# Vehicle Loan Default Prediction

**********************

College of Science and Technology, Bellevue University

DSC630-T302: Data Mining (2237-1)

Sashidhar Bezawada

Aug 09, 2023

## Contents

## Introduction

Financial institutions incur significant losses due to the default of vehicle loans. This has led to the tightening up of vehicle loan underwriting and increased vehicle loan rejection rates. The need for a better credit risk scoring model is also raised by these institutions. This warrants a study to estimate the determinants of vehicle loan default.

Financial institutions have the need to accurately predict the probability of borrower defaulting on a vehicle loan in the first EMI (Equated Monthly Instalments) on the due date. Doing so will ensure that clients capable of repayment are not rejected and important determinants can be identified which can be further used for minimizing the default rates.

Financial institutions are using classification models to predict the default or raise red flags on the accounts which are likely to default. One major problem is of Imbalanced Distribution of classes i.e. the available training data consists more of non-default accounts vs. default accounts.

## Dataset and Sources

The Dataset with the following information regarding the loan and borrower will be used:

- Borrower Information : Demographic data like age, income, Identity proof etc.
- Loan Information : Disbursal details, amount, EMI, loan to value ratio etc.
- Bureau data : Bureau score, number of active accounts, status of other loans, credit history.

I am choosing the dataset from Kaggle as titled " Vehicle Loan Default Prediction" . It contains two datasets ( train and test ) . And I am planning to cover various EDA, Outlier treatment, PCA, Variable Inspection, Classification models, etc

test.csv : has 40 Columns & 112392 records

train.csv : has 41 Columns & 233154 records

Names of columns in train dataset : ['UNIQUEID', 'DISBURSED_AMOUNT', 'ASSET_COST', 'LTV', 'BRANCH_ID', 'SUPPLIER_ID', 'MANUFACTURER_ID', 'CURRENT_PINCODE_ID', 'DATE_OF_BIRTH', 'EMPLOYMENT_TYPE', 'DISBURSAL_DATE', 'STATE_ID', 'EMPLOYEE_CODE_ID', 'MOBILENO_AVL_FLAG', 'AADHAR_FLAG', 'PAN_FLAG', 'VOTERID_FLAG', 'DRIVING_FLAG', 'PASSPORT_FLAG', 'PERFORM_CNS_SCORE', 'PERFORM_CNS_SCORE_DESCRIPTION', 'PRI_NO_OF_ACCTS', 'PRI_ACTIVE_ACCTS', 'PRI_OVERDUE_ACCTS', 'PRI_CURRENT_BALANCE', 'PRI_SANCTIONED_AMOUNT', 'PRI_DISBURSED_AMOUNT', 'SEC_NO_OF_ACCTS', 'SEC_ACTIVE_ACCTS', 'SEC_OVERDUE_ACCTS', 'SEC_CURRENT_BALANCE', 'SEC_SANCTIONED_AMOUNT', 'SEC_DISBURSED_AMOUNT', 'PRIMARY_INSTAL_AMT', 'SEC_INSTAL_AMT', 'NEW_ACCTS_IN_LAST_SIX_MONTHS', 'DELINQUENT_ACCTS_IN_LAST_SIX_MONTHS', 'AVERAGE_ACCT_AGE', 'CREDIT_HISTORY_LENGTH', 'NO_OF_INQUIRIES', 'LOAN_DEFAULT']

