VEHICLE LOAN PREDICTION

Text

Description automatically generated

REPORT BY,

Mrunalini Devineni

Aihan Liu

Sara Sanchez

DATS6103

Professor Amir Jafari

December 6, 2021

TABLE OF CONTENTS

|  |  |
| --- | --- |
| *Introduction* | *3* |
| *Data Description* | ***4*** |
| *EDA* | ***5*** |
| *Data Preprocessing* | ***8*** |
| *Data Modelling* | ***10*** |
| *Conclusion* | ***11*** |
| *Reference* | ***12*** |
| *Appendix* | ***13*** |

INTRODUCTION

Financial institutions incur significant losses due to the default of Vehicle Loans. This situation has led to the constricting of vehicle loan underwriting and increased vehicle loan rejection rates. These institutions also raise the need for a better credit risk scoring model. By doing this, the institutions are trying to accurately predict the Probability of loanee defaulting on a vehicle loan on the due date. In this sense, the credit scores are significant as a tool that can decide which clients can take or not a credit, considering their unique characteristics. The scoring is also a helpful tool for the clients because this can avoid accepting a loan that will not be able to be paid in the future and, consequently, prevent having problems with financial institutions.

Furthermore, the scoring tool is helpful not only for cash loans but also for mortgages and car loans, so this is the main reason that motivated us to choose this topic because it is helpful for many kinds of businesses.

To develop this scoring project, we will analyze an L&T car loan company database from Kaggle. Then we are doing some preprocessing and cleaning of the data to apply the classification models like Decision tree, Random Forest, logistic, and Gradient Boosting. Finally, we will show the results by using the Pyqt5.

The following report will be organized: first, we will describe the data set, then the methodology to be used, and finally, the results and main conclusions.

DATA DESCRIPTION

For this study, we have chosen to analyze an L&T company dataset, in charge of cars' sale from kaggle.com. This dataset is like the one that financial institution must build the scoring models that allow them to forecast the approval or rejection of customers. This dataset does not require any cleaning and is equipped to fuel the analysis of this project. The base consists of 40 variables and 233 154 observations assessing a person's attributes ranging from demographic data (date of birth, etc.) and bureau data, like the amount of loan disbursed and the asset's cost.

The following Information regarding the loan and loanee are provided in the datasets:

* **Loanee Information** (Demographic data like age, Identity proof etc.)
* **Loan Information** (Disbursal details, loan to value ratio etc.)
* **Bureau data & history** (Bureau score, number of active accounts, the status of other loans, credit history etc.)

The Dataset available on Kaggle contains:

* Train.csv: a base that contains the training data with details on loan as described in the last section.
* data\_dictionary.csv: a base containing a brief description of each variable

provided in the training and test set.

EXPLORATORY DATA ANALYSIS (EDA)

From the car loan database, we will develop the Exploratory data analysis (EDA). This kind of analysis permits a better understanding of the data and builds a better model. The target variable is the “loan default,” this gives us information related to the number of persons that defaulted, and for this database, the default ratio is around 27.7%.

