CREDIT SCORE ANALYSIS USING MACHINE LEARNING

A PROJECT REPORT

Submitted by

Dheeraaj P	**(195002030)**
Dinesh S	**(195002033)**
Maanas Karthikeyan S	**(195002069)**

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

Department of

Information Technology

Sri Sivasubramaniya Nadar College of Engineering

(An Autonomous Institution, Affiliated to Anna University)

MAY 2023

Sri Sivasubramaniya Nadar College of Engineering

(An Autonomous Institution, Affiliated to Anna University)

BONAFIDE CERTIFICATE

Certified that this Report titled "**CREDIT SCORE ANALYSIS USING MACHINE LEARNING**" is the bonafide work of **Dheeraaj P (195002030), Dinesh S (195002033) and Maanas Karthikeyan S (195002069)** who carried out the work under my supervision.

Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. C. Aravindan
Head of the Department

Department of Information

Technology

SSN College of Engineering

Kalavakkam – 603 110

SIGNATURE

Dr. K.S. Gayathri
Associate Professor

Department of Information

Technology

SSN College of Engineering

Kalavakkam – 603 110

Submitted for Project Viva-Voce Examination held on...................................

EXTERNAL EXAMINER

INTERNAL EXAMINE

ACKNOWLEDGEMENT

We are honored to acknowledge the below mentioned people who have instrumental in the completion of the project.

We express our deepest gratitude to **Padma Bhushan Dr. Shiv Nadar**, Chairman, **SSN Institutions** for being a source of inspiration and for instilling a great mission and vision in our minds.

We also thank our Principal, **Dr. V.E. Annamalai** for providing us with all the support and encouragement.

We are highly indebted to **Dr. Chandrabose Aravindan**, Head of the Department for providing us the opportunity and facilities to take up this project.

We also thank our project guide, **Dr. K.S. Gayathri**, Associate Professor, Department of Information Technology for providing us with a suitable time schedule to complete our project.

We are truly grateful to our panel members **Dr. S. Karthika**, Associate Professor, **Dr. Mr. B. Senthil Kumar**, Assistant Professor, and **Dr. R. Swathika**, Assistant professor, Department of Information Technology for their patience, constructive comments and support during the entire course of this project work.

TABLE OF CONTENTS

ABSTRACT

There exists a global competition of banks on market share and an antagonism to gain competitive advantage, banks and financial bodies have been seen to suffer from high levels of non-performing loans.

Thus, lenders have long relied on credit scores to make lending decisions for firms and retail clients. Credit scoring refers to the process of evaluating an individual’s credit worthiness that reflects the level of credit risk and determines whether an application of the customer should be approved or declined. This project strives to create a suitable machine learning model for banks or financial institutions to determine a customer’s credit score. Credit scoring refers to the process of evaluating an individual’s creditworthiness that reflects the level of credit risk and determines whether an application of the customer should be approved or declined. Proposes to compare the accuracy and specificity of various machine learning algorithms and demonstrate these algorithms’ efficacy in successfully classifying customers into various risk classes, thus creating a risk analytical model for the financial institutions. The costs of incorrectly classifying a customer can be high, both financially and in terms of reputation. The development of a typical credit scorecard can be represented in three main stages, namely Dataset construction, modelling and documentation. Moreover, it aims to put more focus on the dataset construction and modelling part, while also giving importance to analyzing the results and developing a proper risk and profit model.

LIST OF FIGURES

LIST OF ACRONYMS

1. CTGAN – Conditional Generative Adversarial Network
2. SMOTE – Synthetic Minority Over-sampling Technique
3. MLP – Multi Layer Perceptron
4. SVM – Support Vector Machine
5. RBF – Radial Basis Function
6. POD – Probability of Default
7. ReLU – Rectified Linear Unit

1. INTRODUCTION

1.1. GENERAL

Credit score analysis using machine learning is a project aimed at developing a predictive model that can evaluate the creditworthiness of a borrower based on various features such as credit history, income, employment status, and other relevant factors.

The project involves collecting a dataset of historical credit records, cleaning and preprocessing the data, and then training a machine learning model using techniques such as regression analysis, decision trees, and neural networks to predict the likelihood of a borrower defaulting on their loan.

The ultimate goal of this project is to develop an accurate and reliable credit scoring system that financial institutions can use to assess the creditworthiness of potential borrowers and make informed lending decisions. By using machine learning techniques, the credit scoring system can be made more efficient and effective, allowing lenders to process loan applications faster while reducing the risk of default.

Overall, this project has the potential to revolutionize the lending industry by providing a more accurate and reliable credit scoring system, improving access to credit for borrowers, and reducing the risk of default for lenders.

1.2. NEED FOR STUDY

There are several reasons why the study of credit score analysis using machine learning is important.

Firstly, credit scores are a crucial factor in determining whether individuals and businesses can access credit, such as loans and credit cards. Machine learning can help to improve the accuracy and reliability of credit scoring systems, which can benefit both lenders and borrowers.

Secondly, the traditional methods used for credit scoring have limitations and can be biased, leading to unfair lending practices. Machine learning models can be designed to reduce or eliminate these biases, leading to fairer lending practices.

Thirdly, the study of credit score analysis using machine learning can help to improve financial inclusion by making it easier for individuals and businesses to access credit.

Finally, the study of credit score analysis using machine learning can have broader implications for the field of machine learning and data science. It can help researchers to explore and develop new techniques for handling complex data and making accurate predictions, which can be applied to other areas such as healthcare, transportation, and marketing.

1.3. OBJECTIVES

- Research more on the current existing problems in credit scoring by conversing with people actually in the industry to get a broader view on the current requirements and areas lagging behind in the field of assigning credit worthiness of each customer.
- Pre-process data using suitable data pre-processing techniques from publicly available credit approval dataset from real banks in such a way that the dataset is not imbalanced, thus avoiding the the low-default portfolio problem.
- Develop of a multi-class risk model for credit scoring using machine learning and deep learning techniques with higher accuracies than already existing implementations.
- After running the model on the pre-processed data, the output data is further analyzed to create a risk-profit model.

2. LITERATURE SURVEY

1. Kennedy, K. (2013) Credit scoring using machine learning. Doctoral thesis. Technological University Dublin :

Kennedy’s doctoral thesis “Credit Scoring using Machine Learning” aims to explore the potential of machine learning techniques in improving credit scoring models. The thesis discusses the limitations of traditional credit scoring models and highlights the benefits of using machine learning algorithms in developing more accurate and robust credit scoring models.

Kennedy’s research involves the use of various machine learning techniques such as decision trees, neural networks, support vector machines, and ensemble methods. He compares the performance of these techniques against traditional credit scoring models in terms of their predictive accuracy and stability. The research is conducted on a dataset of over 60,000 credit applications from an Irish financial institution.

The results show that machine learning techniques outperform traditional credit scoring models in terms of predictive accuracy, stability, and generalizability. The thesis concludes that machine learning algorithms have the potential to significantly improve credit scoring models, and that further research is needed to fully realize their potential in this domain. Overall, the thesis provides valuable insights into the use of machine learning in credit scoring and highlights its potential for future research and applications.

2. **Provenzano, A. R., Trifirò, D., Datteo, A., Giada, L., Jean, N., Riciputi, A, & Nordio C. (2020).Machine learning approach for credit scoring :**

The paper “Machine learning approach for credit scoring” by Provenzano et al. presents a machine learning framework for credit scoring based on a dataset of over 20,000 credit applications from an Italian bank. The paper proposes a hybrid machine learning model that combines logistic regression and decision tree algorithms for credit scoring.

The study compares the performance of the hybrid model with traditional credit scoring models, such as logistic regression and discriminant analysis, in terms of their predictive accuracy and stability. The results show that the proposed hybrid model outperforms traditional models in terms of predictive accuracy and stability.

The paper also explores the importance of feature selection in credit scoring models and uses a random forest algorithm to identify the most important variables for credit scoring. The study identifies variables such as income, age, and employment status as the most important predictors for creditworthiness. Overall, the paper demonstrates the potential of machine learning algorithms in improving credit scoring models and highlights the importance of feature selection in developing accurate and robust credit scoring models.

3. Ampountolas, A., Nyarko Nde, T., Date, P., & Constantinescu, C. (2021). A machine learning approach for micro-credit scoring

The paper “A machine learning approach for micro-credit scoring” by Ampountolas et al. presents a machine learning framework for micro-credit scoring. The study focuses on the specific context of micro-credit, which involves lending small amounts of money to individuals or small businesses that do not have access to traditional forms of credit.

The study uses a 15ighly15t of over 1,000 micro-credit loan applications from a microfinance institution in Ghana. The paper proposes a machine learning model that combines logistic regression and decision tree algorithms for micro-credit scoring. The model is trained on a set of features such as age, income, employment status, and loan purpose.

The results of the study show that the proposed machine learning model outperforms traditional credit scoring models such as logistic regression and discriminant analysis in terms of predictive accuracy and stability. The study also identifies age, income, and loan purpose as the most important predictors for micro-creditworthiness.

The paper concludes that the proposed machine learning model has the potential to improve the accuracy and efficiency of micro-credit scoring and can help increase access to credit for individuals and small businesses in emerging economies. Overall, the study highlights the importance of developing context-specific credit scoring models and demonstrates the potential of machine learning algorithms in this domain.

4. **Nyon, E. E., & Matshisela, N. (2018). Credit scoring using machine learning algorithims. Zimbabwe Journal of Science and Technology :**

The paper “Credit scoring using machine learning algorithms” by Nyon and Matshisela presents a machine learning approach for credit scoring based on a dataset of over 2,000 credit applications from a Zimbabwean financial institution. The study focuses on the use of machine learning algorithms such as decision trees, artificial neural networks, and k-nearest neighbor for credit scoring.

The study compares the performance of machine learning algorithms with traditional credit scoring models such as logistic regression and discriminant analysis in terms of their predictive accuracy and stability. The results show that machine learning algorithms outperform traditional models in terms of predictive accuracy, and k-nearest neighbor algorithm was found to be the most accurate model for credit scoring.

The paper also explores the importance of feature selection in credit scoring models and uses the Chi-Square test to identify the most important variables for creditworthiness. The study identifies variables such as employment status, income, and credit history as the most important predictors for creditworthiness.

Overall, the paper demonstrates the potential of machine learning algorithms in improving credit scoring models and highlights the importance of context-specific feature selection in developing accurate and robust credit scoring models.

5. **Abdou, H. A., & Pointon, J. (2011). Credit scoring, statistical techniques and evaluation criteria: a review of the literature. Intelligent systems in accounting, finance and management :**

The paper “Credit scoring, statistical techniques and evaluation criteria: a review of the literature” by Abdou and Pointon presents a comprehensive review of the literature on credit scoring. The study focuses on the use of statistical techniques in credit scoring and the evaluation criteria used to assess the performance of credit scoring models.

The paper provides a detailed overview of the different statistical techniques used in credit scoring, such as logistic regression, discriminant analysis, decision trees, neural networks, and support vector machines. The study compares the performance of these techniques in terms of their predictive accuracy, stability, and interpretability.

The paper also explores the different evaluation criteria used to assess the performance of credit scoring models, such as accuracy, precision, recall, and F1-score. The study discusses the advantages and disadvantages of each evaluation criterion and highlights the importance of choosing appropriate evaluation criteria based on the specific context of the credit scoring application.

Overall, the paper provides valuable insights into the use of statistical techniques in credit scoring and the importance of appropriate evaluation criteria in assessing the performance of credit scoring models. The study is a useful resource for researchers and practitioners interested in credit scoring and provides a foundation for future research in this field.

6. A Abdou, H. A., & Pointon, J. (2009). Credit scoring and decision making in Egyptian public sector banks. International Journal of Managerial Finance :

The paper “Credit scoring and decision making in Egyptian public sector banks” by Abdou and Pointon examines the credit scoring practices and decision-making processes of public sector banks in Egypt. The study focuses on the use of credit scoring models in assessing the creditworthiness of loan applicants and the role of credit scoring in the loan approval process.

The study uses a dataset of over 1,500 loan applications from three public sector banks in Egypt and analyzes the performance of different credit scoring models such as discriminant analysis, logistic regression, and neural networks. The study also examines the factors that influence loan approval decisions, such as collateral, loan size, and borrower characteristics.

The results of the study show that credit scoring models are widely used in public sector banks in Egypt and are considered important tools for assessing creditworthiness. The study also shows that collateral and loan size are the most important factors in loan approval decisions, while borrower characteristics such as age and education level have a relatively small impact.

The paper highlights the importance of context-specific credit scoring models and the need to tailor credit scoring models to the specific characteristics of the lending environment. The study provides valuable insights into the credit scoring practices and decision-making processes of public sector banks in emerging economies such as Egypt and can be useful for policymakers, researchers, and practitioners interested in credit scoring and lending practices in developing countries.

7. **Luo, C., Wu, D., & Wu, D. (2017). A deep learning approach for credit scoring using credit default swaps. Engineering Applications of Artificial Intelligence :**

The paper “A deep learning approach for credit scoring using credit default swaps” by Luo, Wu, and Wu proposes a deep learning-based approach for credit scoring using credit default swaps (CDS) data. The study aims to improve the accuracy of credit scoring by leveraging the information contained in CDS prices, which reflect market participants’ expectations about the creditworthiness of a borrower.

The study uses a da’aset of CDS prices for 78 companies from 2005 to 2015 and applies a deep learning model called a long short-term memory (LSTM) network to predict the probability of default for each company. The study compares the performance of the LSTM model with traditional credit scoring models such as logistic regression and random forest.

The results of the study show that the LSTM model outperforms traditional models in terms of predictive accuracy and stability. The study also shows that the inclusion of CDS data significantly improves the performance of credit scoring models, suggesting that market-based information can be valuable for assessing creditworthiness.

The19ighlyghts the potential of deep learning approaches and market-based data for credit scoring and provides a framework for incorporating market-based information into credit scoring models. The study is useful for researchers and practitioners interested in developing innovative credit scoring models that leverage alternative data sources and advanced machine learning techniques.

8. **Pol, S., & Ambekar, S. S. (2022). Predicting Credit Ratings using Deep Learning Models–An Analysis of the Indian IT Industry. Australasian Accounting, Business and Finance Journal:**

The paper “Predicting Credit Ratings using Deep Learning Models–An Analysis of the Indian IT Industry” by Pol and Ambekar investigates the effectiveness of deep learning models in predicting credit ratings for companies in the Indian IT industry. The study aims to explore whether deep learning models can outperform traditional credit rating models and provide more accurate and timely credit risk assessments.

The study uses a dataset of financial and non-financial information for 42 companies in the Indian IT industry from 2015 to 2020 and applies two deep learning models, namely the long short-term memory (LSTM) network and the convolutional neural network (CNN), to predict credit ratings. The study also compares the performance of the deep learning models with traditional credit rating models such as discriminant analysis and logistic regression.

The results of the study show that the LSTM and CNN models outperform traditional models in terms of predictive accuracy and stability. The study also identifies key financial and non-financial variables that are significant predictors of credit ratings, such as profitability, liquidity, leverage, market capitalization, and industry-specific factors.

The paper highlights the potential of deep learning models for credit rating prediction and provides valuable insights into the credit risk assessment of companies in the Indian IT industry. The study can be useful for researchers and practitioners interested in developing innovative credit rating models that leverage advanced machine learning techniques and alternative data sources.

9. Addo, P. M., Guegan, D., & Hassani, B. (2018). Credit risk analysis using machine and deep learning models :

The paper “Credit Risk Analysis Using Machine and Deep Learning Models” by Addo, Guegan, and Hassani investigates the use of machine and deep learning models for credit risk analysis. The study aims to compare the performance of different machine and deep learning models in predicting credit default risk and identify the key factors that affect credit risk.

The study uses a dataset of financial and non-financial information for 1,000 borrowers from a UK-based lending institution and applies various machine and deep learning models, including logistic regression, random forest, artificial neural networks, and convolutional neural networks, to predict credit default risk. The study also examines the importance of different variables, such as credit score, debt-to-income ratio, and loan purpose, in predicting credit risk.

The results of the study show that deep learning models, especially convolutional neural networks, outperform traditional machine learning models in terms of predictive accuracy and robustness. The study also identifies credit score, debt-to-income ratio, and loan amount as the most important predictors of credit risk.

The paper highlights the potential of deep learning models for credit risk analysis and provides valuable insights into the key factors that affect credit risk. The study can be useful for researchers and practitioners interested in developing innovative credit risk models that leverage advanced machine learning techniques and alternative data sources.

10. **Ferreira, F. A., Spahr, R. W., Gavancha, I. F., & Çipi, A. (2013). Readjusting trade-offs among criteria in internal ratings of credit-scoring: an empirical essay of risk analysis in mortgage loans. Journal of Business Economics and Management :**

The paper "Readjusting Trade-Offs Among Criteria in Internal Ratings of Credit-Scoring: An Empirical Essay of Risk Analysis in Mortgage Loans" by Ferreira, Spahr, Gavancha, and Çipi investigates the use of internal credit-scoring models in mortgage lending. The study aims to explore the trade-offs among different criteria used in internal credit-scoring models and the impact of these trade-offs on credit risk assessment.

The study uses a dataset of mortgage loans originated by a Portuguese bank from 2006 to 2010 and applies various statistical techniques, including logistic regression, discriminant analysis, and neural networks, to model the credit risk of the loans. The study also examines the impact of different criteria, such as borrower characteristics, loan characteristics, and macroeconomic factors, on credit risk assessment.

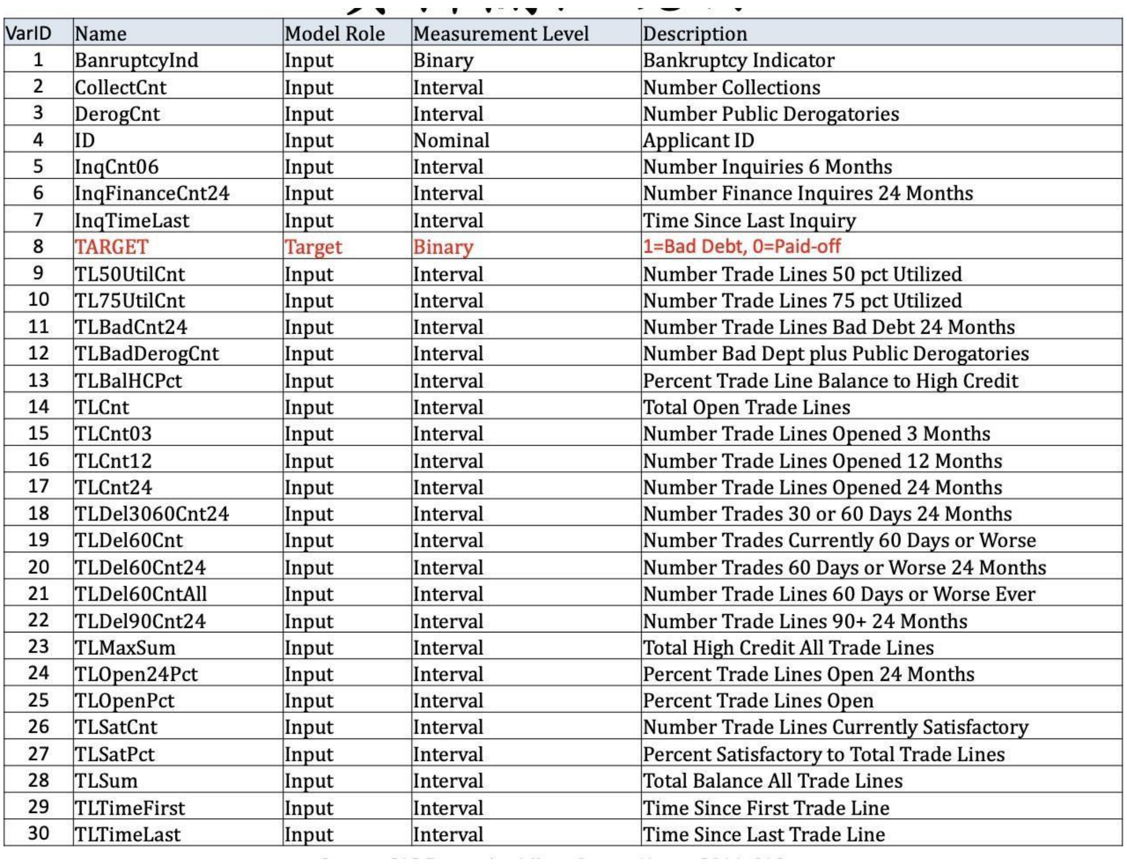
The results of the study show that borrower characteristics, such as income, age, and occupation, are the most important predictors of credit risk in mortgage lending. The study also identifies the trade-offs among different criteria, such as loan-to-value ratio, debt-to-income ratio, and credit score, and their impact on credit risk assessment.

The paper highlights the importance of internal credit-scoring models in mortgage lending and provides valuable insights into the trade-offs among different criteria used in these models.

3. PROPOSED METHODOLOGY:

3.1. DATASET - I

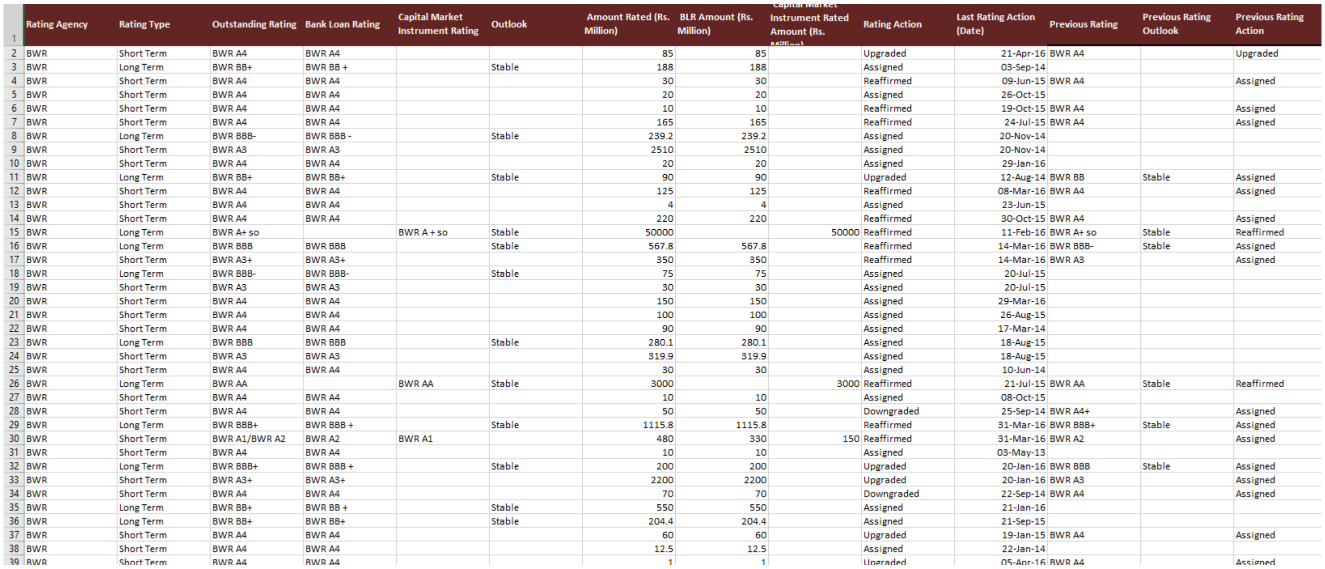
- The dataset chosen – SAS Enterprise Miner’s Credit Risk Dataset.
- Data accessed from a case study conducted by professor Min-Yuh Day of Tamkang University.
- https://mail.tku.edu.tw/myday/teaching/1062/BDM/1062BDM09_Big_Data_Mining.pdf
- 30 Attributes. 3000 instances.
- Has necessary attributes to calculate estimated losses and revenue of bad and good loans respectively.



VarID	Name	Model Role	Measurement Level	Description
1	BanruptcyInd	Input	Binary	Bankruptcy Indicator
2	CollectCnt	Input	Interval	Number Collections
3	DerogCnt	Input	Interval	Number Public Derogatories
4	ID	Input	Nominal	Applicant ID
5	InqCnt06	Input	Interval	Number Inquiries 6 Months
6	InqFinanceCnt24	Input	Interval	Number Finance Inquires 24 Months
7	InqTimeLast	Input	Interval	Time Since Last Inquiry
8	TARGET	Target	Binary	1=Bad Debt, 0=Paid-off
9	TL50UtilCnt	Input	Interval	Number Trade Lines 50 pct Utilized
10	TL75UtilCnt	Input	Interval	Number Trade Lines 75 pct Utilized
11	TLBadCnt24	Input	Interval	Number Trade Lines Bad Debt 24 Months
12	TLBadDerogCnt	Input	Interval	Number Bad Dept plus Public Derogatories
13	TLBalHCPct	Input	Interval	Percent Trade Line Balance to High Credit
14	TLCnt	Input	Interval	Total Open Trade Lines
15	TLCnt03	Input	Interval	Number Trade Lines Opened 3 Months
16	TLCnt12	Input	Interval	Number Trade Lines Opened 12 Months
17	TLCnt24	Input	Interval	Number Trade Lines Opened 24 Months
18	TLDel3060Cnt24	Input	Interval	Number Trades 30 or 60 Days 24 Months
19	TLDel60Cnt	Input	Interval	Number Trades Currently 60 Days or Worse
20	TLDel60Cnt24	Input	Interval	Number Trades 60 Days or Worse 24 Months
21	TLDel60CntAll	Input	Interval	Number Trade Lines 60 Days or Worse Ever
22	TLDel90Cnt24	Input	Interval	Number Trade Lines 90+ 24 Months
23	TLMaxSum	Input	Interval	Total High Credit All Trade Lines
24	TLOpen24Pct	Input	Interval	Percent Trade Lines Open 24 Months
25	TLOpenPct	Input	Interval	Percent Trade Lines Open
26	TLSatCnt	Input	Interval	Number Trade Lines Currently Satisfactory
27	TLSatPct	Input	Interval	Percent Satisfactory to Total Trade Lines
28	TLSum	Input	Interval	Total Balance All Trade Lines
29	TLTimeFirst	Input	Interval	Time Since First Trade Line
30	TLTimeLast	Input	Interval	Time Since Last Trade Line

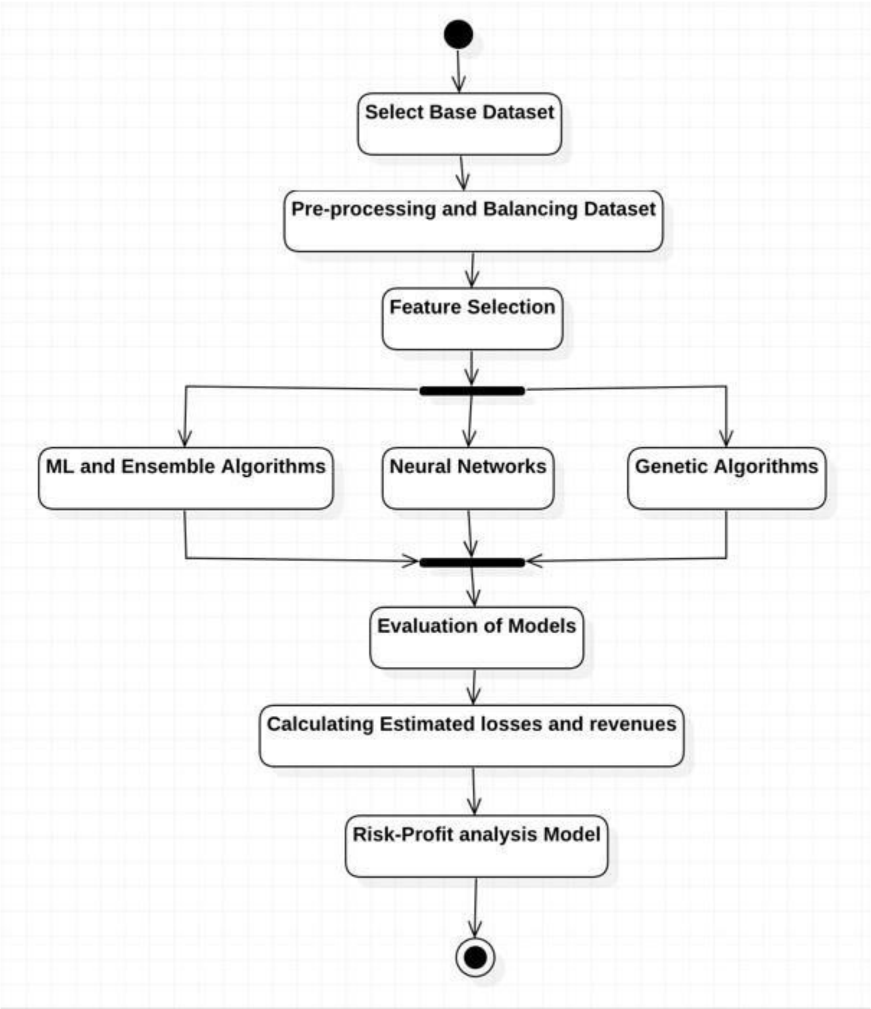
DATASET – II

- Dataset chosen – CRISIL (Credit Rating Agency)
- 40 Attributes. 2185 instances.



