High Level Design (HLD)
Credit Card Default Prediction

Contents:

Page:

Abstract:

The Credit Card Default Prediction project aims to develop a robust machine learning model that accurately predicts the likelihood of a credit card holder defaulting on their payments. By leveraging historical credit card transaction data, demographic information, and other relevant features, this project aims to assist credit card companies, financial institutions, and risk management professionals in identifying customers who are at a high risk of defaulting.

This abstract project document outlines the key objectives, methodologies, and expected outcomes of the Credit Card Default Prediction project. The document provides an overview of the problem statement, the dataset used, the proposed machine learning approach, and the evaluation metrics to assess model performance.

General Description:

Introduction:

The Credit Card Default Prediction Machine Learning project aims to develop a predictive model to assess the likelihood of credit card default by utilizing historical transactional and customer data. This score document provides a concise summary of the project's key objectives, methodology, results, and recommendations.

Problem Statement:

The problem of credit card default is a significant concern for both financial institutions and borrowers. Default occurs when a credit cardholder fails to make the required minimum payments on their credit card balance within a specified timeframe. This situation poses significant financial risks to credit card companies and can lead to substantial losses. Therefore, accurately predicting the likelihood of credit card default is crucial for mitigating these risks and making informed decisions regarding credit card approvals, credit limits, and interest rates.

Need for the Model:

Risk Management: Credit card companies and financial institutions need an effective tool to assess the creditworthiness of customers and identify those at a higher risk of default. By accurately predicting credit card default, the model can assist in proactive risk management by allocating appropriate credit limits, adjusting interest rates, or even declining credit to high-risk individuals.

Profitability: Defaulting customers lead to financial losses for credit card companies. By accurately predicting default, institutions can prevent revenue loss by taking preventive actions, such as offering timely payment reminders, negotiating alternative payment plans, or initiating debt recovery processes.

Customer Protection: Accurate default prediction helps protect borrowers from falling into a cycle of debt and financial distress. By identifying high-risk customers in advance, credit card companies can provide appropriate financial counselling and support to help customers manage their debts effectively.

Regulatory Compliance: Financial institutions must comply with regulations and guidelines set by regulatory bodies. Developing a credit card default prediction model can help institutions demonstrate compliance by implementing risk management practices and making sound lending decisions.

Improved Decision-Making: Accurate default prediction enables credit card companies to make informed decisions regarding credit limits, interest rates, and loan approvals. This leads to more efficient allocation of resources, reduced credit risk exposure, and improved profitability.

In summary, the development of a credit card default prediction model addresses the need for effective risk management, enhanced profitability, customer protection, regulatory compliance, and improved decision-making in the credit card industry. By accurately predicting credit card default, financial institutions can take proactive measures to minimize financial risks and ensure the overall stability of their operations.

Proposed Solution:

The project follows a supervised learning approach, utilizing a dataset containing a comprehensive range of features, including credit limit, payment history, amount owed, age, etc. Various machine learning algorithms, such as logistic regression, decision trees, random forests, and gradient boosting, were explored and compared. Feature engineering techniques, including Weight of Evidence, were employed to enhance model performance. AUC ROC GINI are the metrics used for model validation

Technical Requirements and Tools:

Data Requirements:

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

This dataset has 22 independent variable

- ID column is not any impact on the prediction it is a unique key
- This Dataset have no missing values
- This Dataset has some categoric variable which marked as int type need to convert into string like SEX, MERRAGE
- Numeric continuous variables have suspicious outcome with respect to information value. All these variables need to delt with

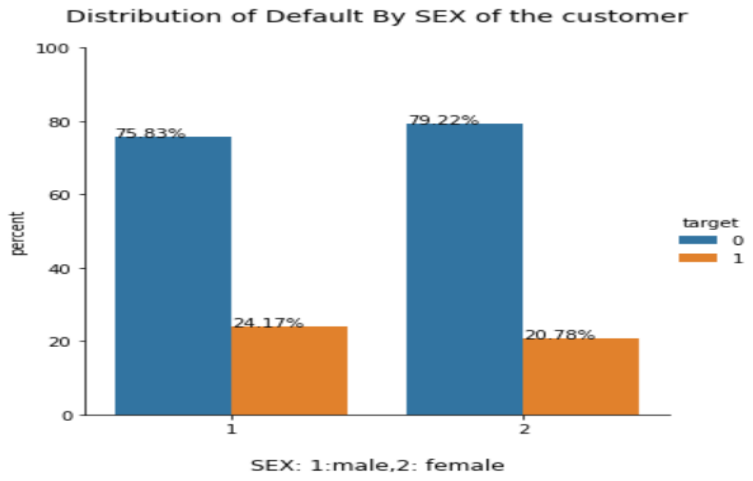
	target	total	%
0	0	23364	77.88
1	1	6636	22.12

