Credit Card Approval Prediction

Diana Paola Vega Feriz

Centennial College

Business Analytics Capstone BA723

Bilal Hasanzadah

August 12, 2022

Contents

Summary

Credit card have become an essential part of everyday life for most people around the world and have completely changed the management of personal finances. According to Shift Credit Card Processing (2021) as per August 2021 data globally estimated 2.8 billion credit cards were in use out of those numbers only United States of America has 1.06 billion in use. Almost without realizing it, each American accumulates an average of 4 different credit cards in his wallet.

Credit cards have become a common payment tool, however the rise in accumulated debts have led issuing companies to an increment of its credit risk and default provisions. Considering that, the process of approval has become more relevant than ever, but still being in many cases manual, time-consuming, and inaccurate impacting negatively businesses.

For that reason, this analysis provides a set of models to develop an accurate analytical tool to derive the drivers for efficient credit card approval and thus decrease the financial loss by not missing out on good candidates and avoid the increased credit risk by approving a bad candidate.

The statistical analysis has been constructed in Python, using a public domain dataset of credit card approvals with demographic and socio-economic variables such as gender, age, income, education, household members, and credit score. Different statistical modeling tools, including random forest, logistic regression, neural network, and gradient boosting classifier have been applied. The model comparison has shown that according to the validation accuracy of each model and the ROC/AUC, the model with the best performance is logistic regression.

Using the logistic regression model, I was able to predict 82.8% accuracy credit card approvals. The key drivers for the successful prediction are Total_Good_Debt,

Applicant_Gender_F, Owned_Realty, Education_Type_Secondary/secondary special, and Income_Type_Working. Overall, it can be noted that having a good debt has the greatest positive impact on the status of approval.

The ultimate recommendation is to collect more data specifically from people who have been rejected for a credit card and moreover, from men, students, pensioners, who do not owned realty, with study level different from secondary, not married, with low income, children, and bad debt so that the dataset will be more balanced thus new different algorithms and techniques could be applied to enhance the performance of the model.

Introduction

1.0 Background

When it comes to evaluating the big risks that can impact the net income and profit of companies there can be a big list but the most important ones are credit, operational, market, and liquidity risk. Organizations and more specifically financial institutions must adhere to government rules and must have well-built risk management infrastructures to reduce the potential risks.

According to The Investopedia Team (2022) credit risk - the biggest risk faced by financial institutions - is a possibility of loss due to borrower's failure to make loan payments or fulfil contractual commitments. It typically refers to the possibility that a lender will not get the principal and interest that is owed, which would disrupt cash flows and raise collection costs. Even while it is impossible to predict who exactly may miss payments, correctly evaluating and managing credit risk can decrease the impact of a loss. A lender's or creditor's interest payments are made from a debt obligation's issuer or borrower.

Consumers credit risk may vary depending on the institution and criteria applied for credit approval but mainly driven by key factors like credit history or score, capacity to repay, capital, the credits conditions, and associated collaterals.

The organization generally have specific department and teams which are responsible for evaluating the credit risk for the current and future clients. In the process, the businesses now have created the capacity to swiftly review the data to determine the risk profile of a consumer added by the technology.

According to Shift Credit Card Processing (2021) as per August 2021 there were 2.8 billion credit cards in use worldwide of which 1.06 billion were in use in United States of America. In research rom TheGlobalEconomy.com (2022) Canada accounted in 2017 the highest percentage of people aged 15+ who have a credit card with 82.58%. Currently there are 4 major credit card companies, Visa, Mastercard, Discover, and American Express being the two first the biggest.

It is not possible for the companies to be fully protected from credit risk as issuing credit cards is part of their core business model though the exposure can be reduced in multiple ways, for example by issuing credit to people with good credit history, manage transactions with proven parties, or ensuring collaterals as back to up to the credits.

2.0. Problem Statement

Taking into account the high and manual steps involve in credit card approvals such as a wrong approval or missing a right candidate, can both have significant negative financial impact for the company.

The objective is to develop a right analytical tool to ensure efficient credit card approval process and therefore, decrease the financial loss by missing to select a good candidate and on the other hand increasing the credit risk by approving a bad candidate.

Considering the high-risk exposure involved for the issuing companies with the credit approval process it is important that there is a well-defined and well-constructed risk management infrastructure to proper control mechanism in place, particularly when it comes to underwriting.

According to The Investopedia Team (2022) underwriting describes the procedure used by financial institutions to evaluate the risk and creditworthiness of applicants. The underwriting procedure takes into account some of the key considerations like credit score, income, credit history, and age, this list is not limited and the specifics can differ significantly from company to company.

Depending on the organization the underwriting procedures may be stricter than others, requiring customers to have longer credit histories, lower debt-to-income ratios, and higher earnings.

When it comes to deciding on whether to approve a credit card application, issuing companies take a big risk. One of the unsecured debts includes credit card debt as there are no collaterals secured loans, such as a mortgage or auto loan.

As a conclusion, the underwriting procedure is incredibly complicated but is constantly becoming automated utilising technology. The use of data science can be helpful to scan detailed customer behavior and provide insight on borrowing and spending behavior of each individual customer and subsequently providing basis to understand the credit authorization needs. The use of analytics has already shown how it can impact the financial performance of the organizations

and everyday it is even more viable to achieve consistent financial growth through the application of data-driven insights.

3.0. Objectives & Measurement

The objective of this project is to build a classification model that predicts if an applicant is a “good” or “bad” client thus issue a credit card or not. This model will impact the business performance of the issuing companies since by making an accurate assessment they will reduce the credit risk and losses caused by the customer´s default and increase earnings by issuing credit cards to those who are good candidates.

The metrics used in this project to measure its success are meant to assess each model for its accuracy which need to be greater than 85% to be considered. The model to be labeled a good model only if it is beyond the required accuracy, the ROC of each one will be assessed, this being the decision factor.

4.0. Assumptions and Limitations

To use the dataset, it was assumed that the data has not omitted any relevant information from the original tables when cleaning. Moreover, it is assumed that the results in the credit scores are accurate and well transformed. Besides, it is also assumed that this dataset is representative of the status quo.

Data Sources

5.0. Data Set Introduction

The present study is a prediction of credit card approval in which it is determined if an applicant is a “good” or “bad” client with reference to the impact on the business performance. Based on this, the “Credit Card Approval Prediction (Cleaned Version)” dataset, owned by Mario Caesar has been used to carry out this analysis. This dataset is a cleaned version of two

original datasets created by Seanny in 2020. The first dataset contains application records and the second one credit records. According to Caesar (n.d.), he merged, cleaned, and transformed these two tables using Pentaho Data Integration. The cleaned version used in this analysis has been downloaded from the Kaggle website.

6.0. Exclusions

For this analysis it is not necessary to make any exclusions due to the fact that the dataset used is an already cleaned version from the original. Moreover, initial data preparation is not required.

7.0. Data Dictionary

The dataset contains 25,128 observations and 21 variables related to the demographic and socio-economic information of the applicants as shown as follow:

Feature name	Explanation
Applicant_ID	Client ID
Applicant_Gender	Gender Male – Female
Owned_Car	The client own a car 0-1
Owned_Realty	The client own a property 0 -1
Total_Children	Number of children
Total_Income	Annual income
Income_Type	Income category
Education_Type	Education level
Family_Status	Marital status
Housing_Type	Way of living

Owned_Mobile_Phone	The client own a mobile phone 0 -1
Owned_Work_Phone	The client has work phone 0 -1
Owned_Phone	The client has phone 0 – 1
Owned_Email	The client has email 0 – 1
Job_Title	Occupation type
Total_Family_Members	Family size
Applicant_Age	Age
Years_of_Working	Years of working
Total_Bad_Debt	Score of bad debt
Total_Good_Debt	Score of good debt
Status	Status of the credit card approval Target: 0: No – 1:Yes

Data Exploration

8.0. Data Exploration Techniques

Diversity of techniques have been used to do the data exploration analysis of this dataset which include variety of visualizations but also some functions of the different libraries.

8.1. Load Data and Data Structure

This analysis has been made entirely in Python which is a high-level programming language and one of the widely used worldwide. Also, different machine learning libraries such as Pandas, NumPy, Matplotlib, and Scikit-Learn have been used quite often. Refer to Figure 1.

The data exploration starts by loading the dataset and reviewing its shape as well as the types of the variables. It is observed that the dataset contains 25,128 rows and 21 columns of

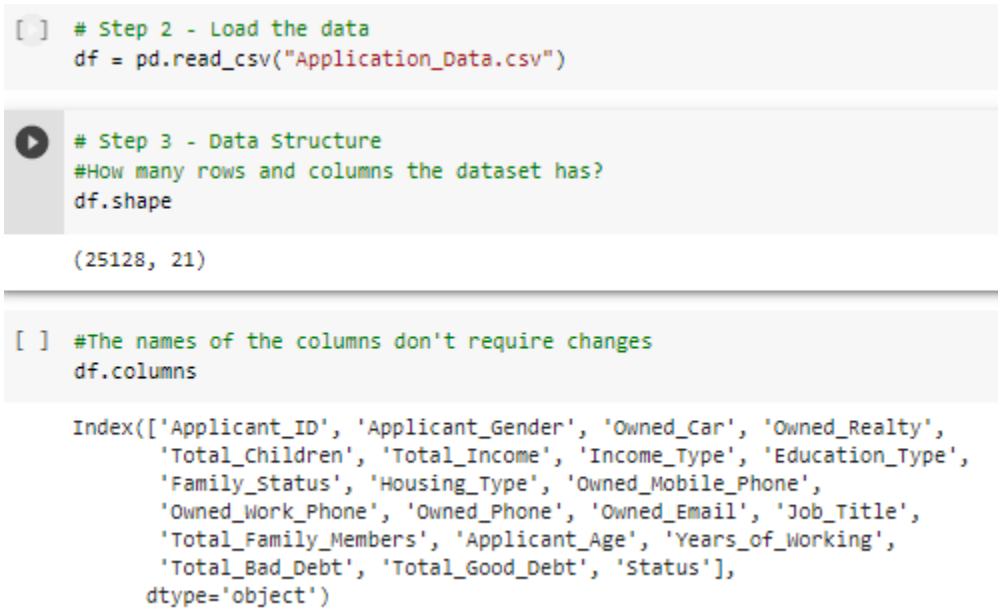
which 15 are stored as integer and 6 as object. The label of the 6 object variables have been changed to “Category” since they are categorical variables. “Refer to Figure 2 and 3.

Figure 1: Packages installed

```
# STEP 1 - Load the packages
!pip install dmba
!pip install imbalanced-learn
!pip install mord

import pandas as pd
import numpy as np
from numpy import mean

from sklearn import metrics
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV, KFold
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.neural_network import MLPClassifier, MLPRegressor
from sklearn.metrics import accuracy_score, recall_score, confusion_matrix, precision_score, f1_score, classification_report
from sklearn.linear_model import LogisticRegression, LogisticRegressionCV
from sklearn.neural_network import MLPClassifier

import matplotlib.pylab as plt
from dmba import plotDecisionTree, classificationSummary, gainsChart, liftChart
import statsmodels.formula.api as sm
import statsmodels.api as sm
import seaborn as sns
from scipy.stats import norm
from imblearn.over_sampling import SMOTE
from collections import Counter
from mord import LogisticIT
from dmba.metric import AIC_score

%matplotlib inline
from pathlib import Path
```

Figure 1. Packages installed

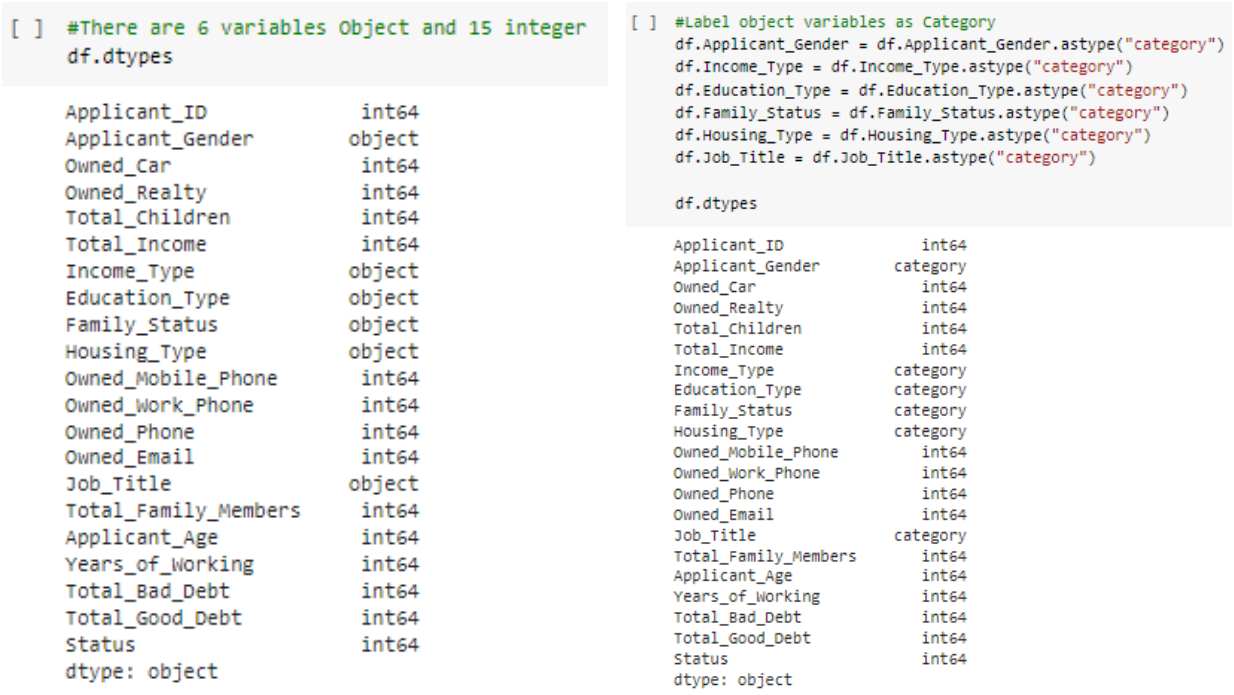
Figure 2: Load data and Data structure



```
[ ]  # Step 2 - Load the data
     df = pd.read_csv("Application_Data.csv")

     # Step 3 - Data Structure
     #How many rows and columns the dataset has?
     df.shape

     (25128, 21)

[ ]  #The names of the columns don't require changes
     df.columns

     Index(['Applicant_ID', 'Applicant_Gender', 'Owned_Car', 'Owned_Realty',
            'Total_Children', 'Total_Income', 'Income_Type', 'Education_Type',
            'Family_Status', 'Housing_Type', 'Owned_Mobile_Phone',
            'Owned_Work_Phone', 'Owned_Phone', 'Owned_Email', 'Job_Title',
            'Total_Family_Members', 'Applicant_Age', 'Years_of_Working',
            'Total_Bad_Debt', 'Total_Good_Debt', 'Status'],
           dtype='object')
```