1	Rating Agency	Rating Type	Outstanding Rating	Bank Loan Rating	Capital Market Instrument Rating	Outlook	Amount Rated (Rs. Million)	BLR Amount (Rs. Million)	Capital Market Instrument Rated Amount (Rs. Million)	Rating Action	Last Rating Action (Date)	Previous Rating	Previous Rating Outlook	Previous Rating Action
2	BWR	Short Term	BWR A4	BWR A4			85	85		Upgraded	21-Apr-16	BWR A4		Upgraded
3	BWR	Long Term	BWR BB+	BWR BB +		Stable	188	188		Assigned	03-Sep-14			
4	BWR	Short Term	BWR A4	BWR A4			30	30		Reaffirmed	09-Jun-15	BWR A4		Assigned
5	BWR	Short Term	BWR A4	BWR A4			20	20		Assigned	26-Oct-15			
6	BWR	Short Term	BWR A4	BWR A4			10	10		Reaffirmed	19-Oct-15	BWR A4		Assigned
7	BWR	Short Term	BWR A4	BWR A4			165	165		Reaffirmed	24-Jul-15	BWR A4		Assigned
8	BWR	Long Term	BWR BBB-	BWR BBB -		Stable	239.2	239.2		Assigned	20-Nov-14			
9	BWR	Short Term	BWR A3	BWR A3			2510	2510		Assigned	20-Nov-14			
10	BWR	Short Term	BWR A4	BWR A4			20	20		Assigned	29-Jan-16			
11	BWR	Long Term	BWR BB+	BWR BB+		Stable	90	90		Upgraded	12-Aug-14	BWR BB	Stable	Assigned
12	BWR	Short Term	BWR A4	BWR A4			125	125		Reaffirmed	08-Mar-16	BWR A4		Assigned
13	BWR	Short Term	BWR A4	BWR A4			4	4		Assigned	23-Jun-15			
14	BWR	Short Term	BWR A4	BWR A4			220	220		Reaffirmed	30-Oct-15	BWR A4		Assigned
15	BWR	Long Term	BWR A+ so		BWR A + so	Stable	50000		50000	Reaffirmed	11-Feb-16	BWR A+ so	Stable	Reaffirmed
16	BWR	Long Term	BWR BBB	BWR BBB		Stable	567.8	567.8		Reaffirmed	14-Mar-16	BWR BBB-	Stable	Assigned
17	BWR	Short Term	BWR A3+	BWR A3+			350	350		Reaffirmed	14-Mar-16	BWR A3		Assigned
18	BWR	Long Term	BWR BBB-	BWR BBB-		Stable	75	75		Assigned	20-Jul-15			
19	BWR	Short Term	BWR A3	BWR A3			30	30		Assigned	20-Jul-15			
20	BWR	Short Term	BWR A4	BWR A4			150	150		Assigned	29-Mar-16			
21	BWR	Short Term	BWR A4	BWR A4			100	100		Assigned	26-Aug-15			
22	BWR	Short Term	BWR A4	BWR A4			90	90		Assigned	17-Mar-14			
23	BWR	Long Term	BWR BBB	BWR BBB		Stable	280.1	280.1		Assigned	18-Aug-15			
24	BWR	Short Term	BWR A3	BWR A3			319.9	319.9		Assigned	18-Aug-15			
25	BWR	Short Term	BWR A4	BWR A4			30	30		Assigned	10-Jun-14			
26	BWR	Long Term	BWR AA		BWR AA	Stable	3000		3000	Reaffirmed	21-Jul-15	BWR AA	Stable	Reaffirmed
27	BWR	Short Term	BWR A4	BWR A4			10	10		Assigned	08-Oct-15			
28	BWR	Short Term	BWR A4	BWR A4			50	50		Downgraded	25-Sep-14	BWR A4+		Assigned
29	BWR	Long Term	BWR BBB+	BWR BBB +		Stable	1115.8	1115.8		Reaffirmed	31-Mar-16	BWR BBB+	Stable	Assigned
30	BWR	Short Term	BWR A1/BWR A2	BWR A2	BWR A1		480	330	150	Reaffirmed	31-Mar-16	BWR A2		Assigned
31	BWR	Short Term	BWR A4	BWR A4			10	10		Assigned	03-May-13			
32	BWR	Long Term	BWR BBB+	BWR BBB +		Stable	200	200		Upgraded	20-Jan-16	BWR BBB	Stable	Assigned
33	BWR	Short Term	BWR A3+	BWR A3+			2200	2200		Upgraded	20-Jan-16	BWR A3		Assigned
34	BWR	Short Term	BWR A4	BWR A4			70	70		Downgraded	22-Sep-14	BWR A4		Assigned
35	BWR	Long Term	BWR BB+	BWR BB +		Stable	550	550		Assigned	21-Jan-16			
36	BWR	Long Term	BWR BB+	BWR BB+		Stable	204.4	204.4		Assigned	21-Sep-15			
37	BWR	Short Term	BWR A4	BWR A4			60	60		Upgraded	19-Jan-15	BWR A4		Assigned
38	BWR	Short Term	BWR A4	BWR A4			12.5	12.5		Assigned	22-Jan-14			
39	BWR	Short Term	BWR A4	BWR A4			1	1		Upgraded	05-Apr-16	BWR A4		Assigned

3.2. SYSTEM DIAGRAM


Select Base Dataset
Pre-processing and Balancing Dataset
Feature Selection
ML and Ensemble Algorithms
Neural Networks
Genetic Algorithms
Evaluation of Models
Calculating Estimated losses and revenues
Risk-Profit analysis Model

3.3. PRE-PROCESSING DATASET

First, to make sure the dataset doesn’t have any NULL values, I start by analysing the value distribution of attributes with missing values. The distribution of almost all the attributes has outliers, which can definitely affect the mean value; hence, the median value is chosen for missing data imputations.

Next, the dataset size is increased by using CTGAN, which is one of the deep learning synthetic data generators for single table data that can draw knowledge from the original data and produce highly accurate artificial data.

Up next, to counter the low-default portfolio problem, the class imbalance has to be mitigated. So two of the most prominent oversampling methods are used, and we choose the one that yields the best accuracy. The first method is Synthetic Minority Oversampling Technique, or SMOTE, which synthetically creates new data based on the trends and design of the old one. The next method is bootstrapping, which randomly chooses attribute values to create new row values. If any row is exactly repeated, it is promptly deleted to avoid overfitting.

3.4. FEATURE ENGINEERING

The next step is to reduce the number of features by reducing lesser correlated features using correlation coefficient analysis. In this model, feats that are highly related to each other contribute the same value to training the model. So to avoid overfitting, one of them is removed. But the pre- requisite for using this method is that the feature values should be normally distributed, which should be checked priorly.

The second feature-reducing method is via the mutual information gain value, which is found for each attribute, and only the top ten attributes are chosen for further processing. The information gain value denotes how much an attribute affects the outcome class of the instance. Therefore, removing the less significant features or attributes is important in training the model better.

Mutual information, which measures the level of knowledge gained about one random variable through the other random variable, is a measure between two random variables, X and Y. It is given by

$$I(X;Y) = \int_X \int_Y p(x,y) \log \frac{p(x,y)}{p(x)p(y)} dxdy,$$

Fig: Mutual Information Formula

3.5. MODELLING

As the outcome classes will be predicted using an ensemble method, the chosen machine learning algorithms for the base learners are logistic regression, support vector machines (using 4 different linear and non-linear type kernels), decision trees, random forests, and extreme gradient boosting (XGBoost).

As discussed in the literature survey, ensemble techniques and deep learning techniques yielded good accuracies in suitable conditions; therefore, the use of a deep learning model such as Multi- Layer Perceptron (MLP) as a base learner for the stacking algorithm will be beneficial, as dataset patterns not found by conventional algorithms could be recognized by a deep learning algorithm like MLP. Then, these base learners will be stacked with XGBoost again as the secondary layer stacking model, where the outputs of the base learners will be fed as inputs to XGBoost again to yield better accuracy.

Finally, basic performance measures such as the accuracy score, precision, recall, f1-score, support, macro average, and weighted average will be used to evaluate. These metrics will be divided for each output class as well as for the overall model. The models mentioned above are described in detail below:

1. Logistic Regression

Logistic regression is a statistical method used to model the relationship between a binary dependent variable (also known as the response variable) and one or more independent variables (also known as predictors). In other words, it is used to predict the probability of an event occurring (such as a customer making a purchase or a person having a disease) based on one or more predictor variables.

The output of a logistic regression model is a logistic function, which is an S-shaped curve that ranges between 0 and 1. This curve represents the predicted probability of the event occurring for different values of the predictor variables. The logistic function is created by applying a transformation (called the logistic transformation) to a linear combination of the predictor variables, where the coefficients of the linear combination are estimated from the data.

The logistic regression model is trained by estimating the coefficients of the linear combination using a method called maximum likelihood estimation. The goal of this method is to find the set of coefficients that maximizes the likelihood of observing the actual response values in the training data, given the predictor variables.

Once the logistic regression model is trained, it can be used to predict the probability of the event occurring for new data points. The predicted probability can then be used to make binary decisions (such as whether to approve or reject a loan application) by applying a decision threshold (such as 0.5) to the predicted probability. If the predicted probability is above the threshold, the event is predicted to occur; if it is below the threshold, the event is predicted not to occur.

2. Decision Tree

A decision tree is a supervised machine learning algorithm that is used for classification and regression tasks. It is a tree-like model where each internal node represents a test on a feature or attribute of the input data, each branch represents the outcome of the test, and each leaf node represents a class label or a numerical value.

The decision tree algorithm builds the tree by recursively splitting the data into subsets based on the value of a particular attribute or feature. The goal of the algorithm is to create a tree that makes accurate predictions on new data points. The tree is constructed by selecting the feature that results in the most information gain or the least impurity at each node.

Once the decision tree is constructed, it can be used to make predictions on new data points by traversing the tree from the root node to a leaf node, based on the values of the input features. The leaf node that is reached represents the predicted class label or numerical value for the input data point.

3. Support Vector Machine

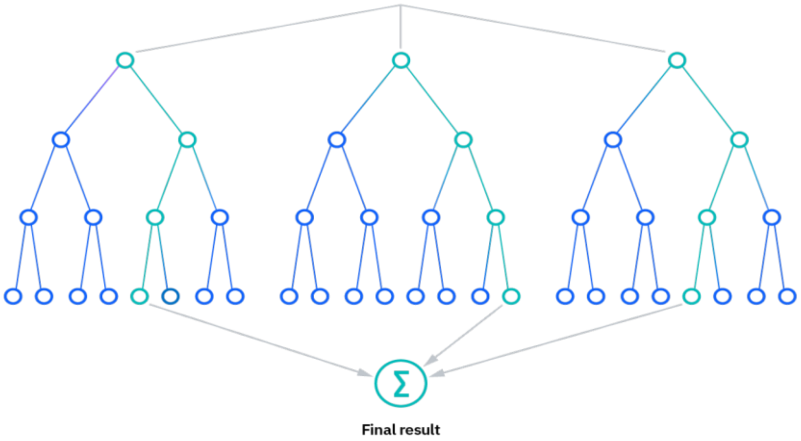
A support vector machine (SVM) is a supervised machine learning algorithm used for classification and regression tasks. It works by finding a hyperplane in a high-dimensional space that maximally separates the classes or fits the regression data.

In the case of binary classification, an SVM aims to find a hyperplane that separates the data into two classes, such that the margin between the hyperplane and the closest data points (called support vectors) of each class is maximized. The hyperplane is defined by a linear combination of the input features, and the coefficients of the linear combination are determined by solving an optimization problem that minimizes a cost function subject to the constraint that the data points are correctly classified.

If the data is not linearly separable, an SVM can use a kernel function to transform the input features into a higher-dimensional space where the classes are separable by a hyperplane. The kernel function calculates the dot product of the transformed features in the high-dimensional space, without actually computing the transformation explicitly. Common kernel functions include linear, polynomial, and radial basis function (RBF) kernels.

In the case of regression, an SVM aims to find a hyperplane that fits the data such that the error between the predicted values and the actual values is minimized. The hyperplane is defined by a linear combination of the input features, and the coefficients of the linear combination are determined by solving an optimization problem that minimizes a cost function subject to the constraint that the error between the predicted values and the actual values is less than a specified value.

4. Random Forest



Random forest is a supervised machine learning algorithm that is used for classification and regression tasks. It is an ensemble method that combines multiple decision trees to improve the accuracy and robustness of the model.

The random forest algorithm works by building a large number of decision trees, each of which is trained on a randomly selected subset of the training data and a randomly selected subset of the input features. This process is called bagging (short for bootstrap aggregating) and helps to reduce overfitting and increase the diversity of the trees.

In addition to bagging, the random forest algorithm introduces another source of randomness called feature randomization. This means that at each node of the decision tree, only a randomly selected subset of the input features is considered for splitting the data. This helps to prevent the algorithm from relying too much on any single feature and encourages the trees to be less correlated with each other.

Once the random forest is trained, it can be used to make predictions on new data points by aggregating the predictions of all the trees in the forest. For classification tasks, the class label with the highest frequency among the trees is chosen as the final prediction. For regression tasks, the average of the predicted values of all the trees is taken as the final prediction.

5. Naïve-Bayes

I Bayes is a supervised machine learning algorithm used for classification tasks. It is based on Bayes' theorem, which describes the probability of a hypothesis given some evidence or data.

I Bayes works by modeling the joint probability distribution of the input features and the class labels. It assumes that the features are conditionally independent given the class label, which means that the probability of observing a particular combination of features is simply the product of the probabilities of each individual feature given the class label.

To classify a new data point, I Bayes calculates the posterior probability of each class label given the observed features, using Bayes' theorem. The class label with the highest posterior probability is chosen as the final prediction.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Naïve Bayes Formula

6. XGBoost

XGBoost (Extreme Gradient Boosting) is a supervised machine learning algorithm used for classification and regression tasks. It is an implementation of the gradient boosting algorithm that uses a tree-based approach.

XGBoost works by building an ensemble of decision trees sequentially, where each new tree is trained to correct the errors of the previous trees. During training, the algorithm calculates a loss function that measures the difference between the predicted values and the actual values, and updates the model parameters (i.e., the weights and biases of the decision tree nodes) to minimize the loss.

Once the XGBoost model is trained, it can be used to make predictions on new data points by aggregating the predictions of all the decision trees in the ensemble. For classification tasks, the class label with the highest probability among the trees is chosen as the final prediction. For regression tasks, the average of the predicted values of all the trees is taken as the final prediction.

7. Multi-Layer perceptions:

Multi-layer perceptron (MLP) is a type of artificial neural network (ANN) that is used for supervised machine learning tasks, such as classification and regression. It consists of multiple layers of interconnected artificial neurons (also known as nodes or units), where each neuron in one layer is connected to all the neurons in the adjacent layer.

The input layer of an MLP receives the input data, and each neuron in the layer represents one input feature. The output layer of the MLP produces the output predictions, and each neuron in the layer represents one class label (for classification tasks) or one continuous variable (for regression tasks).

The hidden layers of the MLP are responsible for transforming the input data into a representation that is more suitable for the given task. Each neuron in a hidden layer receives the weighted sum of the outputs of the neurons in the previous layer, applies an activation function to the sum, and passes the result to the neurons in the next layer.

The activation function introduces nonlinearity into the MLP, which allows it to learn complex and nonlinear relationships between the input and output data. Common activation functions used in MLPs include sigmoid, hyperbolic tangent (tanh), and rectified linear unit (ReLU).

During training, the MLP adjusts the weights of the connections between the neurons using an optimization algorithm such as backpropagation, to minimize a loss function that measures the difference between the predicted and actual values. The weights are updated iteratively using the gradient of the loss function with respect to the weights, until convergence or a stopping criterion is reached.

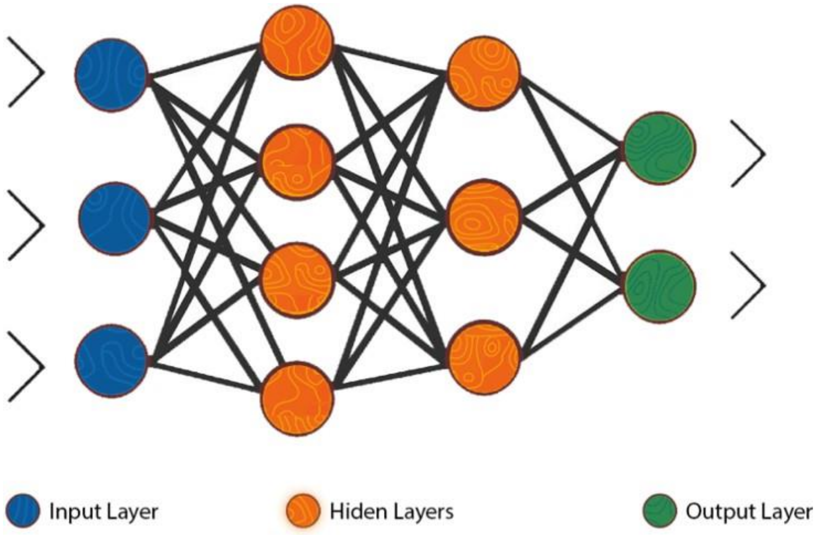


Fig: Multi-Layer Perceptron Visualized

3.6. RISK-PROFIT ANALYSIS

Using the outcome of the model with the highest accuracy from the previous model, the probability of default (POD) is calculated, which is the probability that a borrower fails to pay back the loan or defaults on paying it back.

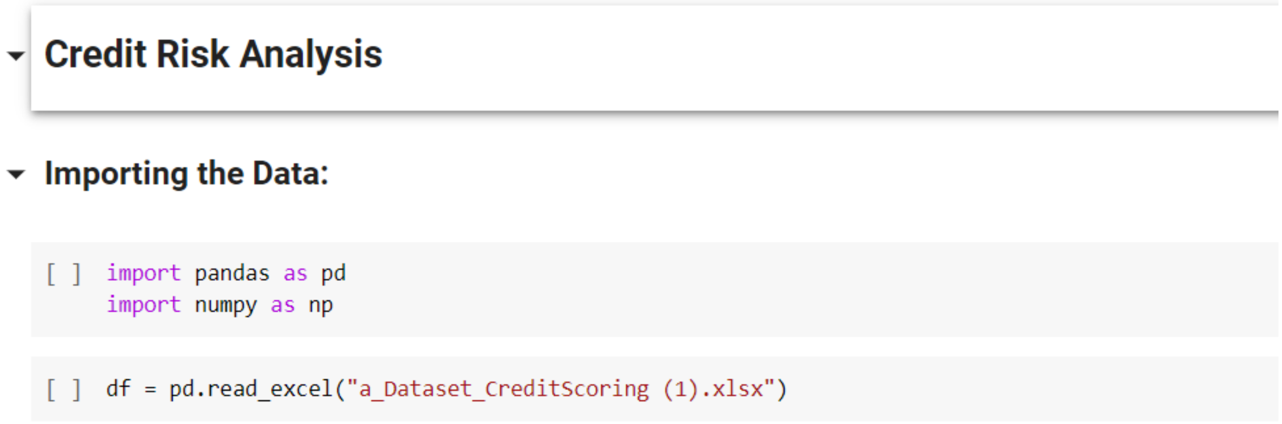
a) FROM THE GIVEN DATASET (FOR GOOD LOANS): Estimated Revenue = TLMaxSum

b) FROM THE GIVEN DATASET (FOR BAD LOANS): Exposure at Default (EAD) = TLSum

Recovery Rate (RR) = (TLMaxSum – TLSum)/(TLMaxSum)
Loss given default (LGD) = EAD * (1– RR)
Estimated Loss (EL) = POD * LGD

Using the estimated losses and revenues, we can use decile and percentile methods to analyses which set of people or up to which percentile or decile of people get granted loans to maximize revenue, as described above in the proposed system module.

4. RESULTS [DATASET -I]



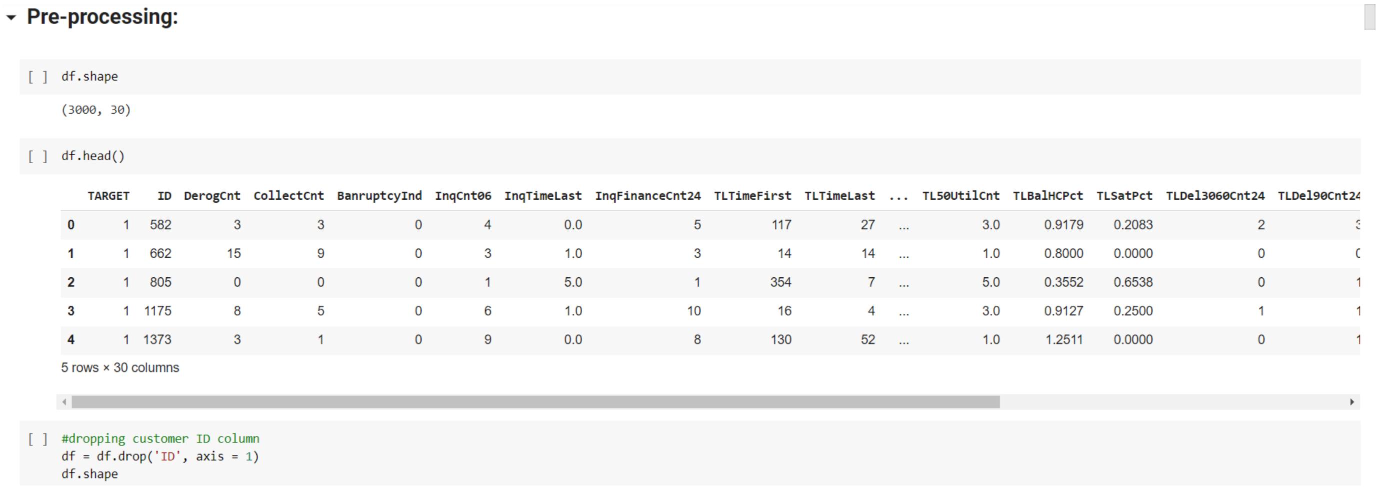
Credit Risk Analysis

Importing the Data:

```
[ ]  import pandas as pd
     import numpy as np
```

```
[ ]  df = pd.read_excel("a_Dataset_CreditScoring (1).xlsx")
```

4.1. PRE-PROCESSING:



Pre-processing:

```
[ ]  df.shape

     (3000, 30)
```

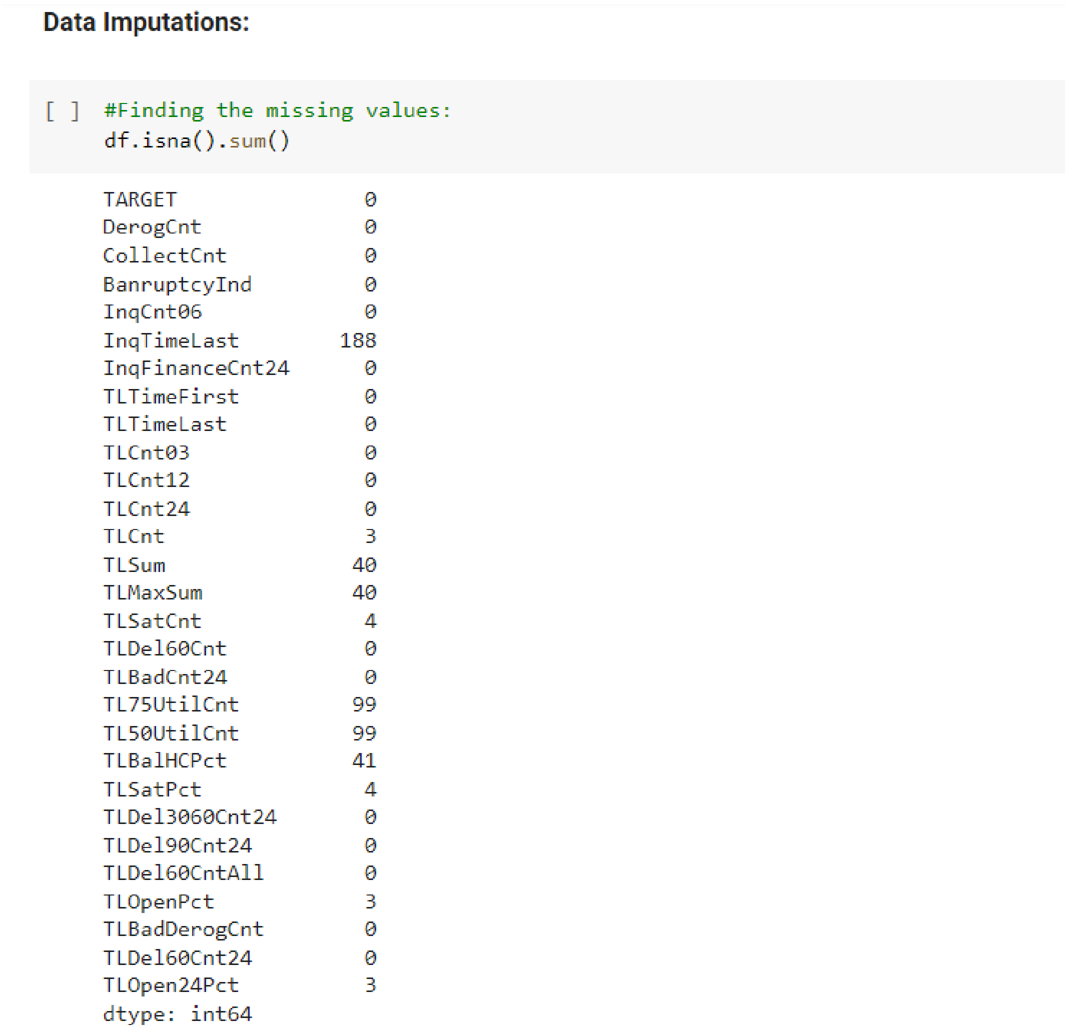
```
[ ]  df.head()
```

	TARGET	ID	DerogCnt	CollectCnt	BanruptcyInd	InqCnt06	InqTimeLast	InqFinanceCnt24	TLTimeFirst	TLTimeLast	...	TL50UtilCnt	TLBalHCPct	TLSatPct	TLDel3060Cnt24	TLDel90Cnt24
0	1	582	3	3	0	4	0.0	5	117	27	...	3.0	0.9179	0.2083	2	3
1	1	662	15	9	0	3	1.0	3	14	14	...	1.0	0.8000	0.0000	0	0
2	1	805	0	0	0	1	5.0	1	354	7	...	5.0	0.3552	0.6538	0	1
3	1	1175	8	5	0	6	1.0	10	16	4	...	3.0	0.9127	0.2500	1	1
4	1	1373	3	1	0	9	0.0	8	130	52	...	1.0	1.2511	0.0000	0	1

5 rows × 30 columns

```
[ ]  #dropping customer ID column
     df = df.drop('ID', axis = 1)
     df.shape
```

