apply this trade-off

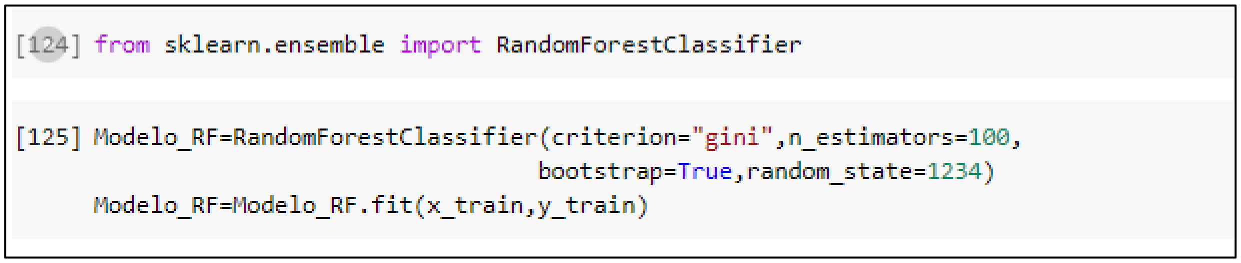
Random Forest Advantages

- There are very few assumptions and therefore data preparation is minimal.
- It can handle up to thousands of input variables and identify the most significant ones. Dimensionality reduction method.
- One of the outputs of the model is the importance of variables.
- Incorporates effective methods to estimate missing values.
- It is possible to use it as an unsupervised method (clustering) and outlier detection.

Random Forest Disadvantages

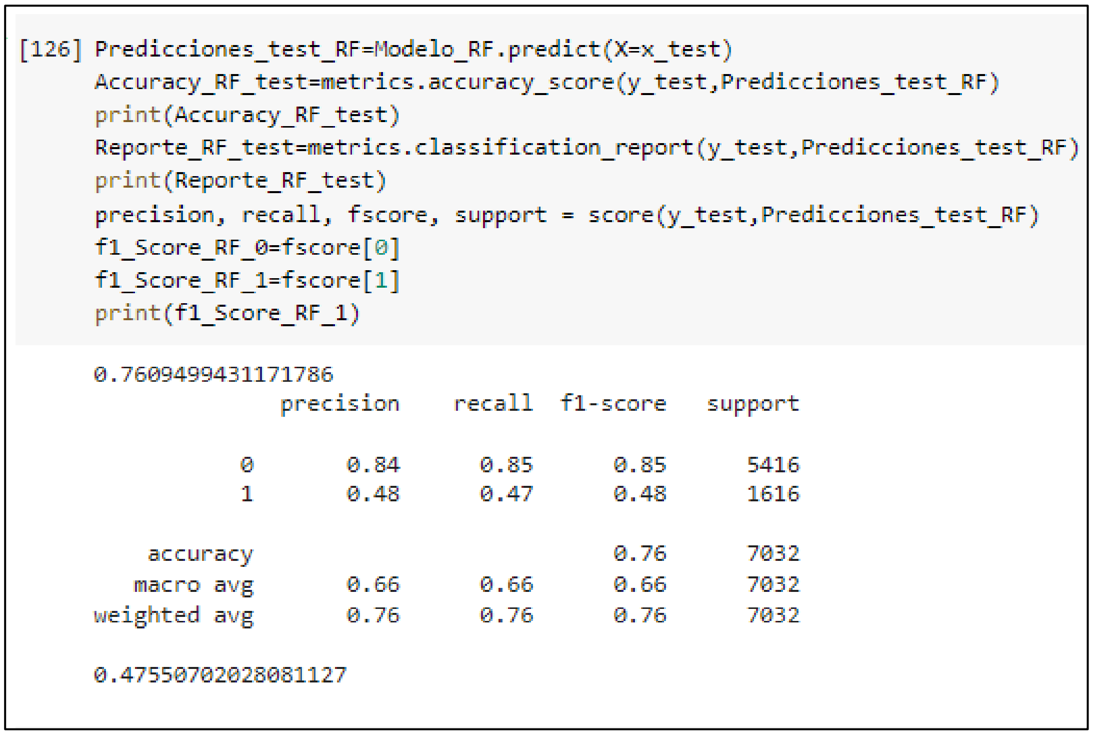
- Loss of interpretation
- Good for classification, not so much for regression. Predictions are not continuous in nature.
- In regression, you cannot predict beyond the range of values of the training set.
- Little control over what the model does (black box model for statistical modelers).

Figure 64



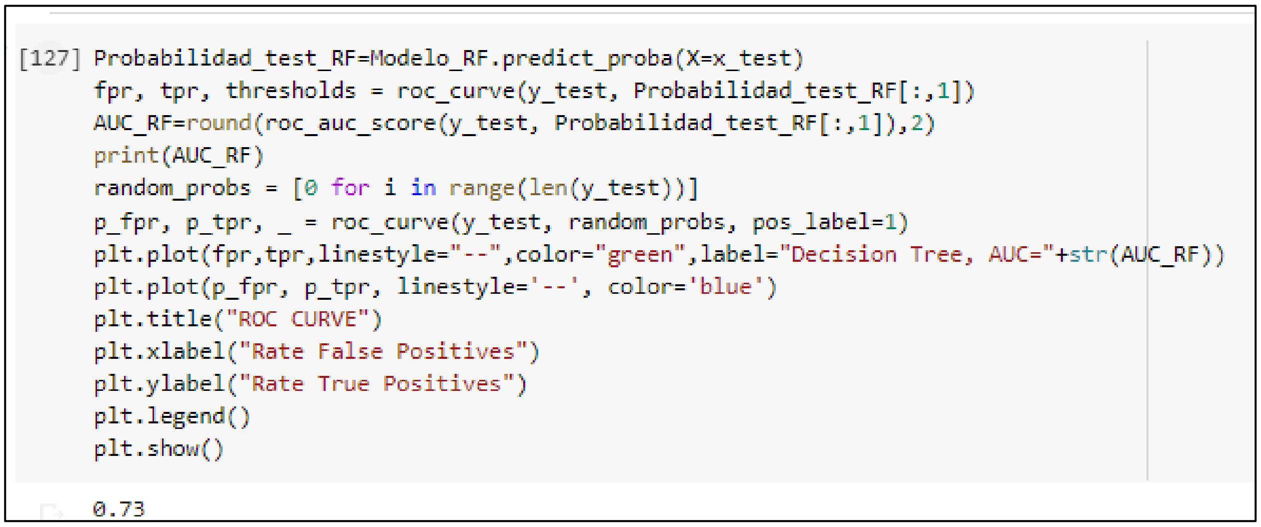
```
[124] from sklearn.ensemble import RandomForestClassifier

[125] Modelo_RF=RandomForestClassifier(criterion="gini",n_estimators=100,
                                      bootstrap=True,random_state=1234)
      Modelo_RF=Modelo_RF.fit(x_train,y_train)
```

Figure 65



```
[126] Predicciones_test_RF=Modelo_RF.predict(X=x_test)
      Accuracy_RF_test=metrics.accuracy_score(y_test,Predicciones_test_RF)
      print(Accuracy_RF_test)
      Reporte_RF_test=metrics.classification_report(y_test,Predicciones_test_RF)
      print(Reporte_RF_test)
      precision, recall, fscore, support = score(y_test,Predicciones_test_RF)
      f1_Score_RF_0=fscore[0]
      f1_Score_RF_1=fscore[1]
      print(f1_Score_RF_1)

      0.7609499431171786
                    precision    recall  f1-score   support

                 0       0.84      0.85      0.85      5416
                 1       0.48      0.47      0.48      1616

          accuracy                           0.76      7032
         macro avg       0.66      0.66      0.66      7032
      weighted avg       0.76      0.76      0.76      7032

      0.47550702028081127
```