#77% of customer falls under non default category and 22% customer falls under default category

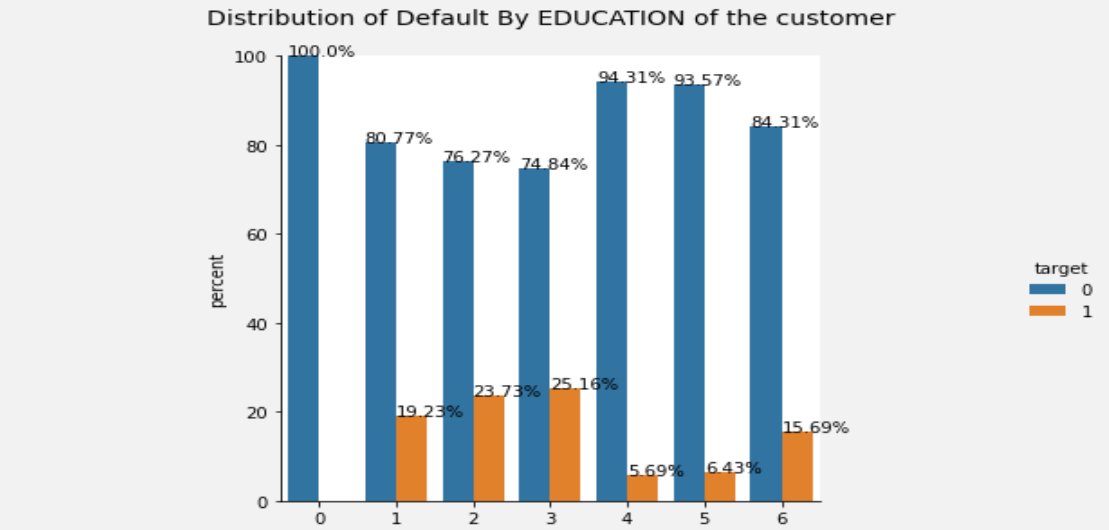
Number of sample: 30000

Explorative Data Analysis:

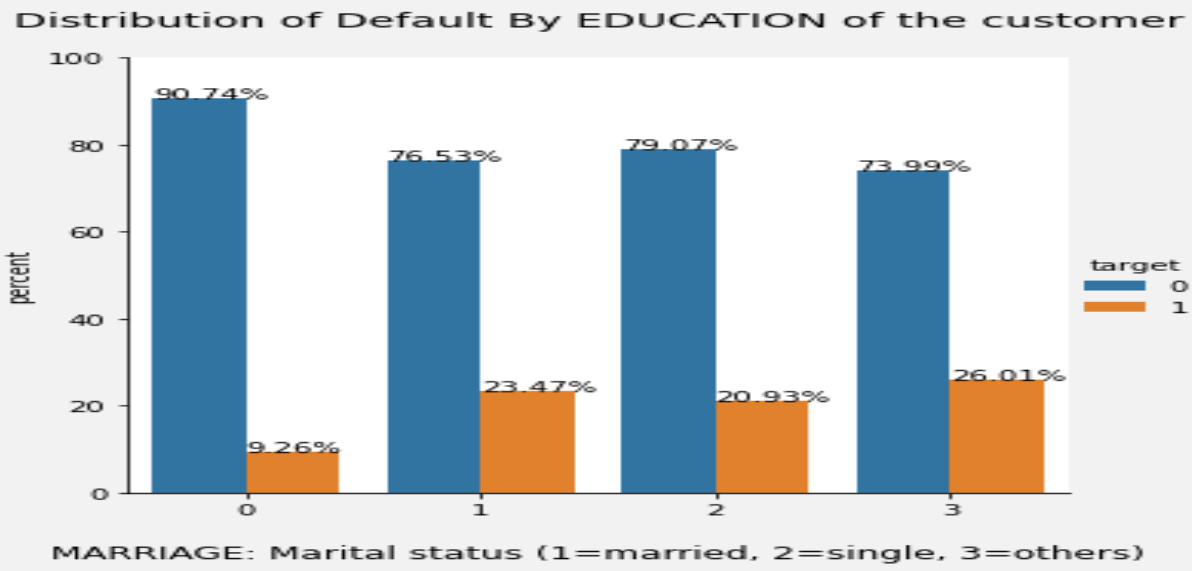
```
Out[8]: 2    0.603733
        1    0.396267
        Name: SEX, dtype: float64
```