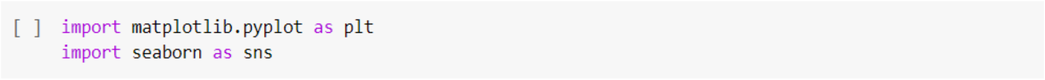
4.1.1. DATA IMPUTATIONS



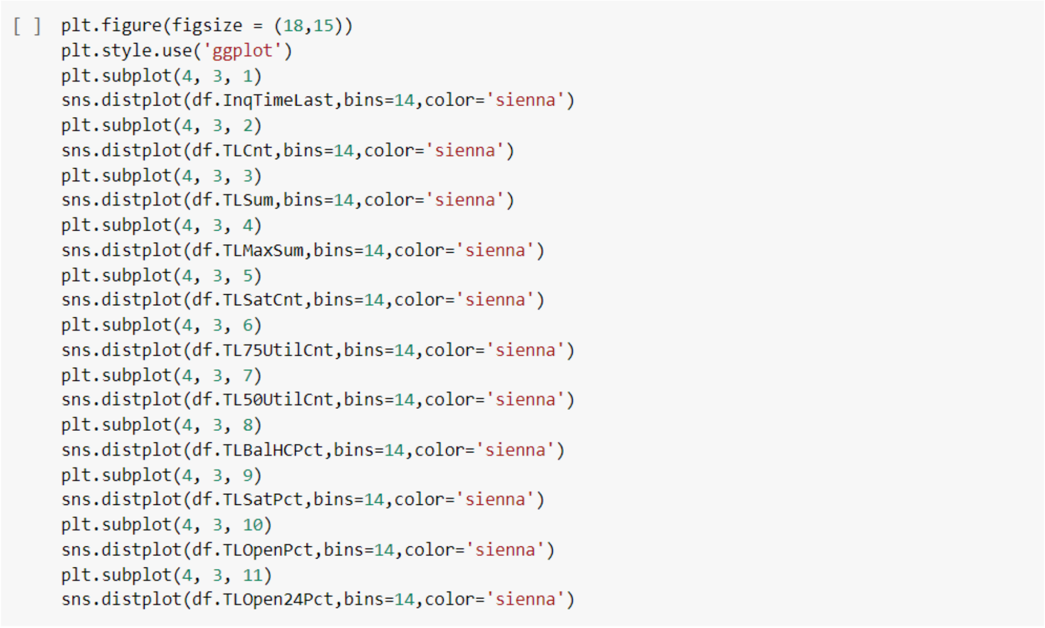
Data Imputations:

```
[ ]  #Finding the missing values:
     df.isna().sum()
```

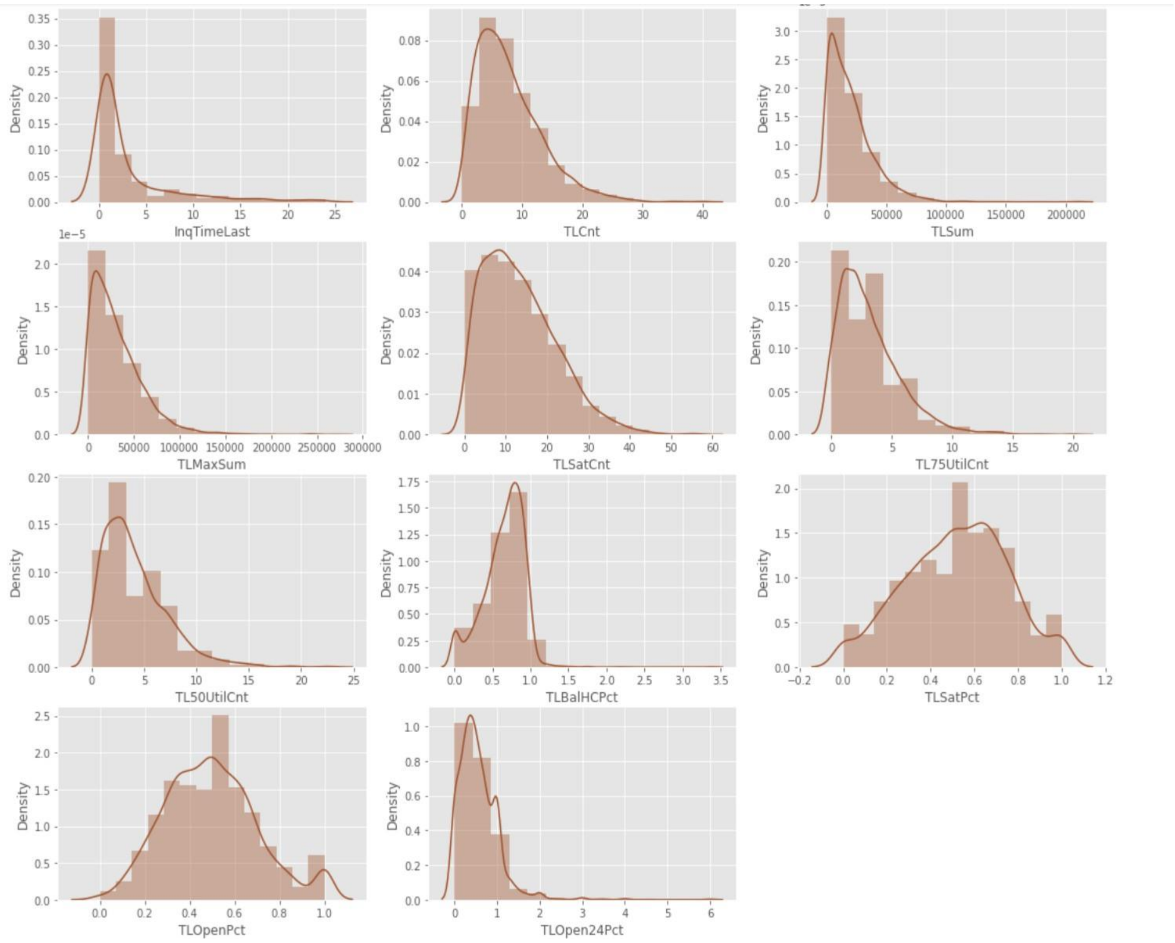
```
TARGET               0
DerogCnt             0
CollectCnt           0
BanruptcyInd         0
InqCnt06             0
InqTimeLast        188
InqFinanceCnt24      0
TLTimeFirst          0
TLTimeLast           0
TLCnt03              0
TLCnt12              0
TLCnt24              0
TLCnt                3
TLSum               40
TLMaxSum            40
TLSatCnt             4
TLDel60Cnt           0
TLBadCnt24           0
TL75UtilCnt         99
TL50UtilCnt         99
TLBalHCPct          41
TLSatPct             4
TLDel3060Cnt24       0
TLDel90Cnt24         0
TLDel60CntAll        0
TLOpenPct            3
TLBadDerogCnt        0
TLDel60Cnt24         0
TLOpen24Pct          3
dtype: int64
```



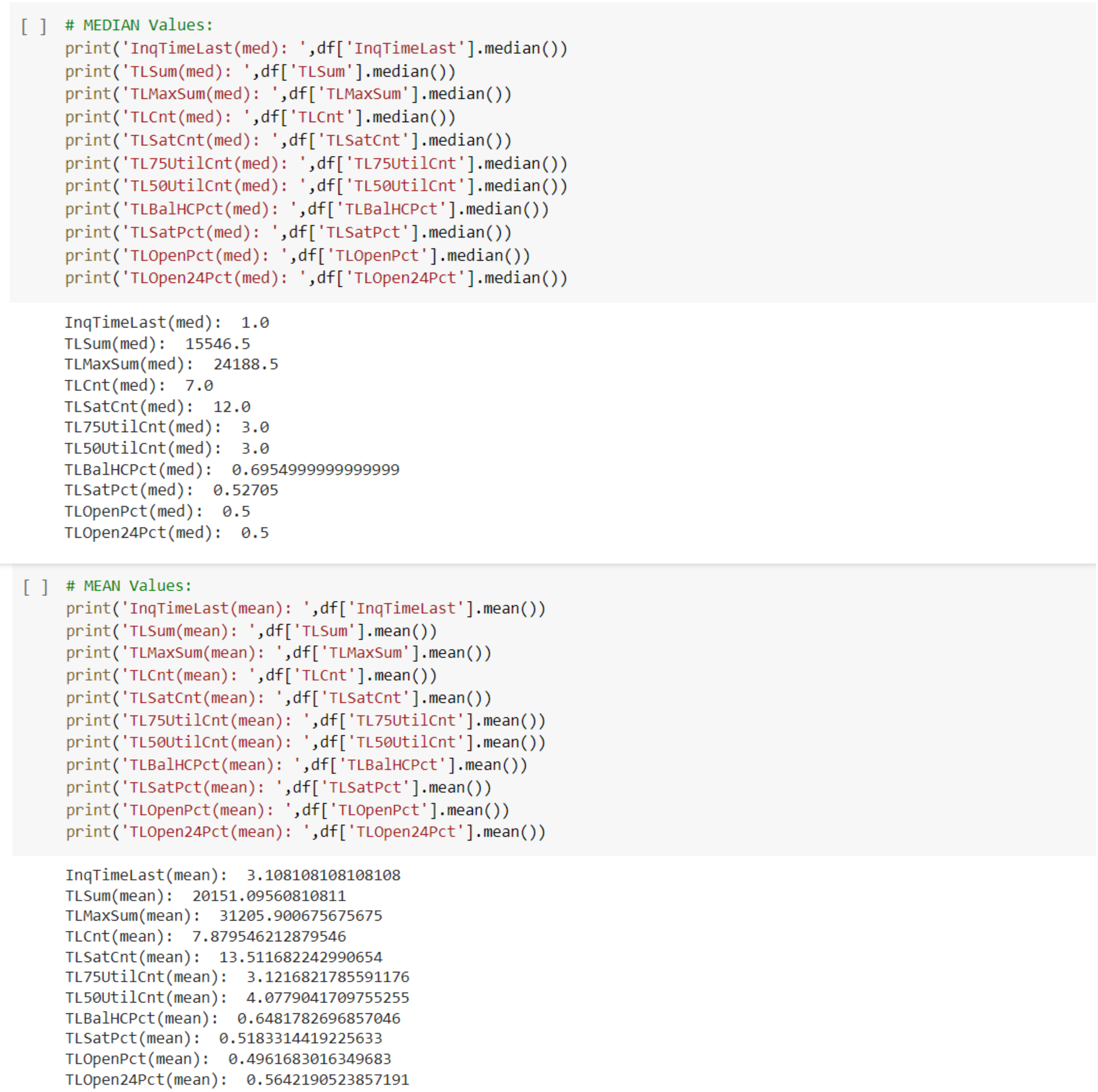
```
[ ]  import matplotlib.pyplot as plt
     import seaborn as sns
```



```
[ ]  plt.figure(figsize = (18,15))
     plt.style.use('ggplot')
     plt.subplot(4, 3, 1)
     sns.distplot(df.InqTimeLast,bins=14,color='sienna')
     plt.subplot(4, 3, 2)
     sns.distplot(df.TLCnt,bins=14,color='sienna')
     plt.subplot(4, 3, 3)
     sns.distplot(df.TLSum,bins=14,color='sienna')
     plt.subplot(4, 3, 4)
     sns.distplot(df.TLMaxSum,bins=14,color='sienna')
     plt.subplot(4, 3, 5)
     sns.distplot(df.TLSatCnt,bins=14,color='sienna')
     plt.subplot(4, 3, 6)
     sns.distplot(df.TL75UtilCnt,bins=14,color='sienna')
     plt.subplot(4, 3, 7)
     sns.distplot(df.TL50UtilCnt,bins=14,color='sienna')
     plt.subplot(4, 3, 8)
     sns.distplot(df.TLBalHCPct,bins=14,color='sienna')
     plt.subplot(4, 3, 9)
     sns.distplot(df.TLSatPct,bins=14,color='sienna')
     plt.subplot(4, 3, 10)
     sns.distplot(df.TLOpenPct,bins=14,color='sienna')
     plt.subplot(4, 3, 11)
     sns.distplot(df.TLOpen24Pct,bins=14,color='sienna')
```



Value distribution of attributes with missing values, to figure out what type of imputation to fill out missing values. Figure 8 showcases that there exists many outliers for each of the attributes, thus median is chosen, because mean values get skewed/biased by these extreme values.

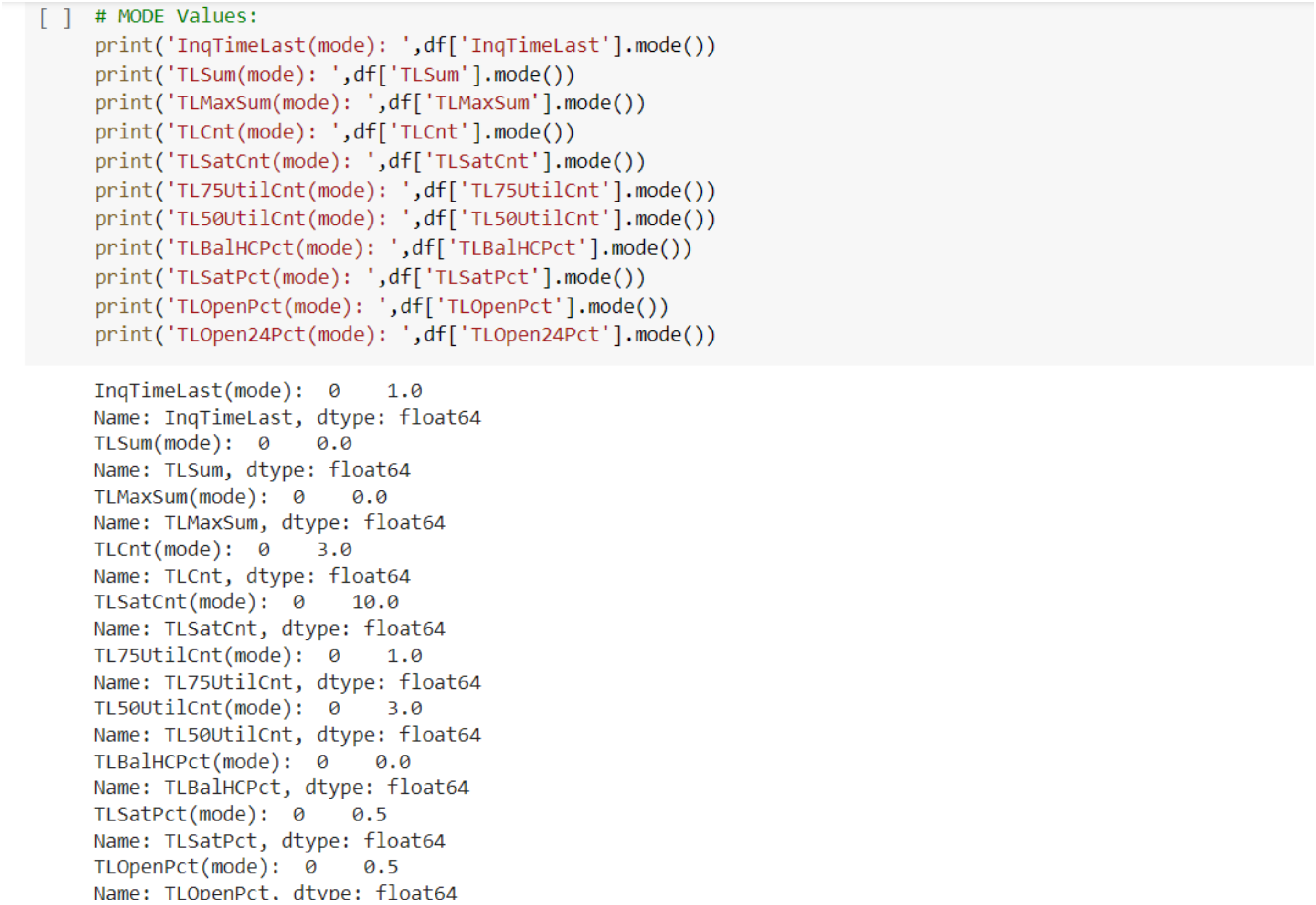


```
[ ]  # MEDIAN Values:
     print('InqTimeLast(med): ',df['InqTimeLast'].median())
     print('TLSum(med): ',df['TLSum'].median())
     print('TLMaxSum(med): ',df['TLMaxSum'].median())
     print('TLCnt(med): ',df['TLCnt'].median())
     print('TLSatCnt(med): ',df['TLSatCnt'].median())
     print('TL75UtilCnt(med): ',df['TL75UtilCnt'].median())
     print('TL50UtilCnt(med): ',df['TL50UtilCnt'].median())
     print('TLBalHCPct(med): ',df['TLBalHCPct'].median())
     print('TLSatPct(med): ',df['TLSatPct'].median())
     print('TLOpenPct(med): ',df['TLOpenPct'].median())
     print('TLOpen24Pct(med): ',df['TLOpen24Pct'].median())
```

```
InqTimeLast(med):  1.0
TLSum(med):  15546.5
TLMaxSum(med):  24188.5
TLCnt(med):  7.0
TLSatCnt(med):  12.0
TL75UtilCnt(med):  3.0
TL50UtilCnt(med):  3.0
TLBalHCPct(med):  0.6954999999999999
TLSatPct(med):  0.52705
TLOpenPct(med):  0.5
TLOpen24Pct(med):  0.5
```

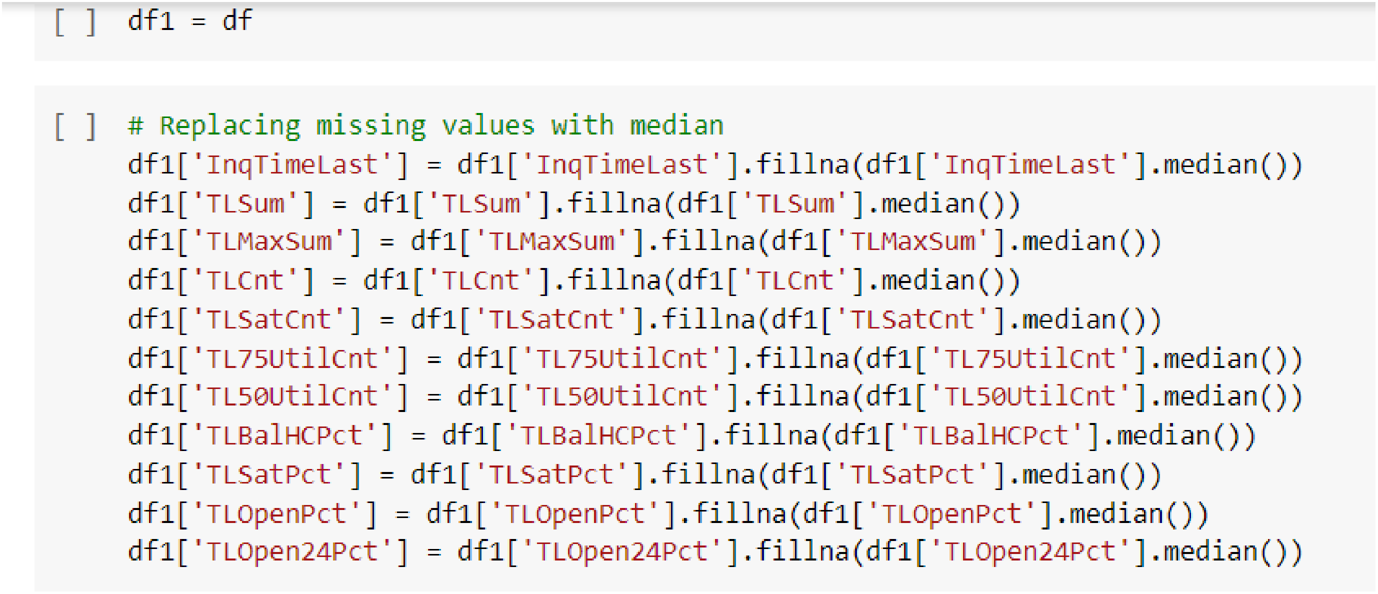
```
[ ]  # MEAN Values:
     print('InqTimeLast(mean): ',df['InqTimeLast'].mean())
     print('TLSum(mean): ',df['TLSum'].mean())
     print('TLMaxSum(mean): ',df['TLMaxSum'].mean())
     print('TLCnt(mean): ',df['TLCnt'].mean())
     print('TLSatCnt(mean): ',df['TLSatCnt'].mean())
     print('TL75UtilCnt(mean): ',df['TL75UtilCnt'].mean())
     print('TL50UtilCnt(mean): ',df['TL50UtilCnt'].mean())
     print('TLBalHCPct(mean): ',df['TLBalHCPct'].mean())
     print('TLSatPct(mean): ',df['TLSatPct'].mean())
     print('TLOpenPct(mean): ',df['TLOpenPct'].mean())
     print('TLOpen24Pct(mean): ',df['TLOpen24Pct'].mean())
```

```
InqTimeLast(mean):  3.108108108108108
TLSum(mean):  20151.09560810811
TLMaxSum(mean):  31205.900675675675
TLCnt(mean):  7.879546212879546
TLSatCnt(mean):  13.511682242990654
TL75UtilCnt(mean):  3.1216821785591176
TL50UtilCnt(mean):  4.0779041709755255
TLBalHCPct(mean):  0.6481782696857046
TLSatPct(mean):  0.5183314419225633
TLOpenPct(mean):  0.4961683016349683
TLOpen24Pct(mean):  0.5642190523857191
```



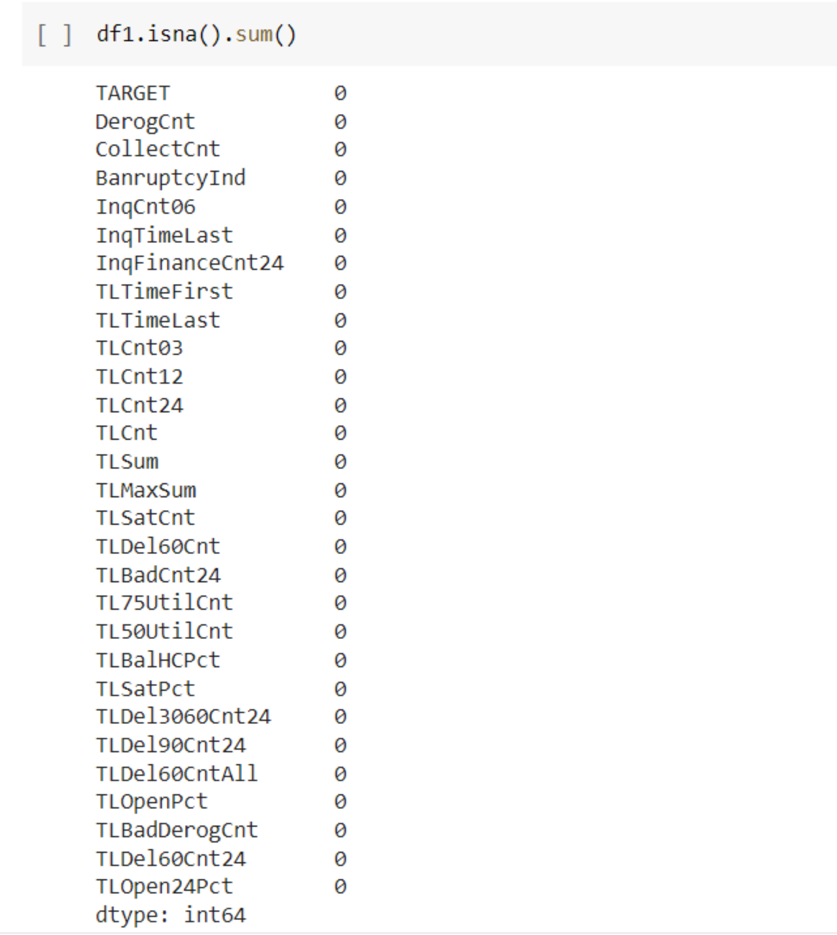
```
[ ]  # MODE Values:
     print('InqTimeLast(mode): ',df['InqTimeLast'].mode())
     print('TLSum(mode): ',df['TLSum'].mode())
     print('TLMaxSum(mode): ',df['TLMaxSum'].mode())
     print('TLCnt(mode): ',df['TLCnt'].mode())
     print('TLSatCnt(mode): ',df['TLSatCnt'].mode())
     print('TL75UtilCnt(mode): ',df['TL75UtilCnt'].mode())
     print('TL50UtilCnt(mode): ',df['TL50UtilCnt'].mode())
     print('TLBalHCPct(mode): ',df['TLBalHCPct'].mode())
     print('TLSatPct(mode): ',df['TLSatPct'].mode())
     print('TLOpenPct(mode): ',df['TLOpenPct'].mode())
     print('TLOpen24Pct(mode): ',df['TLOpen24Pct'].mode())
```

```
InqTimeLast(mode):  0    1.0
Name: InqTimeLast, dtype: float64
TLSum(mode):  0    0.0
Name: TLSum, dtype: float64
TLMaxSum(mode):  0    0.0
Name: TLMaxSum, dtype: float64
TLCnt(mode):  0    3.0
Name: TLCnt, dtype: float64
TLSatCnt(mode):  0    10.0
Name: TLSatCnt, dtype: float64
TL75UtilCnt(mode):  0    1.0
Name: TL75UtilCnt, dtype: float64
TL50UtilCnt(mode):  0    3.0
Name: TL50UtilCnt, dtype: float64
TLBalHCPct(mode):  0    0.0
Name: TLBalHCPct, dtype: float64
TLSatPct(mode):  0    0.5
Name: TLSatPct, dtype: float64
TLOpenPct(mode):  0    0.5
Name: TLOpenPct, dtype: float64
```



```
[ ]  df1 = df
```

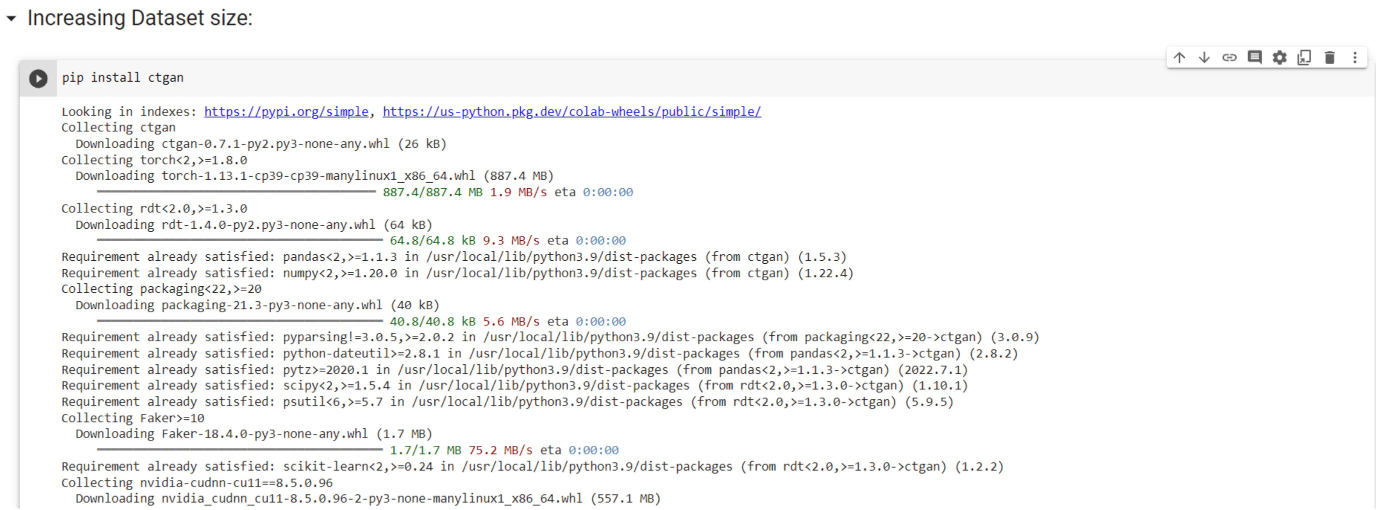
```
[ ]  # Replacing missing values with median
     df1['InqTimeLast'] = df1['InqTimeLast'].fillna(df1['InqTimeLast'].median())
     df1['TLSum'] = df1['TLSum'].fillna(df1['TLSum'].median())
     df1['TLMaxSum'] = df1['TLMaxSum'].fillna(df1['TLMaxSum'].median())
     df1['TLCnt'] = df1['TLCnt'].fillna(df1['TLCnt'].median())
     df1['TLSatCnt'] = df1['TLSatCnt'].fillna(df1['TLSatCnt'].median())
     df1['TL75UtilCnt'] = df1['TL75UtilCnt'].fillna(df1['TL75UtilCnt'].median())
     df1['TL50UtilCnt'] = df1['TL50UtilCnt'].fillna(df1['TL50UtilCnt'].median())
     df1['TLBalHCPct'] = df1['TLBalHCPct'].fillna(df1['TLBalHCPct'].median())
     df1['TLSatPct'] = df1['TLSatPct'].fillna(df1['TLSatPct'].median())
     df1['TLOpenPct'] = df1['TLOpenPct'].fillna(df1['TLOpenPct'].median())
     df1['TLOpen24Pct'] = df1['TLOpen24Pct'].fillna(df1['TLOpen24Pct'].median())
```



```
[ ]  df1.isna().sum()
```

```
TARGET             0
DerogCnt           0
CollectCnt         0
BanruptcyInd       0
InqCnt06           0
InqTimeLast        0
InqFinanceCnt24    0
TLTimeFirst        0
TLTimeLast         0
TLCnt03            0
TLCnt12            0
TLCnt24            0
TLCnt              0
TLSum              0
TLMaxSum           0
TLSatCnt           0
TLDel60Cnt         0
TLBadCnt24         0
TL75UtilCnt        0
TL50UtilCnt        0
TLBalHCPct         0
TLSatPct           0
TLDel3060Cnt24     0
TLDel90Cnt24       0
TLDel60CntAll      0
TLOpenPct          0
TLBadDerogCnt      0
TLDel60Cnt24       0
TLOpen24Pct        0
dtype: int64
```

