A Study On Statistical Techniques For Credit Scoring

A project report submitted
for partial fulfillment of the requirements for the award of the degree of Master of Science

by

Akash Singh
Registration number: 213001818010008
M.Sc. in Applied Statistics and Analytics

Under the supervision of

Dr. Sushovon Jana

MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY, WEST BENGAL

Department of Applied Statistics

MAKAUT, WB

Department of Applied Statistics
MAKAUT, WB

CERTIFICATE

This is to certify that the dissertation report entitled 'A Study On Statistical Techniques For Credit Scoring', submitted by Akash Singh (Reg. No: 213001818010008, Roll No: 3001802108) to MAKAUT, WB, is a record of project work carried out by him/her under my supervision and guidance, and is worthy of consideration for the award of the degree of Master of Science in Applied Statistics and Analytics of the University.

Dr. Sushovon Jana	Ms. Anwesha Sengupta
Supervisor	HoD
Dept. Applied Statistics	Dept. Applied Statistics

MAULANA ABUL KALAM AZAD
UNIVERSITY OF TECHNOLOGY,
WEST BENGAL

Department of Applied Statistics
MAKAUT, WB

Declaration

I declare that, this project report has been composed by me and no part of this project report has formed the basis for the award of any Degree/Diploma or any other similar title to me.

Date:

Student's name:

Reg. No:

Dept. Applied Statistics

MAKAUT, WB

Content

Introduction:

This project aims to build a credit scorecard using machine learning techniques in Python. The credit scorecard is a common industry problem used to assess the credibility of a customer for various financial services, including credit cards, loans, and other financial products. A credit scorecard evaluates the creditworthiness of a customer based on their financial history, income, credit score, and other factors.

The objective of this project is to develop an accurate and explainable credit scorecard that can justify false alarms to non-technical stakeholders such as managers, customers, and business partners. The scorecard should be easy to understand, and the stakeholders should be able to interpret the results with minimal assistance from data scientists.

To achieve this goal, this project utilizes several machine learning techniques, including Logistic Regression, Gradient Boosting, Weight of Evidence (WOE), Information Value (IV), Binning, and Chi-square Binning. These techniques help in feature selection, fine binning, and identifying the most relevant variables for credit scoring.

In addition to the machine learning techniques, this project employs the Toad library, which is a production-to-go library for building scorecards. Toad offers critical features such as exploratory data analysis (EDA), feature engineering, and scorecard generation, which help in streamlining the most critical and time-consuming processes, including feature selection and fine binning.

Overall, this project aims to provide a reliable and explainable credit scorecard that can be used by businesses to assess the creditworthiness of their customers accurately. The scorecard's interpretability will allow stakeholders to make informed decisions and improve business operations.

Dataset Collection:

The Default of Credit Card Clients dataset is a widely-used dataset for building credit scorecards. This dataset contains 30,000 observations of credit card clients in Taiwan from April 2005 to September 2005. The dataset consists of 23 features, including demographic information, credit history, bill statements, and payment history.

The target variable in this dataset is whether or not the client will default on their next payment. This variable is represented by a binary outcome: 1 for default and 0 for non-default. The dataset also contains information about the clients' payment history, including the amount of the bill statement, the amount of the previous payment, and the amount of the current payment.

The dataset was originally sourced from the UCI Machine Learning Repository and was uploaded to Kaggle for use in data science projects. The dataset has been used extensively in academic research and industry applications to develop credit scoring models.

Using this dataset, we can apply various machine learning techniques to build a credit scorecard that predicts the likelihood of a customer defaulting on their next payment. By analyzing the demographic information, credit history, and payment behavior of credit card clients, we can identify patterns and build a model that is accurate and explainable. There are 25 variables: There are 25 variables:

- ID: ID of each client
- 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)
- default.payment.next.month: Default payment (1=yes, 0=no)

Objective:

Building a scorecard system for existing customers and selecting reliable customers for balance transfer offers will help the credit card company minimize losses and promote credit card usage.

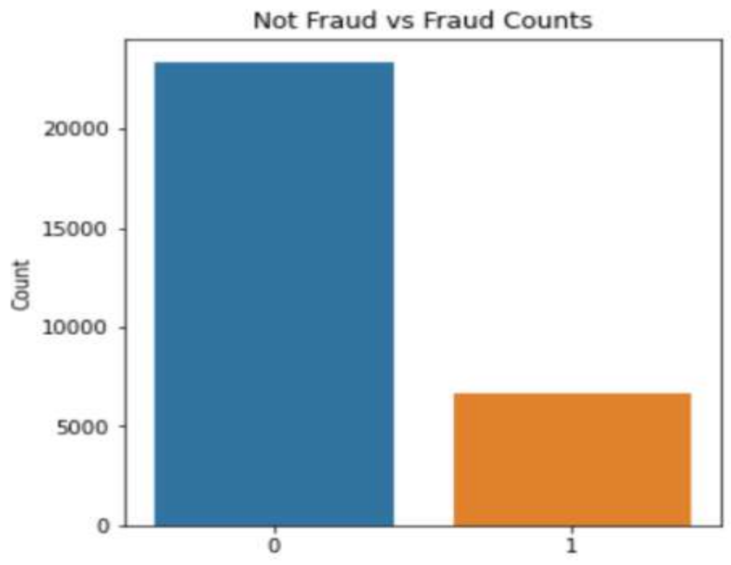
Once I have a reliable credit scorecard, I can segment the existing customers into different risk categories based on their creditworthiness. This segmentation will enable me to target customers with a low risk of defaulting on their payments with balance transfer offers.

Customers with a high credit score are less risky and more likely to pay their balance transfer in full, while customers with a low credit score are considered high-risk and more likely to default. Therefore, customers with a low credit score should be excluded from the balance transfer offer.

Data Pre-Processing:

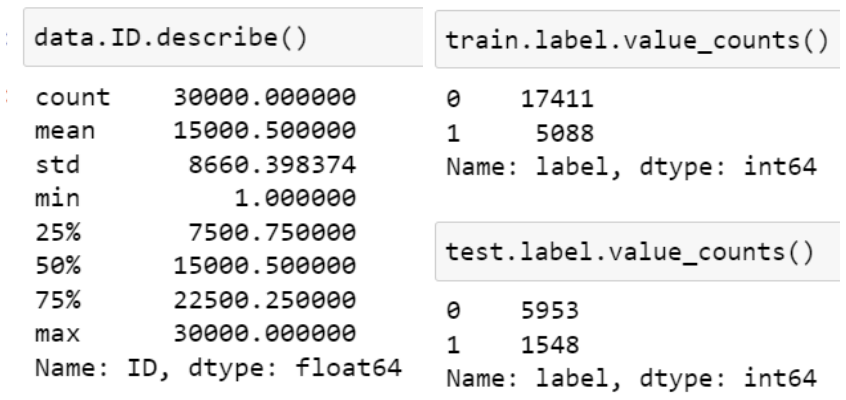
It's a good practice to perform data preprocessing on the dataset before building a scorecard. The dataset has been loaded, and the default label rate has been checked (22%) . It's crucial to note that a high default rate in historical data can indicate a high risk of default in the future.

```
There are total 30000 records in our data.
Is Fraud:
Count: 6636
Proportion (Fraud): 22.0%
Not Fraud:
Count: 23364
Proportion (Not Fraud): 77.9%

'22.0'
```



Train Test Split:

To split the dataset into train and test sets, it's recommended to use a time-based approach instead of a random split. This is because a random split may break the time series and cause overfitting. In this project, we will use the user ID as a proxy for time to perform the train and test split.



```
data.ID.describe()

count    30000.000000
mean     15000.500000
std       8660.398374
min          1.000000
25%       7500.750000
50%      15000.500000
75%      22500.250000
max      30000.000000
Name: ID, dtype: float64
```

```
train.label.value_counts()

0    17411
1     5088
Name: label, dtype: int64
```

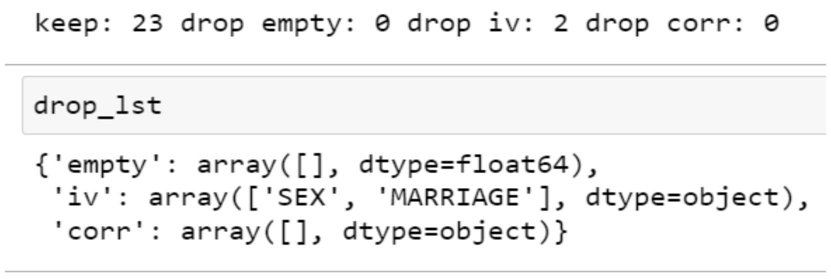
```
test.label.value_counts()

0    5953
1    1548
Name: label, dtype: int64
```

Feature Filtering:

Performing feature filtering is a critical step in building a scorecard system. It involves dropping features that have low information value and high correlation with other features. Low information value features are those that do not contribute significantly to the target variable's prediction. High correlation between features can lead to multicollinearity, making it challenging to interpret the model's coefficients accurately.