## Metadata Dictionary

| Variable Name | Description |
|---|---|
| UniqueID | Identifier for customers |
| loan_default | Payment default in the first EMI on due date |
| disbursed_amount | Amount of Loan disbursed |
| asset_cost | Cost of the Asset |
| ltv | Loan to Value of the asset |
| branch_id | Branch where the loan was disbursed |
| supplier_id | Vehicle Dealer where the loan was disbursed |
| manufacturer_id | Vehicle manufacturer(Hero, Honda, TVS etc.) |
| Current_pincode | Current pincode of the customer |
| Date_of_Birth | Date of birth of the customer |
| Employment_Type | Employment Type of the customer (Salaried/Self Employed) |
| DisbursalDate | Date of disbursement |
| State_ID | State of disbursement |
| Employee_code_ID | Employee of the organization who logged the disbursement |
| MobileNo_Avl_Flag | if Mobile no. was shared by the customer then flagged as 1 |
| Aadhar_flag | if aadhar was shared by the customer then flagged as 1 |
| PAN_flag | if pan was shared by the customer then flagged as 1 |
| VoterID_flag | if voter  was shared by the customer then flagged as 1 |
| Driving_flag | if DL was shared by the customer then flagged as 1 |
| Passport_flag | if passport was shared by the customer then flagged as 1 |
| PERFORM_CNS_SCORE | Bureau Score |
| PERFORM_CNS_SCORE_DESCRIPTION | Bureau score description |
| PRI_NO_OF_ACCTS | count of total loans taken by the customer at the time of disbursement |
| PRI_ACTIVE_ACCTS | count of active loans taken by the customer at the time of disbursement |
| PRI_OVERDUE_ACCTS | count of default accounts at the time of disbursement |
| PRI_CURRENT_BALANCE | total Principal outstanding amount of the active loans at the time of disbursement |
| PRI_SANCTIONED_AMOUNT | total amount that was sanctioned for all the loans at the time of disbursement |
| PRI_DISBURSED_AMOUNT | total amount that was disbursed for all the loans at the time of disbursement |
| SEC_NO_OF_ACCTS | count of total loans taken by the customer at the time of disbursement |
| SEC_ACTIVE_ACCTS | count of active loans taken by the customer at the time of disbursement |
| SEC_OVERDUE_ACCTS | count of default accounts at the time of disbursement |
| SEC_CURRENT_BALANCE | total Principal outstanding amount of the active loans at the time of disbursement |
| SEC_SANCTIONED_AMOUNT | total amount that was sanctioned for all the loans at the time of disbursement |
| SEC_DISBURSED_AMOUNT | total amount that was disbursed for all the loans at the time of disbursement |
| PRIMARY_INSTAL_AMT | EMI Amount of the primary loan |
| SEC_INSTAL_AMT | EMI Amount of the secondary loan |
| NEW_ACCTS_IN_LAST_SIX_MONTHS | New loans taken by the customer in last 6 months before the disbursment |
| DELINQUENT_ACCTS_IN_LAST_SIX_MONTHS | Loans defaulted in the last 6 months |
| AVERAGE_ACCT_AGE | Average loan tenure |
| CREDIT_HISTORY_LENGTH | Time since first loan |
| NO_OF_INQUIRIES | Enquries done by the customer for loans |

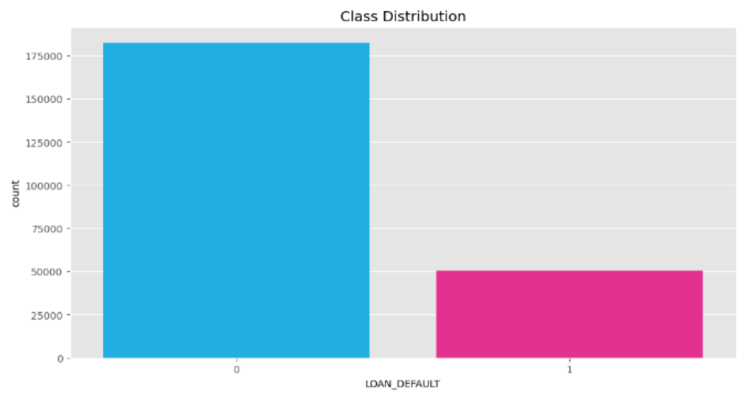
## Data Transformation

In this data Transformation phase,

- I have performed the NULL value inspection and have added the Missing values with default values (“Missing”).
- I have also performed Variable Inspection and changed the types of the below variables.
    - AVERAGE_ACCT_AGE, CREDIT_HISTORY_LENGTH are **object**, changed to **int**.
    - DATE_OF_BIRTH & DISBURSAL_DATE are **object**, changed to **datetime**

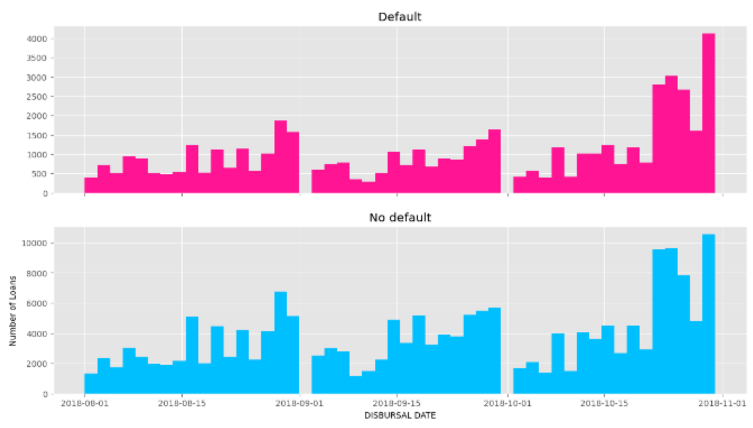
## Exploratory Data Analysis

I have performed Class distribution for the variable “ LOAN_DEFAULT” . This shows the train dataset has about 21.7% of loan defaults. Also, checked the distribution of loan default against other variables like Employment_Type, MobileNo_Avl_Flag, PAN_FLAG,DRIVING_FLAG ,Aadhar_flag, Passport_flag ,etc .



```
% of no defaults        : 78.29288796246257
Number of no defaults      : 182543
% of defaults          : 21.70711203753742
Number of defaults     : 50611
```

I have created hist graphs for Default vs Disbursal dates, which shows that loan defaults were on the higher side during the end of 2018-October.