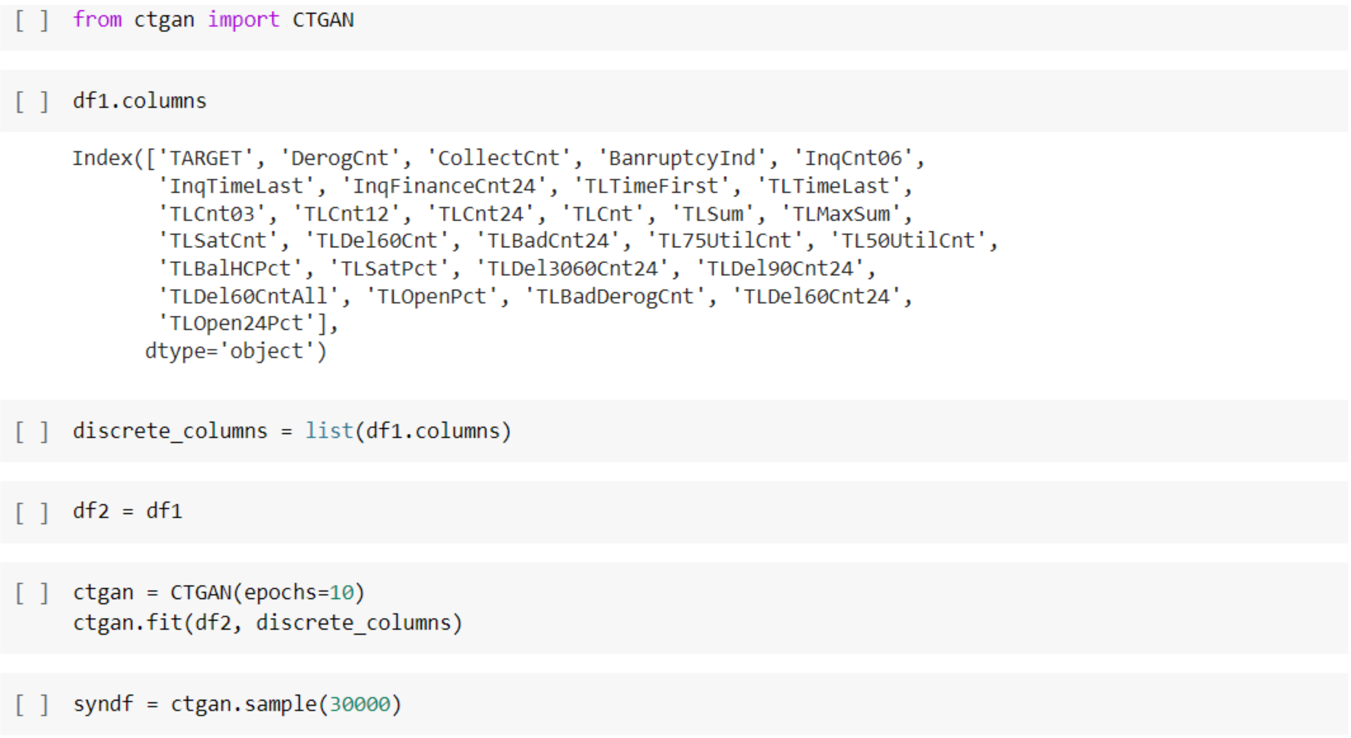
4.1.2. INCREASING DATASET:



Increasing Dataset size:

```
pip install ctgan
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting ctgan
  Downloading ctgan-0.7.1-py2.py3-none-any.whl (26 kB)
Collecting torch<2,>=1.8.0
  Downloading torch-1.13.1-cp39-cp39-manylinux1_x86_64.whl (887.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 887.4/887.4 MB 1.9 MB/s eta 0:00:00
Collecting rdt<2.0,>=1.3.0
  Downloading rdt-1.4.0-py2.py3-none-any.whl (64 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 64.8/64.8 kB 9.3 MB/s eta 0:00:00
Requirement already satisfied: pandas<2,>=1.1.3 in /usr/local/lib/python3.9/dist-packages (from ctgan) (1.5.3)
Requirement already satisfied: numpy<2,>=1.20.0 in /usr/local/lib/python3.9/dist-packages (from ctgan) (1.22.4)
Collecting packaging<22,>=20
  Downloading packaging-21.3-py3-none-any.whl (40 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 40.8/40.8 kB 5.6 MB/s eta 0:00:00
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.9/dist-packages (from packaging<22,>=20->ctgan) (3.0.9)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.9/dist-packages (from pandas<2,>=1.1.3->ctgan) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-packages (from pandas<2,>=1.1.3->ctgan) (2022.7.1)
Requirement already satisfied: scipy<2,>=1.5.4 in /usr/local/lib/python3.9/dist-packages (from rdt<2.0,>=1.3.0->ctgan) (1.10.1)
Requirement already satisfied: psutil<6,>=5.7 in /usr/local/lib/python3.9/dist-packages (from rdt<2.0,>=1.3.0->ctgan) (5.9.5)
Collecting Faker>=10
  Downloading Faker-18.4.0-py3-none-any.whl (1.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.7/1.7 MB 75.2 MB/s eta 0:00:00
Requirement already satisfied: scikit-learn<2,>=0.24 in /usr/local/lib/python3.9/dist-packages (from rdt<2.0,>=1.3.0->ctgan) (1.2.2)
Collecting nvidia-cudnn-cu11==8.5.0.96
  Downloading nvidia_cudnn_cu11-8.5.0.96-2-py3-none-manylinux1_x86_64.whl (557.1 MB)
```



```
[ ]  from ctgan import CTGAN
```

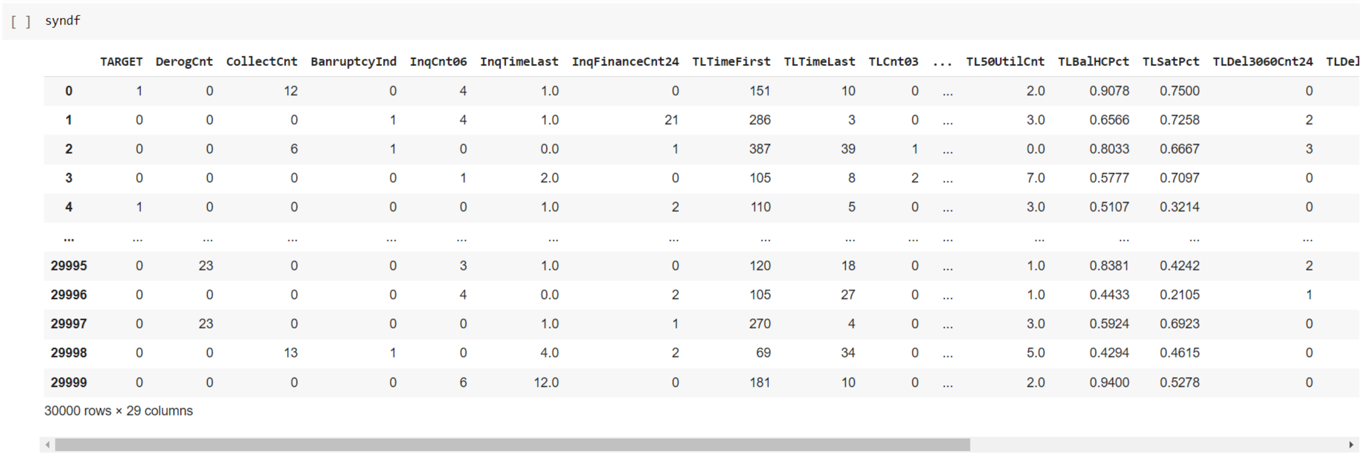
```
[ ]  df1.columns

     Index(['TARGET', 'DerogCnt', 'CollectCnt', 'BanruptcyInd', 'InqCnt06',
            'InqTimeLast', 'InqFinanceCnt24', 'TLTimeFirst', 'TLTimeLast',
            'TLCnt03', 'TLCnt12', 'TLCnt24', 'TLCnt', 'TLSum', 'TLMaxSum',
            'TLSatCnt', 'TLDel60Cnt', 'TLBadCnt24', 'TL75UtilCnt', 'TL50UtilCnt',
            'TLBalHCPct', 'TLSatPct', 'TLDel3060Cnt24', 'TLDel90Cnt24',
            'TLDel60CntAll', 'TLOpenPct', 'TLBadDerogCnt', 'TLDel60Cnt24',
            'TLOpen24Pct'],
           dtype='object')
```

```
[ ]  discrete_columns = list(df1.columns)
```

```
[ ]  df2 = df1
```

```
[ ]  ctgan = CTGAN(epochs=10)
     ctgan.fit(df2, discrete_columns)
```

```
[ ]  syndf = ctgan.sample(30000)
```

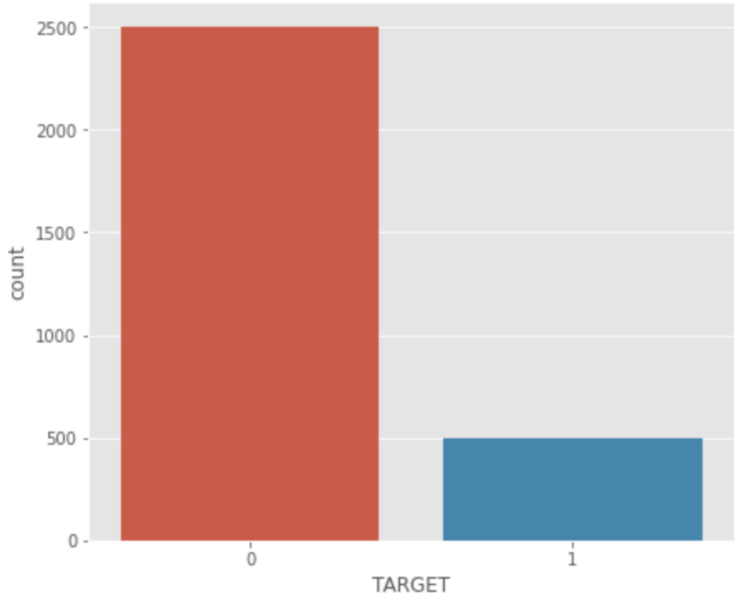


```
[ ]  syndf
```

	TARGET	DerogCnt	CollectCnt	BanruptcyInd	InqCnt06	InqTimeLast	InqFinanceCnt24	TLTimeFirst	TLTimeLast	TLCnt03	...	TL50UtilCnt	TLBalHCPct	TLSatPct	TLDel3060Cnt24	TLDel
0	1	0	12	0	4	1.0	0	151	10	0	...	2.0	0.9078	0.7500	0	
1	0	0	0	1	4	1.0	21	286	3	0	...	3.0	0.6566	0.7258	2	
2	0	0	6	1	0	0.0	1	387	39	1	...	0.0	0.8033	0.6667	3	
3	0	0	0	0	1	2.0	0	105	8	2	...	7.0	0.5777	0.7097	0	
4	1	0	0	0	0	1.0	2	110	5	0	...	3.0	0.5107	0.3214	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
29995	0	23	0	0	3	1.0	0	120	18	0	...	1.0	0.8381	0.4242	2	
29996	0	0	0	0	4	0.0	2	105	27	0	...	1.0	0.4433	0.2105	1	
29997	0	23	0	0	0	1.0	1	270	4	0	...	3.0	0.5924	0.6923	0	
29998	0	0	13	1	0	4.0	2	69	34	0	...	5.0	0.4294	0.4615	0	
29999	0	0	0	0	6	12.0	0	181	10	0	...	2.0	0.9400	0.5278	0	

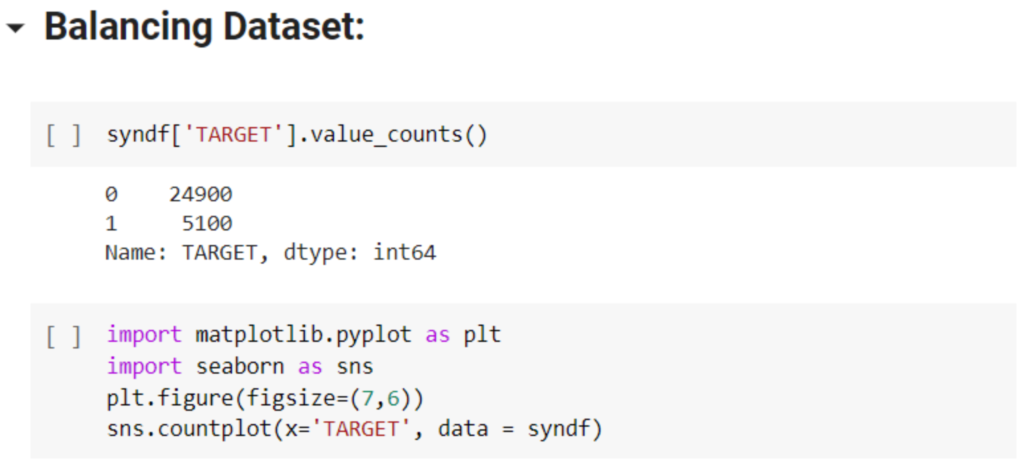
30000 rows × 29 columns

4.1.3. BALANCING DATASET:



CLASS DISTRIBUTION WITHOUT ANY AUGMENTATION

It can be inferred from above figure that, this dataset is imbalanced and is suffering from the issue of low-portfolio default. It needs to be balanced so that, the number of bad loans (1) should almost be equal to good loans (0).

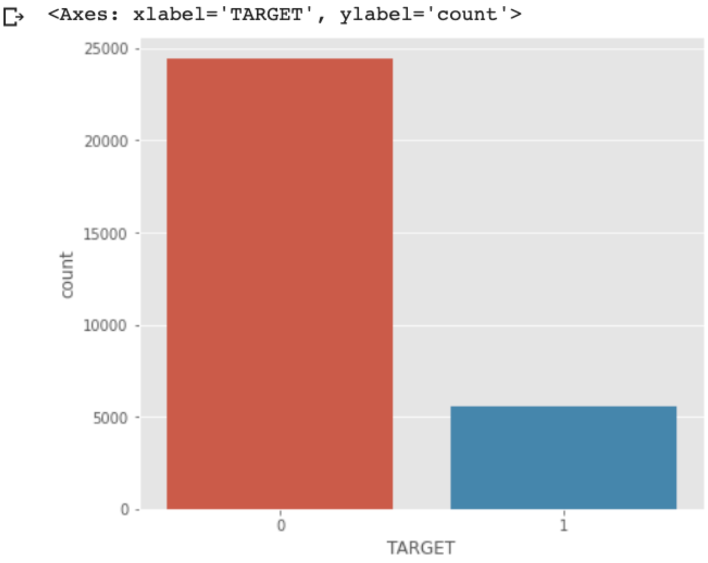


▾ Balancing Dataset:

```
[ ]  syndf['TARGET'].value_counts()
```

```
0    24900
1     5100
Name: TARGET, dtype: int64
```

```
[ ]  import matplotlib.pyplot as plt
     import seaborn as sns
     plt.figure(figsize=(7,6))
     sns.countplot(x='TARGET', data = syndf)
```

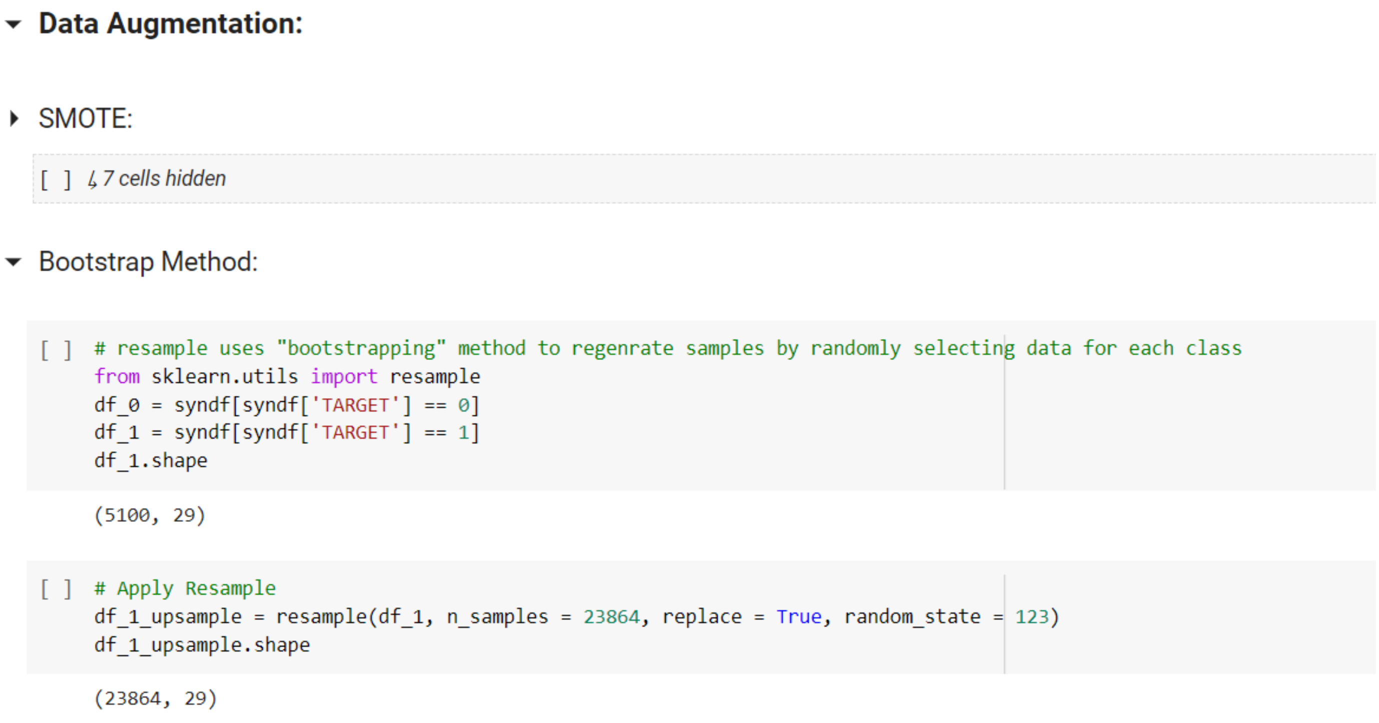


```
<Axes: xlabel='TARGET', ylabel='count'>
```

CLASS DISTRIBUTION AFTER SYNTHETICALLY INCREASING THE DATASET SIZE USING CTGAN

In the above figure, the number of good loans is reduced and the number of bad loans is increased by producing synthetic data and equalizing the number of good and bad loans.

4.1.4. DATA AUGMENTATION



▾ **Data Augmentation:**

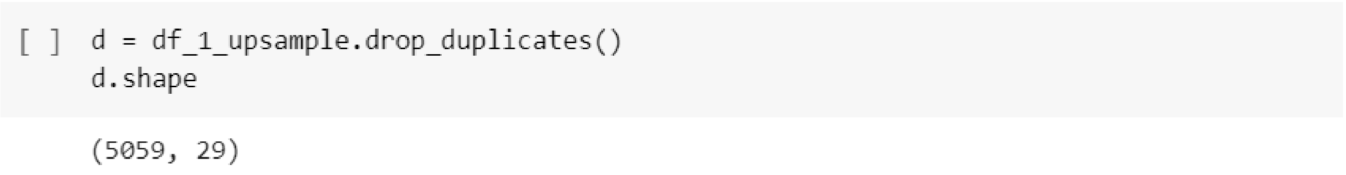
▸ SMOTE:

[ ] ↳ *7 cells hidden*

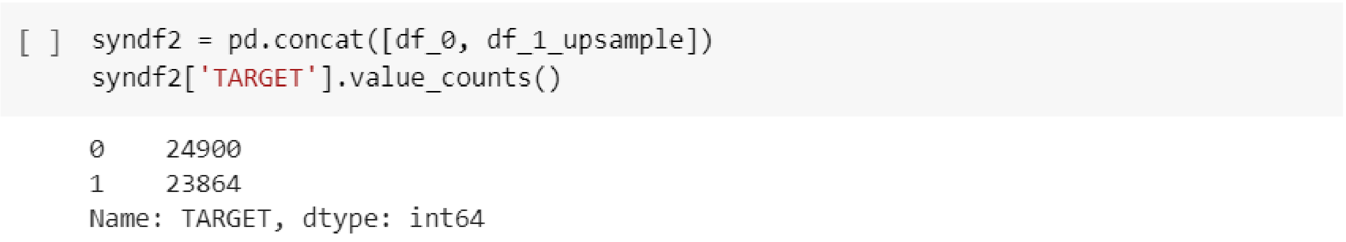
▾ Bootstrap Method:

```
[ ] # resample uses "bootstrapping" method to regenrate samples by randomly selecting data for each class
    from sklearn.utils import resample
    df_0 = syndf[syndf['TARGET'] == 0]
    df_1 = syndf[syndf['TARGET'] == 1]
    df_1.shape

    (5100, 29)
```

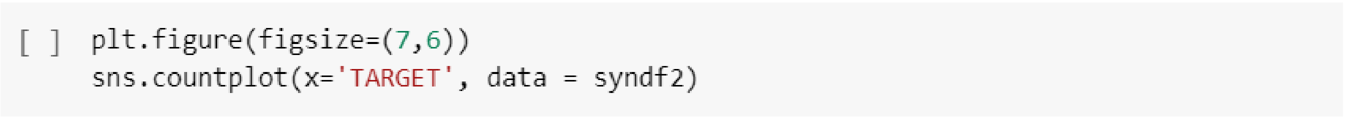
```
[ ] # Apply Resample
    df_1_upsample = resample(df_1, n_samples = 23864, replace = True, random_state = 123)
    df_1_upsample.shape

    (23864, 29)
```



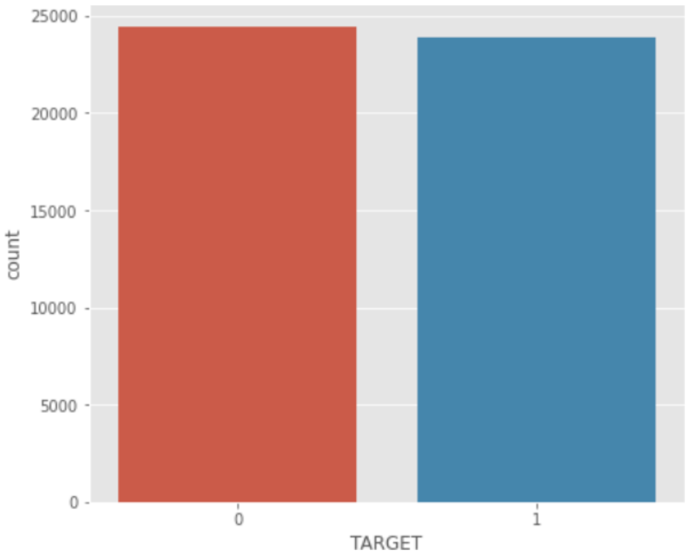
```
[ ] d = df_1_upsample.drop_duplicates()
    d.shape

    (5059, 29)
```



```
[ ] syndf2 = pd.concat([df_0, df_1_upsample])
    syndf2['TARGET'].value_counts()

    0    24900
    1    23864
    Name: TARGET, dtype: int64
```



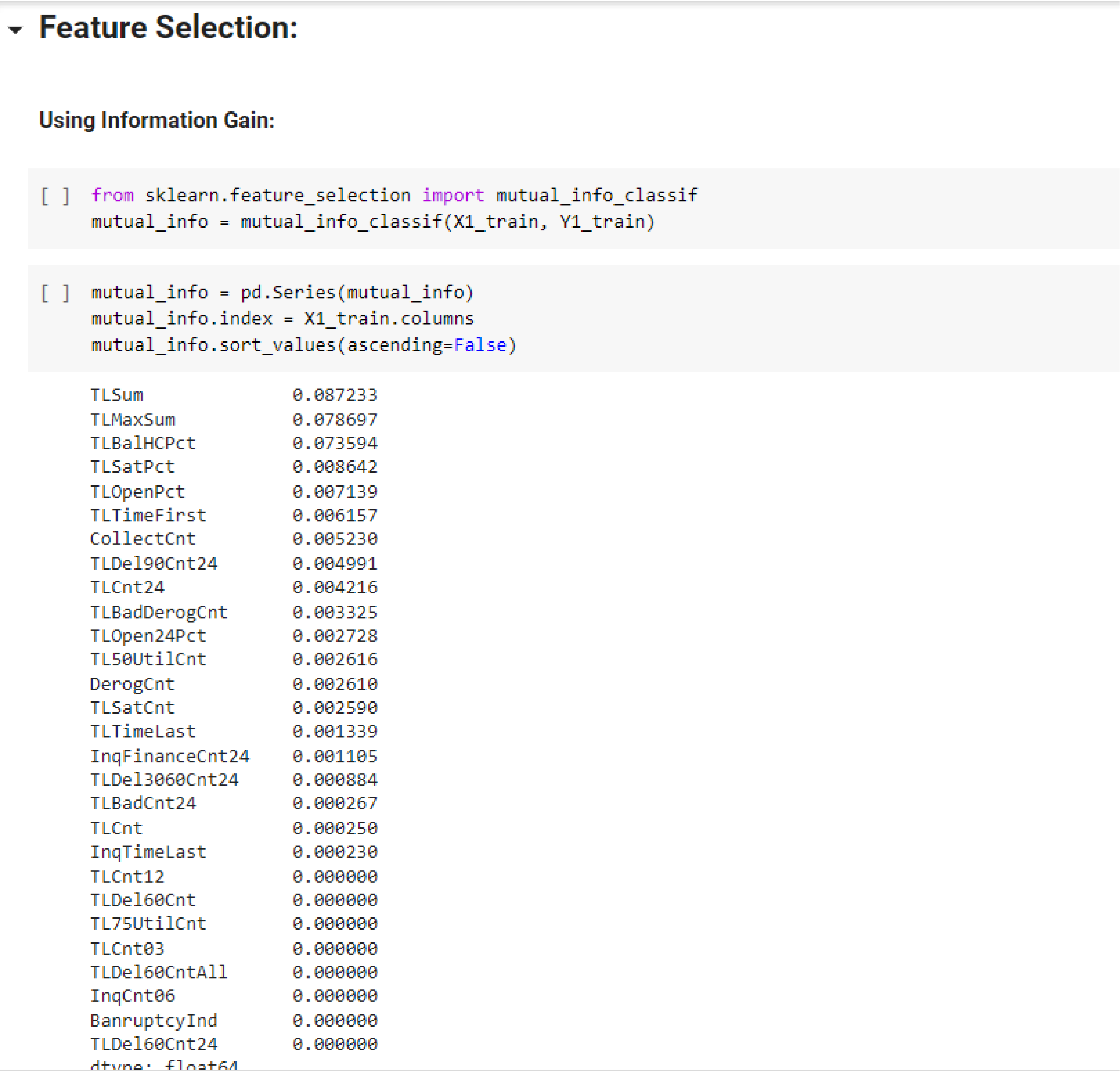
```
[ ] plt.figure(figsize=(7,6))
    sns.countplot(x='TARGET', data = syndf2)
```



CLASS DISTRIBUTION AFTER APPLYING BOOTSTRAP METHOD TO BALANCE THE DATASET USING BAGGING

In above figure, using the bootstrapping method, random values from already existing features are chosen and mix-matched together to form new instances, which are then added to emulate the bad loans output class. If any row gets exactly replicated, it is promptly removed to avoid overfitting.