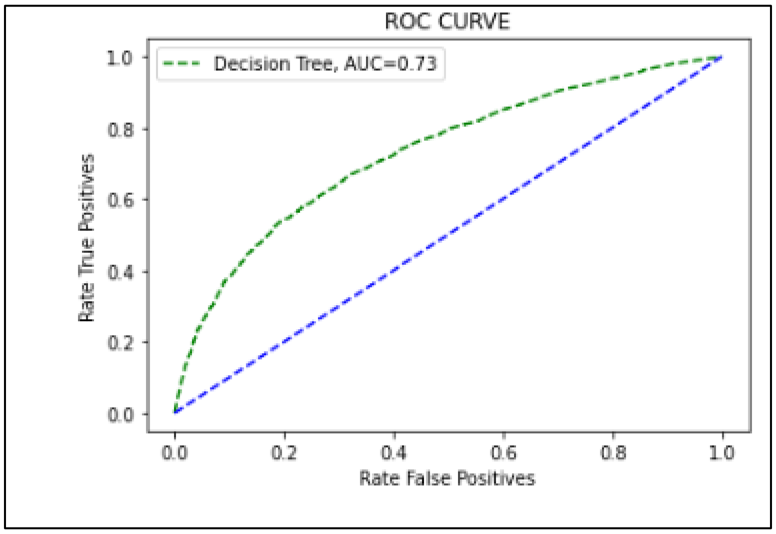
Figure 66



```
[127] Probabilidad_test_RF=Modelo_RF.predict_proba(X=x_test)
      fpr, tpr, thresholds = roc_curve(y_test, Probabilidad_test_RF[:,1])
      AUC_RF=round(roc_auc_score(y_test, Probabilidad_test_RF[:,1]),2)
      print(AUC_RF)
      random_probs = [0 for i in range(len(y_test))]
      p_fpr, p_tpr, _ = roc_curve(y_test, random_probs, pos_label=1)
      plt.plot(fpr,tpr,linestyle="--",color="green",label="Decision Tree, AUC="+str(AUC_RF))
      plt.plot(p_fpr, p_tpr, linestyle='--', color='blue')
      plt.title("ROC CURVE")
      plt.xlabel("Rate False Positives")
      plt.ylabel("Rate True Positives")
      plt.legend()
      plt.show()

      0.73
```

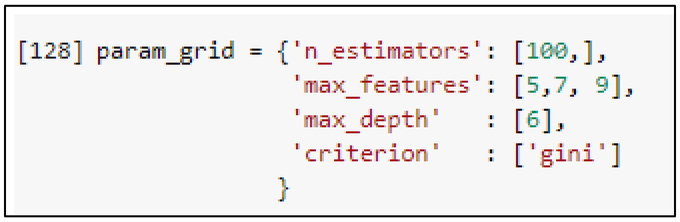
The ROC CURVE for the Random Forest model was 0.73.

Figure 67



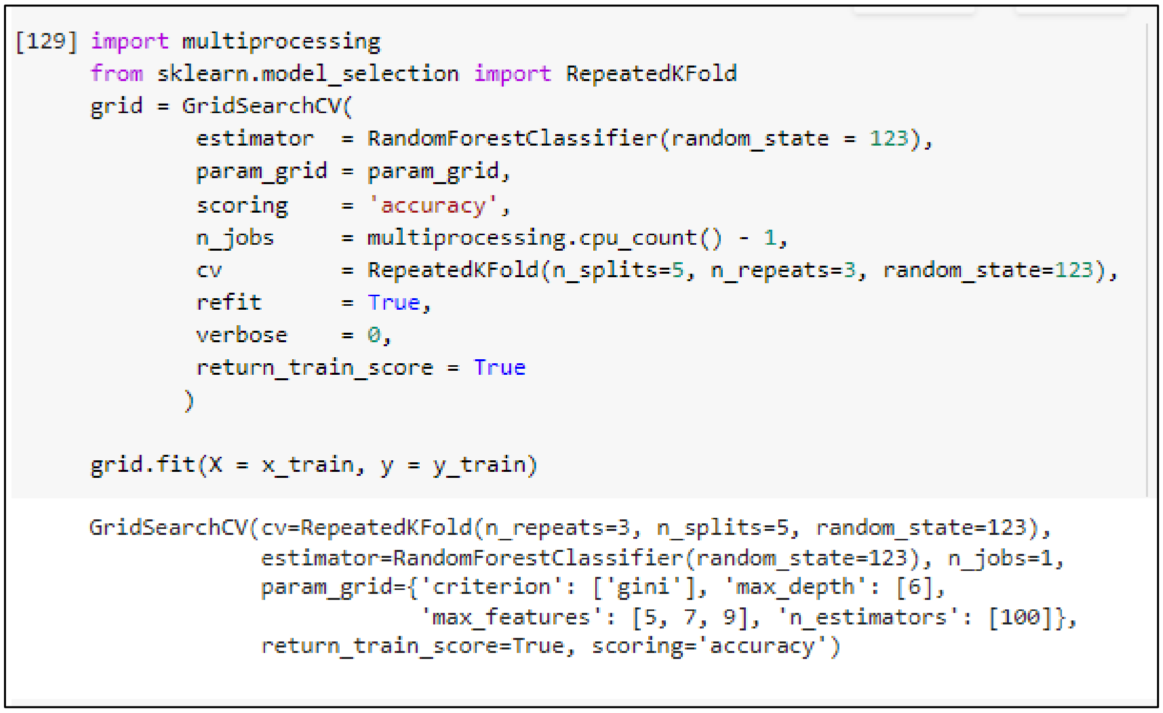
19. Optimization Random Forest

Figure 68



```
[128] param_grid = {'n_estimators': [100,],
                    'max_features': [5,7, 9],
                    'max_depth'   : [6],
                    'criterion'   : ['gini']
                   }
```