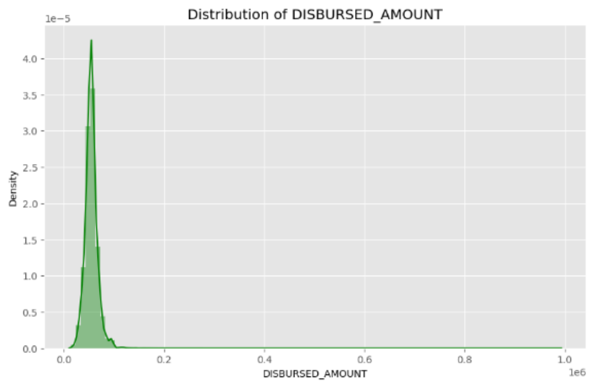


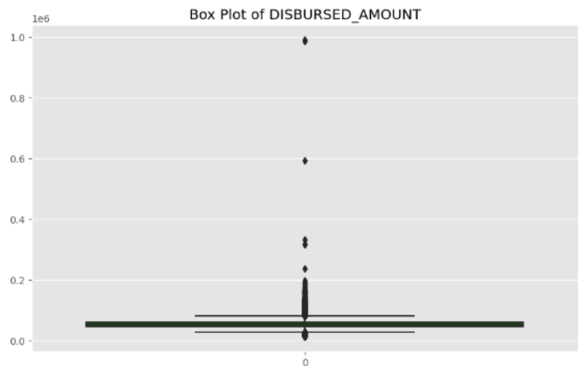
I have also performed Plot distribution & box plot  on high importance variables like :

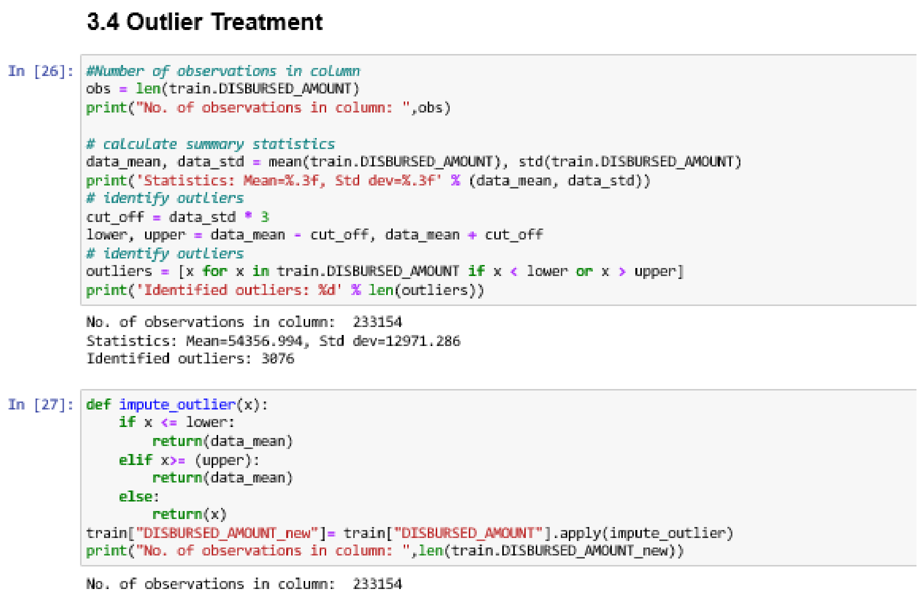
disbursed_amount , asset_cost, LTV , Perform_CNS_Score, PRI_NO_OF_ACCTS, PRI_OVERDUE_ACCTS

As, we have outliers, I have performed – impute the outliers as well as binning. (eg : disbursed_amount)

Below new variables are created : DISBURSED_AMOUNT_new,DISBURSED_AMOUNT_bins, ASSET_COST_new , ASSET_COST_bins , LTV_new , LTV_bins , PERFORM_CNS_SCORE_bins, PRI_NO_OF_ACCTS_bins, PRI_OVERDUE_ACCTS_bins