Figure 2: Load data and Data structure

Figure 3: Data types before and after changes



```
[ ]  #There are 6 variables Object and 15 integer
     df.dtypes

     Applicant_ID             int64
     Applicant_Gender        object
     Owned_Car                int64
     Owned_Realty             int64
     Total_Children           int64
     Total_Income             int64
     Income_Type             object
     Education_Type          object
     Family_Status           object
     Housing_Type            object
     Owned_Mobile_Phone       int64
     Owned_Work_Phone         int64
     Owned_Phone              int64
     Owned_Email              int64
     Job_Title               object
     Total_Family_Members     int64
     Applicant_Age            int64
     Years_of_Working         int64
     Total_Bad_Debt           int64
     Total_Good_Debt          int64
     Status                   int64
     dtype: object

[ ]  #Label object variables as Category
     df.Applicant_Gender = df.Applicant_Gender.astype("category")
     df.Income_Type = df.Income_Type.astype("category")
     df.Education_Type = df.Education_Type.astype("category")
     df.Family_Status = df.Family_Status.astype("category")
     df.Housing_Type = df.Housing_Type.astype("category")
     df.Job_Title = df.Job_Title.astype("category")

     df.dtypes

     Applicant_ID             int64
     Applicant_Gender      category
     Owned_Car                int64
     Owned_Realty             int64
     Total_Children           int64
     Total_Income             int64
     Income_Type           category
     Education_Type        category
     Family_Status         category
     Housing_Type          category
     Owned_Mobile_Phone       int64
     Owned_Work_Phone         int64
     Owned_Phone              int64
     Owned_Email              int64
     Job_Title             category
     Total_Family_Members     int64
     Applicant_Age            int64
     Years_of_Working         int64
     Total_Bad_Debt           int64
     Total_Good_Debt          int64
     Status                   int64
     dtype: object
```

Figure 3: Data types before and after changes

8.2. Quality of Data

The next step is to check the quality of the dataset data by looking for duplicated customer IDs and missing and NA values. If the dataset has missing values, an imputation of those values should be done. Missing values will be replaced by a constant value or any statistics like mean or median. In this case, none of the mentioned kind of values have been found. Refer to Figure 4 and 5.

Figure 4: Data quality – Duplicated IDs

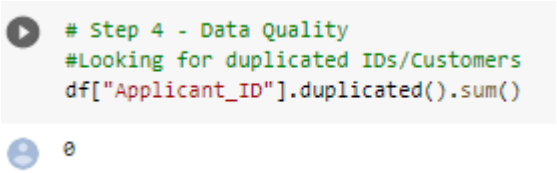


Figure 4: Data quality – Duplicated IDs

Figure 5: Missing and NA values

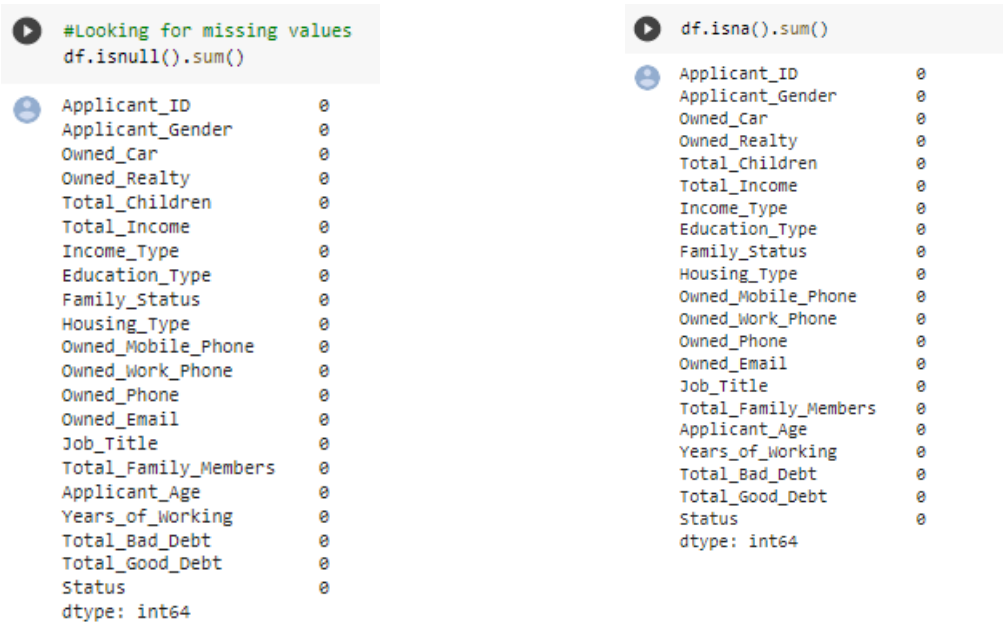


Figure 5: Missing and NA values

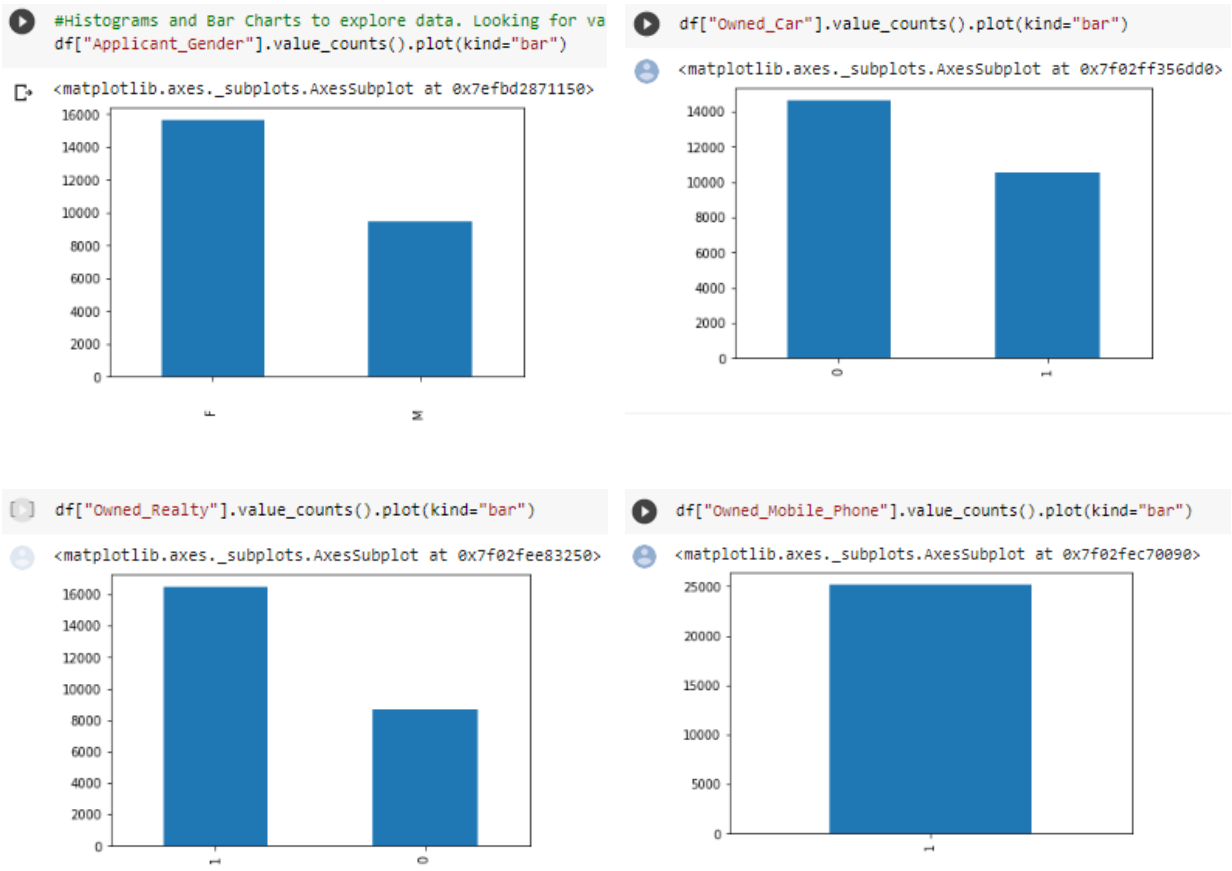
8.3. Target

The binary variable Status has been selected as target since it shows whether a credit card has been approved or not which is the main objective of this project.

8.4. Data Visualizations

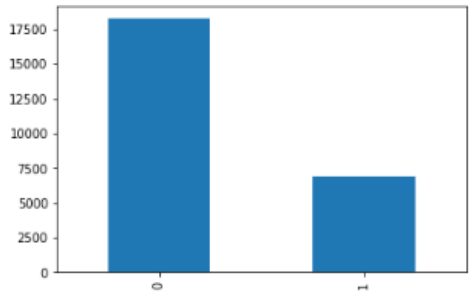
Visualizations are used to get a better understanding of the distribution of the variables. In case of the categorical variables bar charts are plot while for interval variables histograms are used. The main insight from this exploration is the high class imbalanced the dataset has. Variables Owned_Mobile_Phone, Owned_Email, Status, and Housing_Type have more than 85% of concentration of one single class. Refer to Figure 6.

Figure 6: Visualization of variable distribution



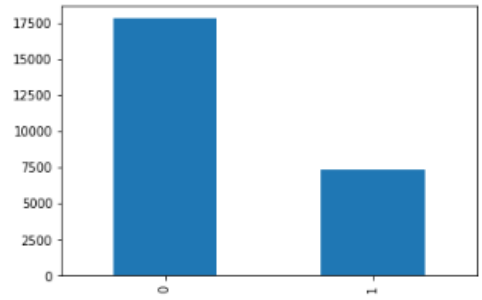
```
df["Owned_Work_Phone"].value_counts().plot(kind="bar")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f02feadb210>
```



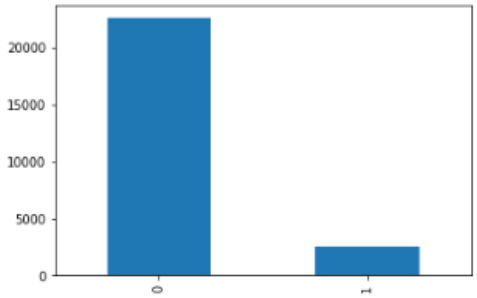

```
[ ] df["Owned_Phone"].value_counts().plot(kind="bar")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f02fea51c10>
```



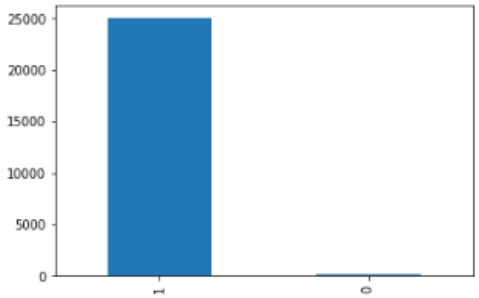

```
[ ] df["Owned_Email"].value_counts().plot(kind="bar")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f02fea45810>
```



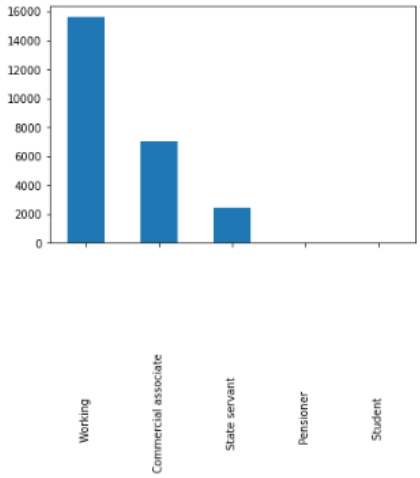

```
df["Status"].value_counts().plot(kind="bar")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f03000468d0>
```



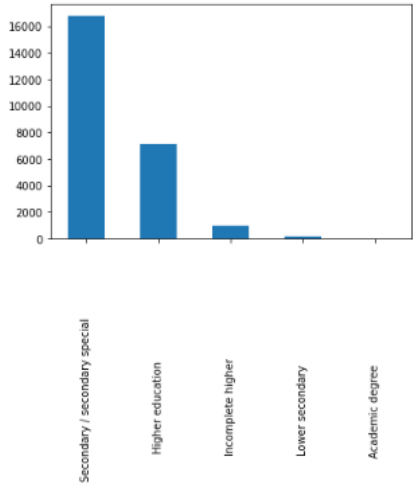

```
df["Income_Type"].value_counts().plot(kind="bar")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f02feca7c90>
```



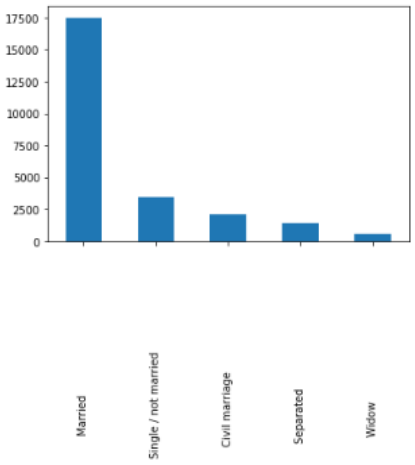

```
[ ] df["Education_Type"].value_counts().plot(kind="bar")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f02feda2610>
```



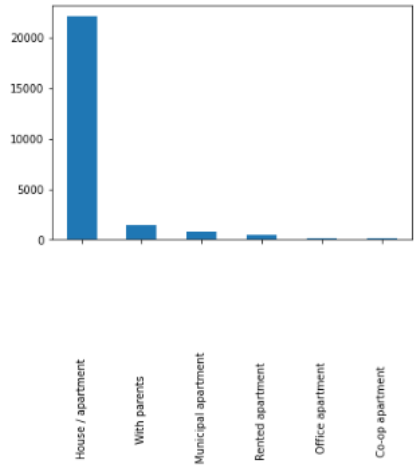
```
df["Family_Status"].value_counts().plot(kind="bar")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f02fec7da10>
```