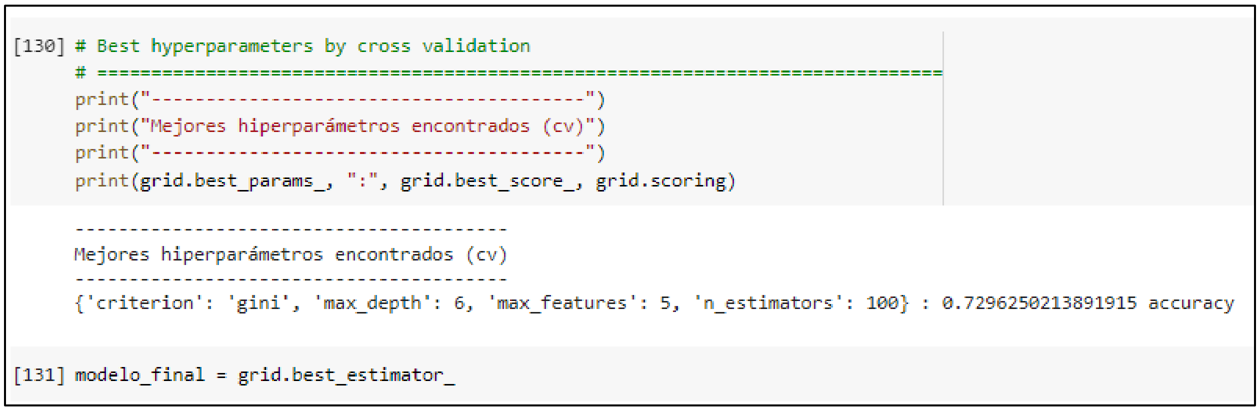
Figure 69



```
[129] import multiprocessing
      from sklearn.model_selection import RepeatedKFold
      grid = GridSearchCV(
              estimator  = RandomForestClassifier(random_state = 123),
              param_grid = param_grid,
              scoring    = 'accuracy',
              n_jobs     = multiprocessing.cpu_count() - 1,
              cv         = RepeatedKFold(n_splits=5, n_repeats=3, random_state=123),
              refit      = True,
              verbose    = 0,
              return_train_score = True
             )

      grid.fit(X = x_train, y = y_train)
```

```
      GridSearchCV(cv=RepeatedKFold(n_repeats=3, n_splits=5, random_state=123),
                   estimator=RandomForestClassifier(random_state=123), n_jobs=1,
                   param_grid={'criterion': ['gini'], 'max_depth': [6],
                               'max_features': [5, 7, 9], 'n_estimators': [100]},
                   return_train_score=True, scoring='accuracy')
```

Figure 70

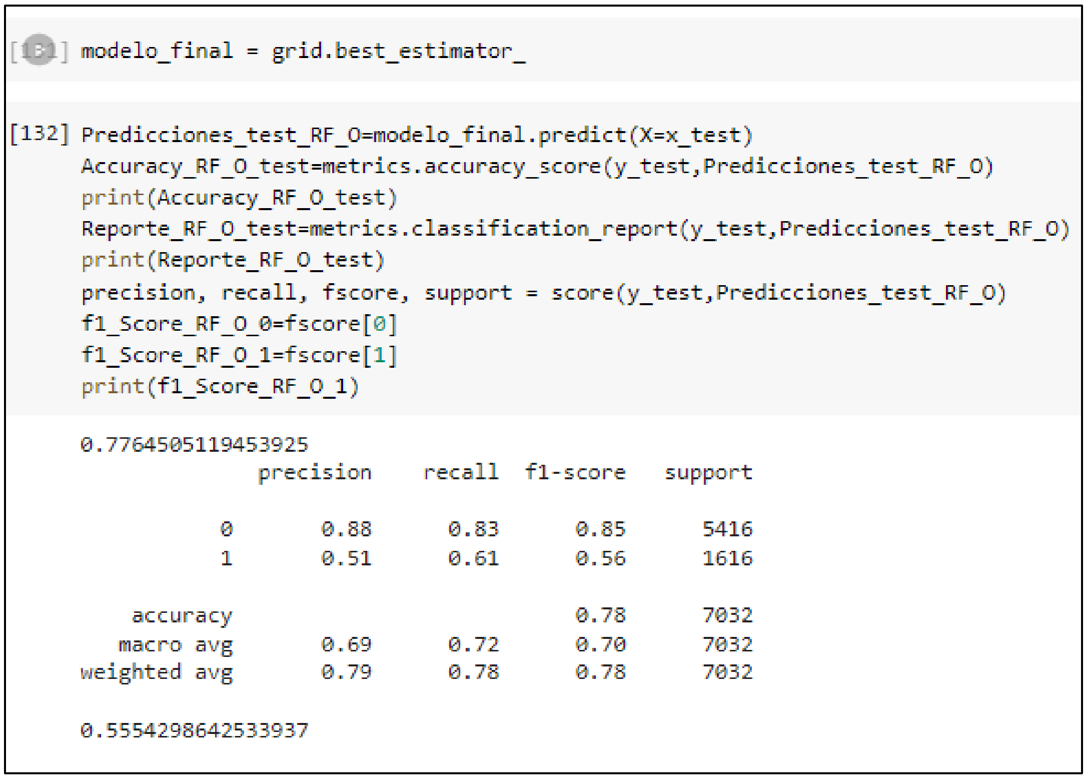


```
[130] # Best hyperparameters by cross validation
      # ==============================================================================
      print("----------------------------------------")
      print("Mejores hiperparámetros encontrados (cv)")
      print("----------------------------------------")
      print(grid.best_params_, ":", grid.best_score_, grid.scoring)
```

```
      ----------------------------------------
      Mejores hiperparámetros encontrados (cv)
      ----------------------------------------
      {'criterion': 'gini', 'max_depth': 6, 'max_features': 5, 'n_estimators': 100} : 0.7296250213891915 accuracy
```

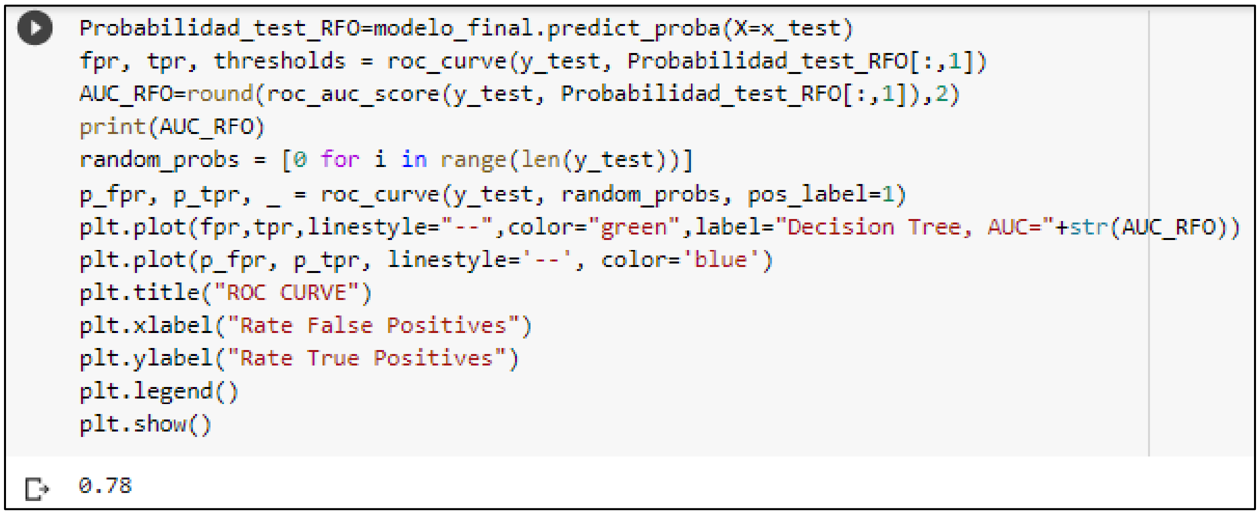
```
[131] modelo_final = grid.best_estimator_
```