### 3.4 Outlier Treatment

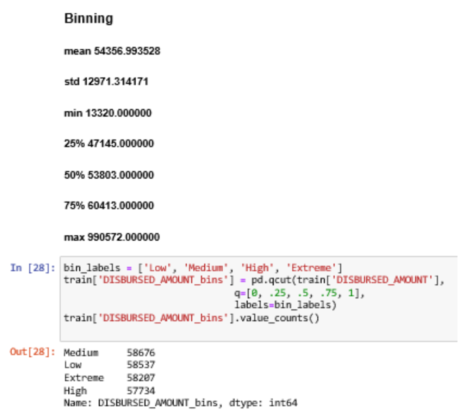
In [26]:
```
#Number of observations in column
obs = len(train.DISBURSED_AMOUNT)
print("No. of observations in column: ",obs)

# calculate summary statistics
data_mean, data_std = mean(train.DISBURSED_AMOUNT), std(train.DISBURSED_AMOUNT)
print('Statistics: Mean=%.3f, Std dev=%.3f' % (data_mean, data_std))
# identify outliers
cut_off = data_std * 3
lower, upper = data_mean - cut_off, data_mean + cut_off
# identify outliers
outliers = [x for x in train.DISBURSED_AMOUNT if x < lower or x > upper]
print('Identified outliers: %d' % len(outliers))
```

```
No. of observations in column:  233154
Statistics: Mean=54356.994, Std dev=12971.286
Identified outliers: 3876
```

In [27]:
```
def impute_outlier(x):
    if x <= lower:
        return(data_mean)
    elif x>= (upper):
        return(data_mean)
    else:
        return(x)
train["DISBURSED_AMOUNT_new"]= train["DISBURSED_AMOUNT"].apply(impute_outlier)
print("No. of observations in column: ",len(train.DISBURSED_AMOUNT_new))
```

```
No. of observations in column:  233154
```



Binning

mean 54356.993528

std 12971.314171

min 13320.000000

25% 47145.000000

50% 53803.000000

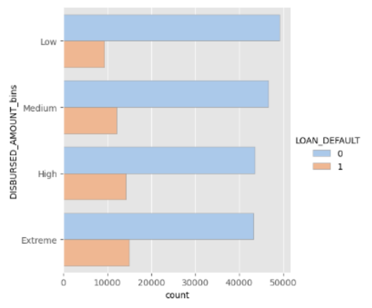
75% 60413.000000

max 990572.000000

```
In [28]: bin_labels = ['Low', 'Medium', 'High', 'Extreme']
         train['DISBURSED_AMOUNT_bins'] = pd.qcut(train['DISBURSED_AMOUNT'],
                                                  q=[0, .25, .5, .75, 1],
                                                  labels=bin_labels)
         train['DISBURSED_AMOUNT_bins'].value_counts()

Out[28]: Medium     58676
         Low        58537
         Extreme    58207
         High       57734
         Name: DISBURSED_AMOUNT_bins, dtype: int64
```

```
In [29]: plot_bar("DISBURSED_AMOUNT_bins")
         <Figure size 1000x5000 with 0 Axes>
```



Employement type vs LOAN_DEFAULT – Catplot indicates it has missing values

I have created Heat maps to identify the correlation & keep the variables for the model development.

Below are the findings –

**Not highly correlated with anyone:** 'PRI_ACTIVE_ACCTS', 'PRI_CURRENT_BALANCE', 'PRI_SANCTIONED_AMOUNT', 'PRI_DISBURSED_AMOUNT','SEC_OVERDUE_ACCTS'

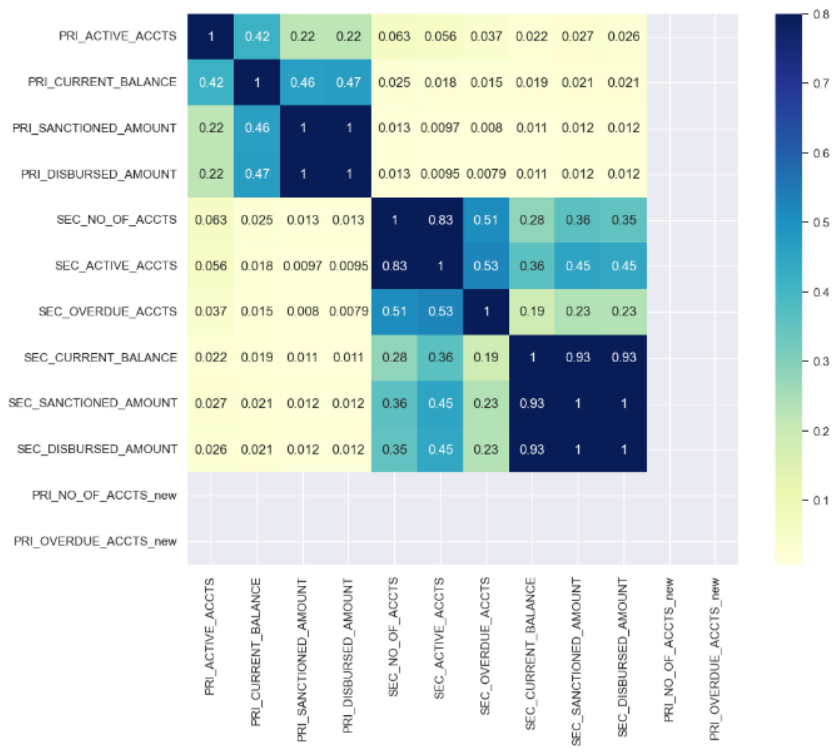
Keeping the variables due to the positive correlation –