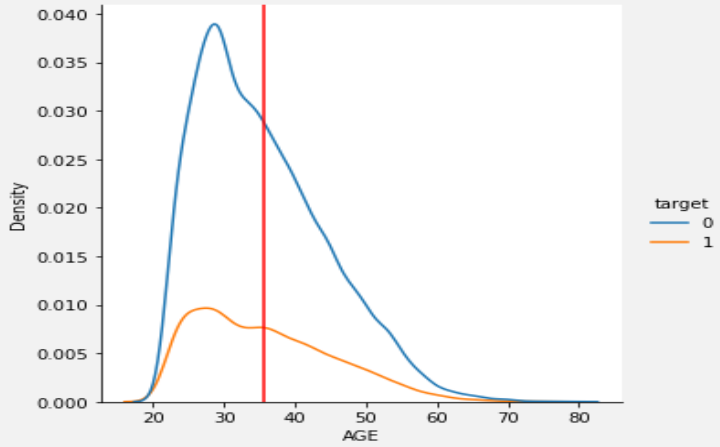
Conclution:
Overall, 60% customer are Female and 40% are male category and out of 60% female 24% defaulted and out of 40% male 20% defaulted. Similar distribution found under this category



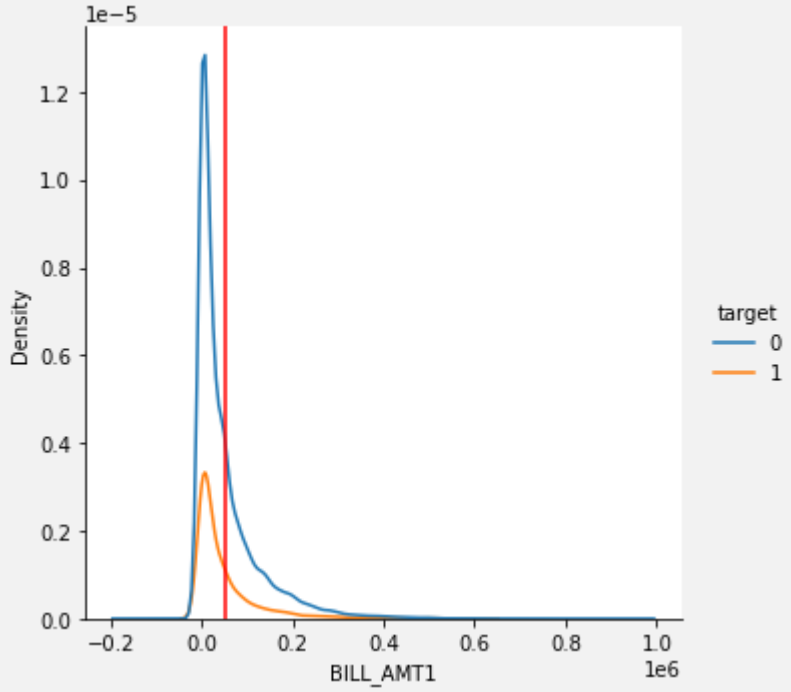
Majority of customer are graduated or above. There is other educational qualification but distribution is very minimal. All category more than 3 will be merged as these has very minimal contribution on the dataset.



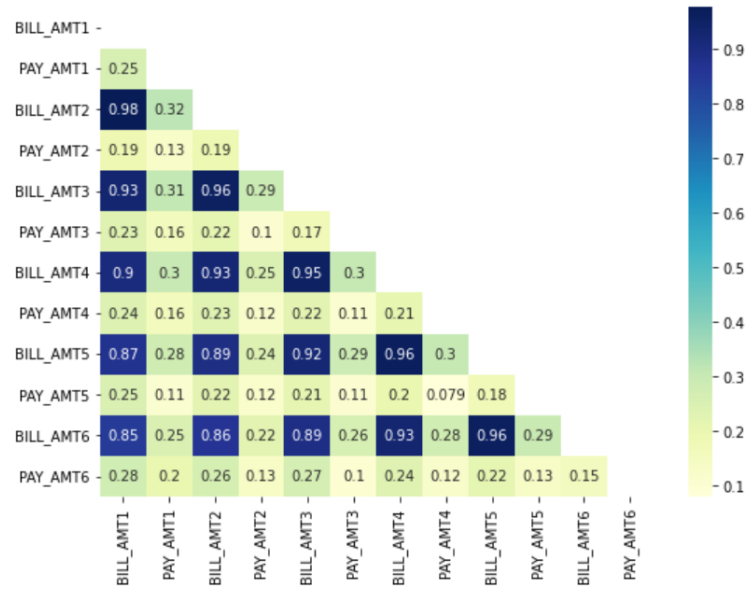
52% unmarried customer and 45% married customer are using the credit card. Bad distribution is equal across this category. As 20% of bad customer lies under these two categories so data cannot explain which segment most vulnerable for default