Figure 71



```
[131] modelo_final = grid.best_estimator_

[132] Predicciones_test_RF_O=modelo_final.predict(X=x_test)
      Accuracy_RF_O_test=metrics.accuracy_score(y_test,Predicciones_test_RF_O)
      print(Accuracy_RF_O_test)
      Reporte_RF_O_test=metrics.classification_report(y_test,Predicciones_test_RF_O)
      print(Reporte_RF_O_test)
      precision, recall, fscore, support = score(y_test,Predicciones_test_RF_O)
      f1_Score_RF_O_0=fscore[0]
      f1_Score_RF_O_1=fscore[1]
      print(f1_Score_RF_O_1)

      0.7764505119453925
                    precision    recall  f1-score   support

                 0       0.88      0.83      0.85      5416
                 1       0.51      0.61      0.56      1616

          accuracy                           0.78      7032
         macro avg       0.69      0.72      0.70      7032
      weighted avg       0.79      0.78      0.78      7032

      0.5554298642533937
```

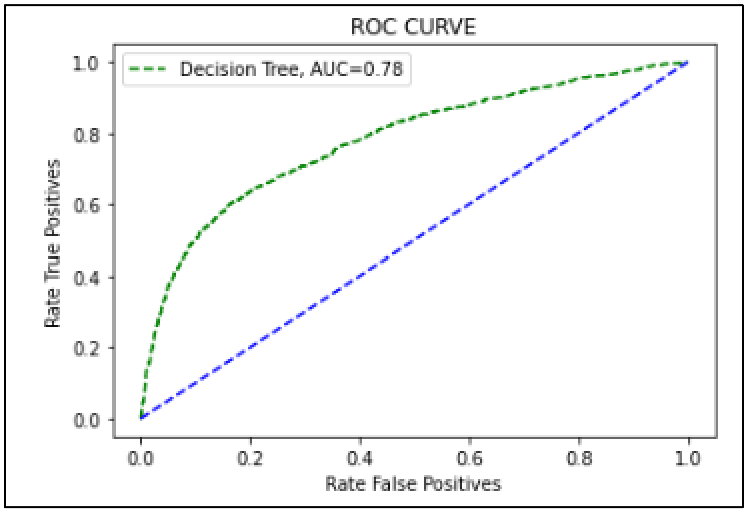
Figure 72



```
Probabilidad_test_RFO=modelo_final.predict_proba(X=x_test)
fpr, tpr, thresholds = roc_curve(y_test, Probabilidad_test_RFO[:,1])
AUC_RFO=round(roc_auc_score(y_test, Probabilidad_test_RFO[:,1]),2)
print(AUC_RFO)
random_probs = [0 for i in range(len(y_test))]
p_fpr, p_tpr, _ = roc_curve(y_test, random_probs, pos_label=1)
plt.plot(fpr,tpr,linestyle="--",color="green",label="Decision Tree, AUC="+str(AUC_RFO))
plt.plot(p_fpr, p_tpr, linestyle='--', color='blue')
plt.title("ROC CURVE")
plt.xlabel("Rate False Positives")
plt.ylabel("Rate True Positives")
plt.legend()
plt.show()

0.78
```

The CURVE ROC result was 0.78

Figure 73



20. Model Technique #3: Gradient Boosting

A gradient boosting model is made up of a set of individual decision trees, trained sequentially, so that each new tree tries to improve the errors of the previous trees. The prediction of a new observation is obtained by adding the predictions of all the individual trees that make up the model.

Advantage

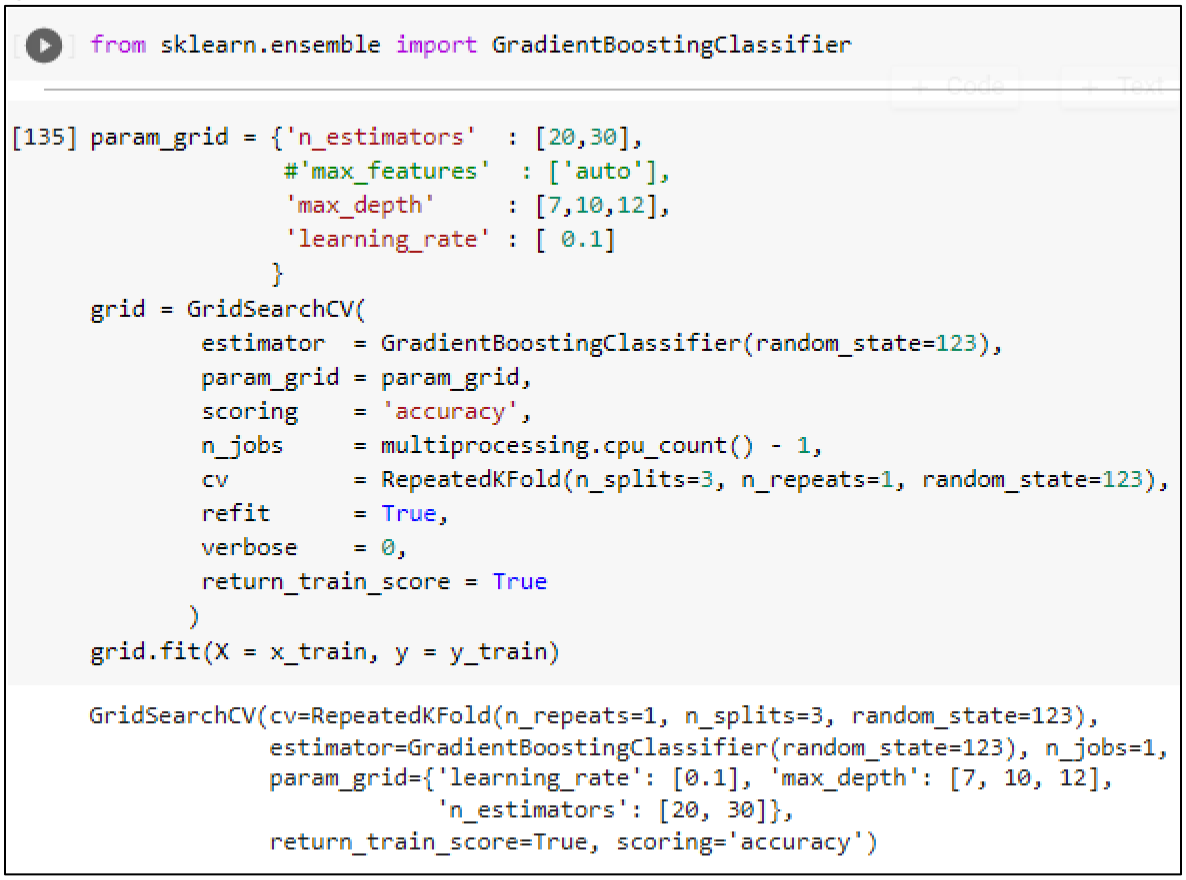
- They are able to select predictors automatically.
- They can be applied to regression and classification problems.
- Trees can, in theory, handle both numerical and categorical predictors without having to create dummy variables or one-hot-encoding. In practice, this depends on the implementation of the algorithm that each library has.
- As these are non-parametric methods, it is not necessary that any specific type of distribution be met.
- They generally require much less data cleaning and pre-processing compared to other statistical learning methods (for example, they do not require standardization).
- They are not very influenced by outliers.

Disadvantages

- When combining multiple trees, the interpretability of models based on a single tree is lost.
- When dealing with continuous predictors, they lose some of their information by categorizing them at the time of node splitting.
- As described later, the creation of the tree branches is achieved by the recursive binary splitting algorithm. This algorithm identifies and evaluates the possible divisions of each predictor according to a certain measure (RSS, Gini, entropy...). Continuous predictors or qualitative predictors with many levels are more likely to contain, just by chance, some optimal cut-off point, so they are usually favored in the creation of trees.