To identify low information value features, we can use the information value (IV) metric, which measures the predictive power of each feature in the dataset. Features with IV scores less than 0.02 are typically considered low information value and can be dropped from the dataset.

To identify highly correlated features, we can use the correlation matrix and drop features that have a correlation coefficient greater than 0.8 or less than -0.8. This helps to reduce the number of features in the dataset, leading to a more interpretable and efficient scorecard model.



```
keep: 23 drop empty: 0 drop iv: 2 drop corr: 0
```

```
drop_lst

{'empty': array([], dtype=float64),
 'iv': array(['SEX', 'MARRIAGE'], dtype=object),
 'corr': array([], dtype=object)}
```

Weight of evidence (WOE) — describes the relationship between a predictive variable and a binary target variable

Information Value (IV) — measures the strength of that relationship based on WOE. The industry level is to drop features with an IV lower than 0.02

$$Information\ Value = \ln\left(\frac{\%good}{\%bad}\right) * (\%good - \%bad)$$

$$Weight\ Of\ Evidence\ = \ln\left(\frac{\%good}{\%bad}\right)$$

This is the IV ranking for all the features. We can see that PAY_0 has the highest IV, which makes sense because this feature indicates the most recent repayment status. EDUCATION and AGE have a low IV compared to payment status and amount.

	name	iv
0	PAY_0	0.864707
1	PAY_2	0.536068
2	PAY_3	0.401513
3	PAY_4	0.349813
4	PAY_5	0.340014
5	PAY_6	0.282692
6	PAY_AMT1	0.189334
7	LIMIT_BAL	0.170973
8	PAY_AMT2	0.168256
9	PAY_AMT3	0.128827
10	PAY_AMT4	0.121719
11	PAY_AMT6	0.106017
12	PAY_AMT5	0.103353
13	BILL_AMT4	0.047998
14	BILL_AMT5	0.046490
15	BILL_AMT3	0.044047
16	EDUCATION	0.036458
17	ID	0.032184
18	AGE	0.027512
19	BILL_AMT6	0.027461
20	BILL_AMT1	0.026191
21	BILL_AMT2	0.022539

Feature Binning:

Feature binning is to transform a continuous or numerical variable into a categorical feature.

Advantages of Feature Binning:

- It simplifies the logistic regression model and reduces the risk of model overfitting
- Logistic regression is a generalized linear model, and its expressive ability is limited; Feature binning can introduce nonlinearity into the model, which can improve the expressive ability of the model and help better model fitting
- The discretized features are very robust to abnormal data: for example, the value of a feature is 1 if age > 30, and 0 otherwise. If the features are not discretized, an abnormal data point “300 years old” will impact the model fitting
- It can treat null data as an individual class .

Steps for feature binning:

Step 1. Initialization: c = toad.transform.Combiner()

Step 2. Training binning:

```
c.fit(dataframe,
    y = 'target',
    method = 'chi',
    min_samples = 0.05,
    n_bins = None,
    empty_separate = False)
```

- y: target column
- method: binning method, supports chi (chi-square binning), dt (decision tree binning), kmean, quantile, step (equal step size binning)
- min_samples: Each box contains the least number of samples, which can be a number or a proportion
- n_bins: the number of bins; If it is not possible to divide so many boxes the maximum number of bins will be divided.
- empty_separate: Whether to separate empty boxes separately

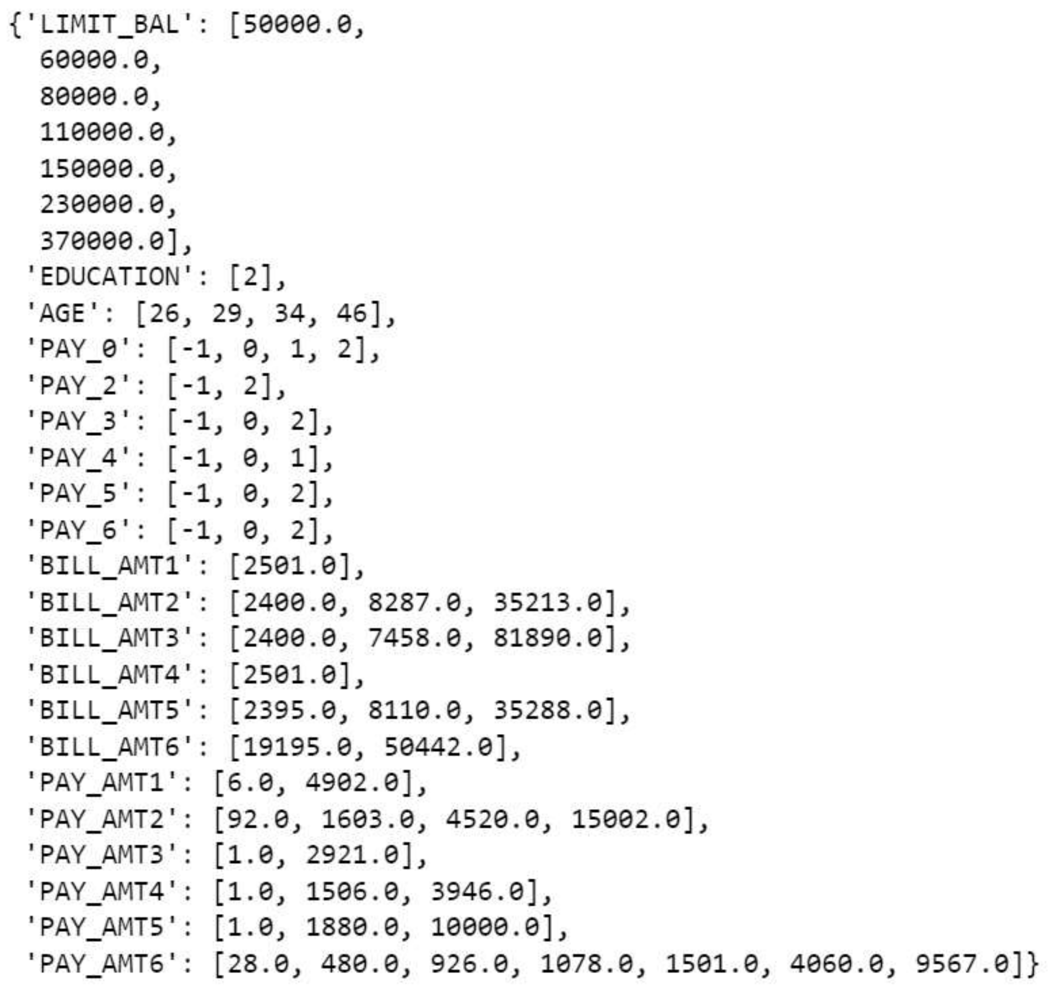
Step 3. check binning nodes: c.export()

Step 4. Manually adjust binning: c.load(dict)

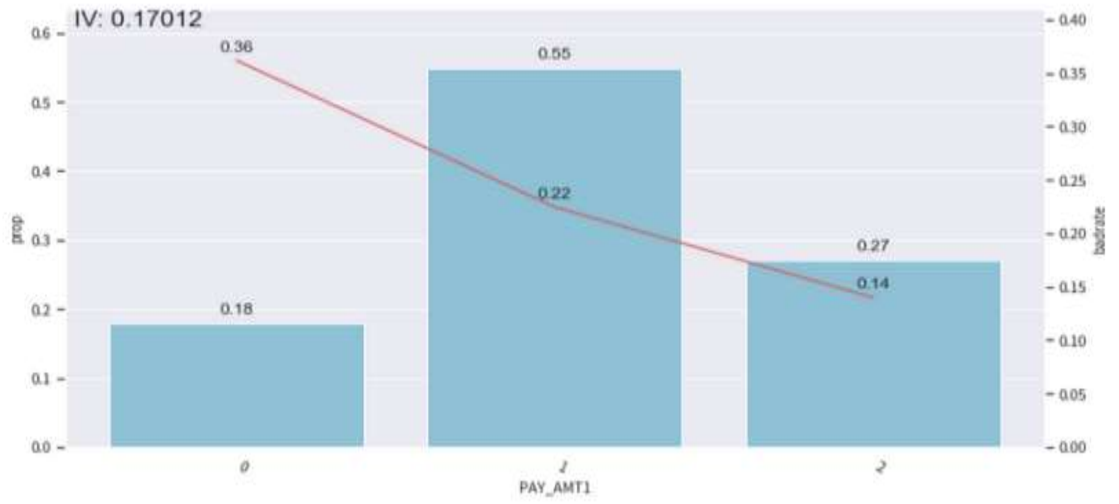
Step 5. Apply binning results: c.transform(dataframe, labels=False)