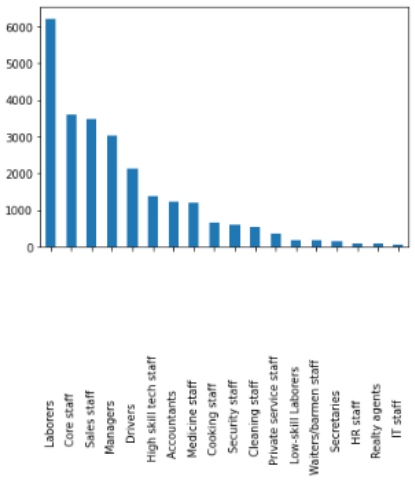

```
df["Housing_Type"].value_counts().plot(kind="bar")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f02fec42bd0>
```

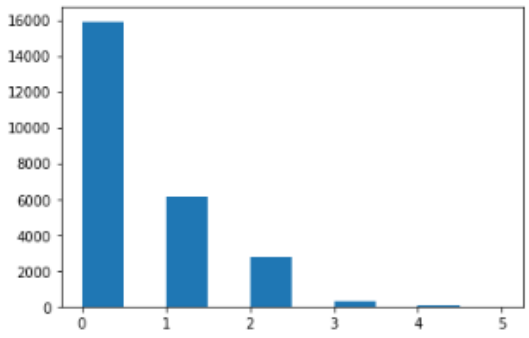



```
df["Job_Title"].value_counts().plot(kind="bar")
```

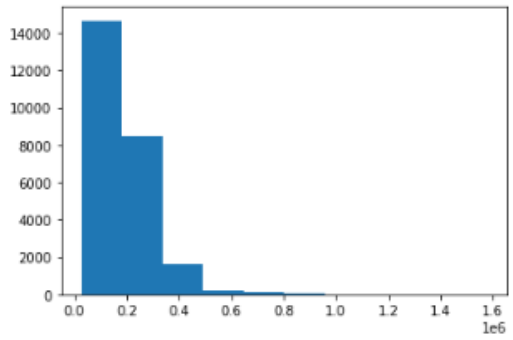
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f02fe9a0b50>
```



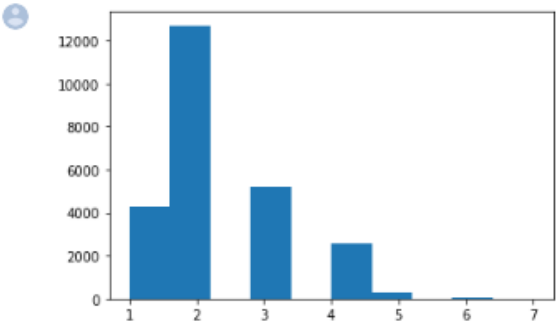

```
[ ] fig, ax=plt.subplots()
    ax.hist(df["Total_Children"])
    plt.show()
```



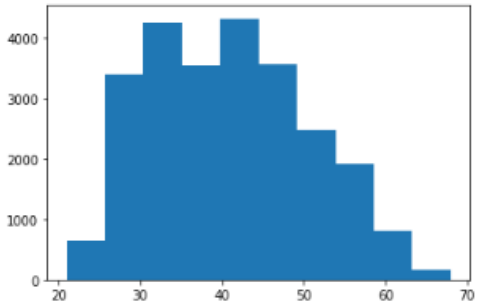
```
[ ]  fig, ax=plt.subplots()
     ax.hist(df["Total_Income"])
     plt.show()
```



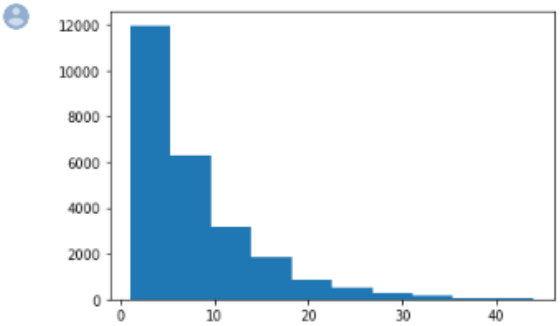

```
fig, ax = plt.subplots()
ax.hist(df["Total_Family_Members"])
plt.show()
```



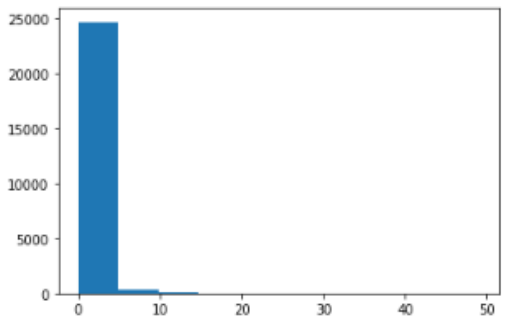

```
fig, ax = plt.subplots()
ax.hist(df["Applicant_Age"])
plt.show()
```




```
fig, ax = plt.subplots()
ax.hist(df["Years_of_Working"])
plt.show()
```




```
fig, ax = plt.subplots()
ax.hist(df["Total_Bad_Debt"])
plt.show()
```




```
[ ]  fig, ax = plt.subplots()
     ax.hist(df["Total_Good_Debt"])
     plt.show()
```

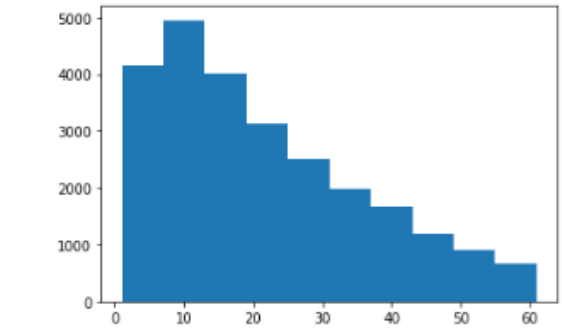


Figure 6: Visualization of variable distribution

8.5. Statistics

Data describe is a useful function to know the main statistics of the variables as it computes and displays the statistics summary. Additionally, this function can be set for categorical variables so it shows mode and frequency. Refer to Figure 7 and 8.

Figure 7: Central tendency and variation

```
[ ] # Step 5 - Variable Properties
    #Central tendency and variation.

    df.describe()
```

	Applicant_ID	Owned_Car	Owned_Realty	Total_Children	Total_Income	Owned_Mobile_Phone	Owned_Work_Phone	Owned_Phone	Owned_Email	Total_Family_Members
count	2.512800e+04	25128.000000	25128.000000	25128.000000	2.512800e+04	25128.0	25128.000000	25128.000000	25128.000000	25128.000000
mean	5.078835e+06	0.418378	0.654927	0.509472	1.948365e+05	1.0	0.273758	0.292741	0.100684	2.291309
std	4.194378e+04	0.493303	0.475402	0.762937	1.045211e+05	0.0	0.445895	0.455030	0.300916	0.928871
min	5.008806e+06	0.000000	0.000000	0.000000	2.700000e+04	1.0	0.000000	0.000000	0.000000	1.000000
25%	5.042226e+06	0.000000	0.000000	0.000000	1.350000e+05	1.0	0.000000	0.000000	0.000000	2.000000
50%	5.079004e+06	0.000000	1.000000	0.000000	1.800000e+05	1.0	0.000000	0.000000	0.000000	2.000000
75%	5.115603e+06	1.000000	1.000000	1.000000	2.250000e+05	1.0	1.000000	1.000000	0.000000	3.000000
max	5.150487e+06	1.000000	1.000000	5.000000	1.575000e+06	1.0	1.000000	1.000000	1.000000	7.000000

Figure 7: Central tendency and variation

Figure 8: Describe for categorical variables