They are not able to extrapolate outside the range of the predictors observed in the training data.

Figure 74

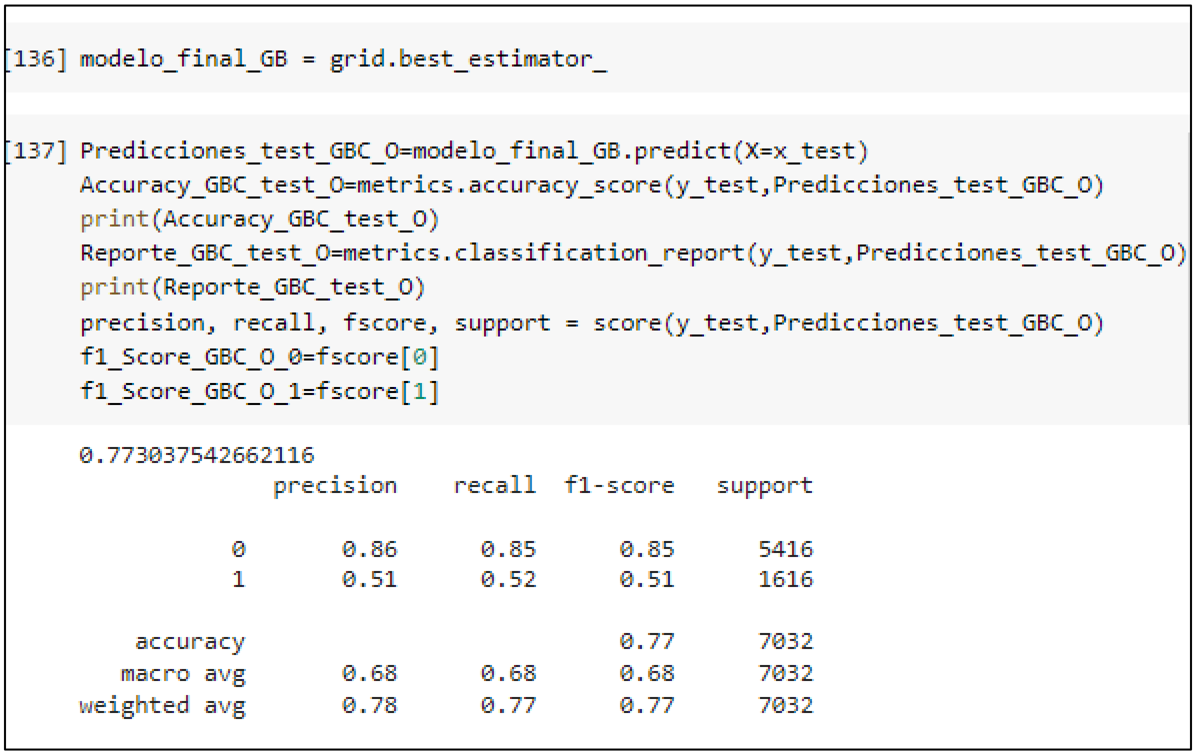


```
from sklearn.ensemble import GradientBoostingClassifier
```

```
[135] param_grid = {'n_estimators'  : [20,30],
                    #'max_features'  : ['auto'],
                    'max_depth'     : [7,10,12],
                    'learning_rate' : [ 0.1]
                   }
      grid = GridSearchCV(
              estimator  = GradientBoostingClassifier(random_state=123),
              param_grid = param_grid,
              scoring    = 'accuracy',
              n_jobs     = multiprocessing.cpu_count() - 1,
              cv         = RepeatedKFold(n_splits=3, n_repeats=1, random_state=123),
              refit      = True,
              verbose    = 0,
              return_train_score = True
             )
      grid.fit(X = x_train, y = y_train)
```

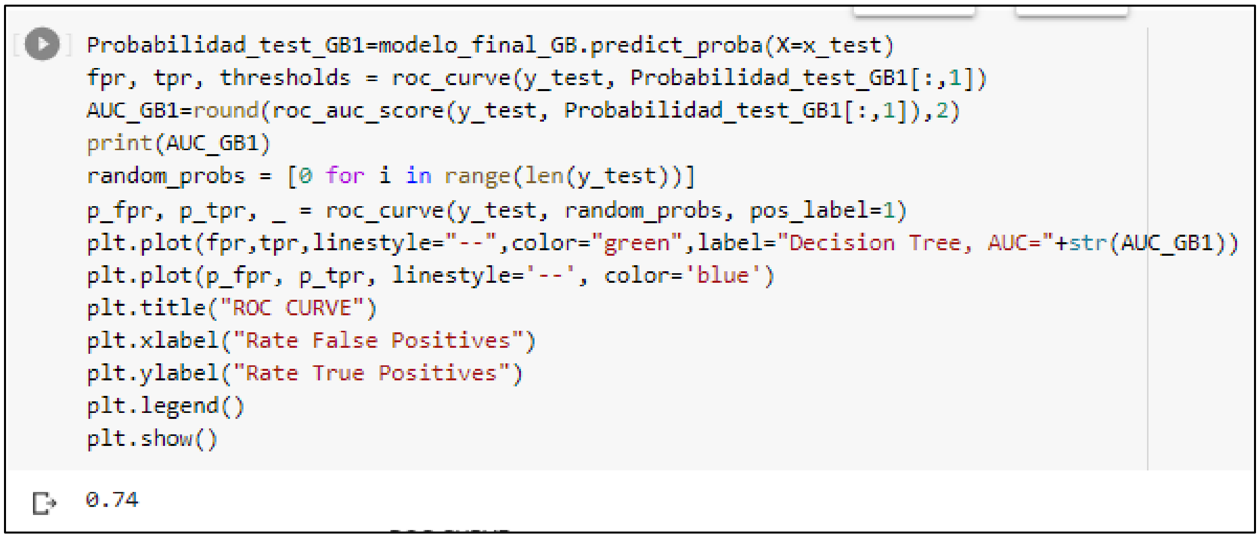
```
GridSearchCV(cv=RepeatedKFold(n_repeats=1, n_splits=3, random_state=123),
             estimator=GradientBoostingClassifier(random_state=123), n_jobs=1,
             param_grid={'learning_rate': [0.1], 'max_depth': [7, 10, 12],
                         'n_estimators': [20, 30]},
             return_train_score=True, scoring='accuracy')
```

Figure 75



```
[136] modelo_final_GB = grid.best_estimator_

[137] Predicciones_test_GBC_O=modelo_final_GB.predict(X=x_test)
      Accuracy_GBC_test_O=metrics.accuracy_score(y_test,Predicciones_test_GBC_O)
      print(Accuracy_GBC_test_O)
      Reporte_GBC_test_O=metrics.classification_report(y_test,Predicciones_test_GBC_O)
      print(Reporte_GBC_test_O)
      precision, recall, fscore, support = score(y_test,Predicciones_test_GBC_O)
      f1_Score_GBC_O_0=fscore[0]
      f1_Score_GBC_O_1=fscore[1]

      0.773037542662116
                    precision    recall  f1-score   support

                 0       0.86      0.85      0.85      5416
                 1       0.51      0.52      0.51      1616

          accuracy                           0.77      7032
         macro avg       0.68      0.68      0.68      7032
      weighted avg       0.78      0.77      0.77      7032
```