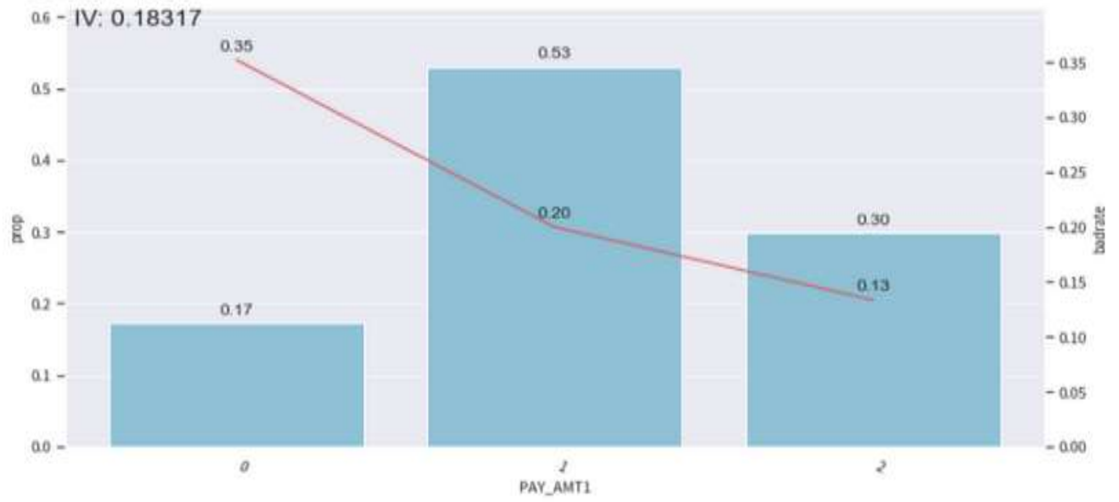
labels: Whether to convert the binning results into box labels. If False, output 0, 1, 2… (discrete variables are sorted according to the proportion), and if True output (-inf, 0], (0,10], (10, inf).

The binning result:



```
{'LIMIT_BAL': [50000.0,
  60000.0,
  80000.0,
  110000.0,
  150000.0,
  230000.0,
  370000.0],
 'EDUCATION': [2],
 'AGE': [26, 29, 34, 46],
 'PAY_0': [-1, 0, 1, 2],
 'PAY_2': [-1, 2],
 'PAY_3': [-1, 0, 2],
 'PAY_4': [-1, 0, 1],
 'PAY_5': [-1, 0, 2],
 'PAY_6': [-1, 0, 2],
 'BILL_AMT1': [2501.0],
 'BILL_AMT2': [2400.0, 8287.0, 35213.0],
 'BILL_AMT3': [2400.0, 7458.0, 81890.0],
 'BILL_AMT4': [2501.0],
 'BILL_AMT5': [2395.0, 8110.0, 35288.0],
 'BILL_AMT6': [19195.0, 50442.0],
 'PAY_AMT1': [6.0, 4902.0],
 'PAY_AMT2': [92.0, 1603.0, 4520.0, 15002.0],
 'PAY_AMT3': [1.0, 2921.0],
 'PAY_AMT4': [1.0, 1506.0, 3946.0],
 'PAY_AMT5': [1.0, 1880.0, 10000.0],
 'PAY_AMT6': [28.0, 480.0, 926.0, 1078.0, 1501.0, 4060.0, 9567.0]}
```





In this plot, the bar plot represents the proportion of the data in the corresponding bin; the red line represents the proportion of default customers.

We need to make sure that the binning has monotonicity, which means the line is trending in the same direction with no sudden jump or drop.

This plot looks ok, if there is a sudden jump or drop, we need to use c.set_rules(dict) to combine the binning.

Transform to WOE and Calculate PSI:

WOE transformation is performed after the binning is Done.

The steps are as follows:

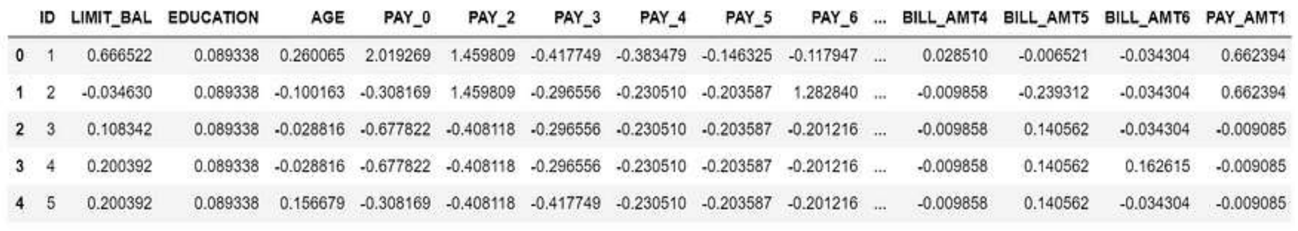
Use the above-adjusted Combiner c to transform the data

Initialize woe transform t: t= toad.transform.WOETransformer()

Training the t: t.fit_transform trains and outputs woe transformed data for the trainset target: target column data (not column name)

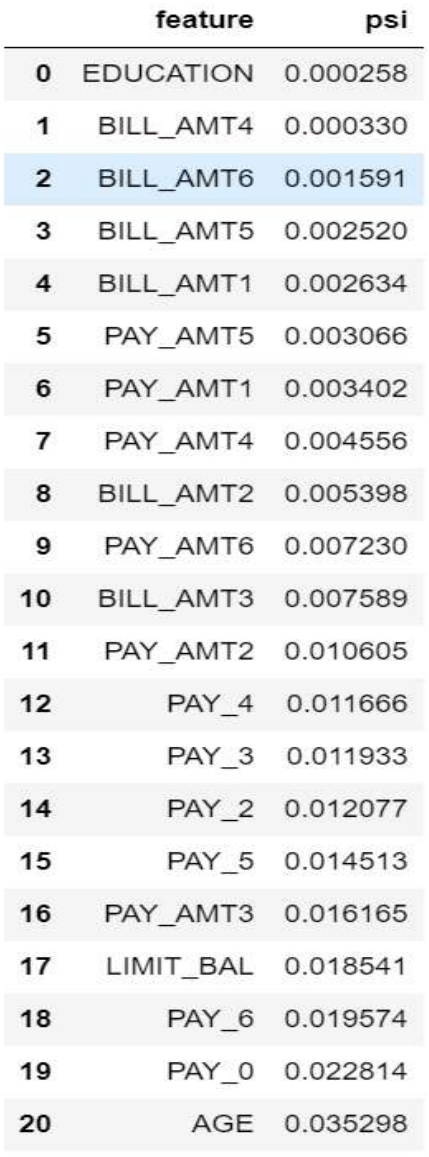
exclude: columns that do not need to be transformed by WOE. Note: All columns will be transformed, including the columns that have not been binning, and the columns that do not need to be converted by WOE will be deleted through exclude, especially the target column.

Transform the test/OOT data: transer.transform



	ID	LIMIT_BAL	EDUCATION	AGE	PAY_0	PAY_2	PAY_3	PAY_4	PAY_5	PAY_6	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1
0	1	0.666522	0.089338	0.260065	2.019269	1.459809	-0.417749	-0.383479	-0.146325	-0.117947	...	0.028510	-0.006521	-0.034304	0.662394
1	2	-0.034630	0.089338	-0.100163	-0.308169	1.459809	-0.296556	-0.230510	-0.203587	1.282840	...	-0.009858	-0.239312	-0.034304	0.662394
2	3	0.108342	0.089338	-0.028816	-0.677822	-0.408118	-0.296556	-0.230510	-0.203587	-0.201216	...	-0.009858	0.140562	-0.034304	-0.009085
3	4	0.200392	0.089338	-0.028816	-0.677822	-0.408118	-0.296556	-0.230510	-0.203587	-0.201216	...	-0.009858	0.140562	0.162615	-0.009085
4	5	0.200392	0.089338	0.156679	-0.308169	-0.408118	-0.417749	-0.230510	-0.203587	-0.201216	...	-0.009858	0.140562	-0.034304	-0.009085