```
df.describe(include="category")
```

	Applicant_Gender	Income_Type	Education_Type	Family_Status	Housing_Type	Job_Title
count	25128	25128	25128	25128	25128	25128
unique	2	5	5	5	6	18
top	F	Working ...	Secondary / secondary special ...	Married ...	House / apartment ...	Laborers ...
freq	15627	15616	16802	17507	22096	6211

Figure 8: Describe for categorical variables

8.6. Kurtosis, Skewness, and Outliers

Moreover, kurtosis, skewness, and outliers are also assessed, being Total_Bad_Debt, Total_Income, Owned_Email, Years_of_Working, and Total_Children the variables that show the greatest right-tailed skewness and outliers. Refer to Figure 9, 10, and 11.

Figure 9: Skewness

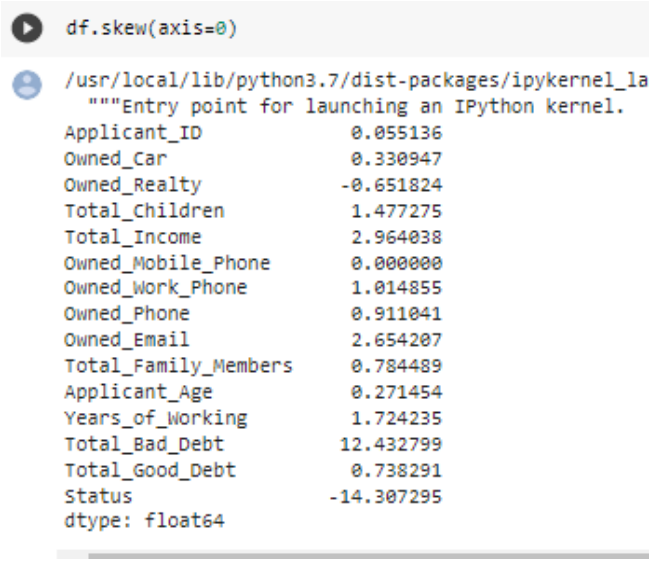


Figure 9: Skewness

Figure 10: Kurtosis

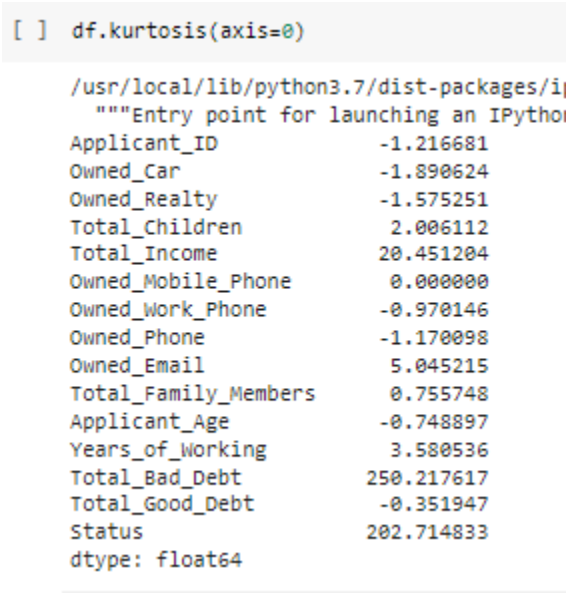


Figure 10: Kurtosis

Figure 11: Outliers

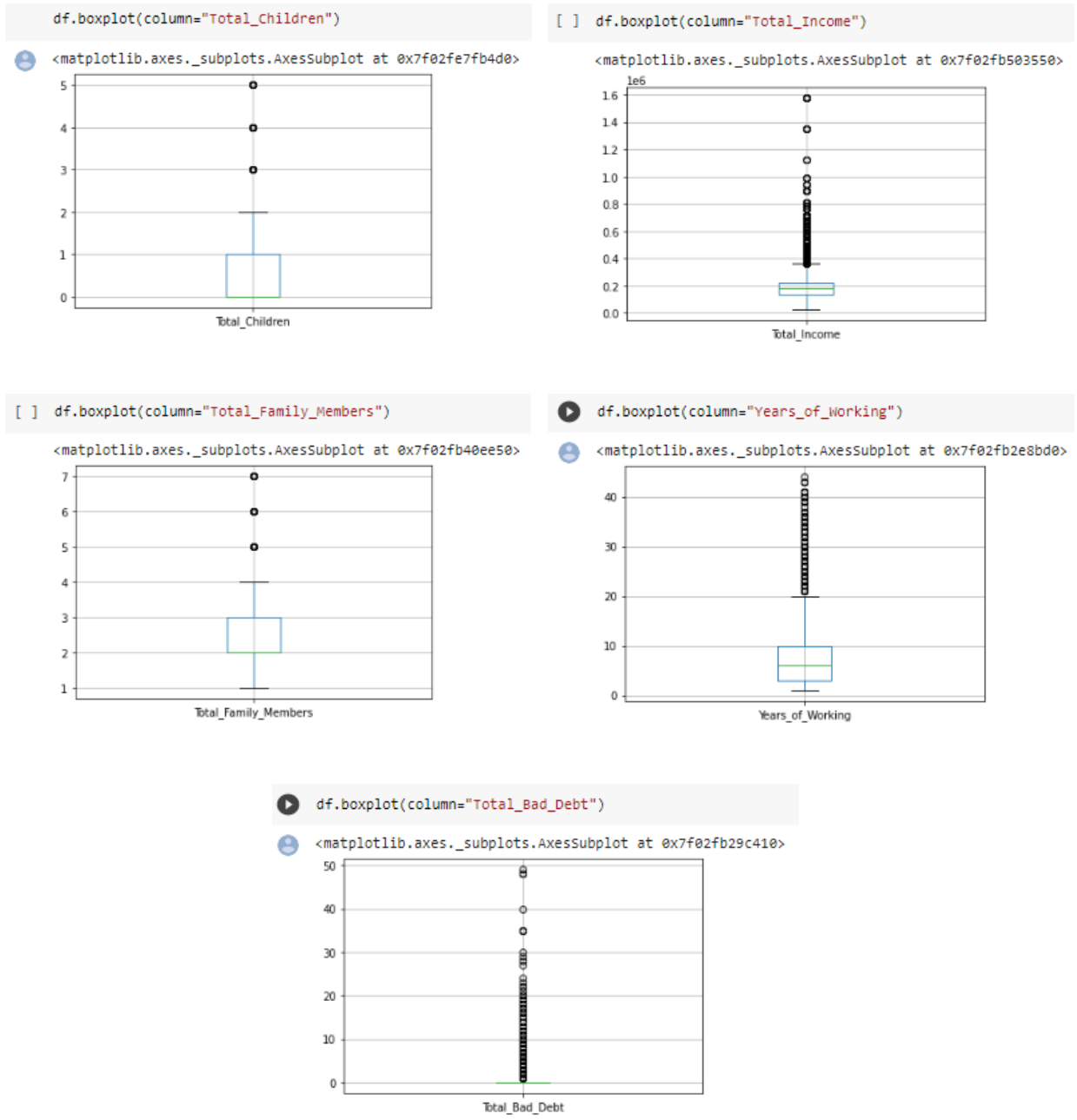


Figure 11: Outliers

8.7. Correlation

Finally, a case of strong correlation and multicollinearity has been found in the heatmap between Total_Children and Total_Family_Members variables with 0.9 and a moderate correlation between variables Total_Bad_Debt and Status with -0.5. Refer to Figure 12 and 13.

Figure 12: Correlation heatmap

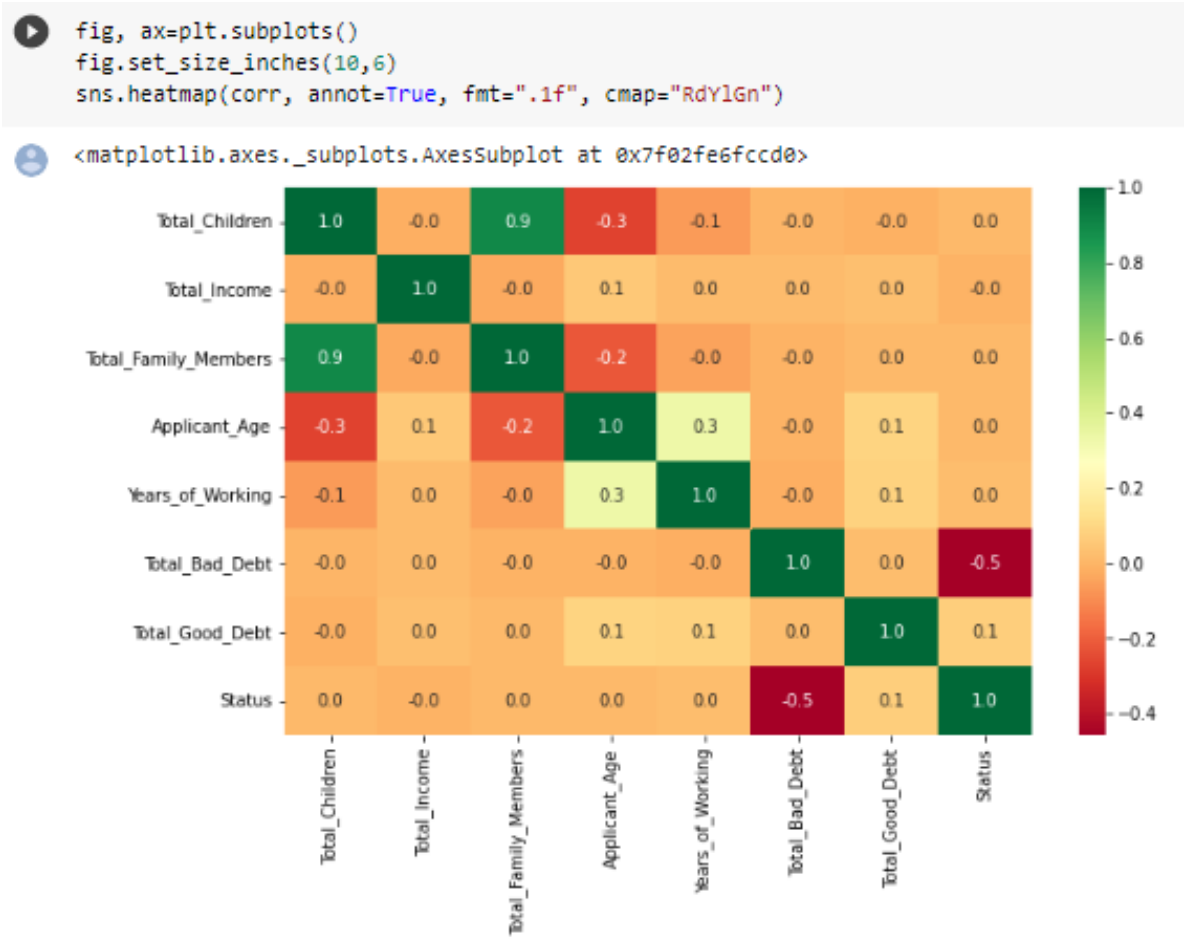


Figure 12: Correlation heatmap

Figure 13: Scatterplot - Total_Family_Member and Total_Children

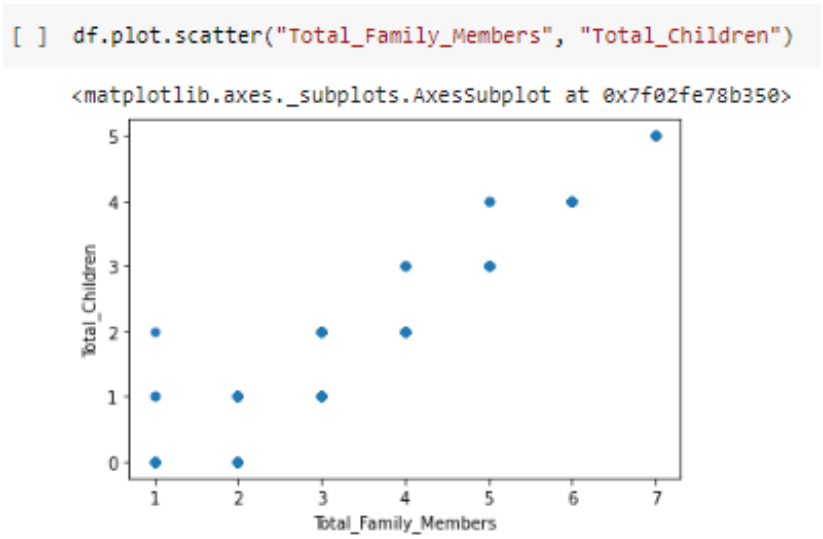
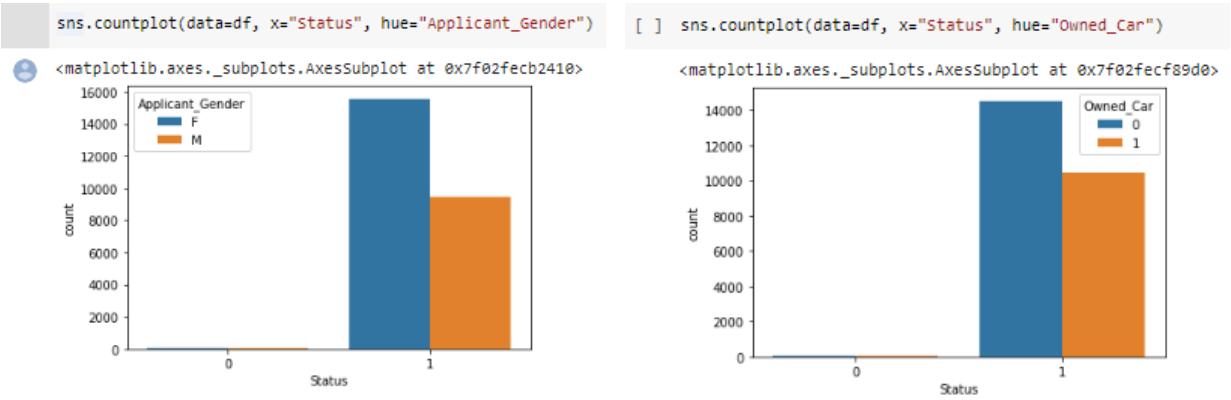


Figure 13: Scatterplot Total_Family_Member and Total_Children

8.8. Bivariate Analysis

The bivariate analysis is performed to all variables in relation to the target to determine the relationship between them. Refer to Figure 14 and 15.

Figure 14: Bivariate analysis for categorical variable versus target



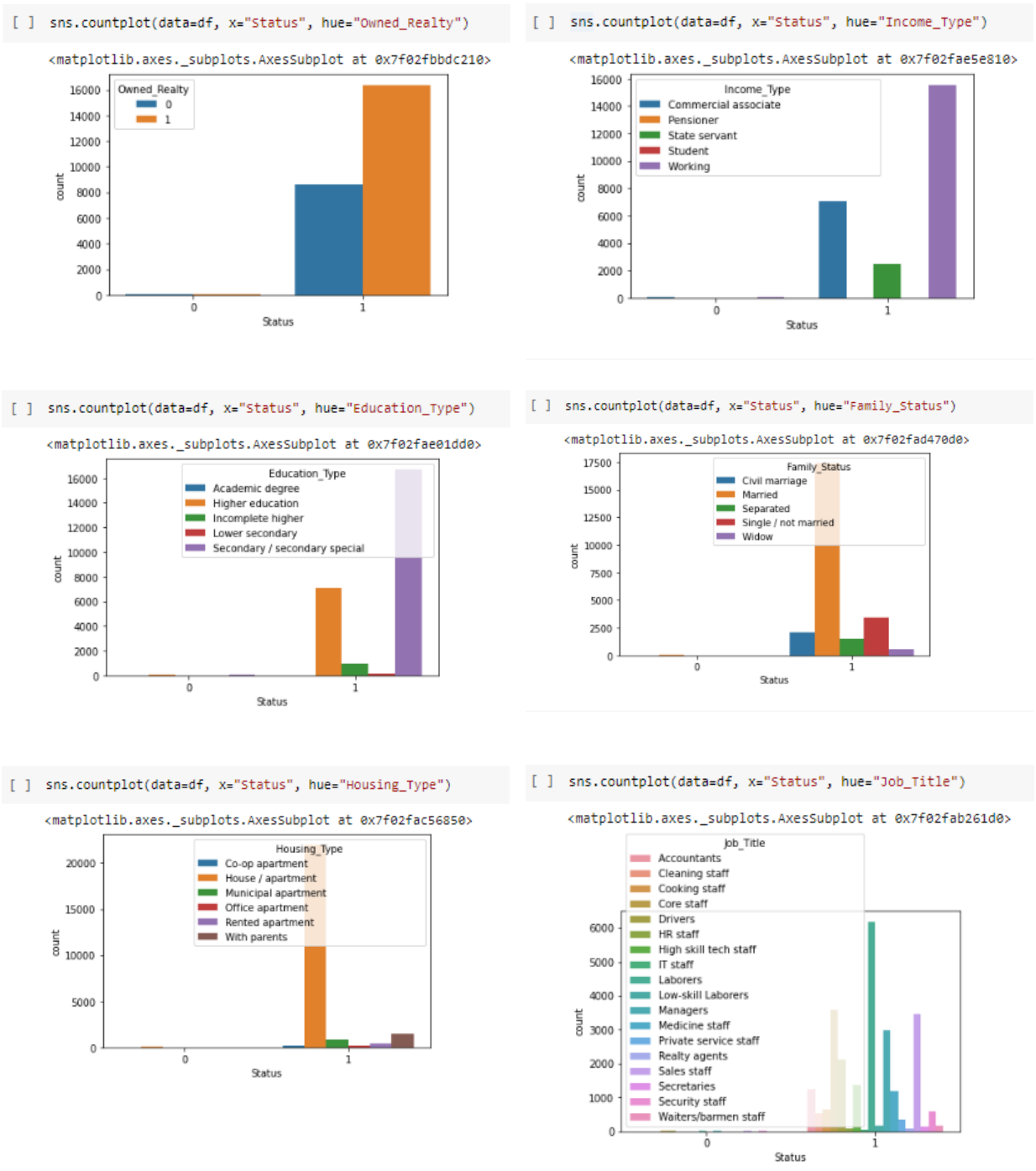


Figure 14: Bivariate analysis for categorical variable versus target

Figure 15: Bivariate analysis for continuous variable versus target

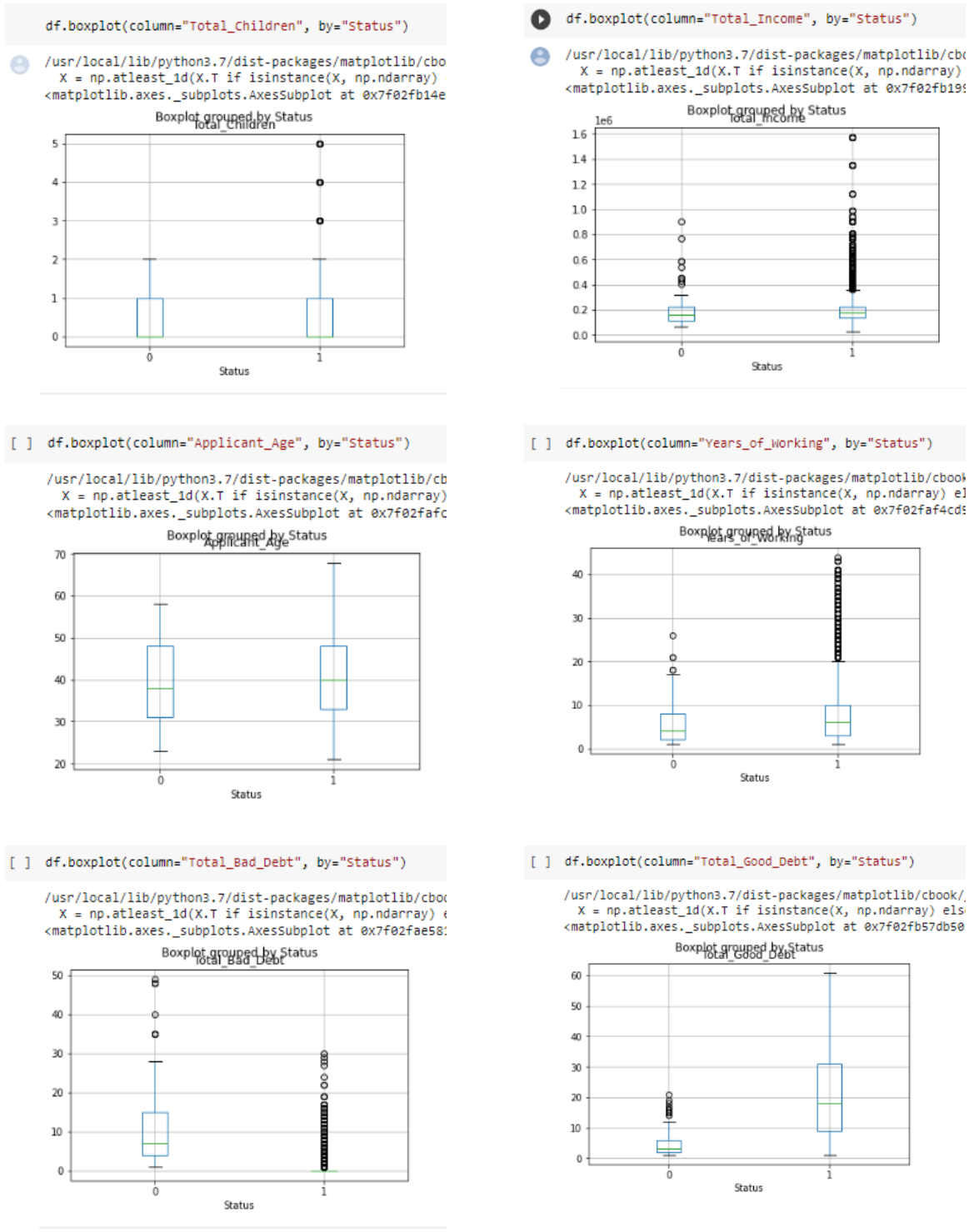


Figure 15: Bivariate analysis for continuous variable versus target

9.0. Data Cleansing

To handle outliers, the records that are outside of the interquartile range have been dropped. Variables Total_Children, Total_Income, and Years_of_Working are assessed with this method with a total of 2,931 records (11.66% of total) dropped. The skewness and outliers issues are mostly solved. Refer to Figure 16, 17, and 18.

Figure 16: Handling outliers