'PRI_NO_OF_ACCTS_new', 'PRI_OVERDUE_ACCTS_new'are perfectly positively correlated and hence keeping one

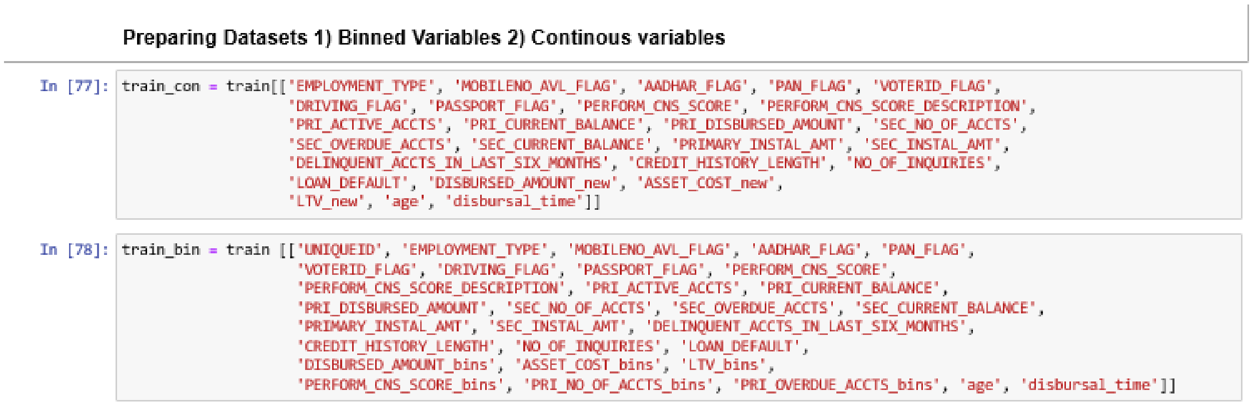
'SEC_NO_OF_ACCTS', 'SEC_ACTIVE_ACCTS' are highly positively correlated, hence keeping one

'SEC_CURRENT_BALANCE', 'SEC_SANCTIONED_AMOUNT', 'SEC_DISBURSED_AMOUNT' are highly positively correlated, hence keeping one

'AVERAGE_ACCT_AGE', 'CREDIT_HISTORY_LENGTH'are highly positively correlated and hence keeping one



Based on the above EDA, these are the final train & test datasets with Binned & continuous variables.



**Preparing Datasets 1) Binned Variables 2) Continous variables**

```
In [77]: train_con = train[['EMPLOYMENT_TYPE', 'MOBILENO_AVL_FLAG', 'AADHAR_FLAG', 'PAN_FLAG', 'VOTERID_FLAG',
                         'DRIVING_FLAG', 'PASSPORT_FLAG', 'PERFORM_CNS_SCORE', 'PERFORM_CNS_SCORE_DESCRIPTION',
                         'PRI_ACTIVE_ACCTS', 'PRI_CURRENT_BALANCE', 'PRI_DISBURSED_AMOUNT', 'SEC_NO_OF_ACCTS',
                         'SEC_OVERDUE_ACCTS', 'SEC_CURRENT_BALANCE', 'PRIMARY_INSTAL_AMT', 'SEC_INSTAL_AMT',
                         'DELINQUENT_ACCTS_IN_LAST_SIX_MONTHS', 'CREDIT_HISTORY_LENGTH', 'NO_OF_INQUIRIES',
                         'LOAN_DEFAULT', 'DISBURSED_AMOUNT_new', 'ASSET_COST_new',
                         'LTV_new', 'age', 'disbursal_time']]
```

```
In [78]: train_bin = train [['UNIQUEID', 'EMPLOYMENT_TYPE', 'MOBILENO_AVL_FLAG', 'AADHAR_FLAG', 'PAN_FLAG',
                          'VOTERID_FLAG', 'DRIVING_FLAG', 'PASSPORT_FLAG', 'PERFORM_CNS_SCORE',
                          'PERFORM_CNS_SCORE_DESCRIPTION', 'PRI_ACTIVE_ACCTS', 'PRI_CURRENT_BALANCE',
                          'PRI_DISBURSED_AMOUNT', 'SEC_NO_OF_ACCTS', 'SEC_OVERDUE_ACCTS', 'SEC_CURRENT_BALANCE',
                          'PRIMARY_INSTAL_AMT', 'SEC_INSTAL_AMT', 'DELINQUENT_ACCTS_IN_LAST_SIX_MONTHS',
                          'CREDIT_HISTORY_LENGTH', 'NO_OF_INQUIRIES', 'LOAN_DEFAULT',
                          'DISBURSED_AMOUNT_bins', 'ASSET_COST_bins', 'LTV_bins',
                          'PERFORM_CNS_SCORE_bins', 'PRI_NO_OF_ACCTS_bins', 'PRI_OVERDUE_ACCTS_bins', 'age', 'disbursal_time']]
```

The above training dataset has been passed through standardization – StandardScaler() & dummy variables are created using get_dummies().