5 rows × 23 columns

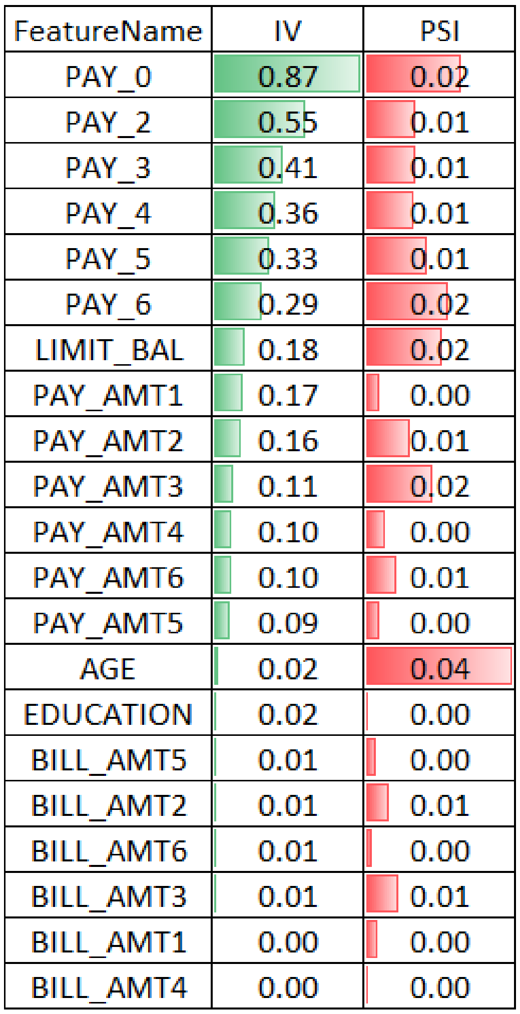


	feature	psi
0	EDUCATION	0.000258
1	BILL_AMT4	0.000330
2	BILL_AMT6	0.001591
3	BILL_AMT5	0.002520
4	BILL_AMT1	0.002634
5	PAY_AMT5	0.003066
6	PAY_AMT1	0.003402
7	PAY_AMT4	0.004556
8	BILL_AMT2	0.005398
9	PAY_AMT6	0.007230
10	BILL_AMT3	0.007589
11	PAY_AMT2	0.010605
12	PAY_4	0.011666
13	PAY_3	0.011933
14	PAY_2	0.012077
15	PAY_5	0.014513
16	PAY_AMT3	0.016165
17	LIMIT_BAL	0.018541
18	PAY_6	0.019574
19	PAY_0	0.022814
20	AGE	0.035298

$$PSI = \sum \left((\%Actual - \%Expected) \times \ln \frac{\%Actual}{\%Expected} \right)$$

Output Final IV:

This step is to output the IV after the WOE transformation, it's a little bit different than the raw features' IV.



FeatureName	IV	PSI
PAY_0	0.87	0.02
PAY_2	0.55	0.01
PAY_3	0.41	0.01
PAY_4	0.36	0.01
PAY_5	0.33	0.01
PAY_6	0.29	0.02
LIMIT_BAL	0.18	0.02
PAY_AMT1	0.17	0.00
PAY_AMT2	0.16	0.01
PAY_AMT3	0.11	0.02
PAY_AMT4	0.10	0.00
PAY_AMT6	0.10	0.01
PAY_AMT5	0.09	0.00
AGE	0.02	0.04
EDUCATION	0.02	0.00
BILL_AMT5	0.01	0.00
BILL_AMT2	0.01	0.01
BILL_AMT6	0.01	0.00
BILL_AMT3	0.01	0.01
BILL_AMT1	0.00	0.00
BILL_AMT4	0.00	0.00

The idea is to get the features with the highest IV and lowest PSI.

Model Tuning:

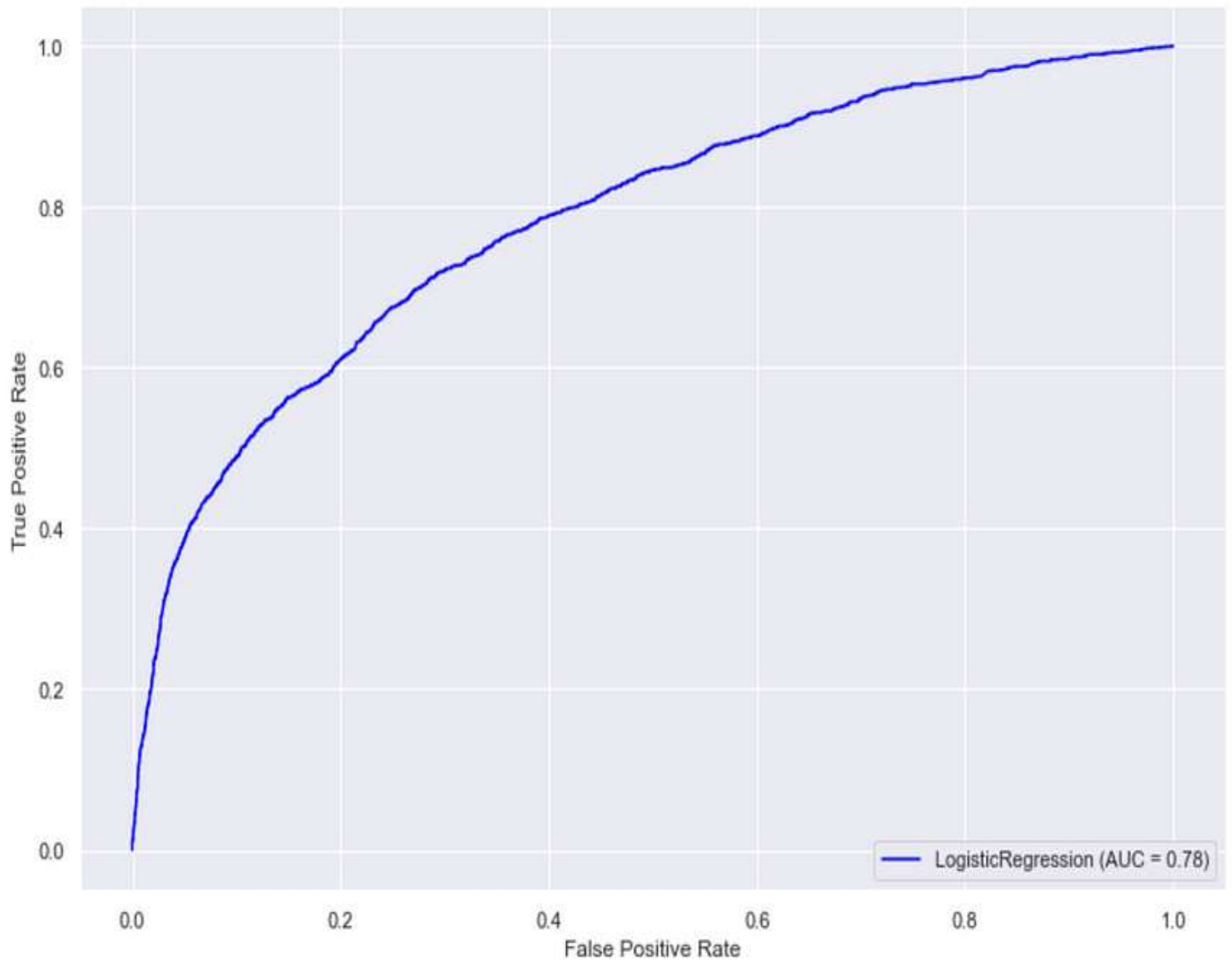
Logistic Regression: The most used algorithm in the credit scorecard modeling process is Logistic Regression. The reasons are as follows:

- Simple linear relationship: the relationship between variables is a linear relationship
- Good Interpretability: the effect of input variables on target variables is readily available
- Give probabilities instead of discriminative classes: the customer's characteristic information (such as marriage, age, historical credit performance, etc.) can be integrated and converted into a probability value, which provides an intuitive basis to

predict whether the customer is good or bad. That is, the larger the value, the smaller the probability that the customer will default in the future.

- Easy to deploy: testing, deployment, monitoring, tuning, etc., are relatively simple

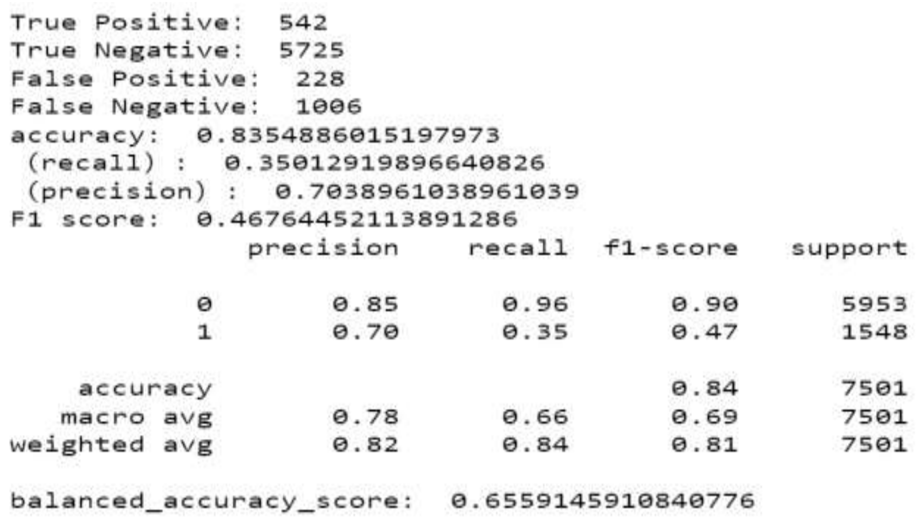
```
train KS 0.40988185783295383
train AUC 0.7703247664492447
Test KS 0.4264763906414198
Test AUC 0.7834675891381715
```