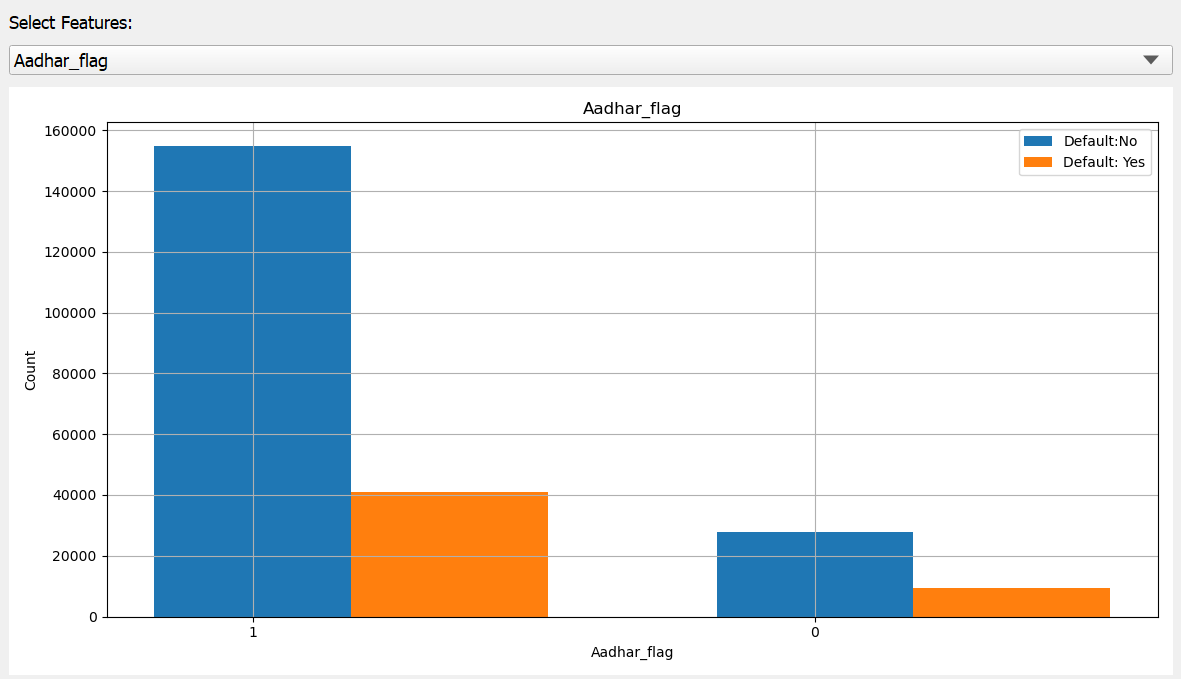
**Figure 1 – Loan\_deafult**

Chart

Description automatically generated

Another essential variable is the Aadhar flag. As this is a database from India, it is crucial to highlight that Aadhar is given to all the citizens from this country. A person who doesn’t have a citizen status will not have an Aadhar. The graph shows that people who have an Aadhar are more likely to be on default than others.