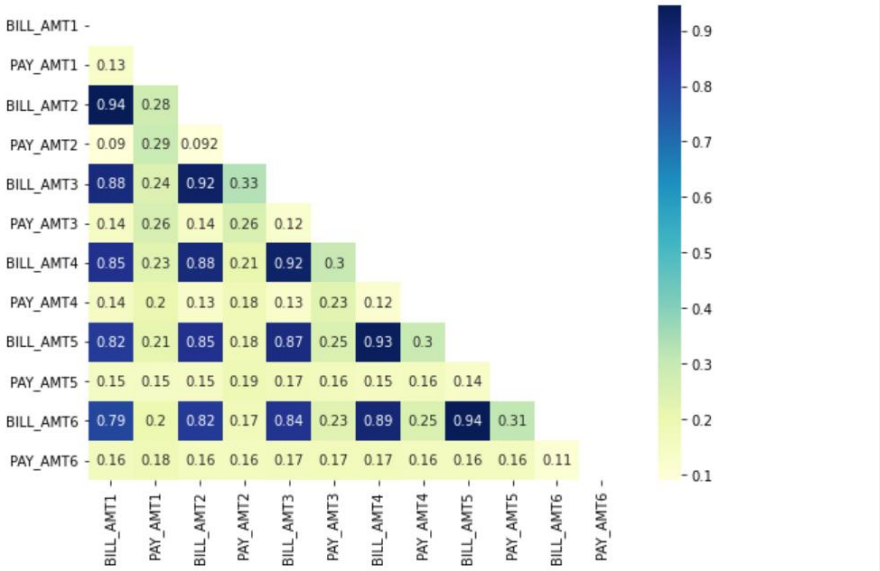
Distribution plotted with respect to target class. Both Good and Bad class has right skewed and med point found at 34. This describes maximum credit card issued for lower aged people. Bad distribution for age slightly flatterer with respect to good class