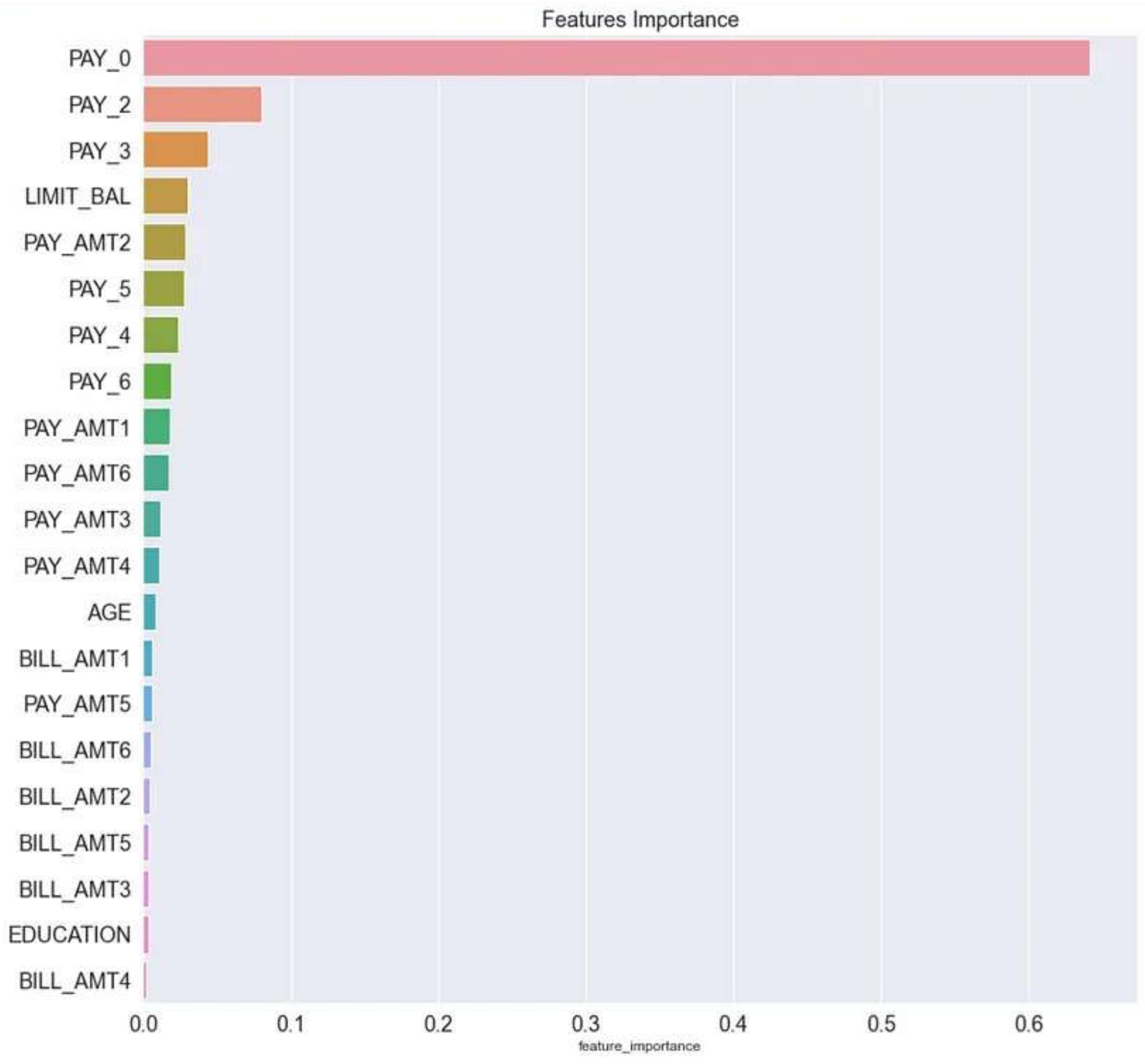
We can see that there is not a big difference between the train AUC and test AUC or train KS and test KS. This means our model does not overfit.

Train a Gradient Boosting Classifier and check the feature importance table

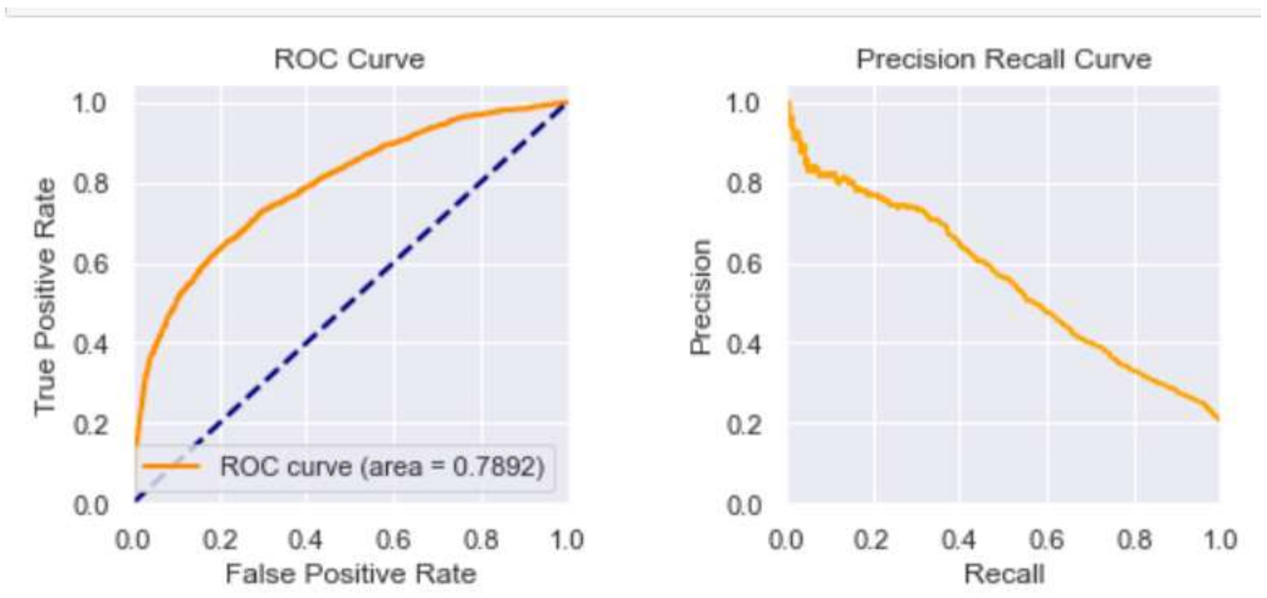
To see if a GBDT model will perform better than LR and compare the feature importance table with IV.



```
True Positive:  542
True Negative:  5725
False Positive:  228
False Negative:  1006
accuracy:  0.8354886015197973
 (recall) :  0.35012919896640826
 (precision) :  0.7038961038961039
F1 score:  0.46764452113891286
              precision    recall  f1-score   support

           0       0.85      0.96      0.90      5953
           1       0.70      0.35      0.47      1548

    accuracy                           0.84      7501
   macro avg       0.78      0.66      0.69      7501
weighted avg       0.82      0.84      0.81      7501

balanced_accuracy_score:  0.6559145910840776
```



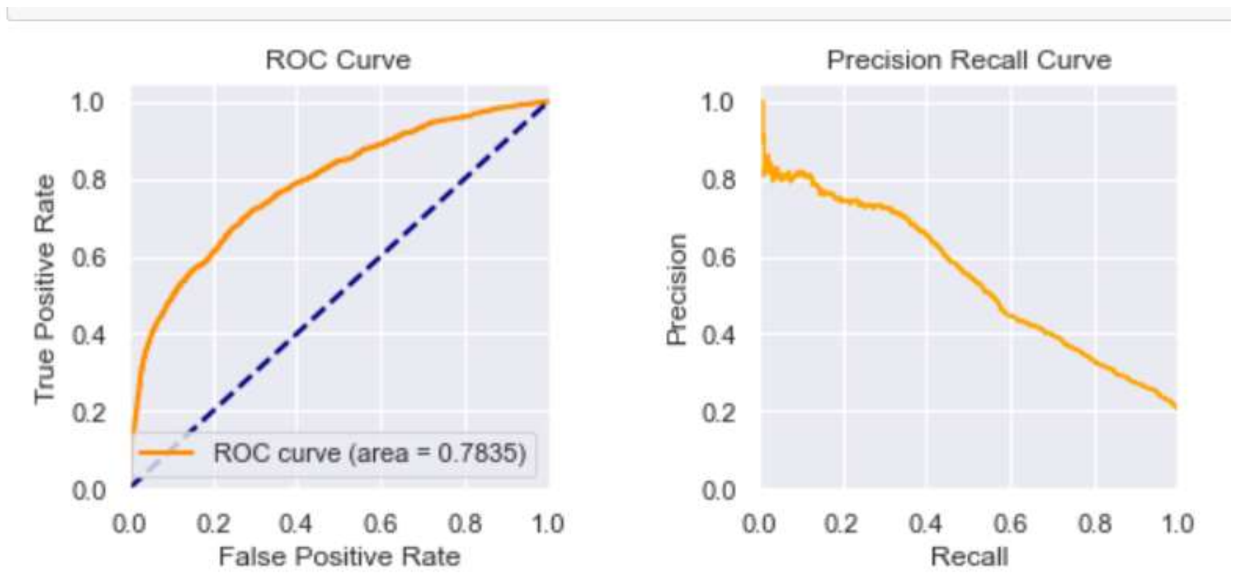
As we can see from the feature importance table, GBDT puts a lot of weight (64%) on the PAY_0 feature.