**Figure 2 – Aadhar flag**

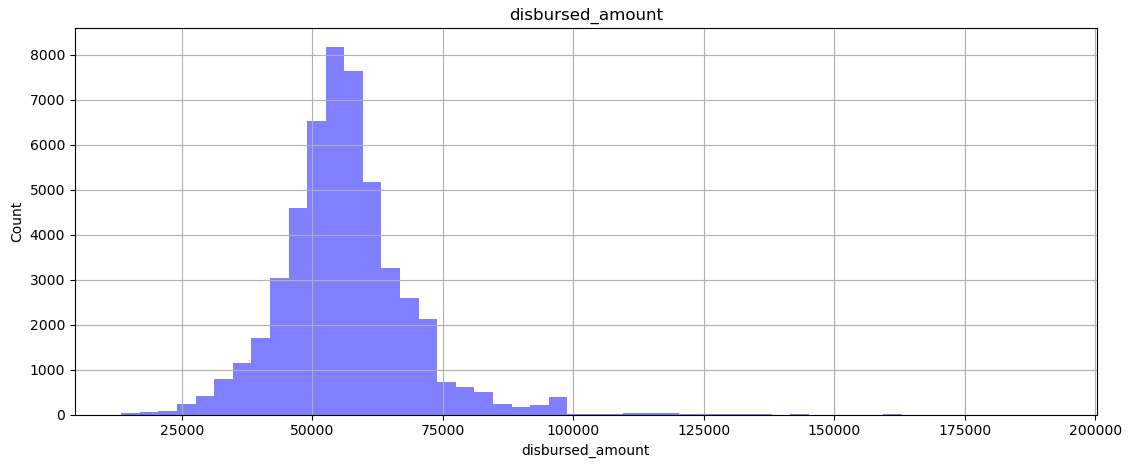
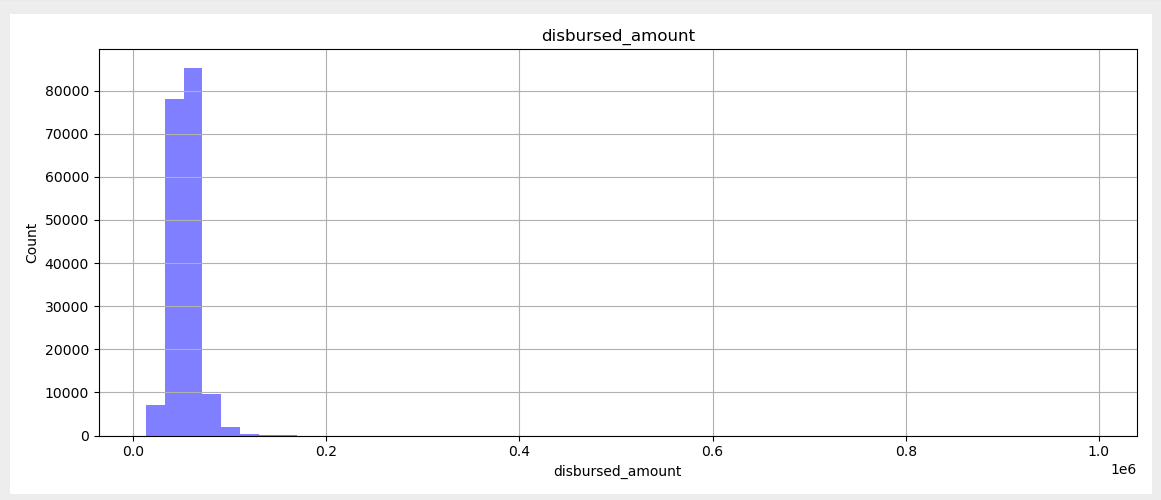


=With Aadhar

=No Aadhar

The disbursed amount is the financial institution's quantity to the borrower. Figure number three shows a significant difference in the amount between those without and those with default. Again, there is a concentration around shorter amounts. This is different from the former who have a normal distribution for the latter.

**Figure 3 – Disbursed amount**

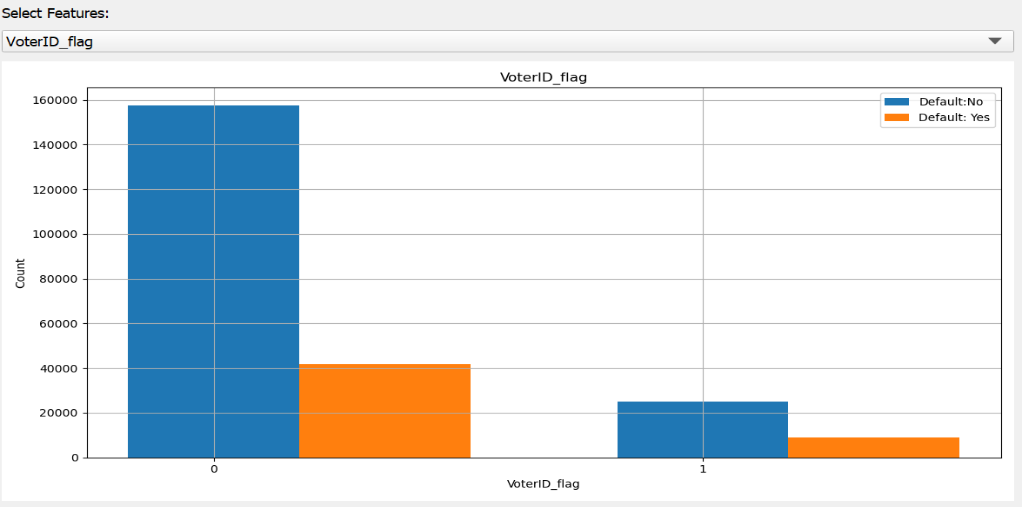


With default

Without default

From figure number four, we can analyze the voter ID flag. This variable is essential because this makes differences between profiles with and without default. As is possible to observe, most people from the database do not have a Voter ID, and, this is the group with the biggest default. Therefore, it is possible to conclude that people who cannot have the biggest default could be an indicator or informality.