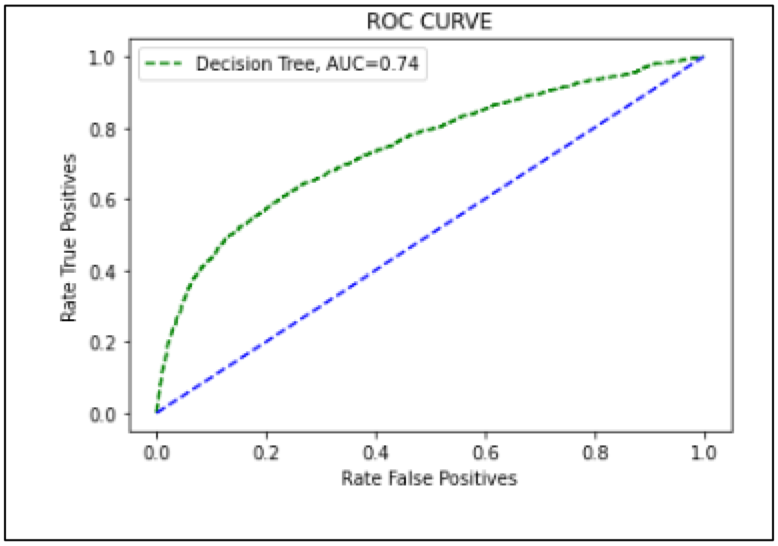
Figure 76



```
Probabilidad_test_GB1=modelo_final_GB.predict_proba(X=x_test)
fpr, tpr, thresholds = roc_curve(y_test, Probabilidad_test_GB1[:,1])
AUC_GB1=round(roc_auc_score(y_test, Probabilidad_test_GB1[:,1]),2)
print(AUC_GB1)
random_probs = [0 for i in range(len(y_test))]
p_fpr, p_tpr, _ = roc_curve(y_test, random_probs, pos_label=1)
plt.plot(fpr,tpr,linestyle="--",color="green",label="Decision Tree, AUC="+str(AUC_GB1))
plt.plot(p_fpr, p_tpr, linestyle='--', color='blue')
plt.title("ROC CURVE")
plt.xlabel("Rate False Positives")
plt.ylabel("Rate True Positives")
plt.legend()
plt.show()

0.74
```

The ROC CURVE result was 0.74

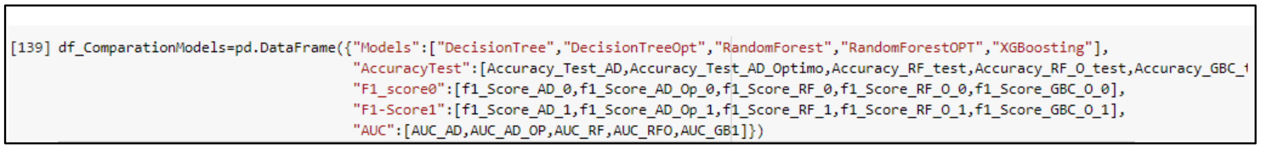
Figure 77



21. Model Comparison

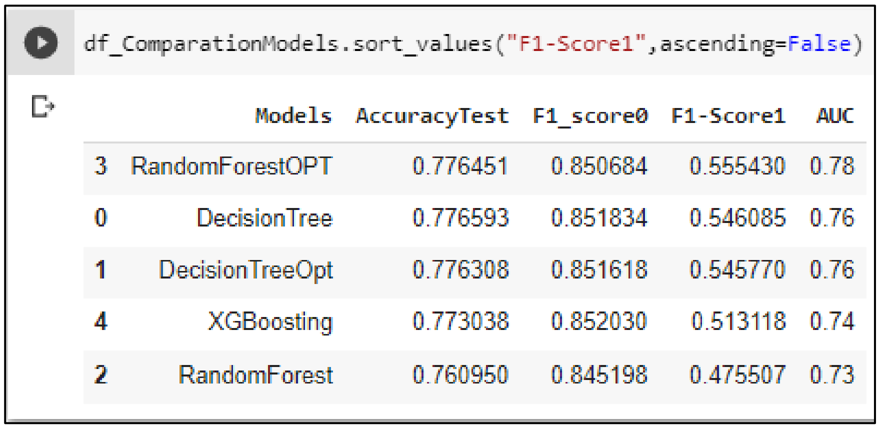
The way to measure performance in these binary classification models is to calculate precision and recall. In this section, the report shows the results of the classification models by accuracy, precision, and recall.

Figure 78



```
[139] df_ComparationModels=pd.DataFrame({"Models":["DecisionTree","DecisionTreeOpt","RandomForest","RandomForestOPT","XGBoosting"],
                                         "AccuracyTest":[Accuracy_Test_AD,Accuracy_Test_AD_Optimo,Accuracy_RF_test,Accuracy_RF_O_test,Accuracy_GBC_
                                         "F1_score0":[f1_Score_AD_0,f1_Score_AD_Op_0,f1_Score_RF_0,f1_Score_RF_O_0,f1_Score_GBC_O_0],
                                         "F1-Score1":[f1_Score_AD_1,f1_Score_AD_Op_1,f1_Score_RF_1,f1_Score_RF_O_1,f1_Score_GBC_O_1],
                                         "AUC":[AUC_AD,AUC_AD_OP,AUC_RF,AUC_RFO,AUC_GB1]})
```

Figure 79

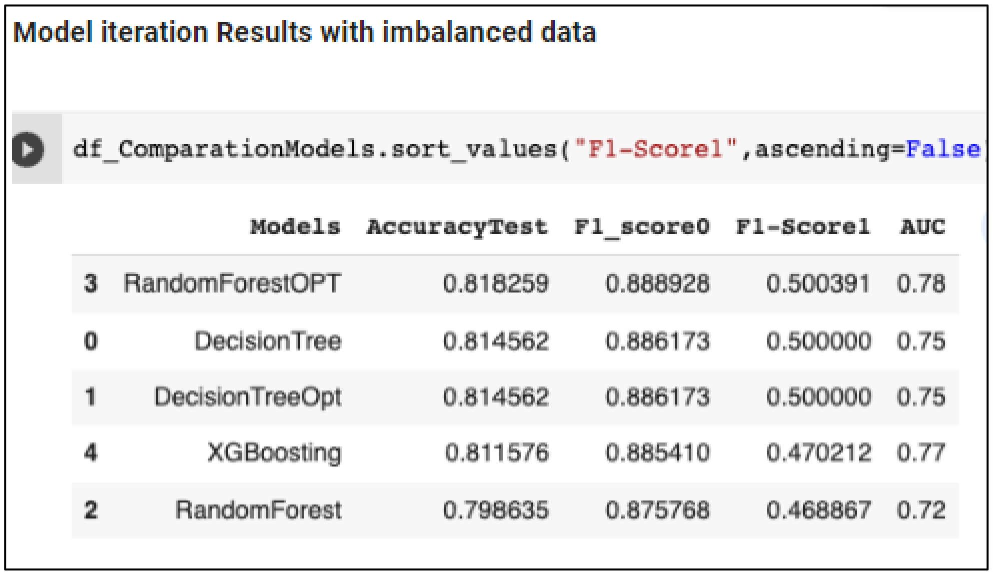


```
df_ComparationModels.sort_values("F1-Score1",ascending=False)
```

	Models	AccuracyTest	F1_score0	F1-Score1	AUC
3	RandomForestOPT	0.776451	0.850684	0.555430	0.78
0	DecisionTree	0.776593	0.851834	0.546085	0.76
1	DecisionTreeOpt	0.776308	0.851618	0.545770	0.76
4	XGBoosting	0.773038	0.852030	0.513118	0.74
2	RandomForest	0.760950	0.845198	0.475507	0.73

Before validating the imbalance data, the accuracy, precision, and recall had lower results. For the model selection F1-Score1 had to show the best results.