```
[ ]  # Handling Outliers

     for x in ["Total_Children"]:
         q75,q25 = np.percentile(df.loc[:,x],[75,25])
         intr_qr = q75-q25

         max = q75+(1.5*intr_qr)
         min = q25-(1.5*intr_qr)

     df.loc[df[x] < min,x] = np.nan
     df.loc[df[x] > max,x] = np.nan
```

```
[ ]  for x in ["Total_Income"]:
         q75,q25 = np.percentile(df.loc[:,x],[75,25])
         intr_qr = q75-q25

         max = q75+(1.5*intr_qr)
         min = q25-(1.5*intr_qr)

     df.loc[df[x] < min,x] = np.nan
     df.loc[df[x] > max,x] = np.nan
```

```
[ ]  for x in ["Years_of_Working"]:
         q75,q25 = np.percentile(df.loc[:,x],[75,25])
         intr_qr = q75-q25

         max = q75+(1.5*intr_qr)
         min = q25-(1.5*intr_qr)

     df.loc[df[x] < min,x] = np.nan
     df.loc[df[x] > max,x] = np.nan
```

```
[ ]  df = df.dropna(axis = 0)
```

Figure 16: Handling outliers

Figure 17: Outliers visualization after cleansing.

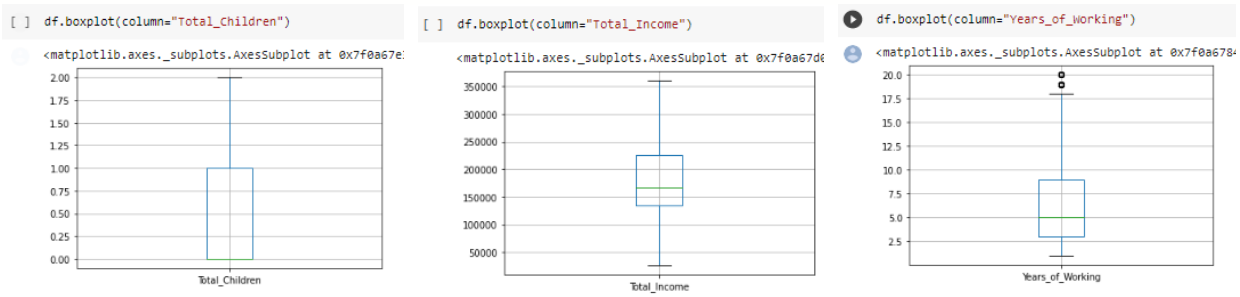


Figure 17: Outliers visualization after cleansing

Figure 18: Skewness after cleansing

```
#Skewness improvement

df.skew(axis=0)
```

```
/usr/local/lib/python3.7/dist-packages/
  This is separate from the ipykernel p
Owned_Car             0.337474
Owned_Realty         -0.623819
Total_Children        1.092711
Total_Income          0.604917
Owned_Work_Phone      1.000345
Owned_Phone           0.933422
Applicant_Age         0.327120
Years_of_Working      0.932117
Total_Good_Debt       0.752681
Status              -14.299688
dtype: float64
```

Figure 18: Skewness after cleansing

Data Preparation and Feature Engineering

10.0. Data Preparation Needs

Different techniques and feature engineering have been implemented in order to prepare the data for the modeling such as get dummies, standardization and SMOTE.

10.1. Drop Variables

In order to clean the data, some variables such as ID, Owned_Mobile_Phone, Total_Family_Members, Owned_Email, and Total_Bad_Debt have been dropped left in total 16 variables. Refer to Figure 19.

In case of Owned_Mobile_Phone, Owned_Email, and Total_Bad_Debt the criteria is the high class imbalanced they have as they account more than 88% of the records with one single class.

Moreover, in case of Total_Bad_Debt it has a correlation of -0.5 with the target variable that can lead to a greater negative impact. Likewise, the criteria for the variable Total_Family_Members is the high positive correlation and multicollinearity it has with Total_Children and the lower correlation with the target variable.

Figure 19: Drop variables

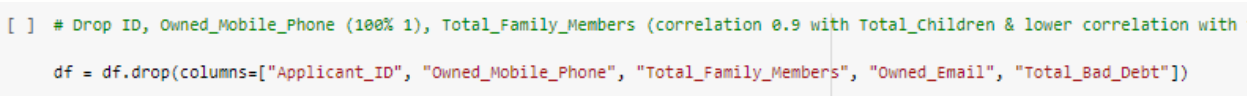


Figure 19: Drop variables

10.2. Get Dummies

Get dummies has been applied to convert categorical variables such as Applicant_Gender, Income_Type, Education_Type, Family_Status, Housing_Type, and Job_Title into dummy variables which leaves a total of 51 variables. Refer to Figure 20.

Figure 20: Get Dummies

```
    df = pd.get_dummies(df)
```

```
[ ] print(df.shape)
    print(df.columns)
```

```
    (22197, 51)
    Index(['Owned_Car', 'Owned_Realty', 'Total_Children', 'Total_Income',
           'Owned_Work_Phone', 'Owned_Phone', 'Applicant_Age', 'Years_of_Working',
           'Total_Good_Debt', 'Status', 'Applicant_Gender_F      ',
           'Applicant_Gender_M      ',
           'Income_Type_Commercial associate                                ',
           'Income_Type_Pensioner                                           ',
           'Income_Type_State servant                                       ',
           'Income_Type_Student                                             ',
           'Income_Type_Working                                             ',
           'Education_Type_Academic degree                                     ',
           'Education_Type_Higher education                                    ',
           'Education_Type_Incomplete higher                                   ',
           'Education_Type_Lower secondary                                     ',
           'Education_Type_Secondary / secondary special                       ',
           'Family_Status_Civil marriage                                    ',
           'Family_Status_Married                                           ',
           'Family_Status_Separated                                         ',
           'Family_Status_Single / not married                              ',
           'Family_Status_Widow                                             ',
           'Housing_Type_Co-op apartment                                    ',
           'Housing_Type_House / apartment                                  ',
           'Housing_Type_Municipal apartment                                ',
           'Housing_Type_Office apartment                                   ',
           'Housing_Type_Rented apartment                                   ',
           'Housing_Type_With parents                                       ',
           'Job_Title_Accountants                                        ',
           'Job_Title_Cleaning staff                                     ',
           'Job_Title_Cooking staff                                      ',
           'Job_Title_Core staff                                         ',
```

Figure 20: Get Dummies

10.3. Standardization

The dataset is also standardized using a StandardScaler to scale each feature to unit variance. This is made since the dataset contains variables with different measure units and to enhance the performance of the algorithms so they could converge faster. For this purpose, before the standardization the dataset must be split into training and validation in a proportion of 60-40. Refer to Figure 21.

Figure 21: Standardization

```
[ ]  # Split the dataset into training and validation data
     X=df.drop(columns=["Status"])
     y=df["Status"]
     train_X, valid_X, train_y, valid_y = train_test_split(X, y,  test_size=0.4, random_state=1)
```

```
[ ]  # Standardization of the data

     ss = StandardScaler()
     ss.fit(train_X)
     train_X = ss.transform(train_X)
     valid_X = ss.transform(valid_X)
```

Figure 21: Standardization

10.4. SMOTE

One of the main concerns in this dataset is the high class imbalanced of the target variable. Status has 99.5% of the records as 1 and 0.5% as 0. To solve this minority-majority issue the technique SMOTE has been applied. SMOTE is the acronym of Synthetic Minority Oversampling Technique which means that the SMOTE synthesizes new examples for the minority class thus the classes of the variable Status become balanced. The number of records after SMOTE is 44,180 with 50% of them for each class. Refer to Figure 22.

Figure 22: SMOTE

```
[ ]  #SMOTE

     sm = SMOTE(random_state=42)

     X_sm, y_sm = sm.fit_resample(X, y)

     print(f'''Shape of X before SMOTE: {X.shape}
     Shape of X after SMOTE: {X_sm.shape}''')

     print('\nBalance of positive and negative classes (%):')
     y_sm.value_counts(normalize=True) * 100
```

```
Shape of X before SMOTE: (22197, 50)
Shape of X after SMOTE: (44180, 50)

Balance of positive and negative classes (%):
1    50.0
0    50.0
Name: Status, dtype: float64
```

```
     counter = Counter(y)
     print(counter)
     counter = Counter(y_sm)
     print(counter)
```

```
Counter({1: 22090, 0: 107})
Counter({1: 22090, 0: 22090})
```

Figure 22: SMOTE

Model Exploration

11.0. Modeling Approach/Introduction

For this analysis different classification models will be built such as Random Forest Classifier, Logistic Regression, Multi-Layer Perceptron and Gradient Boost Classifier to solve the business problem, make the best prediction, and uncover the best drivers. As explain in previous chapters, the metrics to measure the models will be accuracy and ROC. The final goal of the model comparison is to select the best model.

12.0. Model Technique # 1: Random Forest Classifier

In research from IBM Cloud Education (2020) Leo Breiman and Adele Cutler created random forest which is a widely used machine learning technique, the random forest technique produces a single outcome by mixing the output of various decision trees. Because of its high usability and adaptability, the its use is widespread as it can solve the classification and regression issues. The bagging method is extended by the random forest algorithm as it produces an uncorrelated forest of decision trees by using features such as randomness in addition to bagging. In conclusion with regard to this model the advantages of this algorithm are that they are a bit more flexible than other algorithms, they reduce the risk of overfitting, and they are easy to determine feature importance; but they are a bit more complex thus require more time and resources.

12.1. Code

To build this model it is necessary first split the dataset that resulted from the SMOTE, then fit the algorithm, and finally observe the feature importance and its metrics. Refer to Figure 23, 24, 25, and 26.

Figure 23: Split and Fit

```
[ ]  train_X, valid_X, train_y, valid_y = train_test_split(X_sm, y_sm,  test_size=0.4, random_state=17)
```

```
rf = RandomForestClassifier(random_state=42)
rf.fit(train_X, train_y)
```

```
RandomForestClassifier(random_state=42)
```

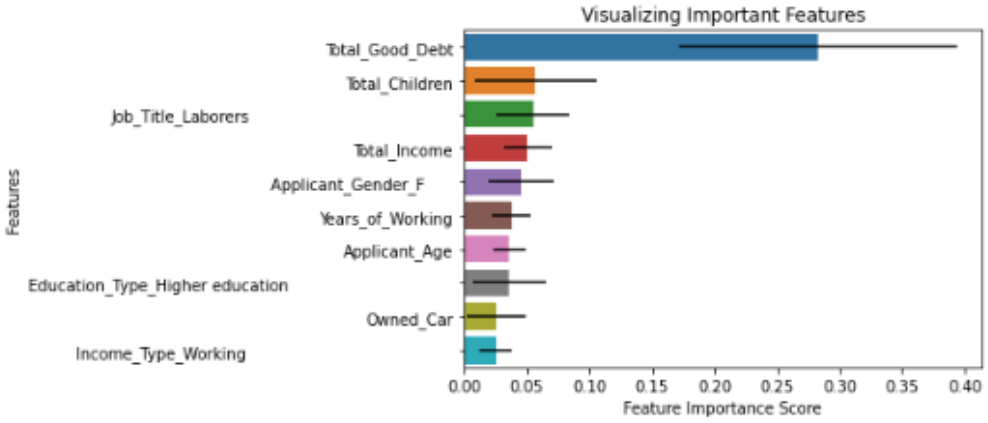
Figure 23: Split and Fit

Figure 24: Feature importance

```
[ ]  importance= rf.feature_importances_
     std=np.std([tree.feature_importances_ for tree in rf.estimators_], axis=0)
```

```
[ ]  std = np.std([tree.feature_importances_ for tree in rf.estimators_], axis=0)
     tf = pd.DataFrame({'feature': train_X.columns, 'importance': rf.feature_importances_, 'std': std})
     tf = tf.sort_values('importance', ascending = False)
     sns.barplot(x = tf['importance'][:10], y = tf['feature'][:10], xerr = tf['std'][:10])
     plt.xlabel('Feature Importance Score')
     plt.ylabel('Features')
     plt.title("Visualizing Important Features")
```

```
Text(0.5, 1.0, 'Visualizing Important Features')
```




```
imp= pd.DataFrame({'feature': train_X.columns, 'importance': importance, 'std':std})
print(imp.sort_values('importance', ascending=False).head())
```

```
                                    feature  importance       std
8                            Total_Good_Debt    0.282459  0.111318
2                             Total_Children    0.057040  0.048870
40  Job_Title_Laborers                   ...    0.055407  0.029046
3                              Total_Income    0.050798  0.019152
```

Figure 24: Feature importance

Figure 25: Metrics and confusion matrix

```
[ ]  Accuracy_rf = accuracy_score(valid_y, rf.predict(valid_X))
     print(Accuracy_rf)
     Precision_rf = precision_score(valid_y, rf.predict(valid_X))
     print(Precision_rf)
     Recall_rf = recall_score(valid_y, rf.predict(valid_X))
     print(Recall_rf)
     F1Score_rf = f1_score(valid_y, rf.predict(valid_X))
     print(F1Score_rf)
```

```
0.9976799456767769
0.9959165154264973
0.9994308480364257
0.9976705868984717
```

```
[ ]  classificationSummary(train_y, rf.predict(train_X))
```

```
Confusion Matrix (Accuracy 0.9999)

       Prediction
Actual     0     1
     0 13200     3
     1     0 13305
```

```
[ ]  classificationSummary(valid_y, rf.predict(valid_X))
```

```
Confusion Matrix (Accuracy 0.9977)

       Prediction
Actual    0    1
     0 8851   36
     1    5 8780
```

Figure 25: Metrics and confusion matrix

Figure 26: ROC / AUC

```
[ ]  scores = cross_val_score(rf, train_X, train_y, scoring='roc_auc', n_jobs=-1)
     print('Mean ROC AUC: %.3f' % mean(scores))