4.2. FEATURE SELECTION



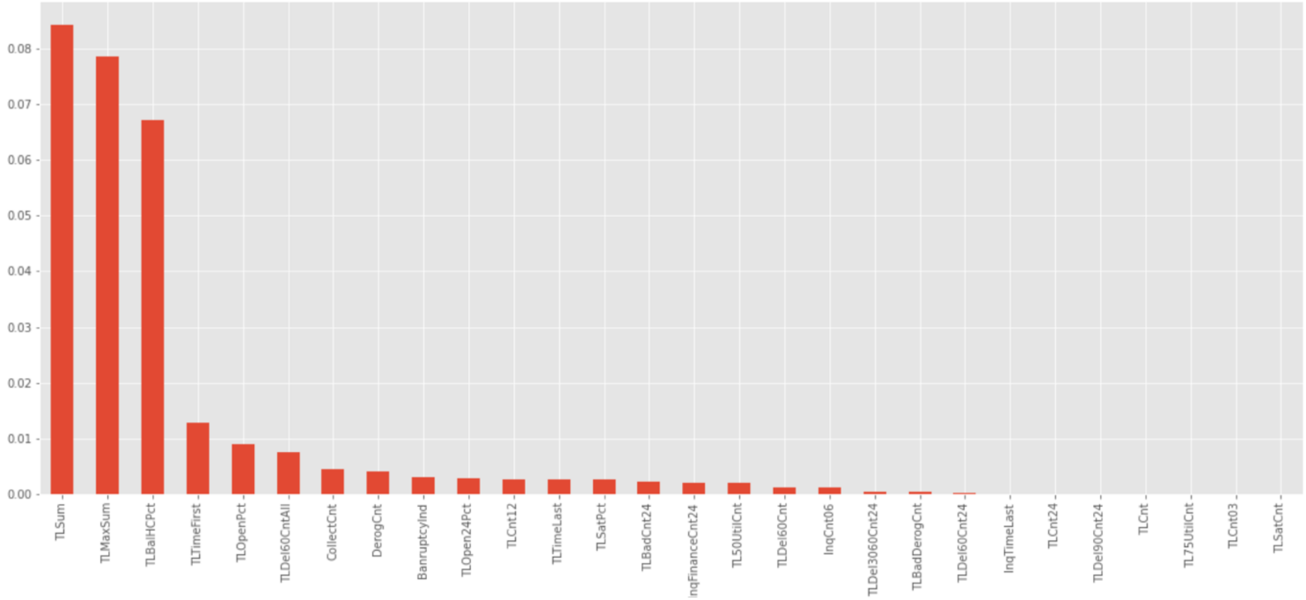
Feature Selection:

Using Information Gain:

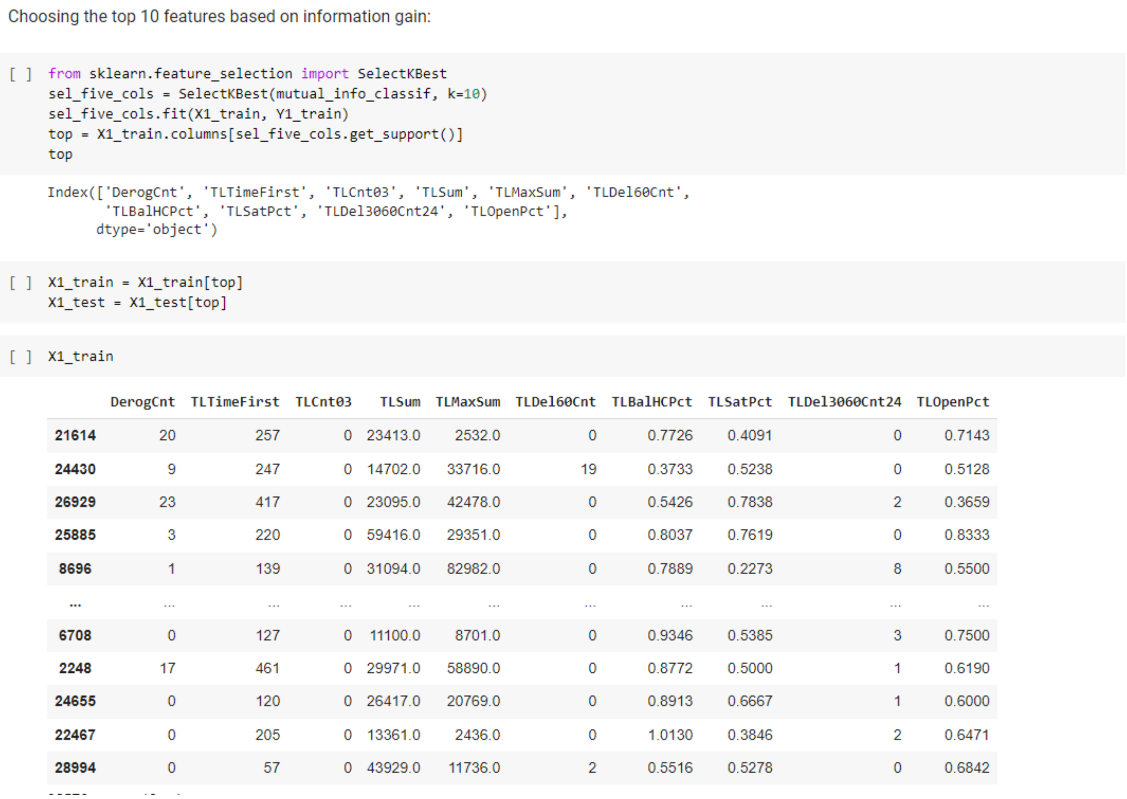
```
[ ]  from sklearn.feature_selection import mutual_info_classif
     mutual_info = mutual_info_classif(X1_train, Y1_train)
```

```
[ ]  mutual_info = pd.Series(mutual_info)
     mutual_info.index = X1_train.columns
     mutual_info.sort_values(ascending=False)
```

```
     TLSum                0.087233
     TLMaxSum             0.078697
     TLBalHCPct           0.073594
     TLSatPct             0.008642
     TLOpenPct            0.007139
     TLTimeFirst          0.006157
     CollectCnt           0.005230
     TLDel90Cnt24         0.004991
     TLCnt24              0.004216
     TLBadDerogCnt        0.003325
     TLOpen24Pct          0.002728
     TL50UtilCnt          0.002616
     DerogCnt             0.002610
     TLSatCnt             0.002590
     TLTimeLast           0.001339
     InqFinanceCnt24      0.001105
     TLDel3060Cnt24       0.000884
     TLBadCnt24           0.000267
     TLCnt                0.000250
     InqTimeLast          0.000230
     TLCnt12              0.000000
     TLDel60Cnt           0.000000
     TL75UtilCnt          0.000000
     TLCnt03              0.000000
     TLDel60CntAll        0.000000
     InqCnt06             0.000000
     BanruptcyInd         0.000000
     TLDel60Cnt24         0.000000
     dtype: float64
```



Here, we can see, the attributes on the left side of the chart having the higher mutual information value are more consequential in predicting the output value then the rest. The top ten attributes alone are chosen. This value is chosen based on the observation that many features end up contributing almost nothing to the outcome class.



Choosing the top 10 features based on information gain:

```
[ ]  from sklearn.feature_selection import SelectKBest
     sel_five_cols = SelectKBest(mutual_info_classif, k=10)
     sel_five_cols.fit(X1_train, Y1_train)
     top = X1_train.columns[sel_five_cols.get_support()]
     top
```

```
Index(['DerogCnt', 'TLTimeFirst', 'TLCnt03', 'TLSum', 'TLMaxSum', 'TLDel60Cnt',
       'TLBalHCPct', 'TLSatPct', 'TLDel3060Cnt24', 'TLOpenPct'],
      dtype='object')
```

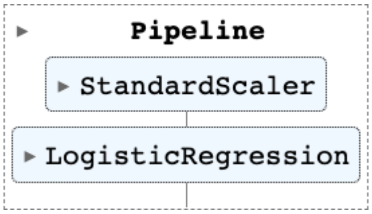
```
[ ]  X1_train = X1_train[top]
     X1_test = X1_test[top]
```

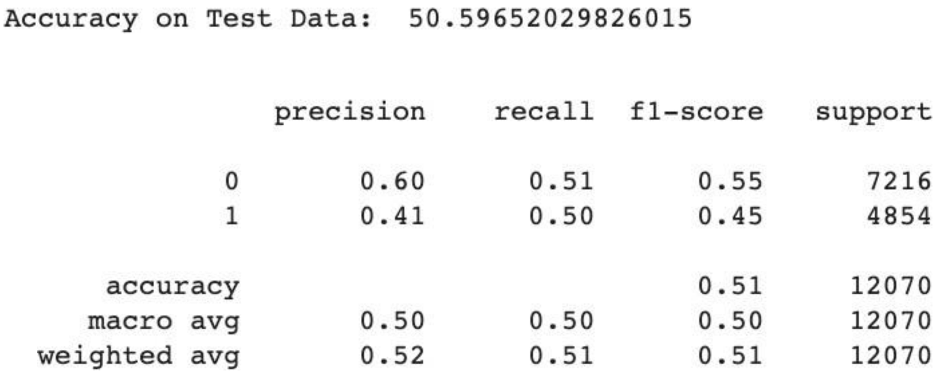
```
[ ]  X1_train
```

	DerogCnt	TLTimeFirst	TLCnt03	TLSum	TLMaxSum	TLDel60Cnt	TLBalHCPct	TLSatPct	TLDel3060Cnt24	TLOpenPct
21614	20	257	0	23413.0	2532.0	0	0.7726	0.4091	0	0.7143
24430	9	247	0	14702.0	33716.0	19	0.3733	0.5238	0	0.5128
26929	23	417	0	23095.0	42478.0	0	0.5426	0.7838	2	0.3659
25885	3	220	0	59416.0	29351.0	0	0.8037	0.7619	0	0.8333
8696	1	139	0	31094.0	82982.0	0	0.7889	0.2273	8	0.5500
...	...	...	...	...	...	...	...	...	...	...
6708	0	127	0	11100.0	8701.0	0	0.9346	0.5385	3	0.7500
2248	17	461	0	29971.0	58890.0	0	0.8772	0.5000	1	0.6190
24655	0	120	0	26417.0	20769.0	0	0.8913	0.6667	1	0.6000
22467	0	205	0	13361.0	2436.0	0	1.0130	0.3846	2	0.6471
28994	0	57	0	43929.0	11736.0	2	0.5516	0.5278	0	0.6842

4.3. MODEL CLASSIFICATION REPORTS:

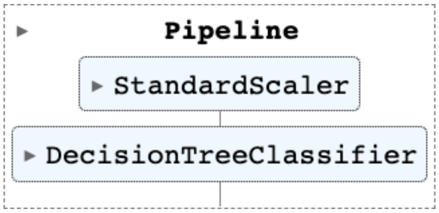
4.3.1. LOGISTIC REGRESSION:

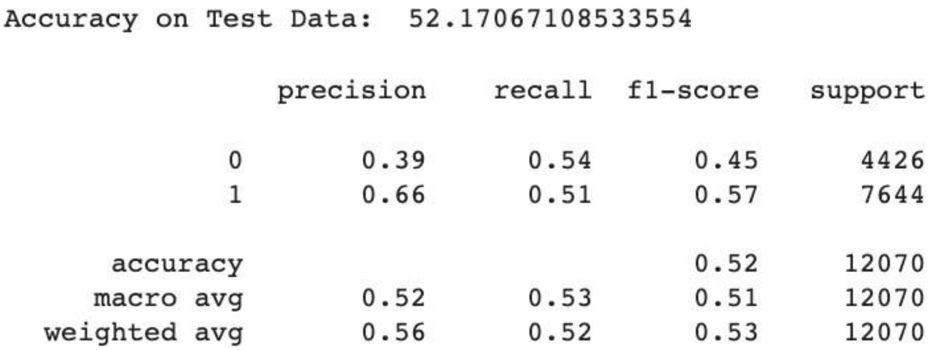




```
Accuracy on Test Data:  50.59652029826015

              precision    recall  f1-score   support

           0       0.60      0.51      0.55      7216
           1       0.41      0.50      0.45      4854

    accuracy                           0.51     12070
   macro avg       0.50      0.50      0.50     12070
weighted avg       0.52      0.51      0.51     12070
```

4.3.2. DECISION TREE

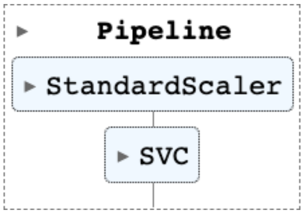


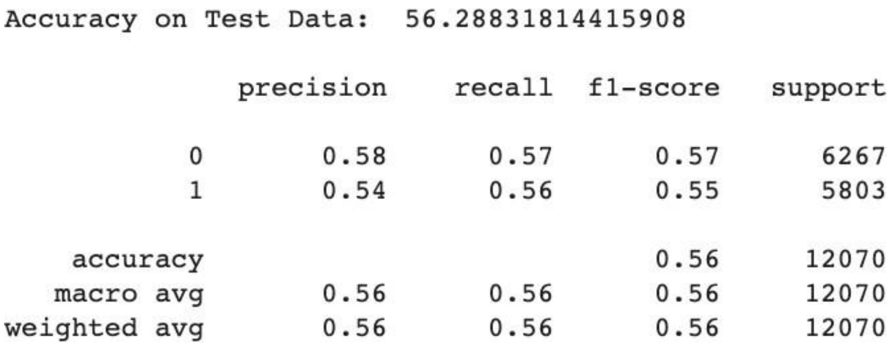


```
Accuracy on Test Data:  52.17067108533554
              precision    recall  f1-score   support

           0       0.39      0.54      0.45      4426
           1       0.66      0.51      0.57      7644

    accuracy                           0.52     12070
   macro avg       0.52      0.53      0.51     12070
weighted avg       0.56      0.52      0.53     12070
```

4.3.3. SUPPORT VECTOR MACHINE

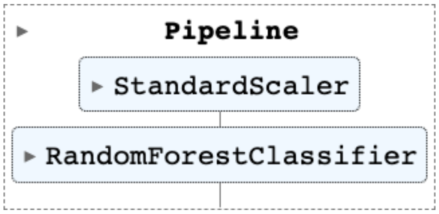
Radial Basis Function Kernel (Best Accuracy):

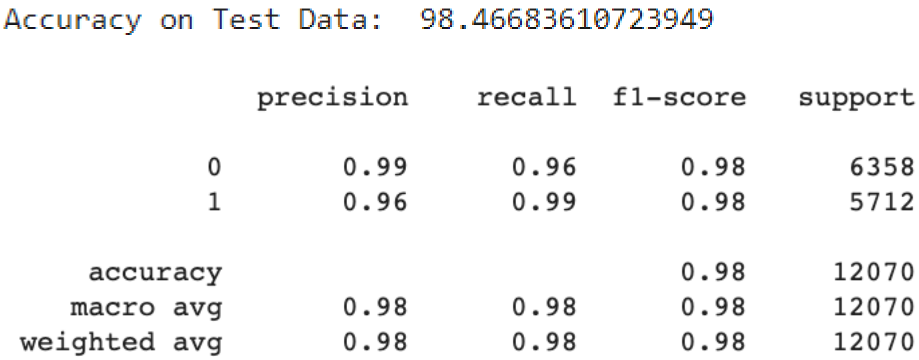




```
Accuracy on Test Data:  56.28831814415908

              precision    recall  f1-score   support

           0       0.58      0.57      0.57      6267
           1       0.54      0.56      0.55      5803

    accuracy                           0.56     12070
   macro avg       0.56      0.56      0.56     12070
weighted avg       0.56      0.56      0.56     12070
```

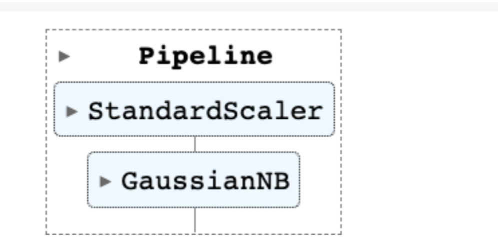
4.3.4. RANDOM FOREST

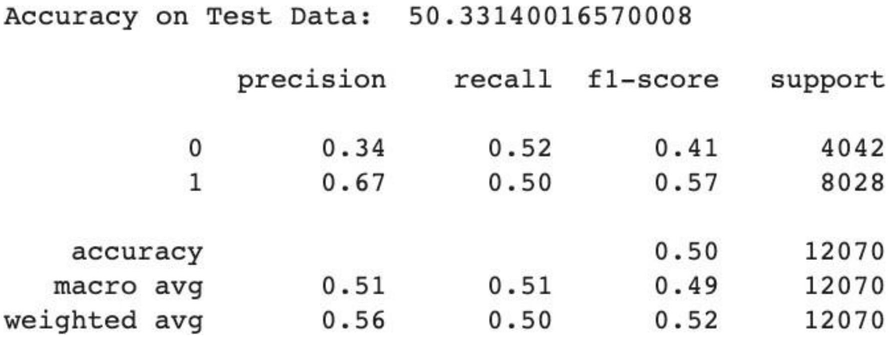




```
Accuracy on Test Data:  98.46683610723949

              precision    recall  f1-score   support

           0       0.99      0.96      0.98      6358
           1       0.96      0.99      0.98      5712

    accuracy                           0.98     12070
   macro avg       0.98      0.98      0.98     12070
weighted avg       0.98      0.98      0.98     12070
```

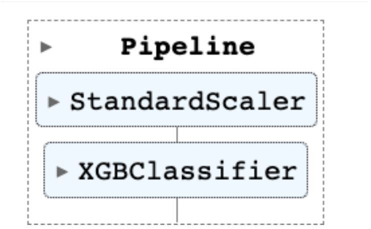
4.3.5. NAÏVE-BAYES

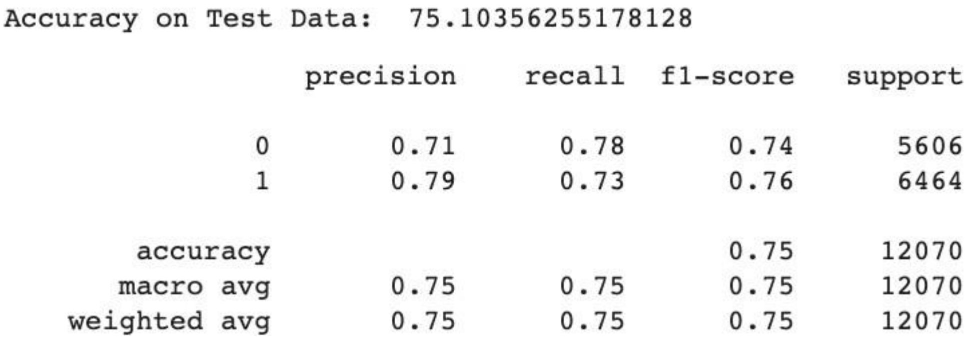




```
Accuracy on Test Data:  50.33140016570008
              precision    recall  f1-score   support

           0       0.34      0.52      0.41      4042
           1       0.67      0.50      0.57      8028

    accuracy                           0.50     12070
   macro avg       0.51      0.51      0.49     12070
weighted avg       0.56      0.50      0.52     12070
```

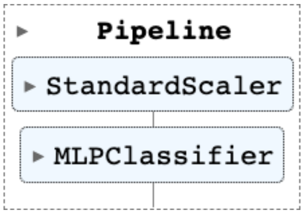
4.3.6. XGBOOST:

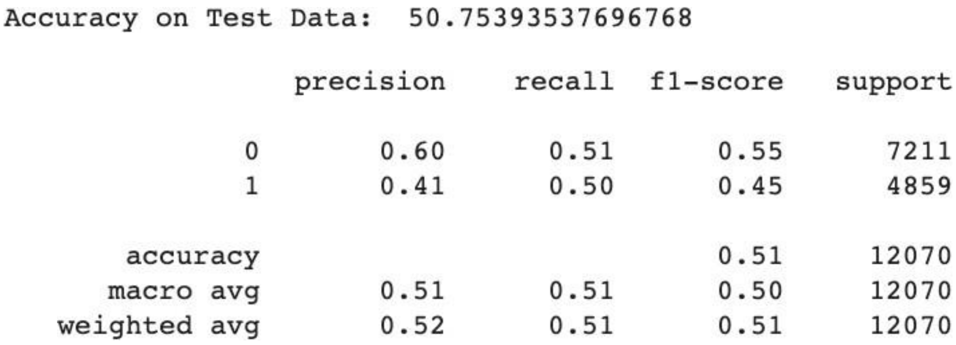




```
Accuracy on Test Data:  75.10356255178128
              precision    recall  f1-score   support

           0       0.71      0.78      0.74      5606
           1       0.79      0.73      0.76      6464

    accuracy                           0.75     12070
   macro avg       0.75      0.75      0.75     12070
weighted avg       0.75      0.75      0.75     12070
```

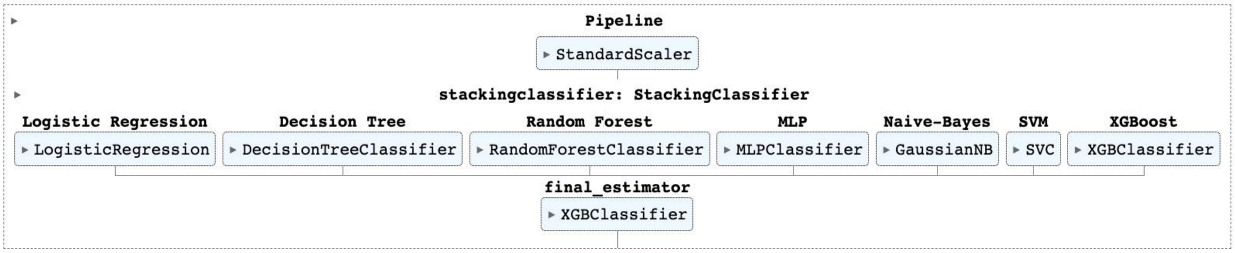
4.3.7. MULTI-LAYER PERCEPTRON

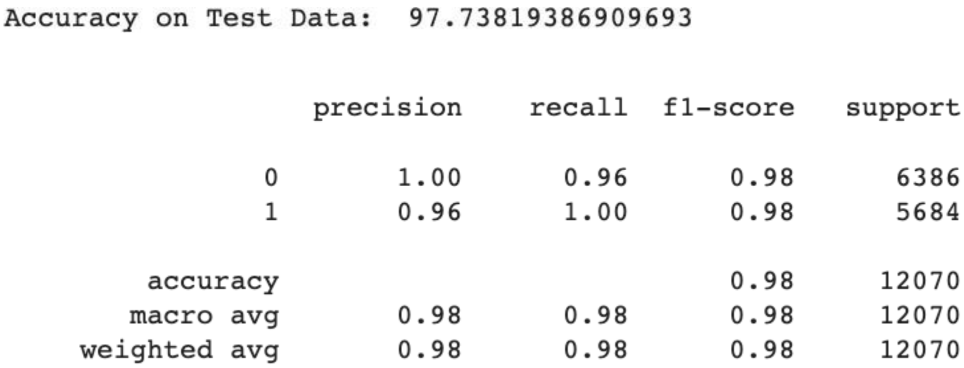




```
Accuracy on Test Data:  50.75393537696768

              precision    recall  f1-score   support

           0       0.60      0.51      0.55      7211
           1       0.41      0.50      0.45      4859

    accuracy                           0.51     12070
   macro avg       0.51      0.51      0.50     12070
weighted avg       0.52      0.51      0.51     12070
```

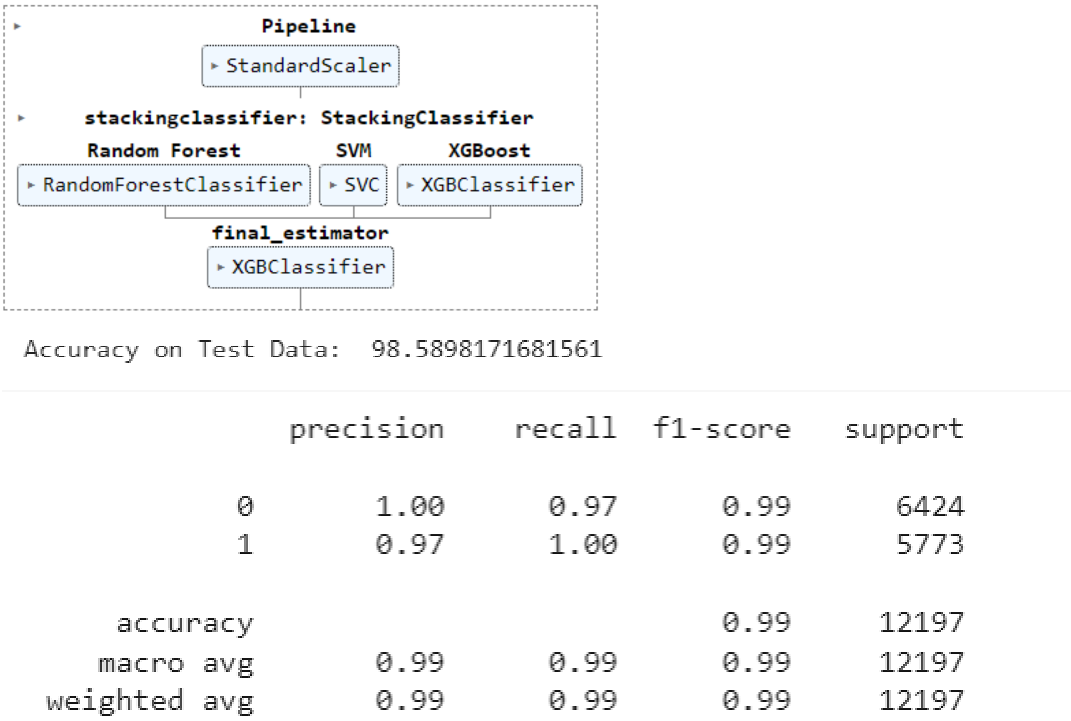
4.3.8. ENSEMBLE - STACKING CLASSIFIER:





```
Accuracy on Test Data:  97.73819386909693

              precision    recall  f1-score   support

           0       1.00      0.96      0.98      6386
           1       0.96      1.00      0.98      5684

    accuracy                           0.98     12070
   macro avg       0.98      0.98      0.98     12070
weighted avg       0.98      0.98      0.98     12070
```

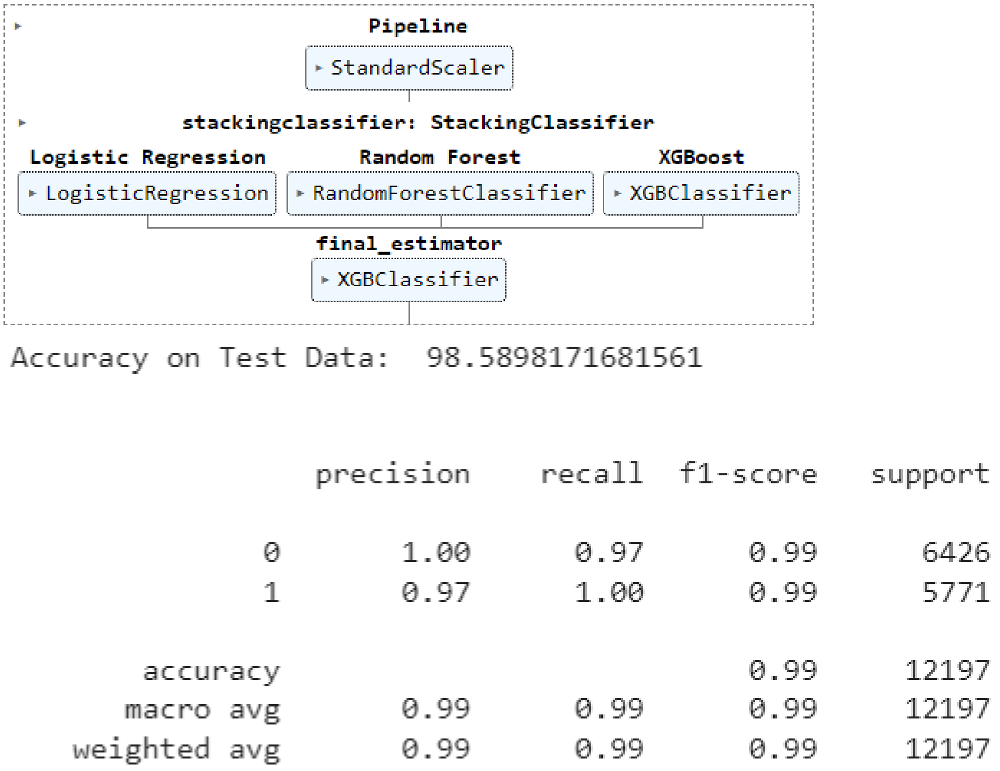
STACKING COMBINATION – 1



```
Accuracy on Test Data:  98.5898171681561
```

```
              precision    recall  f1-score   support

           0       1.00      0.97      0.99      6424
           1       0.97      1.00      0.99      5773

    accuracy                           0.99     12197
   macro avg       0.99      0.99      0.99     12197
weighted avg       0.99      0.99      0.99     12197
```