**Figure 4 – Voter ID**



=No Voter ID

=with VoterID

From this plot, we can observe a positive and strong correlation. Therefore, we choose to select those points with a correlation more significant than 70%. The variables that fulfill this requirement are asset cost, disbursed amount, sanctioned amount, credit history length, and average account age.

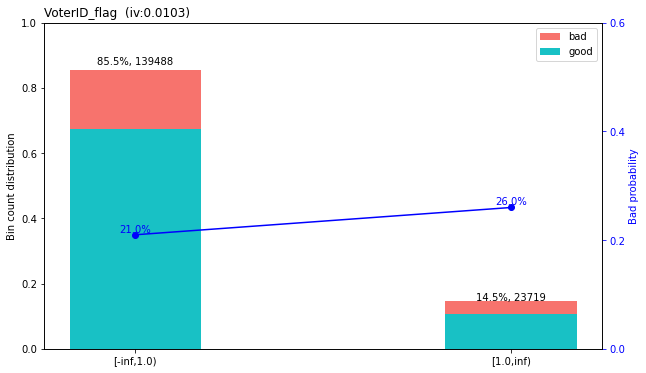
**Figure 5 – Correlation Plot**

Chart, scatter chart

Description automatically generated

The weight of the evidence plot is well known among the risk areas from financial institutions. Therefore, we can interpret this plot as 21 over 100 will default from those with a Voter ID. On the other hand, from those who do not have a Voter ID, 26 over 100 will have a default. This is a crucial plot given that confirms the importance of the variable as a predictor because those who contribute more to the model are the ones that can differentiate the profiles better.

**Figure 6 – Weight of evidence from Voter ID**



=With VoterID

=No VoterID

DATA PREPROCESSING

Data preprocessing is a technique that works for the dirty data that cannot mine directly and transform into the one that is efficiently used. Data preprocessing has several methods: data cleansing, data integration, data transformation, etc. These data processing techniques are used before data mining, significantly improving the quality of the data mining model and reducing the time required to excavate.

The following are the various preprocessing techniques that we applied to our dataset.

**Moving null values:**

There are over 5000 null values in our dataset. To prevent them from affecting the accuracy of the prediction, we remove them from the dataset.

**Format converting:**

Variables' CREDIT.HISTORY. LENGTH' and 'AVERAGE.ACCT.AGE' (credit history length and average account age) have informed date format (such as '2yrs 4mon'), we formatted and calculated them into a month.

**Date of Birth to Age:**

Converted date of birth to age.

**Categorized ID data:**

Made buckets for supplier id, branch id, manufacturer id, and State ID to reduce the levels of these features.

**One hot encoding:**

One hot encoded categorical feature such as employment type and the categorical features we generated earlier.

**Scaling:**

Scaled the features when required. We applied Standard Scaler for the dataset when we used specific algorithms related to the distance calculation, including logistic regression.

**Balanced Data:**

We found that this data was imbalanced since the target feature-loan default was imbalanced. There were fewer defaulters than non-defaulters. To overcome this issue and improve the precision, we applied Synesthetic Minority Oversampling Technique (SMOTE) method to fix the imbalanced data. SMOTE is implemented by finding the k-nearest neighbors for minority class observations and randomly choosing one of the k-nearest neighbors. The new observations it created will be added to the group defaulter.

**Feature Selection**

Since we have over 40 features, we used Random Forest Classifier to find the essential features with a higher step score on the selection iteration. Finally, we have 25 features left in our dataset, which will be applied to the following model generation.

DATA MODELLING

Classification is the process of assigning data points to predefined classes or categories. In this project we have implemented 4 classification Algorithms.