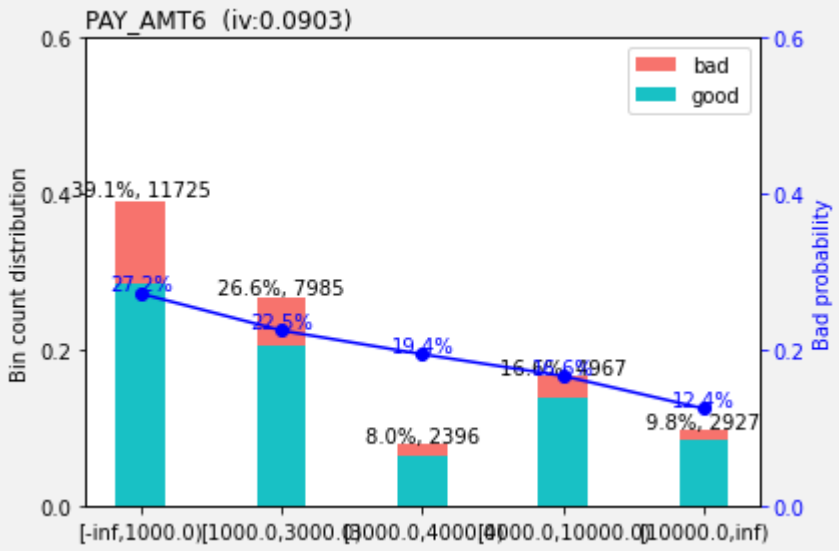


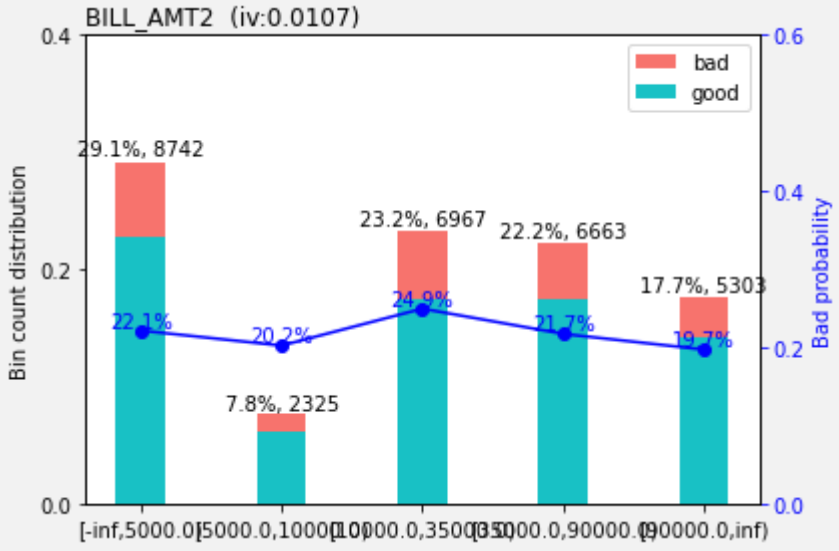
coreletion plot between Bill Amount and Pay amount for all 6 months for bad class

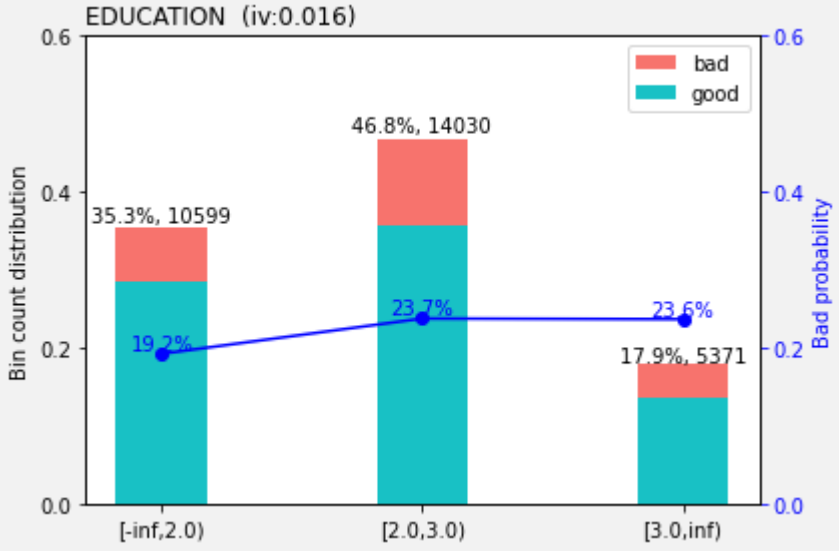


coreletion plot between Bill Amount and Pay amount for all 6 months for good class




PAY_AMT6 (iv:0.0903)
bad
good
Bin count distribution
Bad probability
39.1%, 11725
27.2%
26.6%, 7985
22.5%
19.4%
8.0%, 2396
16.6%, 4967
12.4%
9.8%, 2927
[-inf,1000.0)
[1000.0,3000.0)
[3000.0,4000.0)
[4000.0,10000.0)
[10000.0,inf)


BILL_AMT2 (iv:0.0107)
bad
good
Bin count distribution
Bad probability
29.1%, 8742
22.1%
20.2%
7.8%, 2325
23.2%, 6967
24.9%
22.2%, 6663
21.7%
17.7%, 5303
19.7%
[-inf,5000.0)
[5000.0,10000.0)
[10000.0,35000.0)
[35000.0,90000.0)
[90000.0,inf)


EDUCATION (iv:0.016)
bad
good
Bin count distribution
Bad probability
35.3%, 10599
19.2%
46.8%, 14030
23.7%
23.6%
17.9%, 5371
[-inf,2.0)
[2.0,3.0)
[3.0,inf)

Data Engineering and Feature Selection:

In the credit card default prediction project, data engineering techniques were employed to handle missing values in the dataset using the Weight of Evidence (WoE) imputation method. Additionally, feature selection was performed using the Information Value (IV) and Pearson coefficient.