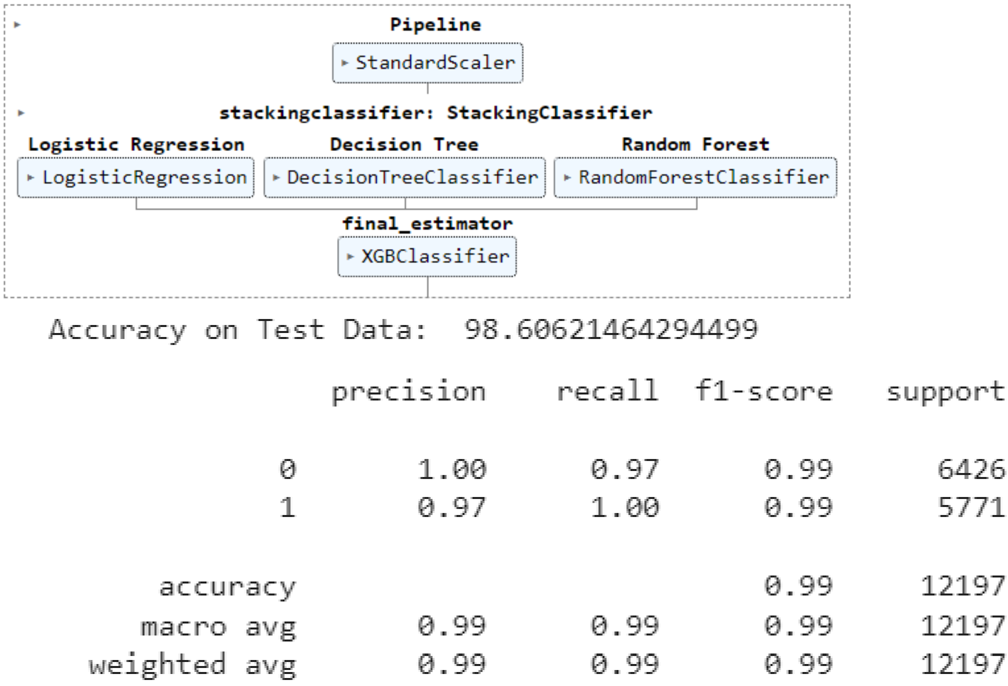
STACKING COMBINATION – 2



```
Accuracy on Test Data:  98.5898171681561
```

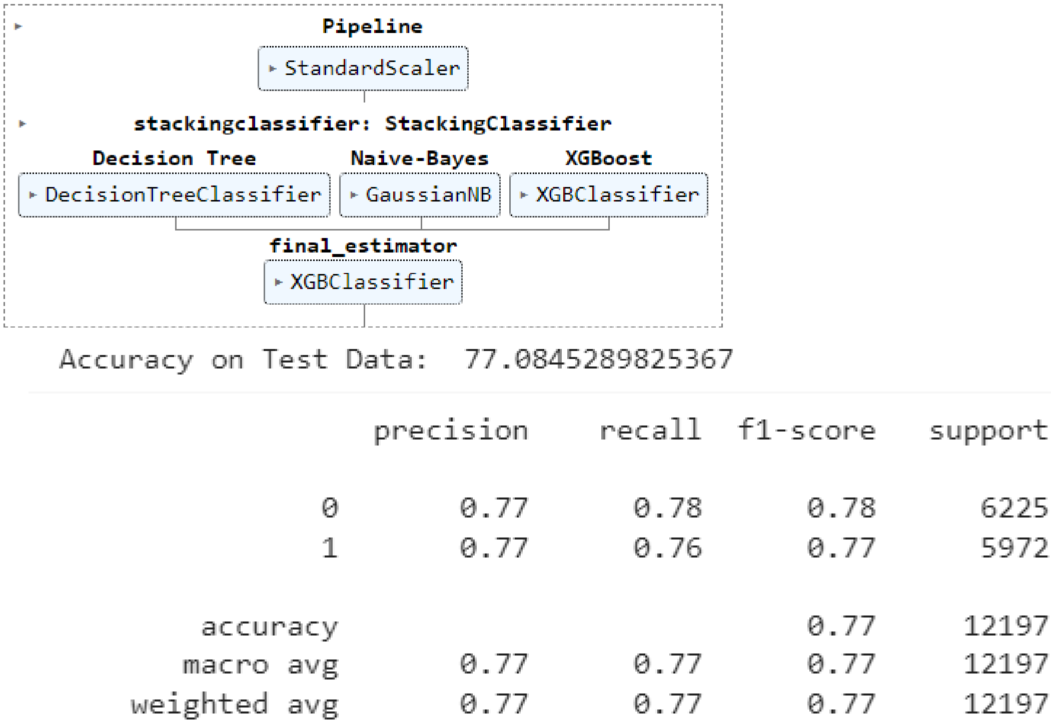
```
              precision    recall  f1-score   support

           0       1.00      0.97      0.99      6426
           1       0.97      1.00      0.99      5771

    accuracy                           0.99     12197
   macro avg       0.99      0.99      0.99     12197
weighted avg       0.99      0.99      0.99     12197
```

STACKING COMBINATION – 3



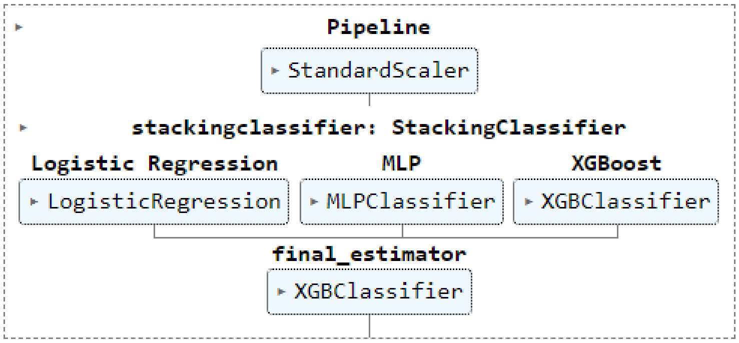
```
Accuracy on Test Data:  98.60621464294499

              precision    recall  f1-score   support

           0       1.00      0.97      0.99      6426
           1       0.97      1.00      0.99      5771

    accuracy                           0.99     12197
   macro avg       0.99      0.99      0.99     12197
weighted avg       0.99      0.99      0.99     12197
```

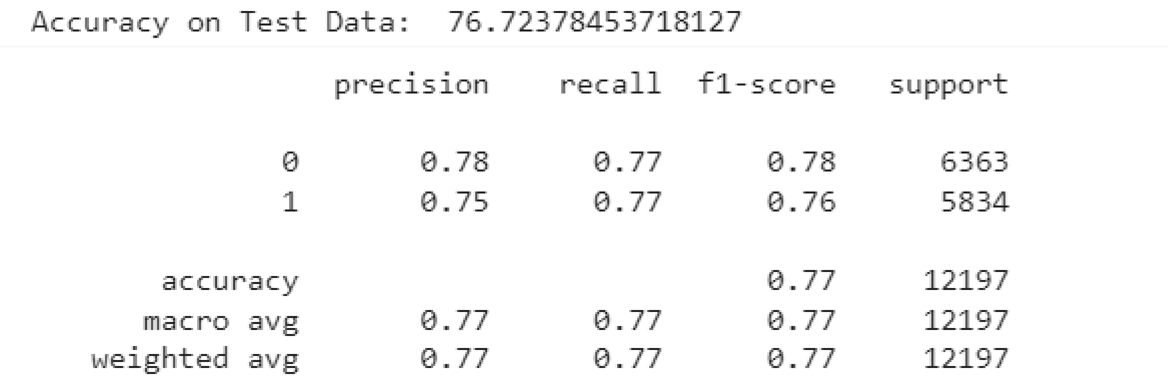
STACKING COMBINATION – 4



```
Accuracy on Test Data:  77.0845289825367

              precision    recall  f1-score   support

           0       0.77      0.78      0.78      6225
           1       0.77      0.76      0.77      5972

    accuracy                           0.77     12197
   macro avg       0.77      0.77      0.77     12197
weighted avg       0.77      0.77      0.77     12197
```

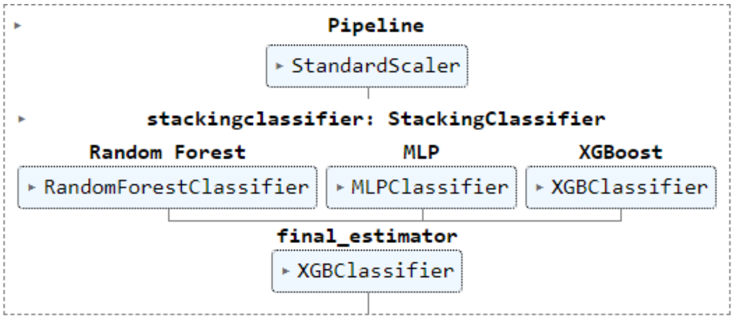
STACKING COMBINATION – 5

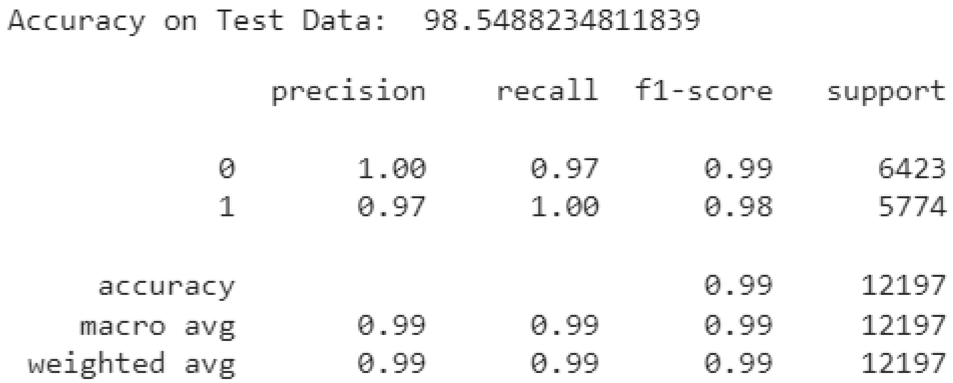




```
 Accuracy on Test Data:  76.72378453718127

              precision    recall  f1-score   support

           0       0.78      0.77      0.78      6363
           1       0.75      0.77      0.76      5834

    accuracy                           0.77     12197
   macro avg       0.77      0.77      0.77     12197
weighted avg       0.77      0.77      0.77     12197
```

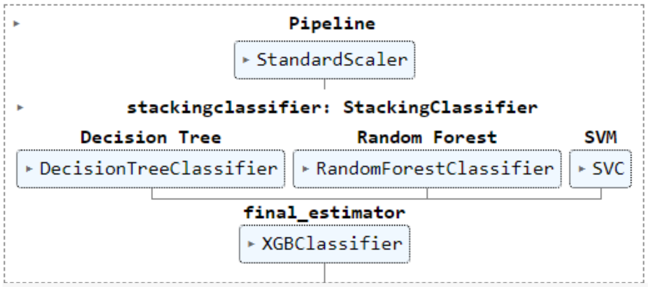
STACKING COMBINATION – 6





```
 Accuracy on Test Data:  98.5488234811839

              precision    recall  f1-score   support

           0       1.00      0.97      0.99      6423
           1       0.97      1.00      0.98      5774

    accuracy                           0.99     12197
   macro avg       0.99      0.99      0.99     12197
weighted avg       0.99      0.99      0.99     12197
```

STACKING COMBINATION – 7



Pipeline

StandardScaler

stackingclassifier: StackingClassifier

Decision Tree

DecisionTreeClassifier

Random Forest

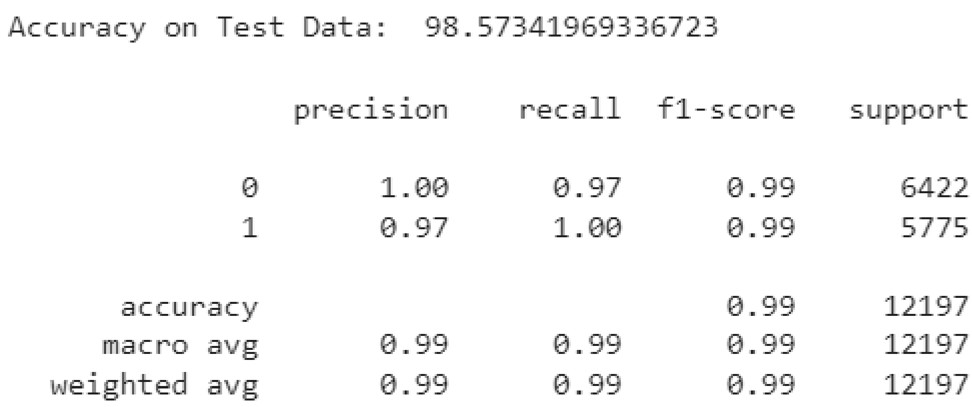
RandomForestClassifier

SVM

SVC

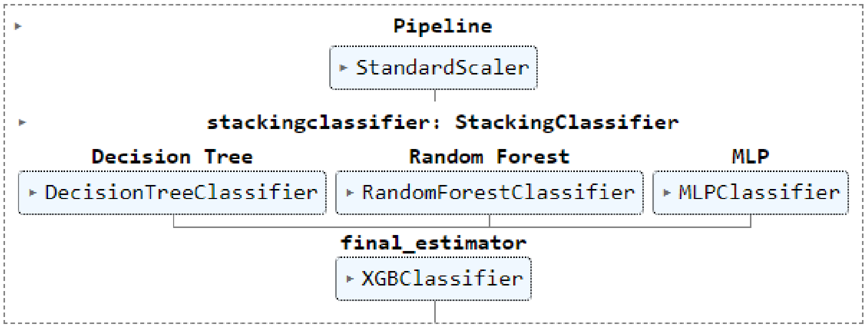
final_estimator

XGBClassifier



```
Accuracy on Test Data:  98.57341969336723

              precision    recall  f1-score   support

           0       1.00      0.97      0.99      6422
           1       0.97      1.00      0.99      5775

    accuracy                           0.99     12197
   macro avg       0.99      0.99      0.99     12197
weighted avg       0.99      0.99      0.99     12197
```

STACKING COMBINATION – 8



Pipeline

StandardScaler

stackingclassifier: StackingClassifier

Decision Tree

DecisionTreeClassifier

Random Forest

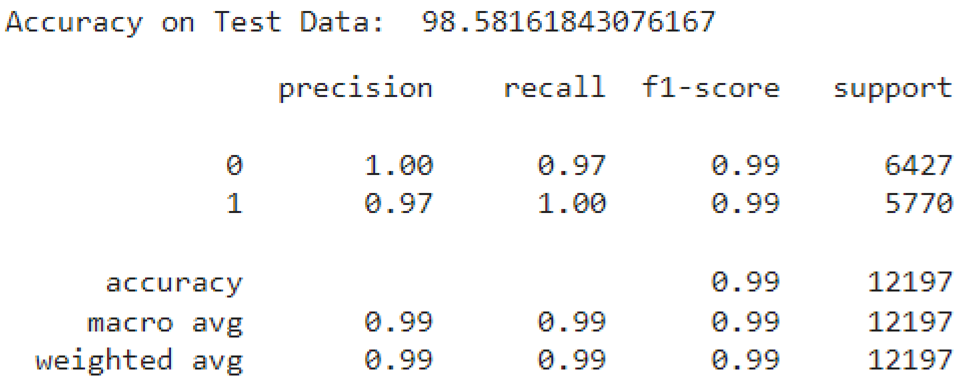
RandomForestClassifier

MLP

MLPClassifier

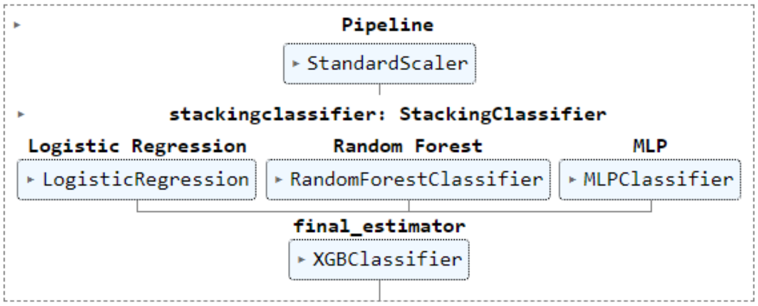
final_estimator

XGBClassifier



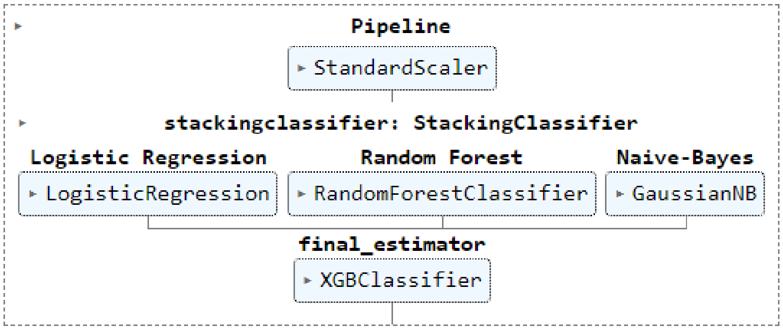
```
Accuracy on Test Data:  98.58161843076167

              precision    recall  f1-score   support

           0       1.00      0.97      0.99      6427
           1       0.97      1.00      0.99      5770

    accuracy                           0.99     12197
   macro avg       0.99      0.99      0.99     12197
weighted avg       0.99      0.99      0.99     12197
```

STACKING COMBINATION – 9



```
  Accuracy on Test Data:  98.56522095597278
              precision    recall  f1-score   support

           0       1.00      0.97      0.99      6423
           1       0.97      1.00      0.99      5774

    accuracy                           0.99     12197
   macro avg       0.99      0.99      0.99     12197
weighted avg       0.99      0.99      0.99     12197
```

STACKING COMBINATION – 10



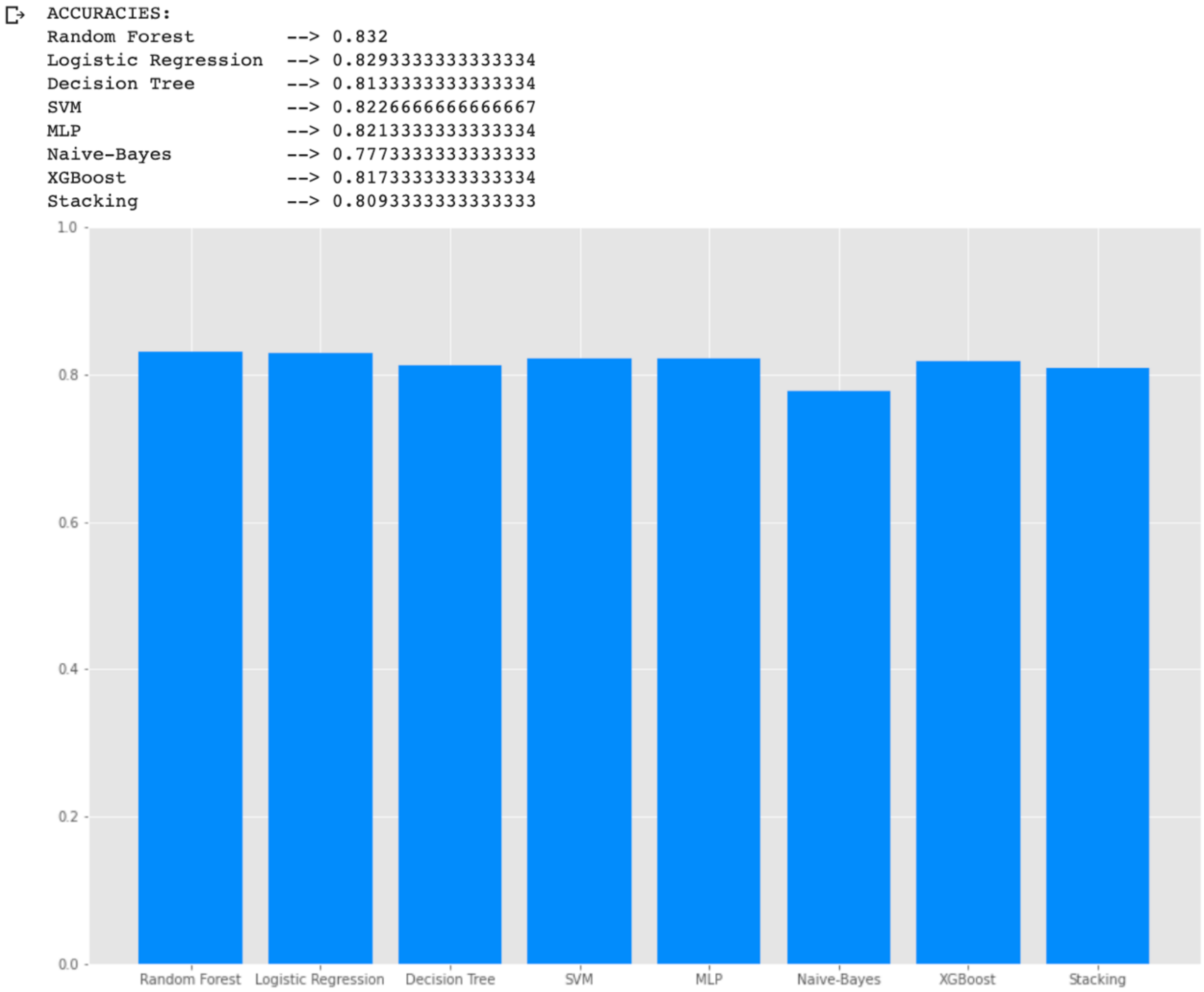
```
  Accuracy on Test Data:  98.55702221857834
              precision    recall  f1-score   support

           0       1.00      0.97      0.99      6430
           1       0.97      1.00      0.98      5767

    accuracy                           0.99     12197
   macro avg       0.99      0.99      0.99     12197
weighted avg       0.99      0.99      0.99     12197
```

ACCURACY TABLE

S.NO	STACKING COMBINATION	ACCURACY
1	Random Forest, SVM, XGBoost	98.5898171681561
2	Logistic Regression, Random Forest, XGBoost	98.5898171681561
3	Logistic Regression, Decision Tree, Random Forest	98.60621464294499
4	Decision Tree, Naive-Bayes, XGBoost	77.0845289825367
5	Logistic Regression, MLP, XGBoost	76.72378453718127
6	Random Forest, MLP, XGBoost	98.5488234811839
7	Decision Tree, Random Forest, SVM	98.57341969336723
8	Decision Tree, Random Forest, MLP	98.58161843076167
9	Logistic Regression, Random Forest, MLP	98.56522095597278
10	Logistic Regression, Random Forest, Naive-Bayes	98.55702221857834

4.4. FINAL ACCURACY GRAPHS:

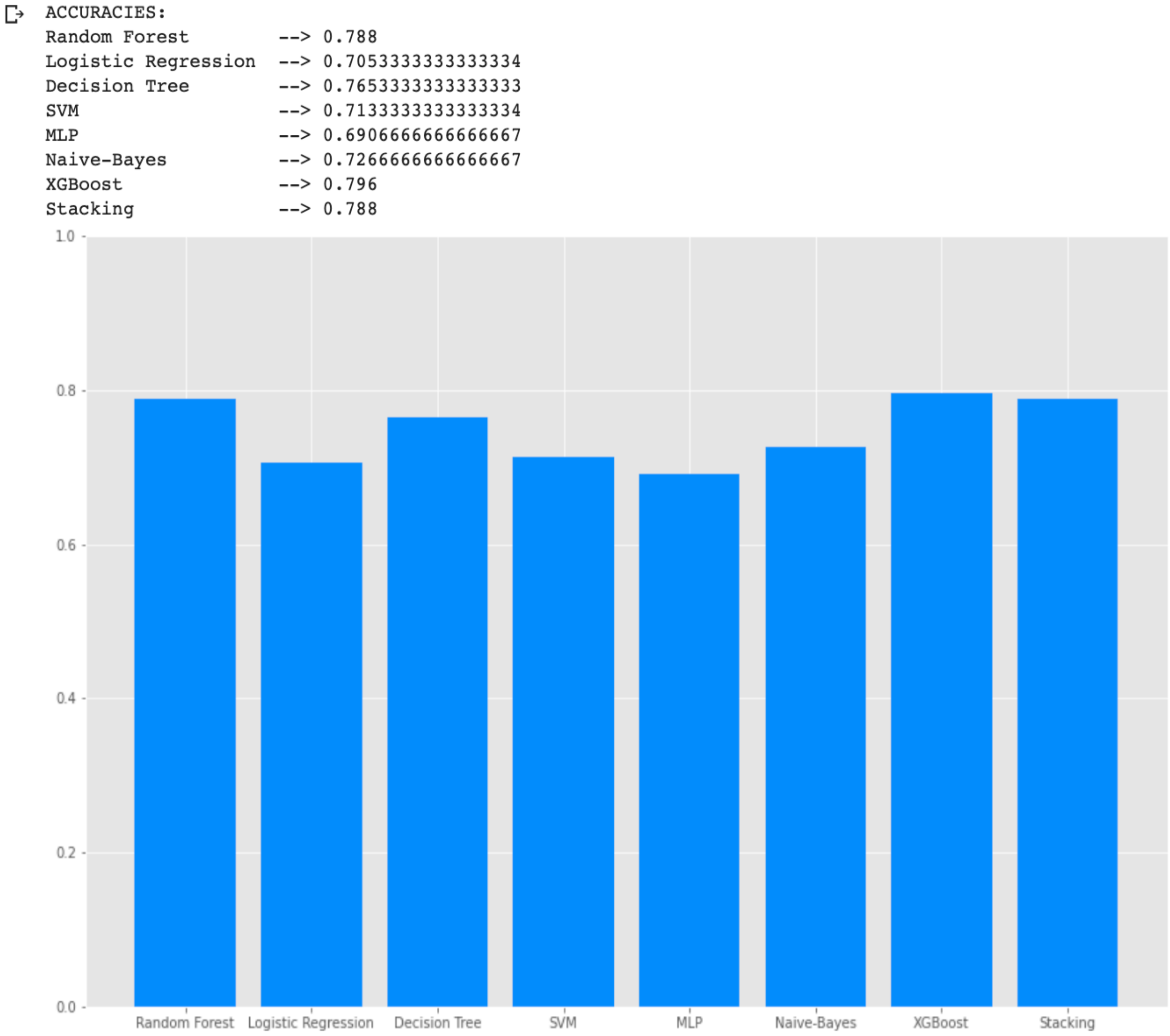
4.4.1. WITHOUT DATA AUGMENTATION:



```
ACCURACIES:
Random Forest        --> 0.832
Logistic Regression  --> 0.8293333333333334
Decision Tree        --> 0.8133333333333334
SVM                  --> 0.8226666666666667
MLP                  --> 0.8213333333333334
Naive-Bayes          --> 0.7773333333333333
XGBoost              --> 0.8173333333333334
Stacking             --> 0.8093333333333333
```