**Decision tree:**

Decision Tree is a tree flowchart like structure that divides the data into different subgroups based on conditions to classify the data. A condition is selected such that the classification is as pure as possible. At each node of the tree a decision is made about how to split the data and how to get the purest nodes. To calculate what attribute to split on, we can use different measures like Gini, entropy or misclassification error. When you travel down the tree, finally at leaf nodes we find the labels of the data of a particular sample.

Chart

Description automatically generated

Graphical user interface, application

Description automatically generated

**Random Forest:**

For random forest, we select random features to check for best split attribute. And we use max voting classifier to classify the data.

Graphical user interface, chart

Description automatically generated

Graphical user interface, application

Description automatically generated

**Logistic Regression:**

Logistic Regression is a Statistical Learning technique. It is one of the Supervised Machine Learning methods used in Classification tasks. We used K-fold Cross validation as one a metric for this classification.

Graphical user interface, chart

Description automatically generated

Graphical user interface, application

Description automatically generated

**Gradient Boosting:**

Gradient boosting is a type of machine learning technique used in classification and regression. It relies on the intuition that the best possible next model, when compared with previous models, minimizes the overall prediction error. This is a special type of ensemble learning technique that works by combining several weak learners into a strong learner. This works by each model paying attention to its predecessor’s mistakes.

The following shows the results of all the Algorithms run so far. We can see that Gradient boosting gives the best prediction with a high precision of 83.4% and accuracy of 76.8%.

Graphical user interface, chart

Description automatically generated

Graphical user interface, application

Description automatically generated

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| METRICS | RANDOM FOREST | LOGISTIC REGRESSION | GRADIENT BOOSTING | DECISION TREE |
| Accuracy | 71.6% | 66.6% | 76.8% | 68.6% |
| F1\_score | 69.8% | 65.2% | 74.3% | 66.0% |
| Precision score | 74.5% | 68.2% | 83.4% | 71.9% |
| Recall | 65.6% | 62.4% | 67.0% | 61.0% |

CONCLUSION

We have significantly improved the accuracy and precision of predicting a loan defaulter. We also found that Gradient Boosting gives the best prediction with a high accuracy of 78.43. Further, we can improve the accuracy of this data by applying PCA or other feature selection techniques. We can also use other ensemble methods to get a better result.

REFERENCES

[1] 2019. LT Vehicle Loan Default Prediction. (2019). <https://www.kaggle.com/mamtadhaker/lt-vehicle-loan-defaultpredictiondata_dictionary.csv>

[2] T. Cover &P. Hart Mickey Haggblade.Nearest neighbor pattern classification, IEEE Transactions on Information Theory 2013.

[3] C.J.C. Burges, “A Tutorial on Support Vector Machines for Pattern Recognition,” submitted to Data Mining and Knowledge Discovery, 1998.

[4] Lloyd, Stuart P. "Least squares quantization in PCM." Information Theory, IEEE Transactions on 28.2 (1982): 129-137.

[5] Nitesh V. Chawla, Kevin W. Bowyer. SMOTE: synthetic minority oversampling technique, Journal of Artificial Intelligence Research Archive Volume 16 Issue 1, January 2002

[6] Herve Abdi, Lynne J. Williams. Principal component analysis, WIREs Computational Statistics, 2010

[7] Aadhar, <https://uidai.gov.in/>

**APPENDIX**

1. Data Download

|  |
| --- |
|  |
|  | ''' |
|  | Please install opendatasets package first |
|  | Please |
|  | ''' |
|  |  |
|  | import opendatasets as od |
|  | od.download(r'https://www.kaggle.com/mamtadhaker/lt-vehicle-loan-default-prediction') |