The ROC and Precision-Recall curve look ok.

Model Production:

Let's train our production model for Logistic Regression



AUC of LR: 0.7835

AUC of GBDT: 0.7892

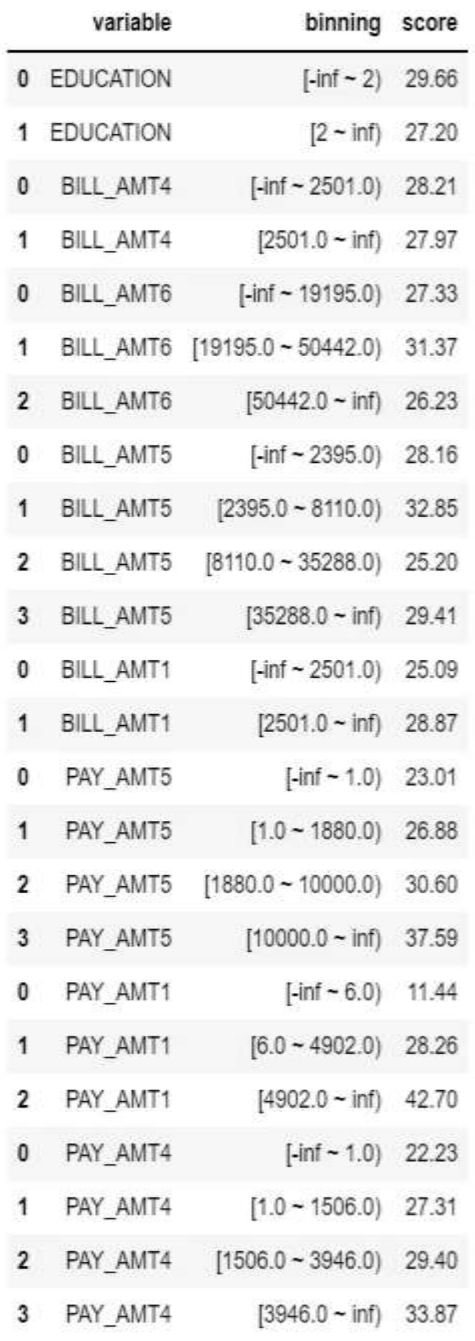
Not a big difference between these models. So it's ok to use LR to build a scorecard.

Scorecard Tuning:

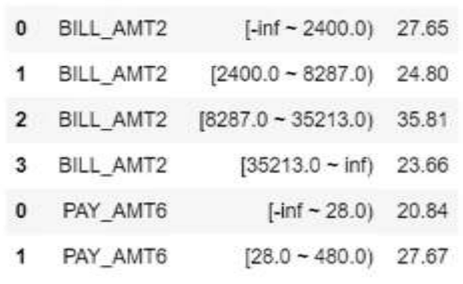
The following parameters are the most important ones for Scorecard Tuning.

base_score = 1000, base_odds = 35 , pdo = 80, rate = 2

The actual meaning is that when the base odds are 35, the benchmark score is 1000, and when the ratio is twice the benchmark, the benchmark score drops by 80 points.



	variable	binning	score
0	EDUCATION	[-inf ~ 2)	29.66
1	EDUCATION	[2 ~ inf)	27.20
0	BILL_AMT4	[-inf ~ 2501.0)	28.21
1	BILL_AMT4	[2501.0 ~ inf)	27.97
0	BILL_AMT6	[-inf ~ 19195.0)	27.33
1	BILL_AMT6	[19195.0 ~ 50442.0)	31.37
2	BILL_AMT6	[50442.0 ~ inf)	26.23
0	BILL_AMT5	[-inf ~ 2395.0)	28.16
1	BILL_AMT5	[2395.0 ~ 8110.0)	32.85
2	BILL_AMT5	[8110.0 ~ 35288.0)	25.20
3	BILL_AMT5	[35288.0 ~ inf)	29.41
0	BILL_AMT1	[-inf ~ 2501.0)	25.09
1	BILL_AMT1	[2501.0 ~ inf)	28.87
0	PAY_AMT5	[-inf ~ 1.0)	23.01
1	PAY_AMT5	[1.0 ~ 1880.0)	26.88
2	PAY_AMT5	[1880.0 ~ 10000.0)	30.60
3	PAY_AMT5	[10000.0 ~ inf)	37.59
0	PAY_AMT1	[-inf ~ 6.0)	11.44
1	PAY_AMT1	[6.0 ~ 4902.0)	28.26
2	PAY_AMT1	[4902.0 ~ inf)	42.70
0	PAY_AMT4	[-inf ~ 1.0)	22.23
1	PAY_AMT4	[1.0 ~ 1506.0)	27.31
2	PAY_AMT4	[1506.0 ~ 3946.0)	29.40
3	PAY_AMT4	[3946.0 ~ inf)	33.87



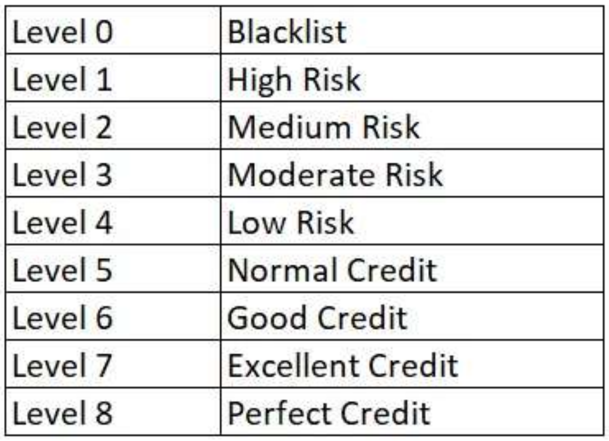
0	BILL_AMT2	[-inf ~ 2400.0)	27.65
1	BILL_AMT2	[2400.0 ~ 8287.0)	24.80
2	BILL_AMT2	[8287.0 ~ 35213.0)	35.81
3	BILL_AMT2	[35213.0 ~ inf)	23.66
0	PAY_AMT6	[-inf ~ 28.0)	20.84
1	PAY_AMT6	[28.0 ~ 480.0)	27.67

Here we have our scorecard; once we get this table, we can throw this CSV file to the developer and let them develop a service to score each customer.

But there is something more we need to do; for a typical scorecard, we need to have a score range, and each range presents a level of trust.

In this way, the business side people are more easily to perform an action against different level customers.

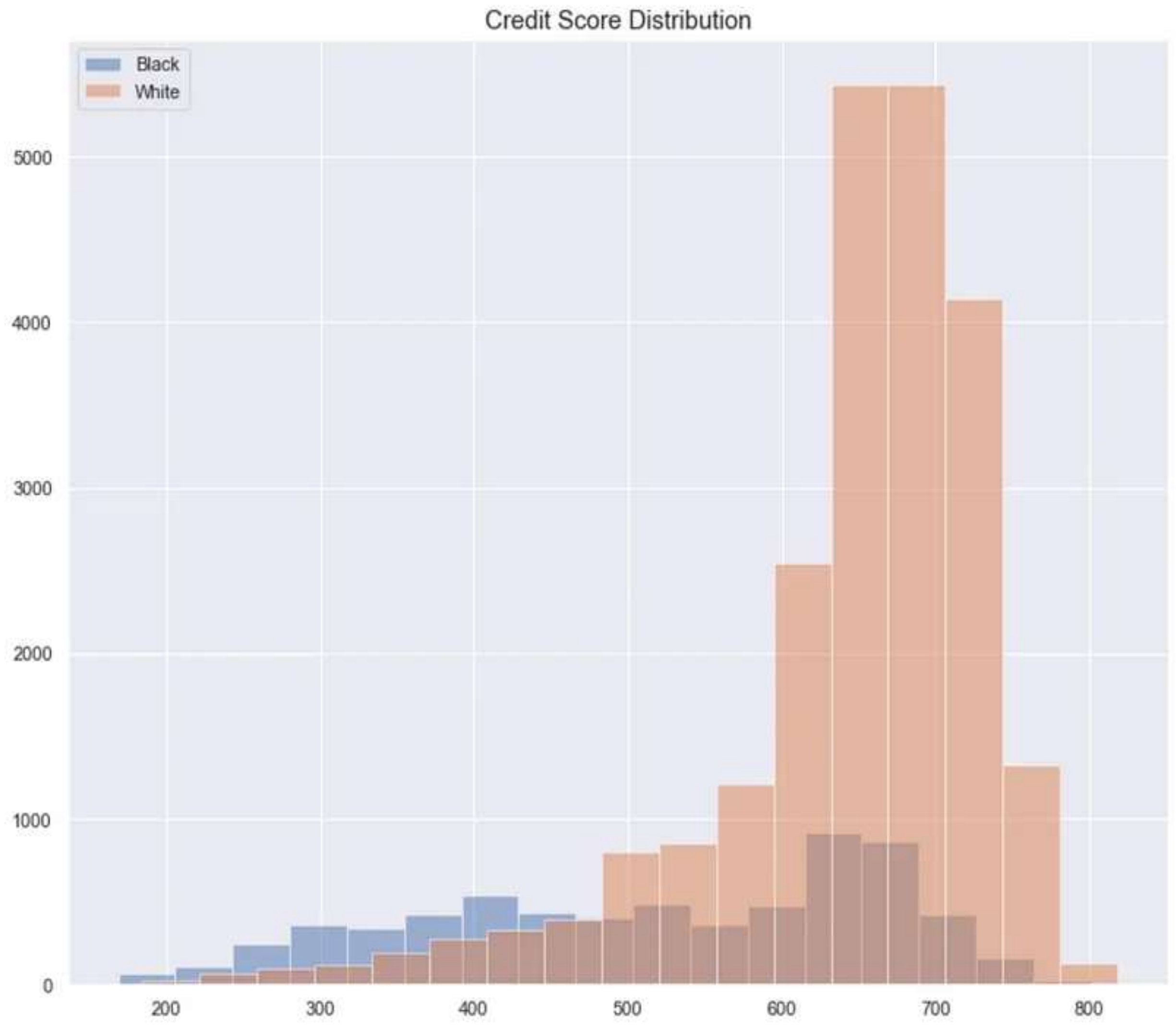
For example, we can set the credit level like this:



Level 0	Blacklist
Level 1	High Risk
Level 2	Medium Risk
Level 3	Moderate Risk
Level 4	Low Risk
Level 5	Normal Credit
Level 6	Good Credit
Level 7	Excellent Credit
Level 8	Perfect Credit

<u>Distribution Analysis:</u>

We need to plot the score distribution of default customers v.s. good customers in order to split the credit levels (level 0 to level 8).



Black means default customers, and white means good customers.

A good model will clearly separate the black/white distribution.

The ideal distribution is a smile shape

Good customers with high credit scores->to the very right

Default customers with low credit scores-> to the very left.

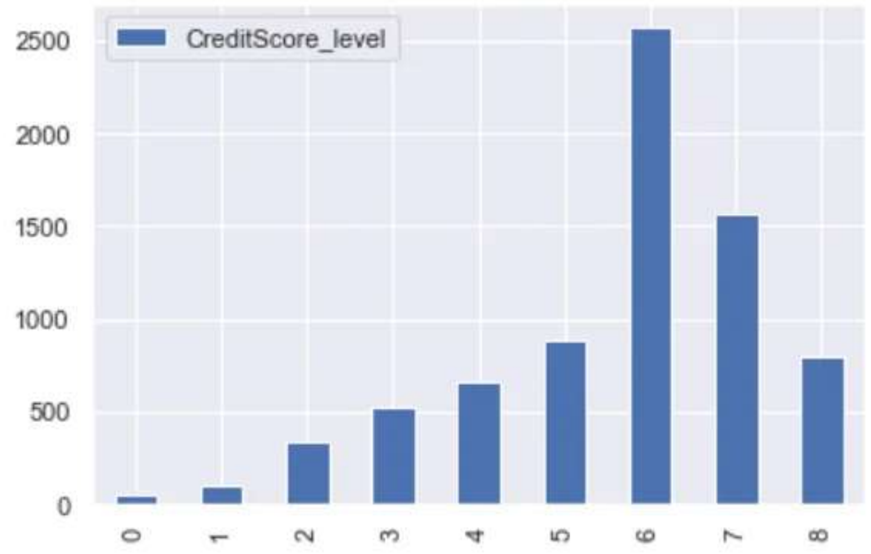
Our current model doesn’t separate these distributions very well. For me, I would go back to explore more features to increase the predictive power. But we can always build a baseline model first and improve based on that.

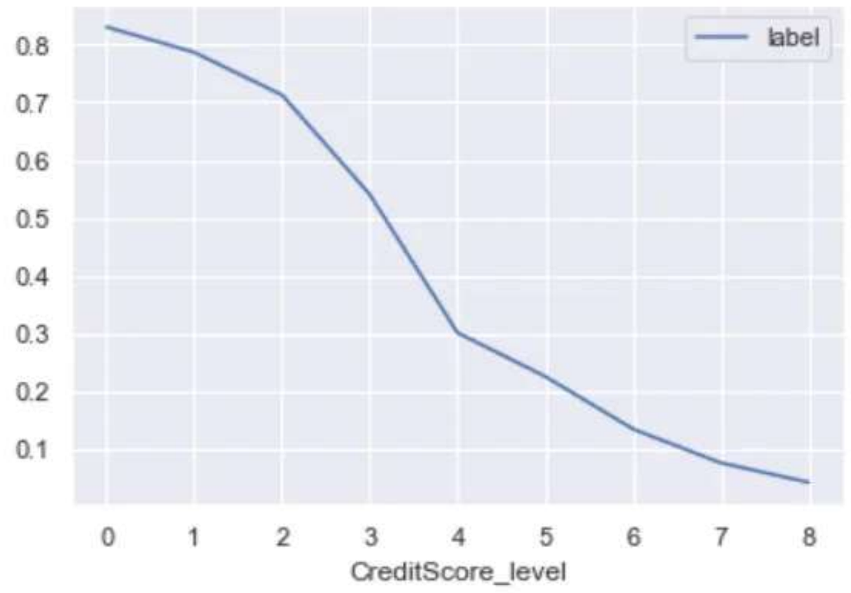
Threshold Tuning

We need to perform threshold tuning for different credit levels, and that's a trade-off between loss & coverage.

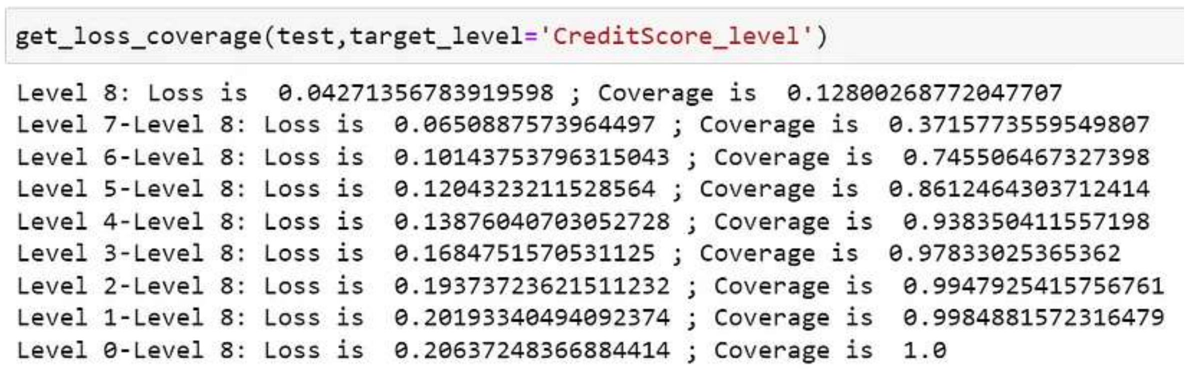
Let's say that company is OK with some loss, but you have to cover 70% of good customers for next month.

In other words, your goal is to find a threshold with ≤10% loss and ≥70% coverage.



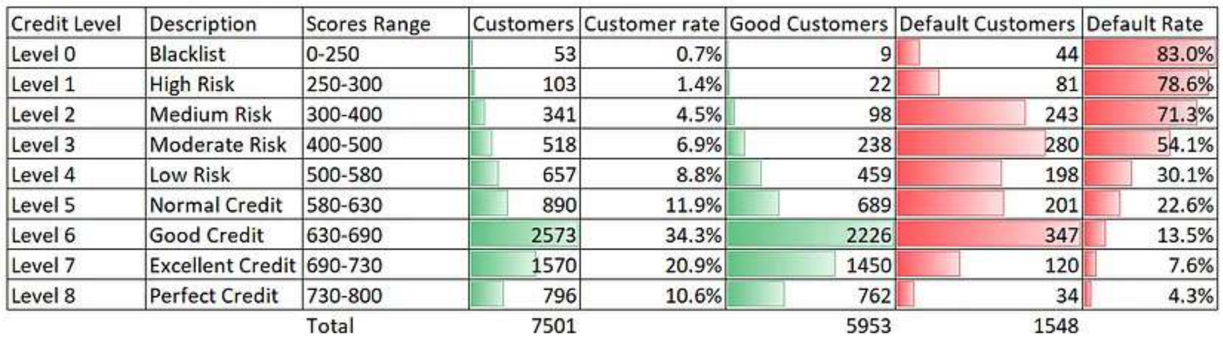


This plot indicates the distribution of each credit level and the credit default rate at that level.



```
get_loss_coverage(test,target_level='CreditScore_level')
```

```
Level 8: Loss is  0.04271356783919598 ; Coverage is  0.12800268772047707
Level 7-Level 8: Loss is  0.0650887573964497 ; Coverage is  0.3715773559549807
Level 6-Level 8: Loss is  0.10143753796315043 ; Coverage is  0.745506467327398
Level 5-Level 8: Loss is  0.1204323211528564 ; Coverage is  0.8612464303712414
Level 4-Level 8: Loss is  0.13876040703052728 ; Coverage is  0.938350411557198
Level 3-Level 8: Loss is  0.1684751570531125 ; Coverage is  0.97833025365362
Level 2-Level 8: Loss is  0.19373723621511232 ; Coverage is  0.9947925415756761
Level 1-Level 8: Loss is  0.20193340494092374 ; Coverage is  0.9984881572316479
Level 0-Level 8: Loss is  0.20637248366884414 ; Coverage is  1.0
```