Data Engineering with Weight of Evidence (WoE) Imputation:
The dataset contained missing values, which needed to be addressed to ensure the quality and integrity of the data. The Weight of Evidence (WoE) imputation technique was applied to handle missing values effectively. WoE measures the predictive power of an independent variable in relation to the dependent variable. By calculating the WoE for each category within a feature, missing values were assigned WoE values based on the distribution of the other non-missing values within the same feature. This approach helped maintain the predictive power of the variable while addressing missing data.

Feature Selection with Information Value (IV) and Pearson Coefficient:
To identify the most informative and relevant features for the credit card default prediction model, feature selection techniques were employed. Information Value (IV) and Pearson coefficient were used as evaluation metrics.

The Information Value (IV) measures the predictive power of a feature by quantifying the relationship between the feature and the target variable. Higher IV values indicate stronger predictive power. Features with low IV values were considered less informative and were potentially removed from the dataset to reduce noise and improve model performance.

Additionally, the Pearson coefficient was calculated to assess the linear correlation between each feature and the target variable. Features with low Pearson coefficients (close to zero) were considered less correlated and had less predictive power. These features were also candidates for removal.

By applying the combination of Information Value (IV) and Pearson coefficient, the feature selection process identified the most relevant features that significantly contributed to predicting credit card default. This approach ensured that the model focused on the most influential factors while reducing the dimensionality of the dataset.

Overall, the use of Weight of Evidence (WoE) imputation for handling missing values and Information Value (IV) and Pearson coefficient for feature selection helped to enhance the quality of the dataset, improve model performance, and increase the interpretability of the credit card default prediction model.

Sample Split and Model Building:

The population data was spited into three categories one is original without converting into woe, second is woe implemented with initial bin and third woe implanted with adjusted bins. All three samples are compared with respect to various ML algorithm

Sample 1(woe implanted with initial woe bins)

```
## S1 ##
compare_classification_model(x_train_woe,y_train_woe,x_test_woe,y_test_woe,classifiers,models)
```

	index	Train_Accuracy	Test_Accuracy	Train_Precision	Test_Precision	Train_Recall	Test_Recall	Train_Fscore	Test_Fscore	Train_AUC	Test_AUC
0	Logistic Regression	0.819667	0.817222	0.672491	0.675222	0.356989	0.3420	0.466394	0.454033	0.653867	0.647500
1	KNN	0.857429	0.777111	0.729074	0.498031	0.563632	0.3795	0.635766	0.430760	0.752147	0.635107
2	Decision Tree	0.967714	0.729556	0.986719	0.388374	0.865401	0.3775	0.922087	0.382860	0.931051	0.603821
3	Random forest	0.967714	0.801667	0.973898	0.588770	0.877265	0.3565	0.923059	0.444098	0.935302	0.642679
4	Ada Boost	0.819952	0.817111	0.688909	0.685535	0.336281	0.3270	0.451950	0.442789	0.646630	0.642071

Sample2 (woe implanted with adjusted bins)

```
### S2 ###
compare_classification_model(x_train_woeadj,y_train_woeadj,x_test_woeadj,y_test_woeadj,classifiers,models)
```

	index	Train_Accuracy	Test_Accuracy	Train_Precision	Test_Precision	Train_Recall	Test_Recall	Train_Fscore	Test_Fscore	Train_AUC	Test_AUC
0	Logistic Regression	0.819476	0.817444	0.672379	0.676558	0.355479	0.3420	0.465077	0.454334	0.653204	0.647643
1	KNN	0.862810	0.780444	0.748795	0.508141	0.569672	0.3745	0.647066	0.431203	0.757764	0.635464
2	Decision Tree	0.980571	0.727333	0.984862	0.388725	0.926230	0.3965	0.954647	0.392574	0.961098	0.609179
3	Random forest	0.980524	0.798778	0.975478	0.577143	0.935289	0.3535	0.954961	0.438450	0.964314	0.639750
4	Ada Boost	0.819952	0.817000	0.688077	0.683281	0.337360	0.3290	0.452743	0.444144	0.647016	0.642714