1. PREPROCESSING

|  |
| --- |
| import pandas as pd |
|  | import numpy as np |
|  | import seaborn as sns |
|  | from scipy import stats |
|  | import matplotlib.pyplot as plt |
|  | import os |
|  | import re |
|  | from sklearn.model\_selection import train\_test\_split |
|  | import random |
|  |  |
|  | import scorecardpy as sc |
|  |  |
|  | # split train into train data and test data |
|  | # os.chdir(r'D:\GWU\Aihan\DATS 6103 Data Mining\Final Project\Code') |
|  |  |
|  |  |
|  | def split\_data(inpath, target\_name, test\_size): |
|  | df = pd.read\_csv(inpath) |
|  | y = df[target\_name] |
|  | #x = df1.loc[:,df1.columns!='loan\_default'] |
|  | x=df.drop(target\_name,axis=1) |
|  | # set a random seed for the data, so that we could get the same train and test set |
|  | random.seed(12345) |
|  | X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=test\_size, random\_state=1, stratify=y) |
|  |  |
|  | training = pd.concat([X\_train, y\_train], axis=1) |
|  | testing = pd.concat([X\_test, y\_test], axis=1) |
|  | return training, testing |
|  |  |
|  |  |
|  | class PreProcessing(): |
|  | def \_\_init\_\_(self, df): |
|  | self.Title = "Preprocessing Start" |
|  | self.df = df |
|  | # checking the null value and drop the null value |
|  | def Null\_value(self): |
|  | self.df.isnull().sum() |
|  | self.df\_new = self.df.dropna() |
|  | return self.df\_new |
|  |  |
|  | # convert the format of 'AVERAGE.ACCT.AGE' and 'CREDIT.HISTORY.LENGTH' from 'xyrs xmon' to numbers that represent month. |
|  | def find\_number(self, text): |
|  | num = re.findall(r'[0-9]+',text) |
|  | return int(num[0])\*12 + int(num[1]) |
|  |  |
|  | def comvert\_format(self, colname): |
|  | colname\_new = self.df[colname].apply(lambda x: self.find\_number(x)) |
|  | self.df[colname] = colname\_new |
|  |  |
|  |  |
|  | # convert categorical string to numbers |
|  | def convert\_cate\_to\_num(self, colname\_list): |
|  | for colname in colname\_list: |
|  | self.df[colname] = self.df[colname].astype('category') |
|  | cat\_columns = self.df.select\_dtypes(['category']).columns |
|  | self.df[cat\_columns] = self.df[cat\_columns].apply(lambda x: x.cat.codes) |
|  |  |
|  | def format\_date(self, colname\_list): |
|  | for colname in colname\_list: |
|  | self.df[colname] = pd.to\_datetime(self.df[colname], format = "%d-%m-%y",infer\_datetime\_format=True) |
|  |  |
|  | def format\_age\_disbursal(self): |
|  | self.df['Date.of.Birth'] = self.df['Date.of.Birth'].where(self.df['Date.of.Birth'] < pd.Timestamp('now'), |
|  | self.df['Date.of.Birth'] - np.timedelta64(100, 'Y')) |
|  | self.df['Age'] = (pd.Timestamp('now') - self.df['Date.of.Birth']).astype('<m8[Y]').astype(int) |
|  | self.df['Disbursal\_months'] = ((pd.Timestamp('now') - self.df['DisbursalDate']) / np.timedelta64(1, 'M')).astype(int) |
|  |  |
|  |  |
|  | def bin\_cutpoint(self, target\_name, colname\_list): |
|  | for colname in colname\_list: |
|  | bins\_disbursed\_amount = sc.woebin(self.df, y=target\_name, x=[colname]) |
|  | sc.woebin\_plot(bins\_disbursed\_amount) |
|  |  |
|  | pd.concat(bins\_disbursed\_amount) |
|  | list\_break = pd.concat(bins\_disbursed\_amount).breaks.astype('float').to\_list() |