Fig: Accuracies 1

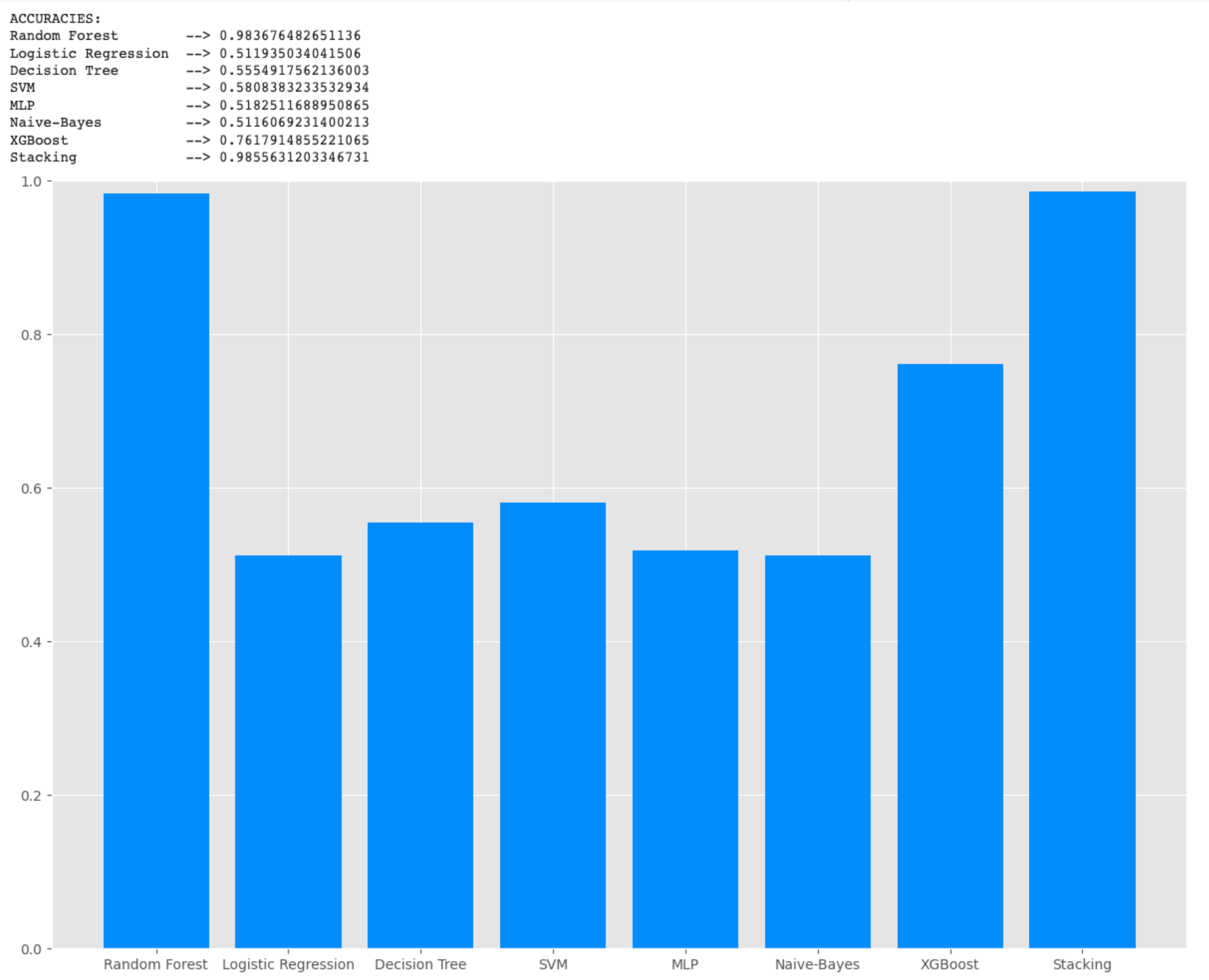
4.4.2. AFTER SMOTE UPSAMPLING:



```
ACCURACIES:
Random Forest          --> 0.788
Logistic Regression    --> 0.7053333333333334
Decision Tree          --> 0.7653333333333333
SVM                    --> 0.7133333333333334
MLP                    --> 0.6906666666666667
Naive-Bayes            --> 0.7266666666666667
XGBoost                --> 0.796
Stacking               --> 0.788
```

Fig: Accuracies 2

4.4.3. AFTER BOOTSTRAPPING UPSAMPLING AND DATASET SIZE INCREASE:

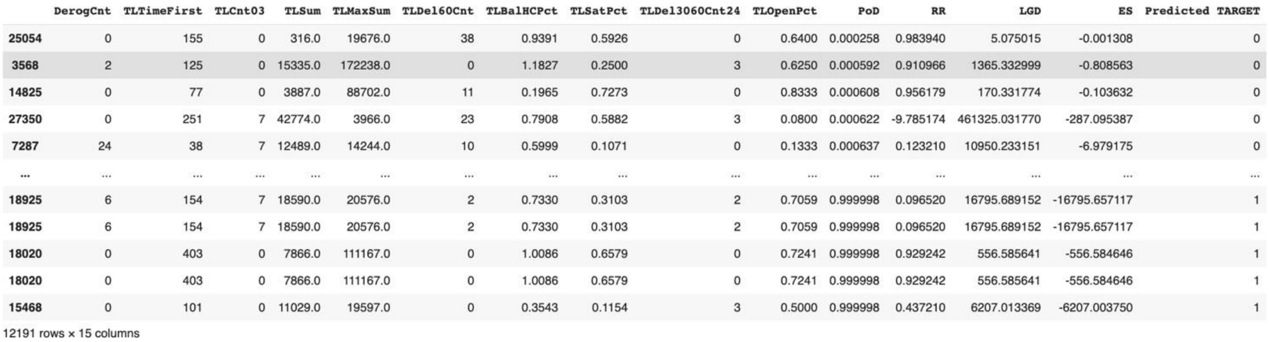


```
ACCURACIES:
Random Forest        --> 0.983676482651136
Logistic Regression  --> 0.511935034041506
Decision Tree        --> 0.5554917562136003
SVM                  --> 0.5808383233532934
MLP                  --> 0.5182511688950865
Naive-Bayes          --> 0.5116069231400213
XGBoost              --> 0.7617914855221065
Stacking             --> 0.9855631203346731
```

Fig 17: Accuracies 3

4.5. PROFIT-RISK ANALYSIS:

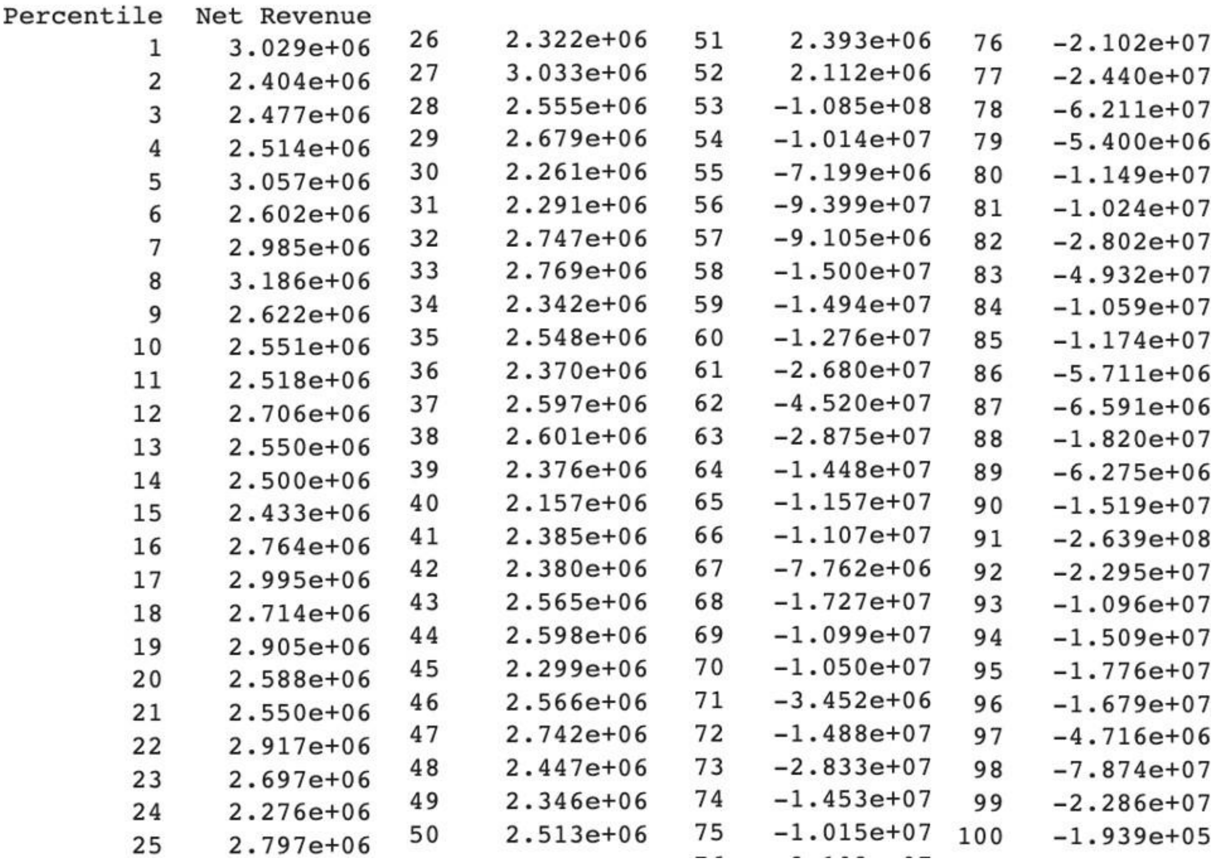
4.5.1. AFTER ESTIMATED LOSS CALCULATION AND SORTING THE DATASET IN ASCENDING ORDER OF PROBABLITY OF DEFAULT VALUES:



	DerogCnt	TLTimeFirst	TLCnt03	TLSum	TLMaxSum	TLDel60Cnt	TLBalHCPct	TLSatPct	TLDel3060Cnt24	TLOpenPct	PoD	RR	LGD	ES	Predicted TARGET
25054	0	155	0	316.0	19676.0	38	0.9391	0.5926	0	0.6400	0.000258	0.983940	5.075015	-0.001308	0
3568	2	125	0	15335.0	172238.0	0	1.1827	0.2500	3	0.6250	0.000592	0.910966	1365.332999	-0.808563	0
14825	0	77	0	3887.0	88702.0	11	0.1965	0.7273	0	0.8333	0.000608	0.956179	170.331774	-0.103632	0
27350	0	251	7	42774.0	3966.0	23	0.7908	0.5882	3	0.0800	0.000622	-9.785174	461325.031770	-287.095387	0
7287	24	38	7	12489.0	14244.0	10	0.5999	0.1071	0	0.1333	0.000637	0.123210	10950.233151	-6.979175	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
18925	6	154	7	18590.0	20576.0	2	0.7330	0.3103	2	0.7059	0.999998	0.096520	16795.689152	-16795.657117	1
18925	6	154	7	18590.0	20576.0	2	0.7330	0.3103	2	0.7059	0.999998	0.096520	16795.689152	-16795.657117	1
18020	0	403	0	7866.0	111167.0	0	1.0086	0.6579	0	0.7241	0.999998	0.929242	556.585641	-556.584646	1
18020	0	403	0	7866.0	111167.0	0	1.0086	0.6579	0	0.7241	0.999998	0.929242	556.585641	-556.584646	1
15468	0	101	0	11029.0	19597.0	0	0.3543	0.1154	3	0.5000	0.999998	0.437210	6207.013369	-6207.003750	1

12191 rows × 15 columns

4.5.2. PERCENTILE RESULTS:



```
Percentile  Net Revenue
         1    3.029e+06   26    2.322e+06   51    2.393e+06   76   -2.102e+07
         2    2.404e+06   27    3.033e+06   52    2.112e+06   77   -2.440e+07
         3    2.477e+06   28    2.555e+06   53   -1.085e+08   78   -6.211e+07
         4    2.514e+06   29    2.679e+06   54   -1.014e+07   79   -5.400e+06
         5    3.057e+06   30    2.261e+06   55   -7.199e+06   80   -1.149e+07
         6    2.602e+06   31    2.291e+06   56   -9.399e+07   81   -1.024e+07
         7    2.985e+06   32    2.747e+06   57   -9.105e+06   82   -2.802e+07
         8    3.186e+06   33    2.769e+06   58   -1.500e+07   83   -4.932e+07
         9    2.622e+06   34    2.342e+06   59   -1.494e+07   84   -1.059e+07
        10    2.551e+06   35    2.548e+06   60   -1.276e+07   85   -1.174e+07
        11    2.518e+06   36    2.370e+06   61   -2.680e+07   86   -5.711e+06
        12    2.706e+06   37    2.597e+06   62   -4.520e+07   87   -6.591e+06
        13    2.550e+06   38    2.601e+06   63   -2.875e+07   88   -1.820e+07
        14    2.500e+06   39    2.376e+06   64   -1.448e+07   89   -6.275e+06
        15    2.433e+06   40    2.157e+06   65   -1.157e+07   90   -1.519e+07
        16    2.764e+06   41    2.385e+06   66   -1.107e+07   91   -2.639e+08
        17    2.995e+06   42    2.380e+06   67   -7.762e+06   92   -2.295e+07
        18    2.714e+06   43    2.565e+06   68   -1.727e+07   93   -1.096e+07
        19    2.905e+06   44    2.598e+06   69   -1.099e+07   94   -1.509e+07
        20    2.588e+06   45    2.299e+06   70   -1.050e+07   95   -1.776e+07
        21    2.550e+06   46    2.566e+06   71   -3.452e+06   96   -1.679e+07
        22    2.917e+06   47    2.742e+06   72   -1.488e+07   97   -4.716e+06
        23    2.697e+06   48    2.447e+06   73   -2.833e+07   98   -7.874e+07
        24    2.276e+06   49    2.346e+06   74   -1.453e+07   99   -2.286e+07
        25    2.797e+06   50    2.513e+06   75   -1.015e+07  100   -1.939e+05
```

Net revenue is observed to be positive until the 52nd percentile. Indicating a profit revenue till the said percentile.

5. RESULTS [DATASET – II]

5.1. PRE-PROCESSING

This dataset contains a lot of categorical data which requires data preparation such as cleaning, transforming and pre-processing raw data into a format that can be used by machine learning algorithms to extract meaningful insights and patterns.

Categorical data refers to data that contains non-numeric values that represent categories or labels, such as gender, occupation, or color. Preprocessing of categorical data is necessary to convert it into a numerical format that can be used in machine learning algorithms. Here are some common preprocessing techniques for categorical data:

Data Encoding: One of the most common techniques for preprocessing categorical data is encoding. There are several encoding techniques that can be used, including:

- One-Hot Encoding: This technique creates binary variables for each category in the data. For example, if there are three categories (red, green, blue) in a variable, three new variables are created with binary values (1 or 0) for each category. This technique works well for nominal variables where the categories have no inherent order.

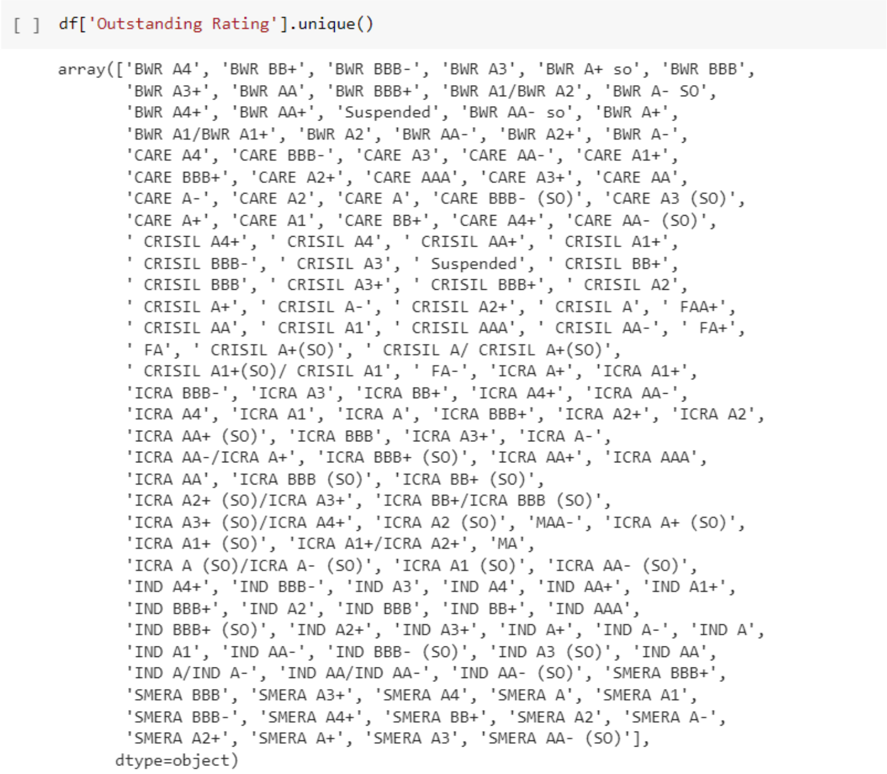
- Label Encoding: This technique assigns a unique integer value to each category in the data. For example, if there are three categories (red, green, blue) in a variable, they can be assigned integer values of 1, 2, and 3. This technique works well for ordinal variables where the categories have an inherent order.

Data Imputation: Categorical data may also contain missing values, and these values need to be imputed before preprocessing. Imputation techniques include replacing missing values with the mode (most frequent category) or creating a new category for missing values.

Data Reduction: Categorical data may also contain a large number of categories, and this can lead to the "curse of dimensionality" problem. To avoid this, the number of categories can be reduced using techniques such as grouping similar categories together or creating new categories based on domain knowledge.

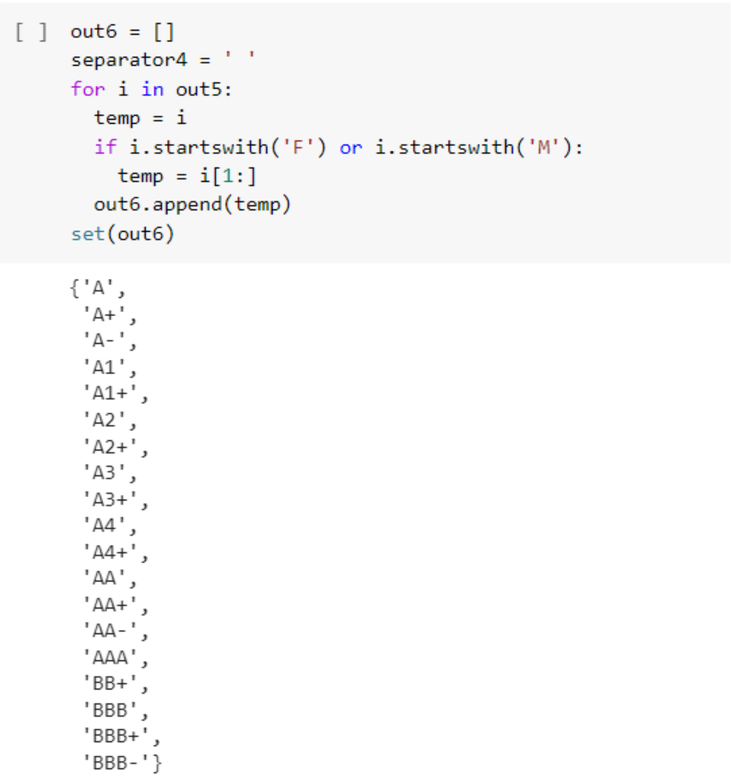
For the following attributes, data encoding has been performed:

- **Outstanding Rating**



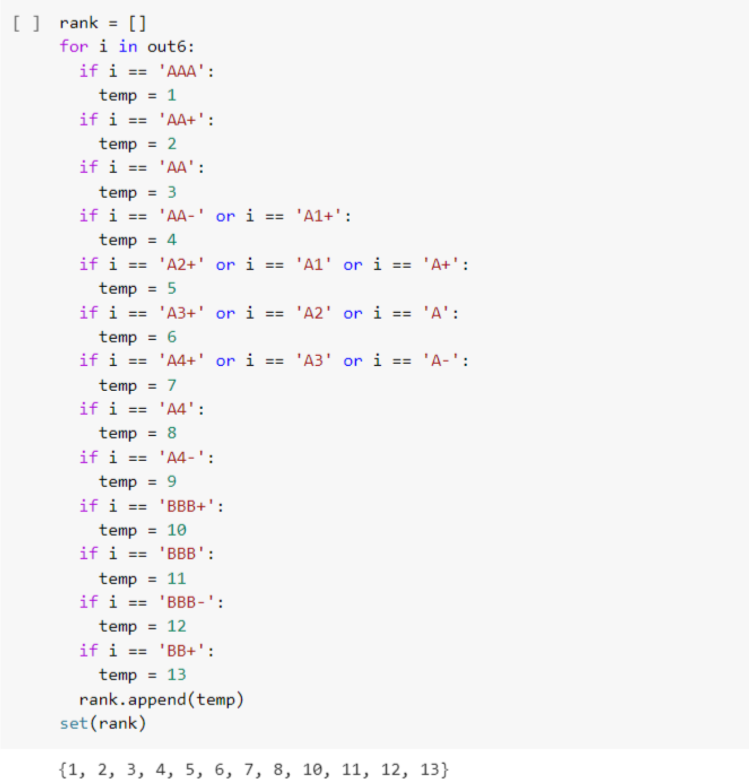
```
[ ]  df['Outstanding Rating'].unique()
```

```
array(['BWR A4', 'BWR BB+', 'BWR BBB-', 'BWR A3', 'BWR A+ so', 'BWR BBB',
       'BWR A3+', 'BWR AA', 'BWR BBB+', 'BWR A1/BWR A2', 'BWR A- SO',
       'BWR A4+', 'BWR AA+', 'Suspended', 'BWR AA- so', 'BWR A+',
       'BWR A1/BWR A1+', 'BWR A2', 'BWR AA-', 'BWR A2+', 'BWR A-',
       'CARE A4', 'CARE BBB-', 'CARE A3', 'CARE AA-', 'CARE A1+',
       'CARE BBB+', 'CARE A2+', 'CARE AAA', 'CARE A3+', 'CARE AA',
       'CARE A-', 'CARE A2', 'CARE A', 'CARE BBB- (SO)', 'CARE A3 (SO)',
       'CARE A+', 'CARE A1', 'CARE BB+', 'CARE A4+', 'CARE AA- (SO)',
       ' CRISIL A4+', ' CRISIL A4', ' CRISIL AA+', ' CRISIL A1+',
       ' CRISIL BBB-', ' CRISIL A3', ' Suspended', ' CRISIL BB+',
       ' CRISIL BBB', ' CRISIL A3+', ' CRISIL BBB+', ' CRISIL A2',
       ' CRISIL A+', ' CRISIL A-', ' CRISIL A2+', ' CRISIL A', ' FAA+',
       ' CRISIL AA', ' CRISIL A1', ' CRISIL AAA', ' CRISIL AA-', ' FA+',
       ' FA', ' CRISIL A+(SO)', ' CRISIL A/ CRISIL A+(SO)',
       ' CRISIL A1+(SO)/ CRISIL A1', ' FA-', 'ICRA A+', 'ICRA A1+',
       'ICRA BBB-', 'ICRA A3', 'ICRA BB+', 'ICRA A4+', 'ICRA AA-',
       'ICRA A4', 'ICRA A1', 'ICRA A', 'ICRA BBB+', 'ICRA A2+', 'ICRA A2',
       'ICRA AA+ (SO)', 'ICRA BBB', 'ICRA A3+', 'ICRA A-',
       'ICRA AA-/ICRA A+', 'ICRA BBB+ (SO)', 'ICRA AA+', 'ICRA AAA',
       'ICRA AA', 'ICRA BBB (SO)', 'ICRA BB+ (SO)',
       'ICRA A2+ (SO)/ICRA A3+', 'ICRA BB+/ICRA BBB (SO)',
       'ICRA A3+ (SO)/ICRA A4+', 'ICRA A2 (SO)', 'MAA-', 'ICRA A+ (SO)',
       'ICRA A1+ (SO)', 'ICRA A1+/ICRA A2+', 'MA',
       'ICRA A (SO)/ICRA A- (SO)', 'ICRA A1 (SO)', 'ICRA AA- (SO)',
       'IND A4+', 'IND BBB-', 'IND A3', 'IND A4', 'IND AA+', 'IND A1+',
       'IND BBB+', 'IND A2', 'IND BBB', 'IND BB+', 'IND AAA',
       'IND BBB+ (SO)', 'IND A2+', 'IND A3+', 'IND A+', 'IND A-', 'IND A',
       'IND A1', 'IND AA-', 'IND BBB- (SO)', 'IND A3 (SO)', 'IND AA',
       'IND A/IND A-', 'IND AA/IND AA-', 'IND AA- (SO)', 'SMERA BBB+',
       'SMERA BBB', 'SMERA A3+', 'SMERA A4', 'SMERA A', 'SMERA A1',
       'SMERA BBB-', 'SMERA A4+', 'SMERA BB+', 'SMERA A2', 'SMERA A-',
       'SMERA A2+', 'SMERA A+', 'SMERA A3', 'SMERA AA- (SO)'],
      dtype=object)
```



```
[ ]  out6 = []
     separator4 = ' '
     for i in out5:
       temp = i
       if i.startswith('F') or i.startswith('M'):
         temp = i[1:]
       out6.append(temp)
     set(out6)
```

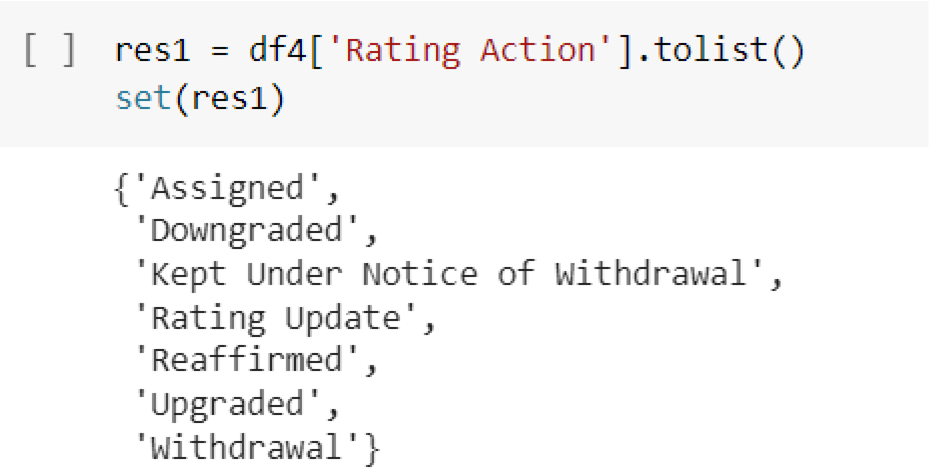
```
{'A',
 'A+',
 'A-',
 'A1',
 'A1+',
 'A2',
 'A2+',
 'A3',
 'A3+',
 'A4',
 'A4+',
 'AA',
 'AA+',
 'AA-',
 'AAA',
 'BB+',
 'BBB',
 'BBB+',
 'BBB-'}
```



```
[ ]  rank = []
     for i in out6:
       if i == 'AAA':
         temp = 1
       if i == 'AA+':
         temp = 2
       if i == 'AA':
         temp = 3
       if i == 'AA-' or i == 'A1+':
         temp = 4
       if i == 'A2+' or i == 'A1' or i == 'A+':
         temp = 5
       if i == 'A3+' or i == 'A2' or i == 'A':
         temp = 6
       if i == 'A4+' or i == 'A3' or i == 'A-':
         temp = 7
       if i == 'A4':
         temp = 8
       if i == 'A4-':
         temp = 9
       if i == 'BBB+':
         temp = 10
       if i == 'BBB':
         temp = 11
       if i == 'BBB-':
         temp = 12
       if i == 'BB+':
         temp = 13
       rank.append(temp)
     set(rank)
```