## Modeling Methodology

On the model evaluation, I have built multiple models with different accuracy, F1 score, Recall score, AUC score, Balanced Accuracy scores. And compared the results to pick the final model with better results.

## Visualizations

Visualizations like Histogram, plot distribution, plot box, Plot bar will help identify the various predictive variables.

Plot distribution compare, categorical plots, Heat map can help visualize relationships among variables.

## Model Evaluation

Models I have used are Logistic regression, Random Forest, Naïve Bayes , Stochastic Gradient Descent, Decision tree classifier to compare the results and pick the better model that gives better Accuracy Score.

## Model versus expectations

I think that Random Forest is the best model for this data, as there appear to be potential nonlinear relationships in the data. And the other baseline models can help predict the datasets used help evaluate the predictive performance.

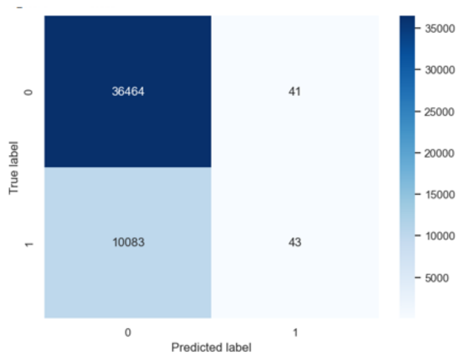
## Risks / Contingencies

Since the dataset is small and simulated dataset, it is highly possible that few models may perform poorly on the data hence as a contingency plan, I have aimed to apply multiple models and compare the results but still results may not be aligned with real world scenario, but I take this opportunity in validating my understanding with end-to-end flow of applying predictive analytics to a problem. In general, such datasets would be imbalanced so I am keeping closed tab on sampling (under/over). And will be using SMOTE, to generate new and synthetic data, which can be used for training our model.