     Mean ROC AUC: 1.000
```

```
[ ]  y_pred_proba = rf.predict_proba(valid_X)[::,1]

     fpr, tpr, _ = metrics.roc_curve(valid_y, y_pred_proba)

     auc = metrics.roc_auc_score(valid_y, y_pred_proba)
     roc_auc_rf = auc

     plt.plot(fpr,tpr,label="test_data(AUC= %0.2f)" % auc, linewidth = 4)
     plt.legend(prop={'size':12},loc='best')
     plt.title('\nROC plot',fontsize = 16)
     plt.xlabel('False positive rate', fontsize = 16)
     plt.ylabel('True positive rate', fontsize = 16)
     plt.show()
```

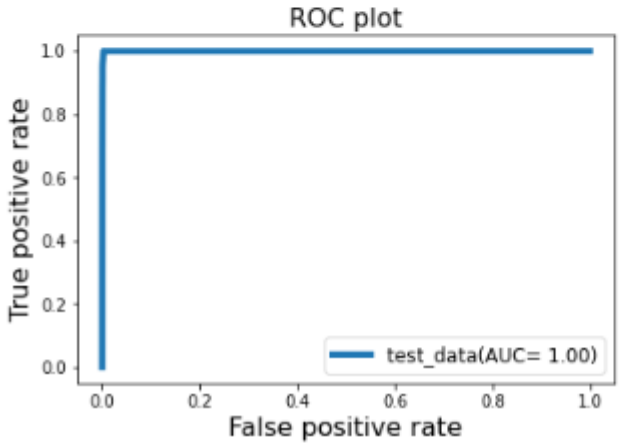


Figure 26: ROC / AUC

According to the feature importance, it is evident that the principal variable that impact this model is Total_Good_Debt with 0.28 followed by Total_Children, Job_Title_Laborers, and Total_Income. However, this model has an accuracy of 99.77% and ROC/AUC of 1 which indicate that the model is overfitted. To solve this issue a cross validation will be fit.

12.2. Cross Validation – Grid Search

According to Beheshti (2022) a cross validation divide the dataset into random groups, reserve one group for testing, and train the model on the remaining groups. For each group retained as the test group, this process is repeated, and the average of the models is used to create the final model. This technique can be applied by using GridSearchCV of SKLearn. Basically, this algorithm search the best parameters among the suggested ones for the new model. Refer to Figure 27, 28, and 29.

Figure 27: Cross Validation / GridSearchCV

```
[ ]  param_grid = {
         'max_depth' : [5, 15, 25, 30],
         'min_samples_split' : [5, 15, 30, 45],
         'min_impurity_decrease' : [0.05, 0.01, 0.001, 0.0001],
         "n_estimators" : [200, 400, 600, 800]
     }
```

```
[ ]  gridsearch = GridSearchCV(RandomForestClassifier(random_state=42), param_grid, cv=3, n_jobs=-1)
```

```
[ ]  gridsearch.fit(train_X, train_y)

     GridSearchCV(cv=3, estimator=RandomForestClassifier(random_state=42), n_jobs=-1,
                  param_grid={'max_depth': [5, 15, 25, 30],
                              'min_impurity_decrease': [0.05, 0.01, 0.001, 0.0001],
                              'min_samples_split': [5, 15, 30, 45],
                              'n_estimators': [200, 400, 600, 800]})
```

```
[ ]  gridsearch.best_score_

     0.9980006035913687
```

```
[ ]  gridsearch.best_params_

     {'max_depth': 30,
      'min_impurity_decrease': 0.0001,
      'min_samples_split': 5,
      'n_estimators': 200}
```

```
[ ]  rf_gs=gridsearch.best_estimator_
     rf_gs

     RandomForestClassifier(max_depth=30, min_impurity_decrease=0.0001,
                            min_samples_split=5, n_estimators=200, random_state=42)
```

Figure 27: Cross Validation / GridSearchCV

Figure 28: Metrics and confusion matrix

```
[ ] Accuracy_rf_gs = accuracy_score(valid_y, rf_gs.predict(valid_X))
    print(Accuracy_rf_gs)
    Precision_rf_gs = precision_score(valid_y, rf_gs.predict(valid_X))
    print(Precision_rf_gs)
    Recall_rf_gs = recall_score(valid_y, rf_gs.predict(valid_X))
    print(Recall_rf_gs)
    F1Score_rf_gs = f1_score(valid_y, rf_gs.predict(valid_X))
    print(F1Score_rf_gs)

    0.9965482118605704
    0.9941096511101042
    0.9989755264655663
    0.9965366490660308

[ ] classificationSummary(train_y, rf_gs.predict(train_X))

    Confusion Matrix (Accuracy 0.9983)

           Prediction
    Actual     0     1
         0 13166    37
         1     9 13296

[ ] classificationSummary(valid_y, rf_gs.predict(valid_X))

    Confusion Matrix (Accuracy 0.9965)

           Prediction
    Actual    0    1
         0 8835   52
         1    9 8776
```

Figure 28: Metrics and confusion matrix

Figure 29: ROC / AUC

```
[ ]  scores = cross_val_score(rf_gs, train_X, train_y, scoring='roc_auc', n_jobs=-1)
     print('Mean ROC AUC: %.3f' % mean(scores))

     Mean ROC AUC: 1.000
```

```
[ ]  y_pred_proba = rf_gs.predict_proba(valid_X)[::,1]

     fpr, tpr, _ = metrics.roc_curve(valid_y, y_pred_proba)

     auc = metrics.roc_auc_score(valid_y, y_pred_proba)
     roc_auc_rf_gs = auc

     plt.plot(fpr,tpr,label="test_data(AUC= %0.2f)" % auc, linewidth = 4)
     plt.legend(prop={'size':12},loc='best')
     plt.title('\nROC plot',fontsize = 16)
     plt.xlabel('False positive rate', fontsize = 16)
     plt.ylabel('True positive rate', fontsize = 16)
     plt.show()
```

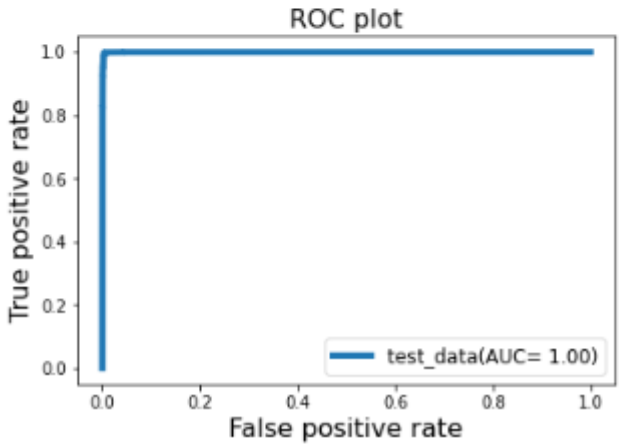


Figure 29: ROC / AUC

Although the cross validation has been executed and the new model has been fitted with the best parameters resulting from the GridSearchCV, the change in the results is not noticeable since the accuracy is 99.65% and ROC/AUC is 1.

13.0. Model Technique # 2: Logistic Regression

Logistic regression or logit model is a supervised machine learning model used for classification and predictive analytics. It predicts the dependent variable based on the analysis if the relationship with one or more independent ones. According to (*What Is Logistic Regression? | IBM*, n.d.) this algorithm calculates the likelihood that a specific event will occur based on a given dataset. Given that the result is a probability, the dependent variable's range is 0 to 1. In logistic regression, the odds—that is, the probability of success divided by the probability of failure—are transformed using the logit formula. When a logit model has high dimensionality, regularisation is often employed to penalise big coefficients in the parameters.

13.1. Code

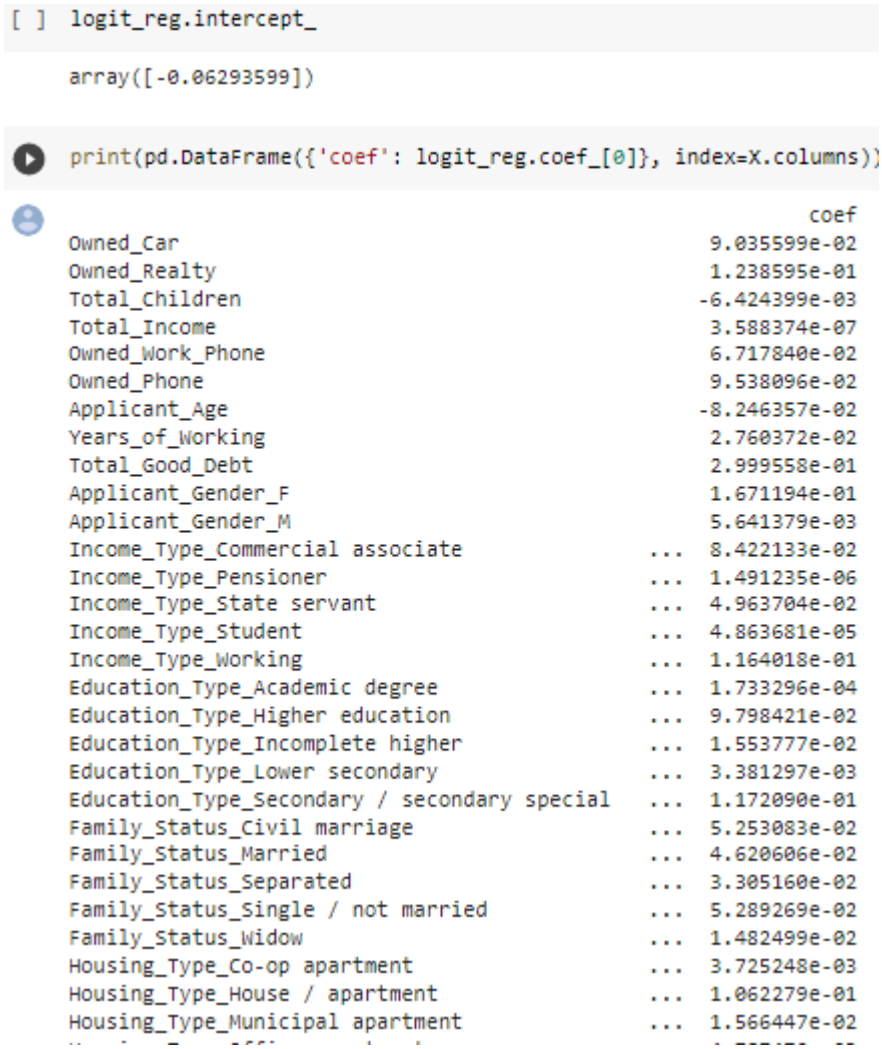
To fit this algorithm, the split and thus the variables from Random Forest are used. Then it can be observed the intercept and coefficients. Finally, with the predictions and probabilities it is possible to measure the accuracy and ROC/AUC. Refer to Figure 30, 31, 32, 33, and 34.

Figure 30: Fit

```
[ ]  logit_reg = LogisticRegression(solver='liblinear', C=1e42, random_state=1)
     logit_reg.fit(train_X, train_y)

     LogisticRegression(C=1e+42, random_state=1, solver='liblinear')
```

Figure 30: Fit

Figure 31: Intercept and Coefficients



```
[ ]  logit_reg.intercept_

     array([-0.06293599])

     print(pd.DataFrame({'coef': logit_reg.coef_[0]}, index=X.columns))

                                                        coef
     Owned_Car                                    9.035599e-02
     Owned_Realty                                 1.238595e-01
     Total_Children                              -6.424399e-03
     Total_Income                                 3.588374e-07
     Owned_Work_Phone                             6.717840e-02
     Owned_Phone                                  9.538096e-02
     Applicant_Age                               -8.246357e-02
     Years_of_Working                             2.760372e-02
     Total_Good_Debt                              2.999558e-01
     Applicant_Gender_F                           1.671194e-01
     Applicant_Gender_M                           5.641379e-03
     Income_Type_Commercial associate        ...  8.422133e-02
     Income_Type_Pensioner                   ...  1.491235e-06
     Income_Type_State servant               ...  4.963704e-02
     Income_Type_Student                     ...  4.863681e-05
     Income_Type_Working                     ...  1.164018e-01
     Education_Type_Academic degree          ...  1.733296e-04
     Education_Type_Higher education         ...  9.798421e-02
     Education_Type_Incomplete higher        ...  1.553777e-02
     Education_Type_Lower secondary          ...  3.381297e-03
     Education_Type_Secondary / secondary special ...  1.172090e-01
     Family_Status_Civil marriage            ...  5.253083e-02
     Family_Status_Married                   ...  4.620606e-02
     Family_Status_Separated                 ...  3.305160e-02
     Family_Status_Single / not married      ...  5.289269e-02
     Family_Status_Widow                     ...  1.482499e-02
     Housing_Type_Co-op apartment            ...  3.725248e-03
     Housing_Type_House / apartment          ...  1.062279e-01
     Housing_Type_Municipal apartment        ...  1.566447e-02
```

Figure 31: Intercept and Coefficients

Figure 32: Probabilities and Predictions

```
logit_reg_prob = logit_reg.predict_proba(valid_X)
logit_reg_prob
```

```
array([[0.00519336, 0.99480664],
       [0.92970607, 0.07029393],
       [0.72985203, 0.27014797],
       ...,
       [0.57301071, 0.42698929],
       [0.87973316, 0.12026684],
       [0.29636304, 0.70363696]])
```

```
[ ] logit_reg_pred = logit_reg.predict(valid_X)
    logit_reg_pred
```

```
array([1, 0, 0, ..., 0, 0, 1])
```

```
[ ] logit_result = pd.DataFrame({'actual' : valid_y,
                                 'p_0' : [p[0] for p in logit_reg_prob],
                                 'p_1' : [p[1] for p in logit_reg_prob],
                                 'predicted': logit_reg_pred})
    logit_result
```

	actual	p_0	p_1	predicted
4043	1	0.005193	0.994807	1
25399	0	0.929706	0.070294	0
22877	0	0.729852	0.270148	0
37141	0	0.278256	0.721744	1
43162	0	0.764932	0.235068	0
...	...	...	...	...
29245	0	0.872658	0.127342	0

Figure 32: Probabilities and Predictions

Figure 33: Metrics and confusion matrix

```
[ ]  Accuracy_logit_reg = accuracy_score(valid_y, logit_reg.predict(valid_X))
     print(Accuracy_logit_reg)
     Precision_logit_reg = precision_score(valid_y, logit_reg.predict(valid_X))
     print(Precision_logit_reg)
     Recall_logit_reg = recall_score(valid_y, logit_reg.predict(valid_X))
     print(Recall_logit_reg)
     F1Score_logit_reg = f1_score(valid_y, logit_reg.predict(valid_X))
     print(F1Score_logit_reg)

     0.8280330466274333
     0.8644089294774226
     0.7757541263517359
     0.8176855240266363