Figure 80



Model iteration Results with imbalanced data

```
df_ComparationModels.sort_values("F1-Score1",ascending=False
```

	Models	AccuracyTest	F1_score0	F1-Score1	AUC
3	RandomForestOPT	0.818259	0.888928	0.500391	0.78
0	DecisionTree	0.814562	0.886173	0.500000	0.75
1	DecisionTreeOpt	0.814562	0.886173	0.500000	0.75
4	XGBoosting	0.811576	0.885410	0.470212	0.77
2	RandomForest	0.798635	0.875768	0.468867	0.72

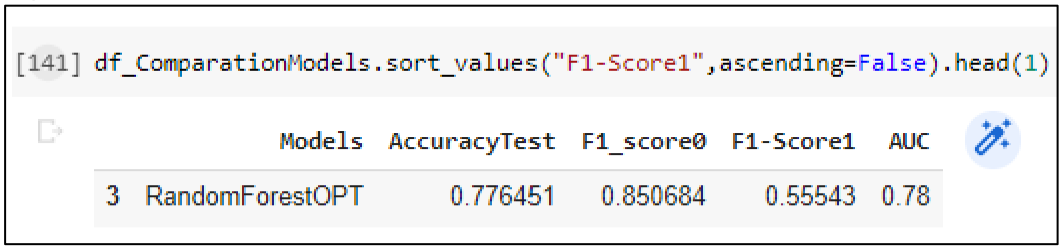
Model Recommendation

22. Model Selection

The best model according to F1-Score is the Optimized Random Forest

The table shows the accuracy test, F1Score for clients who pay on time their obligations and F1-Score1 the clients who will fail to pay their obligations.

Figure 81



```
[141] df_ComparationModels.sort_values("F1-Score1",ascending=False).head(1)
```

	Models	AccuracyTest	F1_score0	F1-Score1	AUC
3	RandomForestOPT	0.776451	0.850684	0.55543	0.78

23. Model Theory

To measure the model’s performance should reflect the accuracy with which the model classifies the credit card clients into two categories, the ones who are going to comply with their debt’s payments and the ones who will fail to pay their debt. The way to measure the performance of these binary classification models was calculating the precision and recall. In this model precision is defined as the number of correctly predicted default of credit card clients by the predicted number of defaults of credit card clients; while recall is defined as the number of correctly predicted default of credit card clients divided by the actual number of defaults of credit card clients. Precision is meant to gauge the number of false positives (clients predicted to fail in their debt’s payment) while recall gauges the number of false negatives (clients predicted to stay current that actually went into default).

It was also considered one statistic that combine precision and recall, the F-Score. The F-score is described as the harmonized mean of precision and recall, and attributes elevated rates to techniques that reach a sufficient balance between precision and recall.

24. Model Assumptions and Limitations

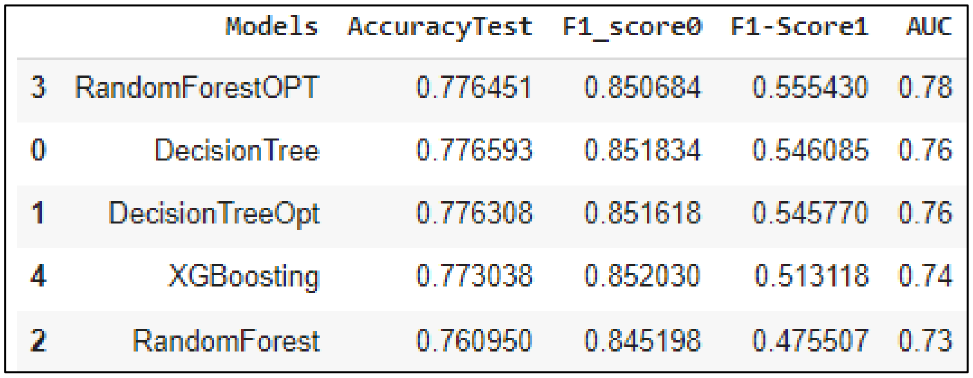
As model limitations we defined the lack of macroeconomic information that could possibly affect the models results, as inflation, unemployment rates, financial crisis, among other key economic factors.

Additionally, to build the model there were not variables related to credit score information of the clients, from other banks, and credit score’s institutions. Despite this data was nor available and was not part of the model, the precision and accuracy of the models presented satisfactory results.

25. Model Sensitivity

Below the model sensitivity is shown as F1-Score1 was the chosen indicator to measure the accuracy and recall of the best model. Random Forest model was proved to be the best model. The accuracy of positive and negative predictions is 0.77. F1 Score: F1_Score0: Fraction of clients paying on time that were correctly identified is 0.85. and F1-Score1 Fraction default clients that were correctly identified is 0.555

Figure 82



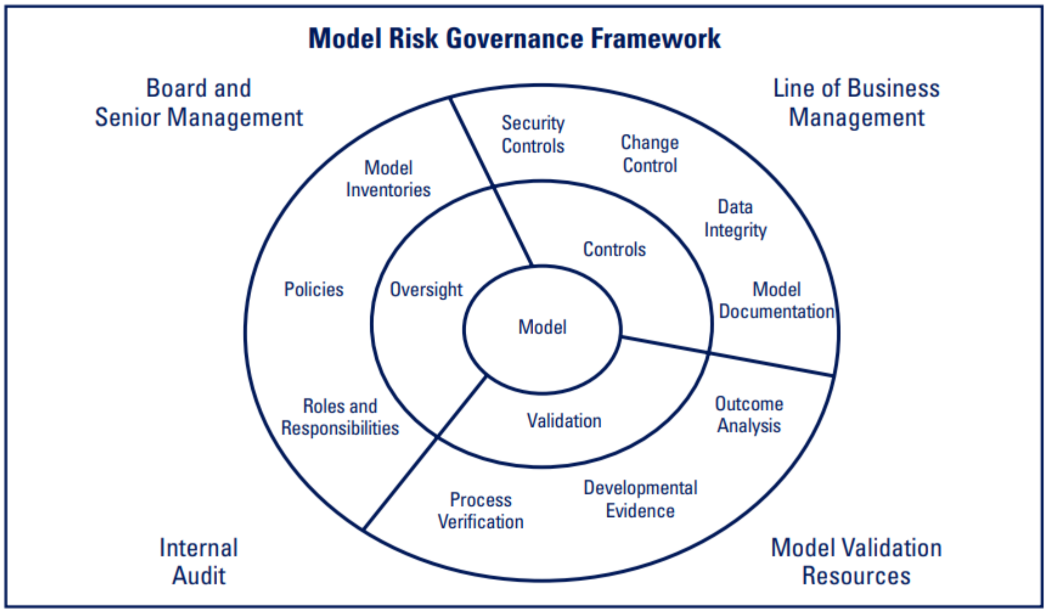
	Models	AccuracyTest	F1_score0	F1-Score1	AUC
3	RandomForestOPT	0.776451	0.850684	0.555430	0.78
0	DecisionTree	0.776593	0.851834	0.546085	0.76
1	DecisionTreeOpt	0.776308	0.851618	0.545770	0.76
4	XGBoosting	0.773038	0.852030	0.513118	0.74
2	RandomForest	0.760950	0.845198	0.475507	0.73

Validation and Governance

26. Model Risk Governance Framework

Business administration will establish period model validations as processes and activities that verify and provide reasonable assurance the model is performed as intended and is a core element of Model Risk Management (MRM). The model evaluation will be taken to the Basel’s Committee and Credit Risk monthly committee. Annually the variables review will be taken Board of Directors meeting with the statistics and completed review, see figure 83 below, (FDIC, 2005). Model performance and stability will be reviewed and testing of outputs against outcomes (Back testing of the real and hypothetical portfolio).