Sample3 (data with original value)

```
## S3 #
compare_classification_model(x_train,y_train,x_test,y_test,classifiers,models)
```

	index	Train_Accuracy	Test_Accuracy	Train_Precision	Test_Precision	Train_Recall	Test_Recall	Train_Fscore	Test_Fscore	Train_AUC	Test_AUC
0	Logistic Regression	0.779190	0.777778	0.333333	0.000000	0.000216	0.0000	0.000431	0.000000	0.500047	0.500000
1	KNN	0.847048	0.737333	0.734057	0.354864	0.481665	0.2225	0.581662	0.273510	0.716114	0.553464
2	Decision Tree	0.998619	0.718778	0.999566	0.377026	0.994176	0.4070	0.996864	0.391440	0.997027	0.607429
3	Random forest	0.998619	0.814444	0.996765	0.645760	0.996980	0.3655	0.996873	0.466794	0.998032	0.654107
4	Ada Boost	0.819762	0.813667	0.694737	0.677668	0.327437	0.3080	0.445096	0.423513	0.643339	0.633071

or Sample1 or S1:

- Logistic Regression is giving accuracy of 0.819667 in train and 0.817222 in test, so it is clear that model is not over/under fit. Precision score is 0.672491 and 0.675222 for train and test respectively. It means ~67% class are actually positive out of all correctly predicted class(TP,TN). Recall score are 0.356989 and 0.3420 respectively. It means out of all ~35% positive class are truly predicted out of all positive class

- Decision Tree and Random Forest is giving better accuracy than logistic regression but test accuracy is not matching with test accuracy. So there has a overfitting issue, Precision and recall score also speak the same.

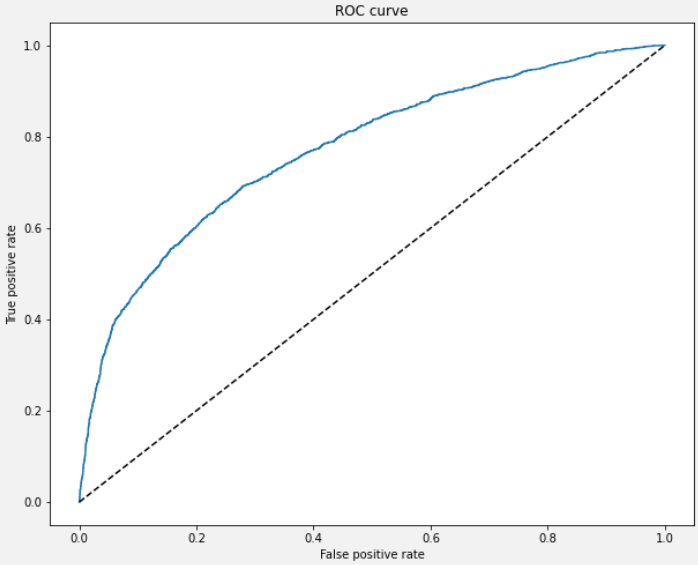
- KNN is giving me better accuracy then logistic regression 0.857429, but test set is 0.777111. though it also has overfitting issue but minimal than Decision Tree and Random Forest. Precision score is 0.777111 but this not incorporate with the test data, score achieved 0.498031. Recall score for test 0.3795 achieved with respective to 0.563632 achieved in train.

- Ada Boost accuracy is 0.819143 achieved in train set and 0.814333 in test. Clearly model is not over fit. Precision score is 0.684095 and 0.674814 for train and test respectively. Recall score is 0.335850, 0.3175 for train and test.

Noe out of the all algorithm Logistic Regression and Ada Boost has more robust result then then others .

For Sample2 or S2:
lt achieved from S2. So this is clear binning adjustment is not adding up any special value

Model Evolution and Model tuning




```
# Calculate the Area Under the Receiver Operating Characteristic Curve (AUROC) on our test set
auroc = roc_auc_score(y_test_proba['y_test_class_actual'], y_test_proba['y_hat_test_proba'])
auroc
```

```
0.7712918571428573
```