[ ]  classificationSummary(train_y, logit_reg.predict(train_X))

     Confusion Matrix (Accuracy 0.8280)

            Prediction
     Actual     0     1
          0 11555  1648
          1  2912 10393

[ ]  classificationSummary(valid_y, logit_reg.predict(valid_X))

     Confusion Matrix (Accuracy 0.8280)

            Prediction
     Actual    0    1
          0 7818 1069
          1 1970 6815
```

Figure 33: Metrics and confusion matrix

Figure 34: ROC/AUC

```
[ ]  scores = cross_val_score(logit_reg, train_X, train_y, scoring='roc_auc', n_jobs=-1)
     print('Mean ROC AUC: %.3f' % mean(scores))

     Mean ROC AUC: 0.916
```

```
[ ]  y_pred_proba = logit_reg.predict_proba(valid_X)[::,1]

     fpr, tpr, _ = metrics.roc_curve(valid_y, y_pred_proba)

     auc = metrics.roc_auc_score(valid_y, y_pred_proba)
     roc_auc_logit_reg = auc

     plt.plot(fpr,tpr,label="test_data(AUC= %0.2f)" % auc, linewidth = 4)
     plt.legend(prop={'size':12},loc='best')
     plt.title('\nROC plot',fontsize = 16)
     plt.xlabel('False positive rate', fontsize = 16)
     plt.ylabel('True positive rate', fontsize = 16)
     plt.show()
```

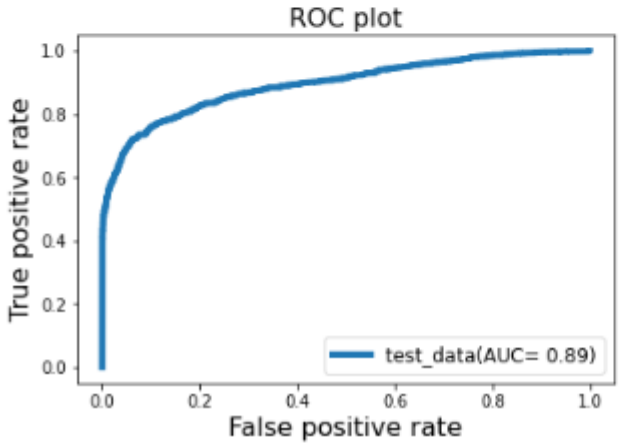


Figure 34: ROC/AUC

This model has an accuracy of 82.80% and ROC/AUC of 0.89 which are good results as it is not overfitted and has good performance.

14.0. Model Technique # 3: Neural Network – Multi-Layer Perceptron Classifier

According to Bento (2021) “Multi-Layer Perceptron (MLP) is a neural network where the mapping between inputs and outputs is non-linear”. It has input layer, output layer, hidden units with neurons stacked together, and they can use any activation function. This algorithm is

considered a feedforward since inputs and initial weights are mixed in a weighted sum and are both subject to the activation function but each linear combination is propagated over the following layer feeding the next one with its internal computation.

14.1. Code

This neural network has to be fitted with the same dataset split and its variables so then the accuracy and ROC/AUC will be shown. Refer to Figure 35, 36, and 37.

Figure 35: Fit

```
[ ]  clf = MLPClassifier(activation='logistic',solver='lbfgs',hidden_layer_sizes=3,random_state=17)
     clf.fit(train_X,train_y)

     MLPClassifier(activation='logistic', hidden_layer_sizes=3, random_state=17,
                   solver='lbfgs')
```

Figure 35: Fit

Figure 36: Metrics and confusion matrix

```
[ ]  Accuracy_clf = accuracy_score(valid_y, clf.predict(valid_X))
     print(Accuracy_clf)
     Precision_clf = precision_score(valid_y, clf.predict(valid_X))
     print(Precision_clf)
     Recall_clf = recall_score(valid_y, clf.predict(valid_X))
     print(Recall_clf)
     F1Score_clf = f1_score(valid_y, clf.predict(valid_X))
     print(F1Score_clf)

     0.4971140787686736
     0.4971140787686736
     1.0
     0.664096458404203

[ ]  classificationSummary(train_y, clf.predict(train_X))

     Confusion Matrix (Accuracy 0.5019)

            Prediction
     Actual     0     1
          0     0 13203
          1     0 13305

[ ]  classificationSummary(valid_y, clf.predict(valid_X))

     Confusion Matrix (Accuracy 0.4971)

            Prediction
     Actual    0    1
          0    0 8887
          1    0 8785
```

Figure 36: Metrics and confusion matrix

Figure 37: ROC/AUC

```
[ ]  scores = cross_val_score(clf, train_X, train_y, scoring='roc_auc', n_jobs=-1)
     print('Mean ROC AUC: %.3f' % mean(scores))

     Mean ROC AUC: 0.500
```

```
[ ]  y_pred_proba = clf.predict_proba(valid_X)[::,1]

     fpr, tpr, _ = metrics.roc_curve(valid_y, y_pred_proba)

     auc = metrics.roc_auc_score(valid_y, y_pred_proba)
     roc_auc_clf = auc

     plt.plot(fpr,tpr,label="test_data(AUC= %0.2f)" % auc, linewidth = 4)
     plt.legend(prop={'size':12},loc='best')
     plt.title('\nROC plot',fontsize = 16)
     plt.xlabel('False positive rate', fontsize = 16)
     plt.ylabel('True positive rate', fontsize = 16)
     plt.show()
```

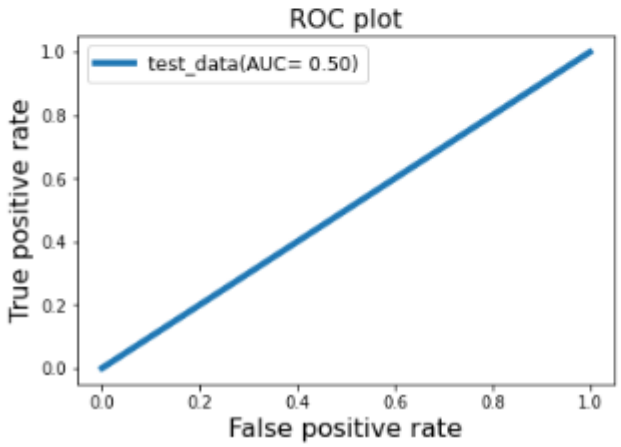


Figure 37: ROC/AUC

This model has an accuracy of 49.71% and ROC/AUC of 0.50 which are weak results enough to be considered as a bad model with a poor performance.

15.0. Model Technique # 4: Gradient Boosting Classifier

In research from Aliyev (2020) in the gradient boosting classifier algorithm each prediction aims to outperform the one before it by lowering the errors. It really fits a new predictor to the residual errors created by the preceding predictor rather than fitting a prediction to the data at each iteration. This algorithm is used specifically for classification tasks and it was created when trying to enhanced weaker models.

15.1. Code

This model has to be fitted with the same dataset split and its variables. The only different thing is that it need to be added the learning rate. Then the accuracy and ROC/AUC will be shown. Refer to Figure 38, 39, and 40.

Figure 38: Fit

```
[ ]  gbc = GradientBoostingClassifier(learning_rate=0.1)
     gbc.fit(train_X,train_y)

     GradientBoostingClassifier()
```

Figure 38: Fit

Figure 39: Metrics and confusion matrix

```
[ ]  Accuracy_gbc = accuracy_score(valid_y, gbc.predict(valid_X))
     print(Accuracy_gbc)
     Precision_gbc = precision_score(valid_y, gbc.predict(valid_X))
     print(Precision_gbc)
     Recall_gbc = recall_score(valid_y, gbc.predict(valid_X))
     print(Recall_gbc)
     F1Score_gbc = f1_score(valid_y, gbc.predict(valid_X))
     print(F1Score_gbc)

     0.9839293798098687
     0.9840564855938959
     0.9836084234490609
     0.9838324035067745

[ ]  classificationSummary(train_y, gbc.predict(train_X))

     Confusion Matrix (Accuracy 0.9871)

            Prediction
     Actual     0     1
          0 13042   161
          1   181 13124

[ ]  classificationSummary(valid_y, gbc.predict(valid_X))

     Confusion Matrix (Accuracy 0.9839)

            Prediction
     Actual    0    1
          0 8747  140
          1  144 8641
```

Figure 39: Metrics and confusion matrix

Figure 40: ROC/AUC

```
[ ]  scores = cross_val_score(gbc, train_X, train_y, scoring='roc_auc', n_jobs=-1)
     print('Mean ROC AUC: %.3f' % mean(scores))

     Mean ROC AUC: 0.999
```

```
[ ]  y_pred_proba = gbc.predict_proba(valid_X)[::,1]

     fpr, tpr, _ = metrics.roc_curve(valid_y, y_pred_proba)

     auc = metrics.roc_auc_score(valid_y, y_pred_proba)
     roc_auc_gbc = auc

     plt.plot(fpr,tpr,label="test_data(AUC= %0.2f)" % auc, linewidth = 4)
     plt.legend(prop={'size':12},loc='best')
     plt.title('\nROC plot',fontsize = 16)
     plt.xlabel('False positive rate', fontsize = 16)
     plt.ylabel('True positive rate', fontsize = 16)
     plt.show()
```

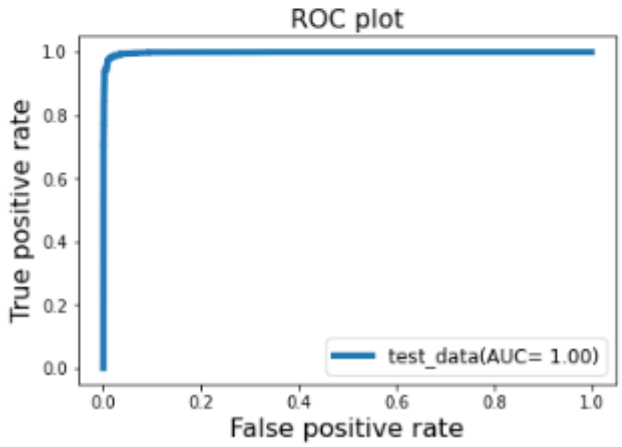


Figure 40: ROC/AUC

This model has an accuracy of 98.39% and ROC/AUC of 1 which indicate that the model is overfitted.

16.0. Model Comparison

After reviewing the models, it is necessary to make a comparison of the metrics of each model. The objective of the model comparison is to choose which one is the best model that can help to solve the business problem of the accurate approval of credit cards and consequently reduced of credit risk and increase of earnings.

To make the comparison new data frames with the metrics that want to be included must be created. After that, the data frames concatenate each other into a table. Refer to Figure 41 and 42.

Figure 31: Creating a new data frame

```
[ ] column_labels = ["Model", "Accuracy", "Recall", "Precision", "F1-Score", "ROC"]
```

```
[ ] df_1 = pd.DataFrame([['Random_Forest', Accuracy_rf, Recall_rf, Precision_rf, F1Score_rf, roc_auc_rf]], columns = column_labels)
    df_2 = pd.DataFrame([['Grid_Search_Random_Forest', Accuracy_rf_gs, Recall_rf_gs, Precision_rf_gs, F1Score_rf_gs, roc_auc_rf_gs]], columns = column_labels)
    df_3 = pd.DataFrame([['Logistic_Regression', Accuracy_logit_reg, Recall_logit_reg, Precision_logit_reg, F1Score_logit_reg, roc_auc_logit_reg]], columns = column_labels)
    df_4 = pd.DataFrame([['Neural_Network', Accuracy_clf, Recall_clf, Precision_clf, F1Score_clf, roc_auc_clf]], columns = column_labels)
    df_5 = pd.DataFrame([['Gradient_Boost_Classifier', Accuracy_gbc, Recall_gbc, Precision_gbc, F1Score_gbc, roc_auc_gbc]], columns = column_labels)
```

Figure 41: Creating a new data frame

Figure 42: Model Comparison

```
[ ] mc = pd.concat([df_1, df_2, df_3, df_4, df_5])
    mc
```

	Model	Accuracy	Recall	Precision	F1-Score	ROC
0	Random_Forest	0.997680	0.999431	0.995917	0.997671	0.999577
0	Grid_Search_Random_Forest	0.996548	0.998976	0.994110	0.996537	0.999601
0	Logistic_Regression	0.828033	0.775754	0.864409	0.817686	0.894708
0	Neural_Network	0.497114	1.000000	0.497114	0.664096	0.500000
0	Gradient_Boost_Classifier	0.983929	0.983608	0.984056	0.983832	0.998527

Figure 42: Model Comparison

From the table above, Random Forest, Cross validation Grid_Search Random Forest, and Gradient Boost Classifier are discarded due to overfitting. Neural Network MLP has a quite poor performance while Logistic Regression meets all the requirements to be the best model. It is not

over or under fitted and has an excellent performance over 80% both in accuracy and in ROC/AUC.

Model Recommendation

17.0 Model Selection

This analysis has been made to help issuing credit card companies to solve the problem of the accurate approval of the credit cards and consequently reduce the credit risk of the company as well as increase earnings by approving credit cards to good potential clients. With this in mind five different classification models have been run in Python: Random Forest, Cross Validation for Random Forest, Logistic Regression, Neural Network Multi-Layer Perceptron, and Gradient Boosting Classifier.

These models have been measured by its predicting accuracy and the ROC/AUC that shows the performance of a classification model at all thresholds. The model comparison table shows that the models Random Forest, Cross Validation for Random Forest, and Gradient Boosting Classifier are overfitted. The neural network Multi-Layer Perceptron has a weak performance with an accuracy of 49.7% and a ROC/AUC of 0.50. These four models have been discarded. The Logistic Regression model has an accuracy of 82.8% and a ROC/AUC of 0.89 thus being the model with the best performance and the one recommended after this detailed analysis. Refer to Figure 42: Model Comparison.

18.0 Model Theory

Logistic Regression is a statistical model that models the probability of an event to occur (dependent variable) based on the predictors (independent variables). When talking about a binary model, the dependent variable or target typically labels its values as “0” and “1” while the predictors can be categorical or continuous variables. Since the outcome is a probability, the

dependent or target variable takes values in a range of 0 and 1. According to (*What Is Logistic Regression? | IBM*, n.d.) in this algorithm the odds—the probability of success divided by the probability of failure—are transformed using the logit formula and creating the logarithm of odds (log odds).

The logistic formula is represented by:

$$\text{Logit}(pi) = 1/(1+ \exp(-pi))$$

$$\ln(pi/(1-pi)) = \text{Beta}\_0 + \text{Beta}\_1*X\_1 + \ldots + B\_k*K\_k$$

In these formulas Logit(pi) is the target variable, X the predictors, and Beta is the coefficient. This formula assesses different values of beta through several iterations to enhance for the best fit of log odds. The final goal of those iterations is to find the best parameter estimate.

However, log odds are difficult to interpret so they must be transformed into odds ratio by exponentiating the beta estimates. The odds ratio explains the probability that an outcome occurs given a specific event.

18.1 Model Assumptions and Limitations

Logistic regressions in comparison with other types of algorithms are more demanding in terms of data quality since they do not handle missing or extreme values well creating skewed results with limited predictive capabilities. Fortunately, this dataset does not contain missing values. Logistic regressions are also prone to overfitting. When a logit model has high dimensionality, regularisation is often employed to penalise big coefficients in the parameters. Also, as said before, because the data is being transformed there are a less representative description of each record than what the original dataset has hence the need to create the odds ratio.

In this sense, some techniques have been applied to handle skewness and outliers in the variables Total_Children, Total_Income, and Years_of_Working with excellent results since the skewness was reduced significantly.

Moreover, since the dataset has a problem of minority-majority, an oversampling technique SMOTE has been applied with great results and solving the issue of imbalance. Also, a standardization of the data and dummies of the categorical variables have been implemented.

19.0 Model Sensitivity to Key Drivers

After running the model, the odds ratio have been calculated. Refer to Figure 43.

Figure 43: Odds ratio