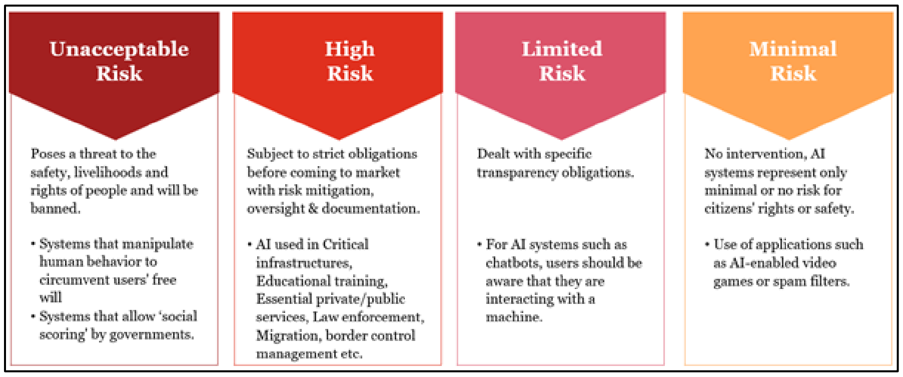
Figure 83



Model Security and change control procedures will be stablished in the supervisory framework. Control and validation, also password protection and double control protection will be required. This will ensure Protection of PII or any personal data. Validators require to assess the model libraries, and if configurations are suitable for the operation while pondering the prospective affects from potential release.

The framework for validating the IA Model a risk indicator should be defined to quantify any potential damage/ impair in the model. As it is indicated by the European Commission by publishing a guideline for the risk tiering associated to the model (Towards Data Science, 2021) , see Figure 84 below:

Figure 84



Unacceptable Risk	High Risk	Limited Risk	Minimal Risk
Poses a threat to the safety, livelihoods and rights of people and will be banned.	Subject to strict obligations before coming to market with risk mitigation, oversight & documentation.	Dealt with specific transparency obligations.	No intervention, AI systems represent only minimal or no risk for citizens' rights or safety.
• Systems that manipulate human behavior to circumvent users' free will • Systems that allow 'social scoring' by governments.	• AI used in Critical infrastructures, Educational training, Essential private/public services, Law enforcement, Migration, border control management etc.	• For AI systems such as chatbots, users should be aware that they are interacting with a machine.	• Use of applications such as AI-enabled video games or spam filters.

Model supervisory review will include a robust validation as:

- Stablishing and document a validation framework, with policies, processes, roles and responsibilities, functions, and activities to the model validation
- Designing and stablishing the type of analysis to validate de model. Validators should evaluate the bias & objectivity properly centered on above stated risk tiering.
- Back testing as data quality, comparing real portfolio with hypothetical one, completeness and integrity of the data, sources, ensure sensitivity analysis has been thoroughly performed).
- The validation should be independent, by an stablished Unit testing, with at least monthly frequency. The initial and periodic reviews should be communicated and documented.
- Report of findings, problems and findings and any issues identify in the model.
- Follow up and amending any issues identified related to the model. Any founding should be monitoring, escalated in necessary cases, and solved to warranty the model operates in timely manner and maintain effectiveness.
- Periodic reviews performed by the Internal Auditor for accuracy and output report and compliance with regulatory reporting.
- All parts involved should follow Code of Conduct of the company and internal policies.

- Training policies should be implemented, and training courses completed at least each 6 months. Evidence of completion and evaluation with at least 80% of approval should be documented and stored.
- Credit Risk Management unit will assure the compliance of all Regulatory Reports and regulations related to the Model.

27. Variable Level Monitoring

The Variables will be validated according to the values of variables in the existing performance environment. If the environment changes in the course of performance and the variable board will be replaced with a table suitable to the latest environment.
Missing variables part of the model will be allowed if the accuracy and precision don't decrease or decrease maximum 0.4.
The variables used in the model weren't identified as stationary.

28. Variable Drift Monitoring

Model drift is the fact that the statistical developments have changed, in the data from a trained model, the target variable and the independent variables relationship changes with time and the model starts to become unstable and the predictions keep on developing inaccurate over time.

Due to this model was created for the financial industry it will be continuously re-fit and re-developed to mitigate the risks associated with drift. Also, data will be weighted, and the necessary parameters will be incorporated to give more weight to the most recent transactions; patterns of seasonality will be checked, and the model will be re-trained according to the identified seasons; Tolerance for drift of each variable will be defined and checked monthly.
The re-train process will be completed as necessary, it could be weekly, monthly, quarterly, etc.

28.1. Model Health & Stability

Model monitoring describes the process of tracking the performance of the models in production environment. It allows to identify bad quality predictions and poor technical performance, data distribution changes, data quality issues. As a result, your machine learning models deliver the best performance.

The method to measure stability and drift will be Kolmogorov–Smirnov test (KS Statistic)).

Input data: the data and files in the pipeline will be reviewed in its schema, and completeness Data validation tests and checks will be operating with monitoring any changes or alerts in the data.

Output data: the output of the model accuracy will be reviewed.

Data quality metrics will be tracked to as mean, standard deviation, correlations, and distance metrics (KL divergence, Kolmogorov-Smirnov statistic). The statistical system of measurement used will be mainly dependent on the dimension of data expected.

Conclusion and Recommendations

29. Impacts on Business Problem

The best model according to F1-Score is the Optimized Random Forest

The table shows the accuracy test, F1Score for clients who pay on time their obligations and F1-Score1 the clients who will fail to pay their obligations.

30. Recommended Next Steps

Process this model every month with updated data of the credit card's clients.

Using this model administration will have better and effective credit risk management for predicting clients who will fail on the credit card obligations and creating appropriate collection policies.

Adequate and timely training to the analysts who are running the model, to interpret results effectively.

References

31. References

CFI. (2021, August 28). Retrieved from corporatefinanceinstitute: https://corporatefinanceinstitute.com/resources/careers/companies/top-banks-in-taiwan/

FDIC. (2005). Model Governance. *FDIC*, 8.

Towards Data Science. (2021, July 7). *Towards Data Science*. Retrieved from https://towardsdatascience.com/ai-ml-model-validation-framework-13dd3f10e824

UCI Machine Learning Repository. (2016, 01 26). (S. o. University of California, Producer) Retrieved from https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients: https://www.kaggle.com/datasets/uciml/default-of-credit-card-clients-dataset