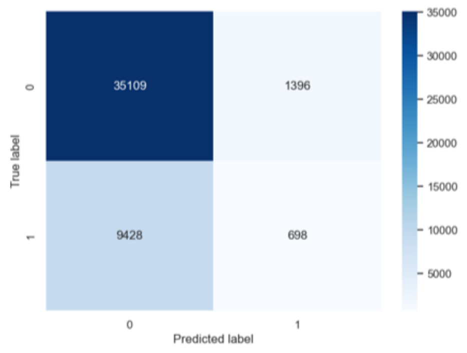
## Models – Confusion Matrices

Confusion matrix for various models are listed below

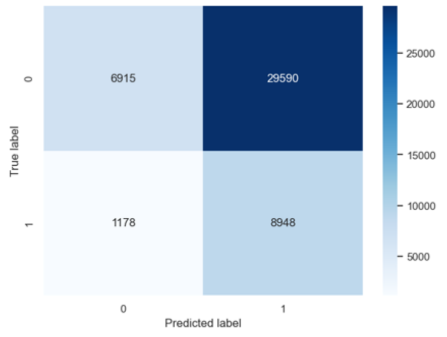
Logistic Regression



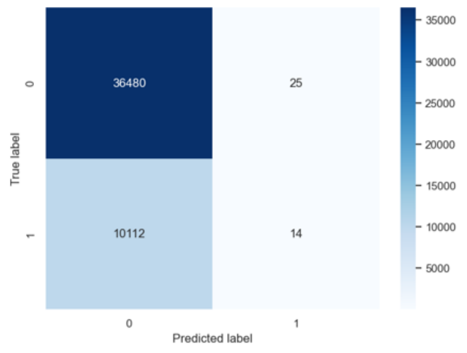
Random Forest



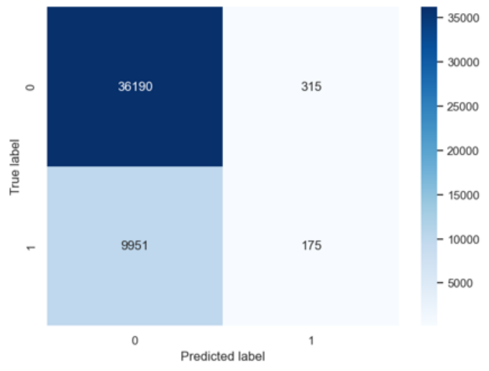
Naïve Bayes



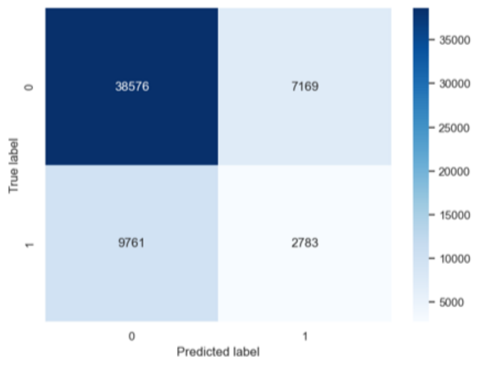
Stochastic Gradient Descent



Decision Tree

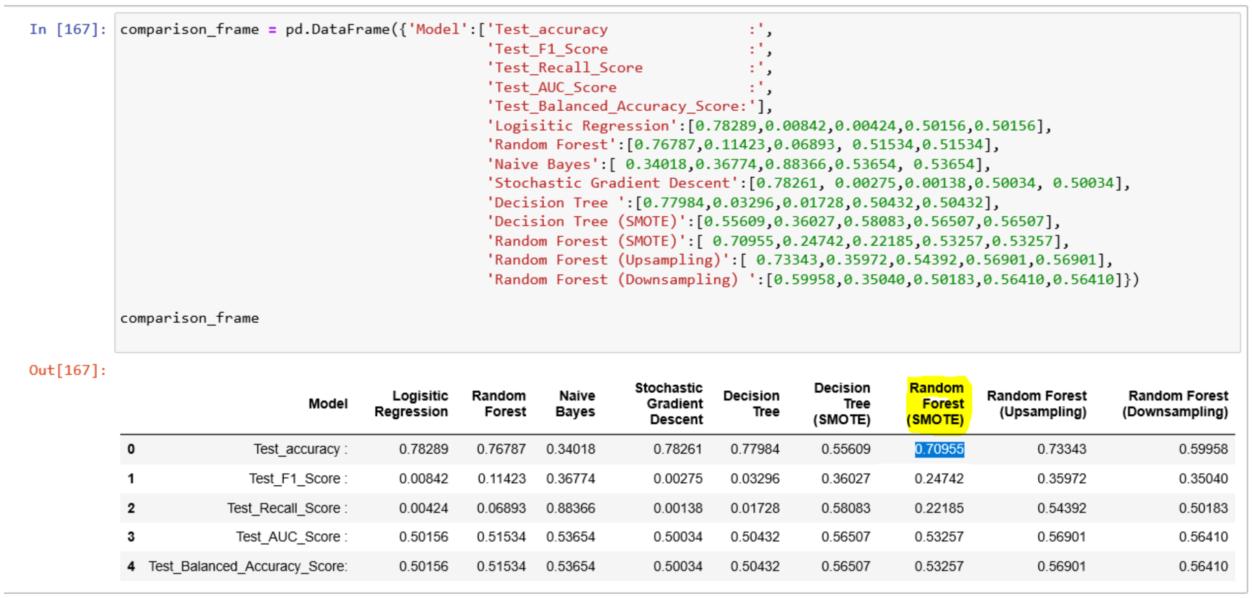


Random Forest ( SMOTE )



## Result Interpretation

### Comparing all the models based on Model Performance



```
In [167]: comparison_frame = pd.DataFrame({'Model':['Test_accuracy                  :',
                                                   'Test_F1_Score                  :',
                                                   'Test_Recall_Score              :',
                                                   'Test_AUC_Score                 :',
                                                   'Test_Balanced_Accuracy_Score:'],
                                                   'Logisitic Regression':[0.78289,0.00842,0.00424,0.50156,0.50156],
                                                   'Random Forest':[0.76787,0.11423,0.06893, 0.51534,0.51534],
                                                   'Naive Bayes':[ 0.34018,0.36774,0.88366,0.53654, 0.53654],
                                                   'Stochastic Gradient Descent':[0.78261, 0.00275,0.00138,0.50034, 0.50034],
                                                   'Decision Tree ':[0.77984,0.03296,0.01728,0.50432,0.50432],
                                                   'Decision Tree (SMOTE)':[0.55609,0.36027,0.58083,0.56507,0.56507],
                                                   'Random Forest (SMOTE)':[ 0.70955,0.24742,0.22185,0.53257,0.53257],
                                                   'Random Forest (Upsampling)':[ 0.73343,0.35972,0.54392,0.56901,0.56901],
                                                   'Random Forest (Downsampling) ':[0.59958,0.35040,0.50183,0.56410,0.56410]})

          comparison_frame
```