```
[75] odd = pd.DataFrame({"odds": np.e**logit_reg.coef_[0]}, index=X.columns)
     odd.sort_values(by=['odds'], ascending=False)
```

	odds
Total_Good_Debt	1.349799
Applicant_Gender_F	1.181895
Owned_Realty	1.131857
Education_Type_Secondary / secondary special	1.124354
Income_Type_Working	1.123447
Job_Title_Laborers	1.115702
Housing_Type_House / apartment	1.112075
Education_Type_Higher education	1.102945
Owned_Phone	1.100078
Owned_Car	1.094564
Income_Type_Commercial associate	1.087870
Job_Title_Sales staff	1.080293

Figure 43: Odds ratio

The following table contains the complete list of predictors with their respective odd ratio and the chance for a person to be approved for a credit card.

Variable	Odd-Ratio	Chances for a person to be approved for a credit card
Total_Good_Debt	1.3497991	34.98%
Applicant_Gender_F	1.1818953	18.19%
Owned_Realty	1.1318568	13.19%
Education_Type_Secondary / secondary special	1.1243544	12.44%
Income_Type_Working	1.1234471	12.34%
Job_Title_Laborers	1.1157023	11.57%
Housing_Type_House / apartment	1.1120753	11.21%
Education_Type_Higher education	1.1029454	10.29%
Owned_Phone	1.1000779	10.01%
Owned_Car	1.0945639	9.46%
Income_Type_Commercial associate	1.0878696	8.79%
Job_Title_Sales staff	1.0802934	8.03%
Owned_Work_Phone	1.0694863	6.95%
Job_Title_Core staff	1.0661711	6.62%
Family_Status_Single / not married	1.0543165	5.43%
Family_Status_Civil marriage	1.0539351	5.39%
Income_Type_State servant	1.0508896	5.09%
Family_Status_Married	1.0472902	4.73%
Job_Title_Managers	1.0462264	4.62%
Job_Title_Drivers	1.039719	3.97%
Family_Status_Separated	1.0336039	3.36%
Job_Title_High skill tech staff	1.0300657	3.01%
Job_Title_Accountants	1.0284816	2.85%
Years_of_Working	1.0279882	2.80%
Housing_Type_With parents	1.0236847	2.37%
Job_Title_Cleaning staff	1.0176116	1.76%
Housing_Type_Municipal apartment	1.0157878	1.58%
Education_Type_Incomplete higher	1.0156591	1.57%
Family_Status_Widow	1.0149354	1.49%
Job_Title_Cooking staff	1.0147978	1.48%
Job_Title_Medicine staff	1.0129957	1.30%
Housing_Type_Rented apartment	1.0106351	1.06%
Job_Title_Private service staff	1.0090169	0.90%
Job_Title_Waiters/barmen staff	1.0067927	0.68%
Applicant_Gender_M	1.0056573	0.57%
Job_Title_Low-skill Laborers	1.0040332	0.40%
Housing_Type_Co-op apartment	1.0037322	0.37%

Education_Type_Lower secondary	1.003387	0.34%
Job_Title_Secretaries	1.0030448	0.30%
Job_Title_HR staff	1.0028279	0.28%
Job_Title_Realty agents	1.0024804	0.25%
Education_Type_Academic degree	1.0001733	0.02%
Income_Type_Student	1.0000486	0.00%
Income_Type_Pensioner	1.0000015	0.00%
Total_Income	1.0000004	0.00%
Housing_Type_Office apartment	0.995224	-0.48%
Job_Title_Security staff	0.9946784	-0.53%
Total_Children	0.9935962	-0.64%
Job_Title_IT staff	0.9791231	-2.09%
Applicant_Age	0.920845	-7.92%

According to the table, the variables with the greatest positive impact on the outcome are Total_Good_Debt, Applicant_Gender_F, Owned_Realty, Education_Type_Secondary/secondary special, and Income_Type_Working.

Total_Good_Debt: if the "Total_Good_Debt" variable is increased by a single unit, a person's chance of being approved for a credit card increased by 35%.

Applicant_Gender_F: this suggests that being female increments the chance of being approved for a credit card by 18%.

Owned_Realty: this suggests that own realty increments the chance of being approved for a credit card by 13%.

Education_Type_Secondary/secondary special: this suggests that having the education level Secondary increments the chance of being approved for a credit card by 12%.

Income_Type_Working: this suggests that if the income type working increments the chance of being approved for a credit card by 12%.

On the contrary, the variable that highest negative impact in the outcome is Applicant_Age.

Applicant_Age: if the "Applicant_Age" variable is increased by a single unit, a person's chance of being approved for a credit card decreased by 8%.

There are not unusual odds ratio.

Validation and Governance

20.0. Variable Level Monitoring

It is important that certain levels be established for each of the model variables, on which a monthly monitoring will be carried out to find out if these variables have drifted, directly impacting the model.

20.1. Build Statistics

Variable	Mean	Standard Deviation	Mode
Applicant_Gender			F
Owned_Car			0
Owned_Realty			1
Total_Children	0.51	0.76	
Total_Income	194,836	104,521	
Income_Type			Working
Education_Type			Secondary/Secondary Special
Family_Status			Married
Housing_Type			House/Apartment
Owned_Mobile_Phone			1
Owned_Work_Phone			0
Owned_Phone			0
Owned_Email			0
Job_Title			Laborers
Total_Family_Members	2.29	0.93	
Applicant_Age	40.99	9.55	
Years_of_Working	7.69	6.41	
Total_Bad_Debt	0.33	1.56	
Total_Good_Debt	21.05	14.73	

20.2. Missing Values

The dataset does not have missing values. However, in case that any variable contains missing values, an imputation of those values should be done. Missing values will be replaced by the mean in case of numerical variables and the mode in case of categorical ones.

20.3. Variable Drift Monitoring

Considering that the principal driver of the model is Total_Good_Debt, there are three external macroeconomic variables that potentially could affect the model, especially in these times where the COVID-19 pandemic has affected the economies worldwide and inflationary environments are seeing.

On one side unemployment rate measures the unemployed people as a percentage of the labor force and it is seasonally adjusted. In Canada, the market expectations for unemployment rate for 2022 is 5%. If this rate increases significantly, it may have an impact on Total_Good_Debt since some of the clients could lose their jobs hence the capacity for payment.

On the other hand, inflation is the rate of increase in prices over a given period of time, being 8% the forecast for 2022 in Canada. If this rate increases significantly, it may also impact Total_Good_Debt since the cost of life is greater and the payment capacity will be reduced. Additionally, the income of the population usually do not go up in the same rate that inflation.

Finally, the interest rate could also affect the driver because it is a mechanism that Central Bank has to tackle inflation by raising the interest rate and in consequence making credits more expensive. In the same way that other variables, it may impact the payment capacity of the clients generating possible defaults in their financial obligations which reduces its Total_Good_Debt.

For the above reasons, it is important to make a monthly monitoring of the model variables to observe if one or some variables have drifted. In case it occurs, immediately action will be taken.

20.4. Tolerance for Drift of Each Variable

Total_Good_Debt: this is the most relevant driver for this model, so it will have a tolerance for drift of 1 standard deviation. For the rest of the variables a tolerance of 3 standard deviations change will be allowed.

21.0. Model Health & Stability

In order to maintain the stability of the model, every month the parameters accuracy and ROC/AUC will be performed on the model. If it is observed a 10% drift on the acceptable values, immediate action will be triggered.

22.0. Model Fit Statistics

The statistics considered for this model are accuracy and ROC/AUC as shown in the following table.

Parameter	Current Value	Acceptable Value
1. Accuracy	82.80%	80%
2. ROC/AUC	0.89	0.80

23.0. Risk Tiering

The potential harms will be classified according to the following four levels.

No Action	Report	Refit	Rebuild
• If there is no sign of drift. • Continue with the monthly monitoring.	• If a little drift, lower than the baseline values are seeing. • Observe if there is any impact of that variable in the parameters.	• If there is evidence of a drift, the model will be refit with the same variables to observe if there are new coefficients that can fit this better. • Use Grid Search.	• If the drift persists after refit or it exceeds too much. • Try new different algorithms and techniques as well as additional relevant dimensions.

A validation and governance plan is extremely important to ensure the stability and consistency in long term of the model. Doing the monthly monitoring of the variables and the parameters will allow the companies to take action on time and avoid potential losses.

Conclusion and Recommendations

24.0. Impact on Business Problem

The solution that have been provided might not fully resolve the business problem because the accuracy of the selected model is 82%, but it will hugely impact the process by bringing more efficiency and overall accuracy in the assessment of potential clients profile for a credit card approval especially because the process will not continue to be manual but more automated.

Although there is one model that have been selected as the best model because of its good performance, this is not a comprehensive solution for the business. It is clear that it will help to assess in a better way the customers, and therefore classify if a customer is “good” or “bad”. But

since the objective is reduce the credit risk and losses and increase earnings by approving all good applicants and rejecting all bad applicants, there are some other variables that would be considered in the assessment such as solvency ratio, debt level, and legal background issues.

25.0. Recommended Next Steps

Considering the big imbalance issue that this dataset has, the recommendation is to collect more data specifically from people who have been rejected for a credit card. It is extremely important to have a real sample from them. Moreover, because the imbalance is present not only in the status of the approval but also in some predictors, it is necessary to collect data that balance a bit those variables. For example, should look to include men in the sample, people who owned a car, who do not owned realty, not only workers but also students, pensioners, and state servants, people with study level different from secondary, who are not married and that do not live in a house or apartment, with a job title different from laborer, low income, children, and bad debt. This analysis would be more interesting, real, and accurate if the data set was more balanced and could really be observed the profile of the people who were rejected.

Regarding the techniques used in this analysis and since the dataset is small, it is recommended to use K-Fold cross validation on all models to evaluate its skill on unseen data by resampling. This procedure splits the data into groups, being the number of those groups the single parameter K. This procedure is very popular due to its simplicity and the accurate and less optimistic estimates it gives. Additionally, it is recommended that additional algorithms shall be added to make this analysis even more robust.

References

Aliyev, V. (2020, May 9). *Gradient Boosting Classification explained through Python*. Towards Data Science. Retrieved August 10, 2022, from https://towardsdatascience.com/gradient-boosting-classification-explained-through-python-60cc980eeb3d

Beheshti, N. (2022, February 12). *Cross Validation and Grid Search - Towards Data Science*. Towards Data Science. Retrieved August 10, 2022, from https://towardsdatascience.com/cross-validation-and-grid-search-efa64b127c1b

Bento, C. (2021, September 21). *Multilayer Perceptron Explained with a Real-Life Example and Python Code: Sentiment Analysis*. Towards Data Science. Retrieved August 10, 2022, from https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141

Brownlee, J. (2021, March 16). *SMOTE for Imbalanced Classification with Python*. Machine Learning Mastery. Retrieved August 8, 2022, from https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

Caesar, M. (n.d.). *Credit Card Approval Prediction (Cleaned Version)*. Kaggle. Retrieved August 8, 2022, from https://www.kaggle.com/datasets/caesarmario/application-data/metadata

CFI Team. (2022, January 27). *Major Risks for Banks*. Corporate Finance Institute. Retrieved August 8, 2022, from https://corporatefinanceinstitute.com/resources/knowledge/finance/major-risks-for-banks/

Deloitte Canada. (2018, August 3). *The new science of credit risk*. Retrieved August 8, 2022, from https://www2.deloitte.com/ca/en/pages/finance/articles/creditrisk.html

IBM Cloud Education. (2020, December 7). *Random Forest*. IBM. Retrieved August 9, 2022, from https://www.ibm.com/cloud/learn/random-forest#:%7E:text=Random%20forest%20is%20a%20commonly,both%20classification%20and%20regression%20problems.

Owainati, H. (2022, April 27). *10 reasons why a credit card application is denied (and what to do about it)*. Ratehub.Ca. Retrieved August 8, 2022, from https://www.ratehub.ca/blog/declined-credit-application-reasons/

S. (2020, March 24). *Credit Card Approval Prediction*. Kaggle. Retrieved August 8, 2022, from https://www.kaggle.com/datasets/rikdifos/credit-card-approval-prediction

SAS Institute Inc. (2022). *Credit Risk Management: What it is and why it matters*. SAS. Retrieved August 8, 2022, from https://www.sas.com/en_ca/insights/risk-management/credit-risk-management.html

Shift Credit Card Processing. (2021, August). *Credit Card Statistics [Updated August 2021] Shift Processing*. Retrieved July 14, 2022, from https://shiftprocessing.com/credit-card/

The imbalanced-learn developers. (2022). *SMOTE — Version 0.9.1*. Imbalanced-Learn. Retrieved August 8, 2022, from https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html

The Investopedia Team. (2022, March 15). *What Is Credit Risk?* Investopedia. Retrieved August 8, 2022, from https://www.investopedia.com/terms/c/creditrisk.asp

TheGlobalEconomy.com. (2022). *Percent people with credit cards by country, around the world*. Retrieved July 14, 2022, from https://www.theglobaleconomy.com/rankings/people_with_credit_cards/

What is Logistic regression? | IBM. (n.d.). IBM. Retrieved August 10, 2022, from https://www.ibm.com/topics/logistic-regression?mhsrc=ibmsearch_a&mhq=logistic%20regression