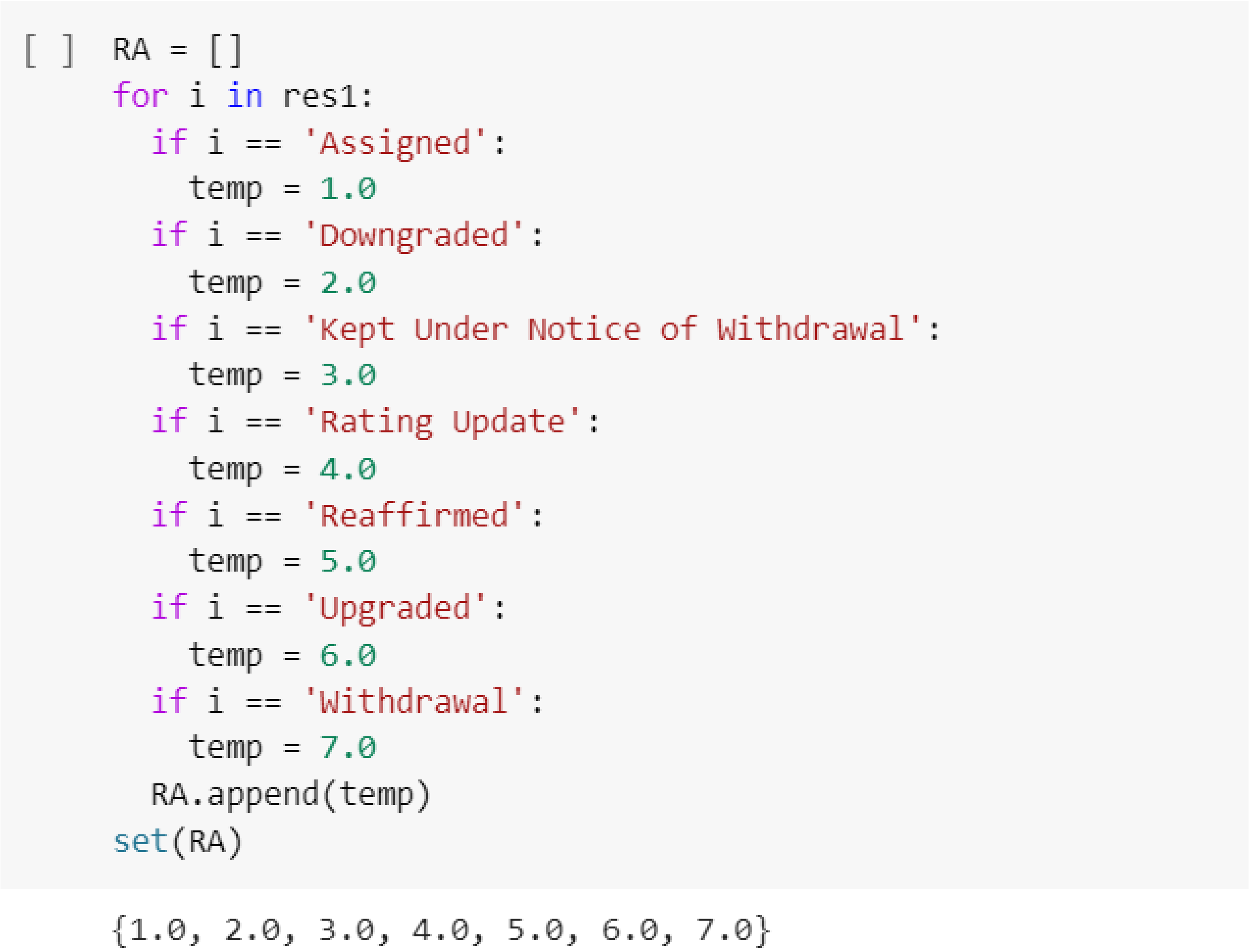
```
{1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13}
```

- **Rating Action**



```
res1 = df4['Rating Action'].tolist()
set(res1)
```

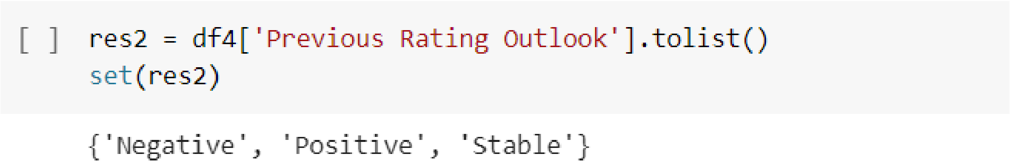
```
{'Assigned',
 'Downgraded',
 'Kept Under Notice of Withdrawal',
 'Rating Update',
 'Reaffirmed',
 'Upgraded',
 'Withdrawal'}
```



```
RA = []
for i in res1:
  if i == 'Assigned':
    temp = 1.0
  if i == 'Downgraded':
    temp = 2.0
  if i == 'Kept Under Notice of Withdrawal':
    temp = 3.0
  if i == 'Rating Update':
    temp = 4.0
  if i == 'Reaffirmed':
    temp = 5.0
  if i == 'Upgraded':
    temp = 6.0
  if i == 'Withdrawal':
    temp = 7.0
  RA.append(temp)
set(RA)
```

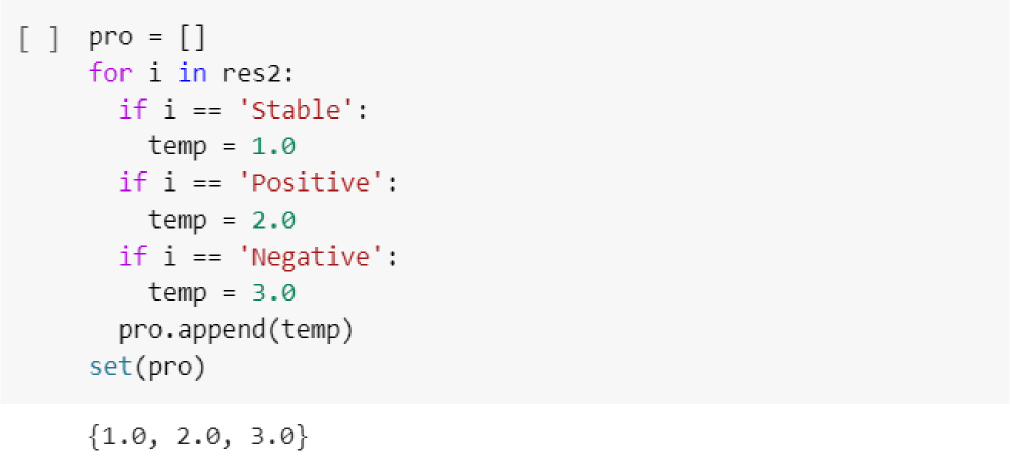
```
{1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0}
```

- **Previous Rating Outlook**



```
res2 = df4['Previous Rating Outlook'].tolist()
set(res2)
```

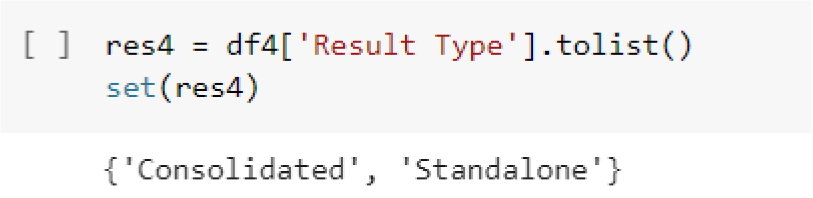
```
{'Negative', 'Positive', 'Stable'}
```



```
pro = []
for i in res2:
  if i == 'Stable':
    temp = 1.0
  if i == 'Positive':
    temp = 2.0
  if i == 'Negative':
    temp = 3.0
  pro.append(temp)
set(pro)
```

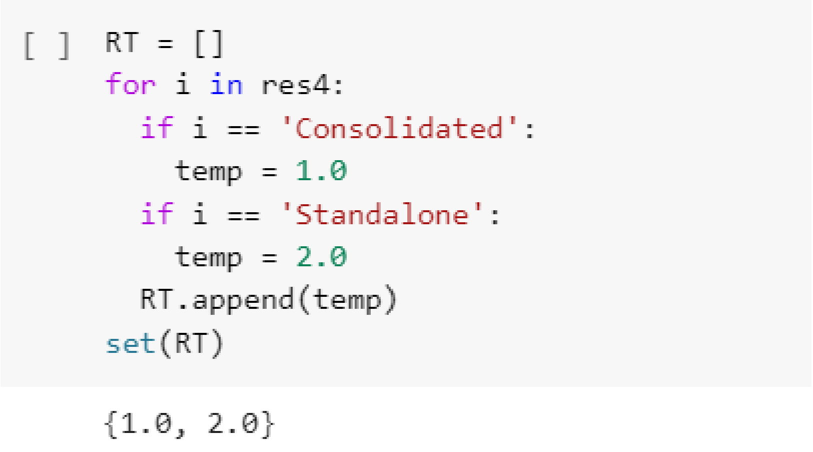
```
{1.0, 2.0, 3.0}
```

- **Result Type**



```
res4 = df4['Result Type'].tolist()
set(res4)
```

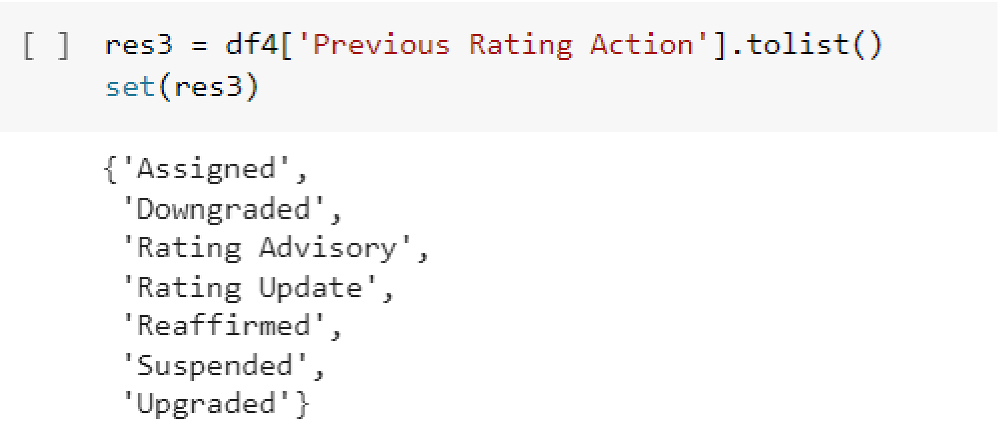
```
{'Consolidated', 'Standalone'}
```



```
RT = []
for i in res4:
  if i == 'Consolidated':
    temp = 1.0
  if i == 'Standalone':
    temp = 2.0
  RT.append(temp)
set(RT)
```

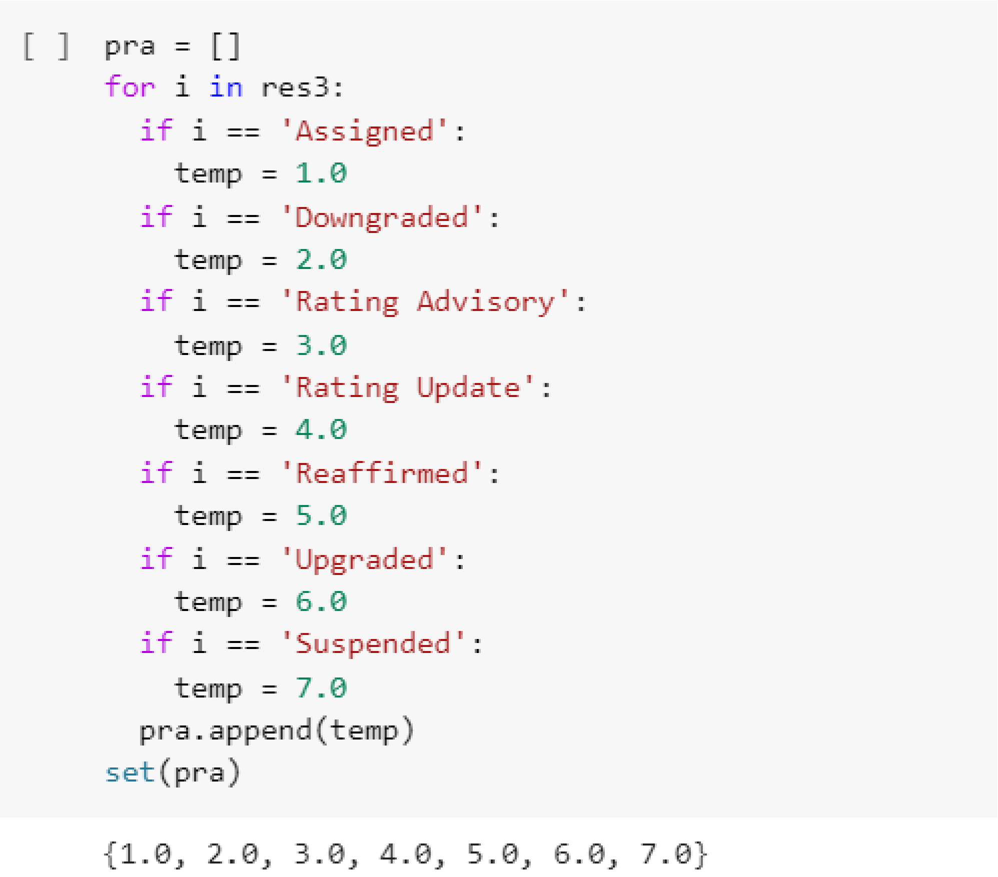
```
{1.0, 2.0}
```

- **Previous Rating Action**



```
res3 = df4['Previous Rating Action'].tolist()
set(res3)
```

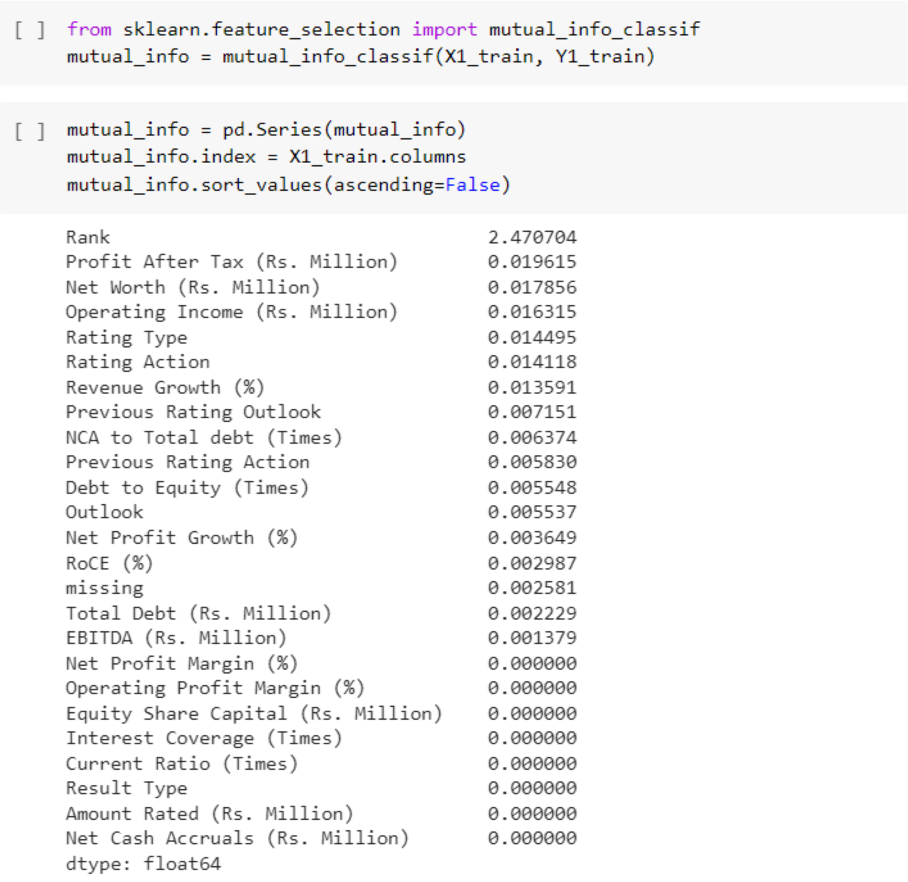
```
{'Assigned',
 'Downgraded',
 'Rating Advisory',
 'Rating Update',
 'Reaffirmed',
 'Suspended',
 'Upgraded'}
```



```
pra = []
for i in res3:
  if i == 'Assigned':
    temp = 1.0
  if i == 'Downgraded':
    temp = 2.0
  if i == 'Rating Advisory':
    temp = 3.0
  if i == 'Rating Update':
    temp = 4.0
  if i == 'Reaffirmed':
    temp = 5.0
  if i == 'Upgraded':
    temp = 6.0
  if i == 'Suspended':
    temp = 7.0
  pra.append(temp)
set(pra)
```

```
{1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0}
```

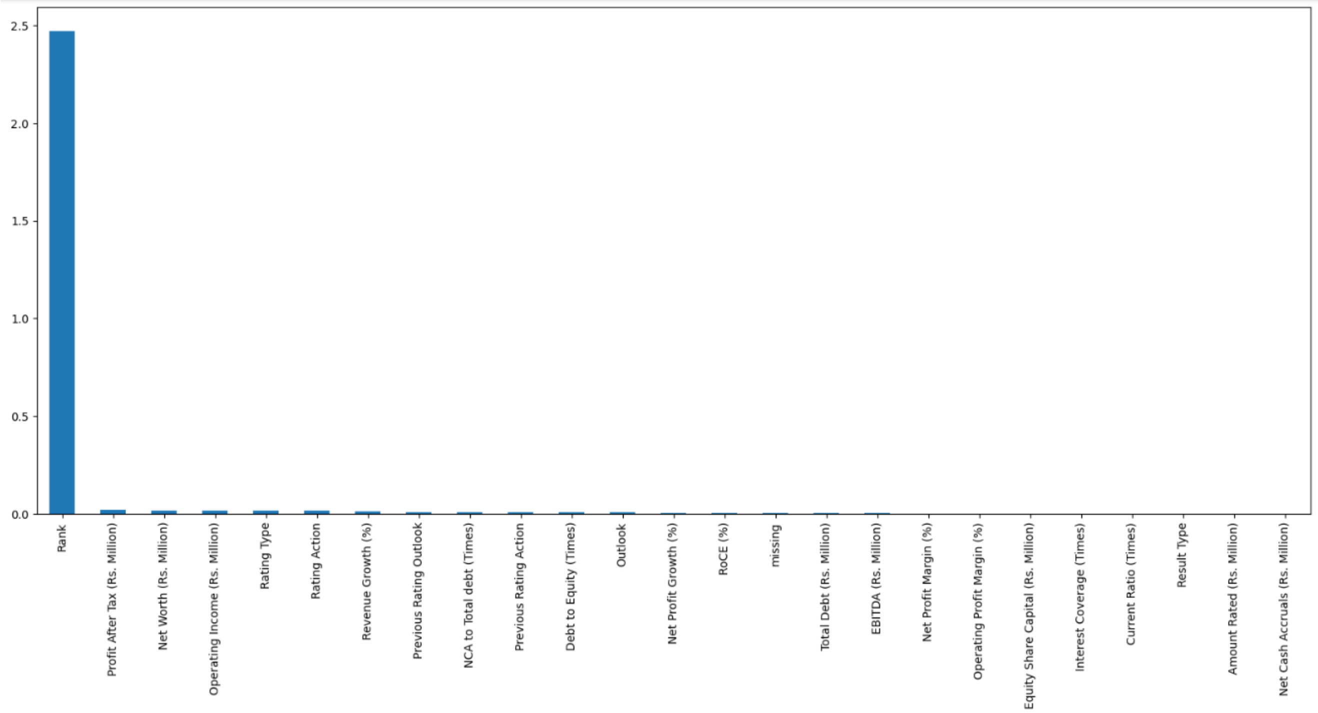
5.2. FEATURE SELECTION



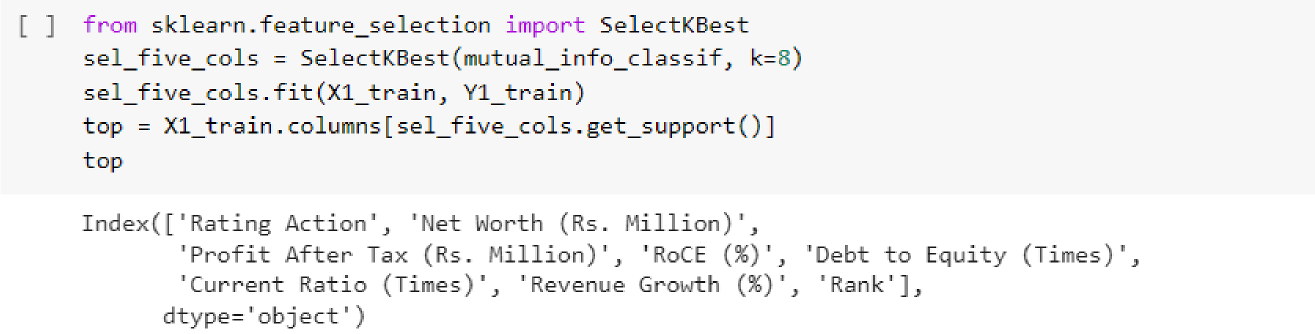
```
[ ]  from sklearn.feature_selection import mutual_info_classif
     mutual_info = mutual_info_classif(X1_train, Y1_train)
```

```
[ ]  mutual_info = pd.Series(mutual_info)
     mutual_info.index = X1_train.columns
     mutual_info.sort_values(ascending=False)
```

```
Rank                                2.470704
Profit After Tax (Rs. Million)      0.019615
Net Worth (Rs. Million)             0.017856
Operating Income (Rs. Million)      0.016315
Rating Type                         0.014495
Rating Action                       0.014118
Revenue Growth (%)                  0.013591
Previous Rating Outlook             0.007151
NCA to Total debt (Times)           0.006374
Previous Rating Action              0.005830
Debt to Equity (Times)              0.005548
Outlook                             0.005537
Net Profit Growth (%)               0.003649
RoCE (%)                            0.002987
missing                             0.002581
Total Debt (Rs. Million)            0.002229
EBITDA (Rs. Million)                0.001379
Net Profit Margin (%)               0.000000
Operating Profit Margin (%)         0.000000
Equity Share Capital (Rs. Million)  0.000000
Interest Coverage (Times)           0.000000
Current Ratio (Times)               0.000000
Result Type                         0.000000
Amount Rated (Rs. Million)          0.000000
Net Cash Accruals (Rs. Million)     0.000000
dtype: float64
```



Here, we can see, the attributes on the left side of the chart having the higher mutual information value are more consequential in predicting the output value then the rest. The top eight attributes alone are chosen. This value is chosen based on the observation that many features end up contributing almost nothing to the outcome class.



```
[ ]  from sklearn.feature_selection import SelectKBest
     sel_five_cols = SelectKBest(mutual_info_classif, k=8)
     sel_five_cols.fit(X1_train, Y1_train)
     top = X1_train.columns[sel_five_cols.get_support()]
     top

     Index(['Rating Action', 'Net Worth (Rs. Million)',
            'Profit After Tax (Rs. Million)', 'RoCE (%)', 'Debt to Equity (Times)',
            'Current Ratio (Times)', 'Revenue Growth (%)', 'Rank'],
           dtype='object')
```

5.3. MODELLING

Since the dataset is very skewed in nature, it creates an overfitting problem under most machine learning algorithms. But here is where random forest comes into picture.

Why is Random Forest a better ML model?

- Impressive in Versatility

Whether you have a regression or classification task, random forest is an applicable model for your needs. It can handle binary features, categorical features, and numerical features. There is very little pre-processing that needs to be done. The data does not need to be rescaled or transformed.

- Great with High dimensionality

Random forests is great with high dimensional data since we are working with subsets of data.

- Quick Prediction/Training Speed

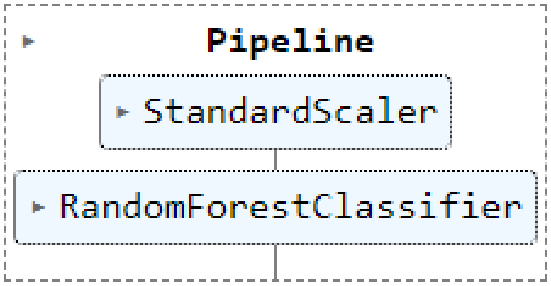
It is faster to train than decision trees because we are working only on a subset of features in this model, so we can easily work with hundreds of features. Prediction speed is significantly faster than training speed because we can save generated forests for future uses.

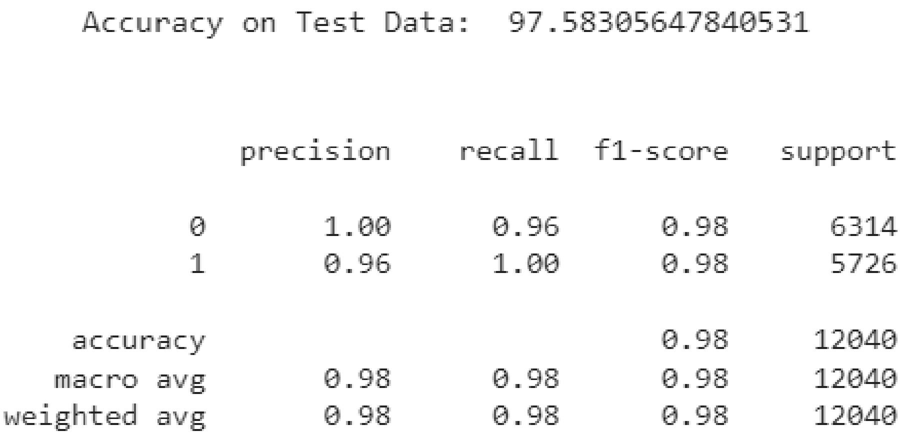
- Handles Unbalanced Data

It has methods for balancing error in class population unbalanced data sets. Random forest tries to minimize the overall error rate, so when we have an unbalance data set, the larger class will get a low error rate while the smaller class will have a larger error rate.

- Low Bias, Moderate Variance

Each decision tree has a high variance, but low bias. But because we average all the trees in random forest, we are averaging the variance as well so that we have a low bias and moderate variance model.




```
Accuracy on Test Data:  97.58305647840531
```

```
              precision    recall  f1-score   support

           0       1.00      0.96      0.98      6314
           1       0.96      1.00      0.98      5726

    accuracy                           0.98     12040
   macro avg       0.98      0.98      0.98     12040
weighted avg       0.98      0.98      0.98     12040
```

Thus, the above accuracy is the highest accuracy obtained using Random Forest model.

6. CONCLUSION AND FUTURE WORK

Thus, from the above results, we can see that, up until the 52nd percentile, the net revenue is positive, indicating a profit for the financial institute, and after that, losses start coming up. Therefore, the bank or financial institute can formulate a policy that loans will be given to people up to the 52nd percentile when sorted in ascending order of probability of default.

After the dataset size is increased to counteract the small size of the existing dataset, the class imbalance issue is fixed using the bootstrapping method. SMOTE, although popular to rectify class imbalance issues, fails to increase accuracy from the default case. It is also found that the ensemble stacking algorithm provides the best accuracy of 98.55%, which is very exceptional out of all the base learners, and is thus adopted to predict the probability of default values.

In the future, better and bigger datasets will need to be tested out with the current model to further test the model's potency. Further, in this work, only estimated losses or profits on revenue are calculated. With more information, such as the principal amount, interest rate, number of times the loan is compounded, etc., available, a more accurate measure of profit-loss values can be calculated.

REFERENCES:

[1] Kennedy, K. (2013). **Credit scoring using machine learning. Doctoral thesis. Technological University Dublin.**

[2] Provenzano, A. R., Trifirò, D., Datteo, A., Giada, L., Jean, N., Riciputi, A., ... & Nordio, C. (2020). **Machine learning approach for credit scoring.**

[3] Ampountolas, A., Nyarko Nde, T., Date, P., & Constantinescu, C. (2021).
A machine learning approach for micro-credit scoring.

[4] Nyon, E. E., & Matshisela, N. (2018). **Credit scoring using machine learning algorithims. Zimbabwe Journal of Science and Technology**.

[5] Abdou, H. A., & Pointon, J. (2011). **Credit scoring, statistical techniques and evaluation criteria: a review of the literature. Intelligent systems in accounting, finance and management.**

[6] Abdou, H. A., & Pointon, J. (2009). **Credit scoring and decision making in Egyptian public sector banks. International Journal of Managerial Finance.**

[7] Luo, C., Wu, D., & Wu, D. (2017). **A deep learning approach for credit scoring using credit default swaps. Engineering Applications of Artificial Intelligence**.

[8] Pol, S., & Ambekar, S. S. (2022). **Predicting Credit Ratings using Deep Learning Models–An Analysis of the Indian IT Industry. Australasian Accounting, Business and Finance Journal**.

[9] Addo, P. M., Guegan, D., & Hassani, B. (2018). **Credit risk analysis using machine and deep learning models.**

[10] Ferreira, F. A., Spahr, R. W., Gavancha, I. F., & Çipi, A. (2013). **Readjusting trade-offs among criteria in internal ratings of credit-scoring: an empirical essay of risk analysis in mortgage loans. Journal of Business Economics and Management.**