|  | list\_break.insert(0, float('-inf')) |
|  | # list\_break |
|  |  |
|  | self.df[colname] = pd.cut(self.df[colname], list\_break) |
|  |  |
|  | def delet\_columns(self, delete\_list): |
|  | df\_new = self.df.drop(delete\_list, axis=1) |
|  | return df\_new |
|  |  |
|  | def save\_csv(self, outpath): |
|  | self.df.to\_csv(outpath) |
|  |  |
|  |  |
|  |  |
|  |  |
|  | ''' |
|  | # format the date variable |
|  | training['Date.of.Birth'] = pd.to\_datetime(training['Date.of.Birth']).dt.strftime('%d/%m/%Y') |
|  | training['DisbursalDate'] = pd.to\_datetime(training['DisbursalDate'], format = "%d-%m-%y",infer\_datetime\_format=True) |
|  | # covert Date of birth to age |
|  |  |
|  |  |
|  | def age(born): |
|  | born\_date = datetime.strptime(born, "%d/%m/%Y").date() |
|  | today = datetime.now() |
|  | return relativedelta(today, born\_date).years |
|  |  |
|  | training['Age'] = training['Date.of.Birth'].apply(age) |
|  | training['Disbursal\_months'] = ((pd.Timestamp('now') - training['DisbursalDate'])/np.timedelta64(1,'M')).astype(int) |
|  |  |
|  | ''' |
|  |  |
|  |  |
|  | if \_\_name\_\_ == "\_\_main\_\_": |
|  | inpath = r'lt-vehicle-loan-default-prediction/train.csv' |
|  | target\_name = 'loan\_default' |
|  | outpath\_train = r'lt-vehicle-loan-default-prediction/final\_train.csv' |
|  | outpath\_test = r'lt-vehicle-loan-default-prediction/final\_test.csv' |
|  | training, testing = split\_data(inpath, target\_name, test\_size=0.3) |
|  | # checking the format of each variable |
|  | print(training.dtypes) |
|  |  |
|  | print(PreProcessing(training).Title) |
|  | df\_new = PreProcessing(training).Null\_value() |
|  |  |
|  | # There are 5375 missing value |
|  |  |
|  | PreProcessing(df\_new).comvert\_format('AVERAGE.ACCT.AGE') |
|  | PreProcessing(df\_new).comvert\_format('CREDIT.HISTORY.LENGTH') |
|  | # comvert\_format(training, 'AVERAGE.ACCT.AGE') |
|  | # comvert\_format(training, 'CREDIT.HISTORY.LENGTH') |
|  |  |
|  | PreProcessing(df\_new).convert\_cate\_to\_num(['Employment.Type', 'PERFORM\_CNS.SCORE.DESCRIPTION']) |
|  |  |
|  | # Create Age and Disbursal\_months |
|  | PreProcessing(df\_new).format\_date(['Date.of.Birth', 'DisbursalDate']) |
|  | PreProcessing(df\_new).format\_age\_disbursal() |
|  | df\_all = PreProcessing(df\_new).delet\_columns(['UniqueID', 'Date.of.Birth', 'DisbursalDate', 'PERFORM\_CNS.SCORE.DESCRIPTION', 'Employee\_code\_ID', 'Current\_pincode\_ID']) |
|  |  |
|  | # Traditional Credit Scoring |
|  | # PreProcessing(df\_new).bin\_cutpoint(target\_name, ["disbursed\_amount", "asset\_cost", "ltv", "PERFORM\_CNS.SCORE", "PRI.NO.OF.ACCTS",\ |
|  | # "PRI.ACTIVE.ACCTS", "PRI.OVERDUE.ACCTS", "PRI.CURRENT.BALANCE", "PRI.SANCTIONED.AMOUNT",\ |
|  | # "PRI.DISBURSED.AMOUNT", "PRIMARY.INSTAL.AMT", "NEW.ACCTS.IN.LAST.SIX.MONTHS", \ |
|  | # "DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS", "AVERAGE.ACCT.AGE", "CREDIT.HISTORY.LENGTH",\ |
|  | # "Age", "Disbursal\_months"]) |
|  |  |
|  | PreProcessing(df\_all).save\_csv(outpath\_train) |
|  |  |
|  |  |
|  |  |
|  | ''' |
|  | # FINISH FOR NOW |
|  | ''' |

1. MODELLING
2. GUI