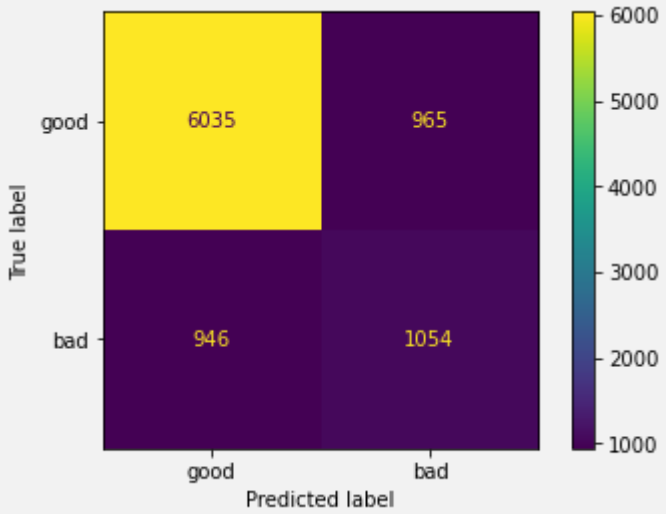
```
# calculate Gini from AUROC
gini = auroc * 2 - 1
gini
```

```
0.5425837142857146
```

```
Optimization terminated successfully.
         Current function value: 0.556409
         Iterations 6
                         Results: Logit
=================================================================
Model:              Logit            Pseudo R-squared: -0.054
Dependent Variable: target           AIC:              23399.1647
Date:               2023-05-24 02:43 BIC:              23518.4489
No. Observations:   21000            Log-Likelihood:   -11685.
Df Model:           14               LL-Null:          -11085.
Df Residuals:       20985            LLR p-value:      1.0000
Converged:          1.0000           Scale:            1.0000
No. Iterations:     6.0000
-----------------------------------------------------------------
                   Coef.  Std.Err.    z    P>|z|   [0.025  0.975]
-----------------------------------------------------------------
BILL_AMT3_woe     -0.4137   0.2781 -1.4874 0.1369 -0.9589  0.1314
AGE_woe            0.4506   0.1125  4.0067 0.0001  0.2302  0.6710
PAY_2_woe          0.0933   0.0392  2.3801 0.0173  0.0165  0.1702
PAY_AMT4_woe       0.1298   0.0676  1.9191 0.0550 -0.0028  0.2623
PAY_3_woe          0.2548   0.0424  6.0129 0.0000  0.1717  0.3378
PAY_0_woe          0.9460   0.0286 33.0886 0.0000  0.8900  1.0020
BILL_AMT1_woe     -1.1853   0.2717 -4.3631 0.0000 -1.7177 -0.6528
PAY_AMT3_woe       0.1545   0.0609  2.5360 0.0112  0.0351  0.2738
PAY_6_woe          0.3311   0.0473  7.0009 0.0000  0.2384  0.4239
...
PAY_AMT6_woe       0.1837   0.0648  2.8338 0.0046  0.0566  0.3107
PAY_AMT2_woe       0.1808   0.0528  3.4263 0.0006  0.0774  0.2842
=================================================================
```

After dropping the variables, the achieved result has been improved. Accuracy , precision, recall auc and gini index has been checked with respect to different threshold. I have applied probability 0.10 to 0.50 and 0.26 came out the optimal for the modelling with accuracy 0.787889, precision 0.522403, recall 0.5305, ruc 0.695964 and gini 0.391929. This is clearly seen that recall value has improved from 0.35 to 0.53 and AUC from 0.64 to 0.69

	threshold	accuracy	precision	recall	auroc	gini
0	0.10	0.415667	0.267248	0.9355	0.601321	0.202643
1	0.15	0.654222	0.365245	0.7535	0.689679	0.379357
2	0.20	0.755556	0.461744	0.6035	0.701250	0.402500
3	0.25	0.783667	0.512625	0.5380	0.695929	0.391857
4	0.26	0.787667	0.522041	0.5270	0.694571	0.389143
5	0.27	0.789444	0.526827	0.5155	0.691607	0.383214
6	0.28	0.791889	0.533229	0.5095	0.691036	0.382071
7	0.29	0.794444	0.540279	0.5030	0.690357	0.380714
8	0.30	0.796111	0.545656	0.4930	0.687857	0.375714
9	0.31	0.798778	0.554031	0.4845	0.686536	0.373071
10	0.32	0.800111	0.559292	0.4740	0.683643	0.367286
11	0.33	0.801778	0.565534	0.4660	0.681857	0.363714
12	0.34	0.807222	0.585209	0.4550	0.681429	0.362857
13	0.35	0.809000	0.594486	0.4420	0.677929	0.355857
14	0.40	0.818000	0.643879	0.4050	0.670500	0.341000
15	0.45	0.818333	0.664563	0.3685	0.657679	0.315357
16	0.50	0.816667	0.675351	0.3370	0.645357	0.290714



Final cutoff has been chosen on 0.26 threshold. So if model throw probability more than 0.26 then model will decide the customer is a risky customer other wise it declare non risky customer

Model Deployment:

This model has deployed into AWS using Flask with CICD pipeline process.

1. Create Docker Image
2. Create an instance for deploying
3. Create an IAM user
4. Create ECR repository
5. Set GitHub actions
6. Set AWS credentials in GitHub