Out[167]:

| | Model | Logisitic Regression | Random Forest | Naive Bayes | Stochastic Gradient Descent | Decision Tree | Decision Tree (SMOTE) | Random Forest (SMOTE) | Random Forest (Upsampling) | Random Forest (Downsampling) |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Test_accuracy : | 0.78289 | 0.76787 | 0.34018 | 0.78261 | 0.77984 | 0.55609 | 0.70955 | 0.73343 | 0.59958 |
| **1** | Test_F1_Score : | 0.00842 | 0.11423 | 0.36774 | 0.00275 | 0.03296 | 0.36027 | 0.24742 | 0.35972 | 0.35040 |
| **2** | Test_Recall_Score : | 0.00424 | 0.06893 | 0.88366 | 0.00138 | 0.01728 | 0.58083 | 0.22185 | 0.54392 | 0.50183 |
| **3** | Test_AUC_Score : | 0.50156 | 0.51534 | 0.53654 | 0.50034 | 0.50432 | 0.56507 | 0.53257 | 0.56901 | 0.56410 |
| **4** | Test_Balanced_Accuracy_Score: | 0.50156 | 0.51534 | 0.53654 | 0.50034 | 0.50432 | 0.56507 | 0.53257 | 0.56901 | 0.56410 |

Six models were evaluated - Logistic Regression, Random Forest, Naive Bayes, Stochastic Gradient Descent & Decision Tree Classifier, Random Forest(with SMO. ( Some of them were done with SMOTE, UPsampling & Undersampling )

- Logisitic Regression - Accuracy score is good, however the model is not predicting the Defaults well
- Random Forest - Accuracy score is good, however the model is predicting the Defaults better than Logistic regression
- Naive Bayes - Model accuracy is very poor
- Stochastic Gradient Descent - Accuracy score is good, however the model is not predicting the Defaults well
- Decision Tree - Accuracy score is good, however the model is not predicting the Defaults well
- Random Forest (SMOTE) - The accuracy of RF might have gone down by 7% but is predicting defaults better. (SMOTE uses a nearest neighbors' algorithm to generate new and synthetic data we can use for training our model.)

*Note: Upsampling & Undersampling results , i haven't considered due to the following reasons*

*Upsampling can be defined as adding more copies of the minority class. Upsampling can be a good choice when you don't have a ton of data to work with. (Not a good choice here though)*

*Undersampling can be defined as removing some observations of the majority class. Undersampling can be a good choice when you have a ton of data -think millions of rows. But a drawback is that we are removing information that may be valuable. This could lead to underfitting and poor generalization to the test set.*

The Logistic Regression model achieved higher accuracy of 78.289% compared to other models. however the model is not predicting the Defaults well.

However, Random Forest models performed better than other models in predicting the Defaults, indicating they have learned something useful.

The Random Forest model ( SMOTE ) had higher F1-Score of 24.724%  than Random Forest model ( without SMOTE) with 11.423%.

The Random Forest model ( SMOTE ) had higher Recall Score of 22.185%  than Random Forest model ( without SMOTE) with 10.995%.. When the model predicted positive, the Decision Tree was slightly more likely to be correct.

The ROC AUC score was higher for Logistic Regression (53.257%) compared to the other models considered useful.

The confusion matrices and classification metrics like false positive rate illustrate the types of errors made by each model. The Random Forest model had fewer false positives than Stochastic Gradient Descent & Decision Tree models.

## Conclusion

Based on the metrics from different Model, my conclusion is that the best result is obtained by Random Forest after deploying [ SMOTE ], as it has better Recall score (22.18%) along with 70.95% accuracy.

Few key insights to  highlight are ASSET_COST, LTV, PERFORM_CNS_SCORE, PRI_NO_OF_ACCTS, PRI_OVERDUE_ACCTS.

Chances of a customer to default a loan is more with Self employed is more than Salaried.

Customer with just more than one account have less chance to default.

Customer with no credit bureau are more likely to default.

Customers with LTV high have more likely chance to default.

As for directions for future work, we can analyze in detail and consider larger dataset and additional variables that can contribute to the Loan Defaulting.

## Ethical Concerns

In this study, I am using the test & train dataset of a vehicle loan servicing bank. Ethical approval is not required.

## References

1. Avik Paul. " Vehicle Loan Default Prediction “

https://www.kaggle.com/datasets/avikpaul4u/vehicle-loan-default-prediction

2. AMAN MIGLANI. “Predicting Loan Default (Classification)”.

https://www.kaggle.com/code/datatattle/predicting-loan-default-classification#Best-result-is-obtained-by-Random-Forest-after-deploying--SMOTE.

## Appendix

Term project - Python Code Notebook Attachment : BezawadaSashidhar_ DSC630_Term_Project