Credit Level	Description	Scores Range	Customers	Customer rate	Good Customers	Default Customers	Default Rate
Level 0	Blacklist	0-250	53	0.7%	9	44	83.0%
Level 1	High Risk	250-300	103	1.4%	22	81	78.6%
Level 2	Medium Risk	300-400	341	4.5%	98	243	71.3%
Level 3	Moderate Risk	400-500	518	6.9%	238	280	54.1%
Level 4	Low Risk	500-580	657	8.8%	459	198	30.1%
Level 5	Normal Credit	580-630	890	11.9%	689	201	22.6%
Level 6	Good Credit	630-690	2573	34.3%	2226	347	13.5%
Level 7	Excellent Credit	690-730	1570	20.9%	1450	120	7.6%
Level 8	Perfect Credit	730-800	796	10.6%	762	34	4.3%
		Total	7501		5953	1548	

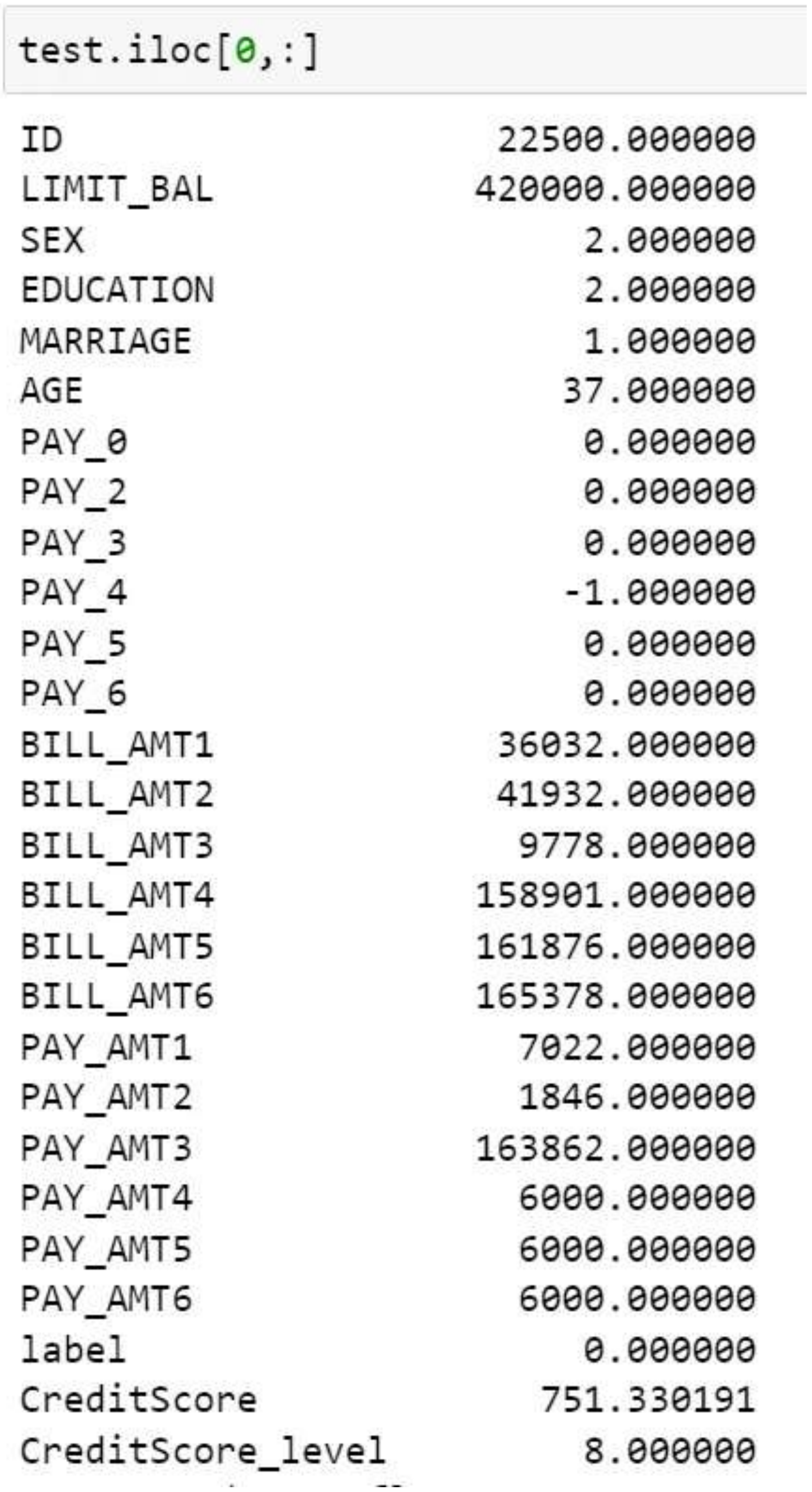
By examining each level, you will have a loss and coverage table. For example, if you send the balance transfer offers to all the people 7501 (L0-L8) you will incur a 21% loss rate (1548/7501).

To reach the goal (≤10% loss and ≥70% coverage), you need to pick L6-L8 with a 10.1% loss (347+120+34)/(2573+1570+796) and a 75% coverage (2226+1450+762)/5953.

Basically, next month (assuming that the test set is next month's data) the business-side people will send the balance transfer offers to customers with a Level 6 rating or above, a total of 4939 customers.

Manually Test our Scorecard

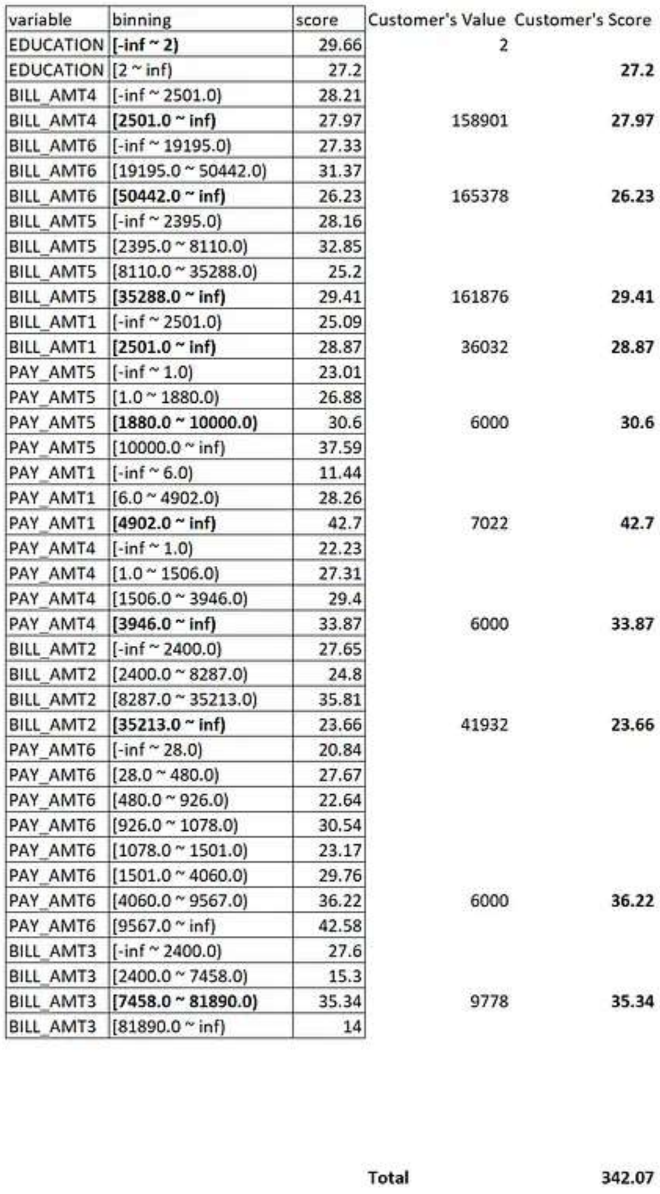
Can we send a balance transfer offer to a customer with the following information next month?



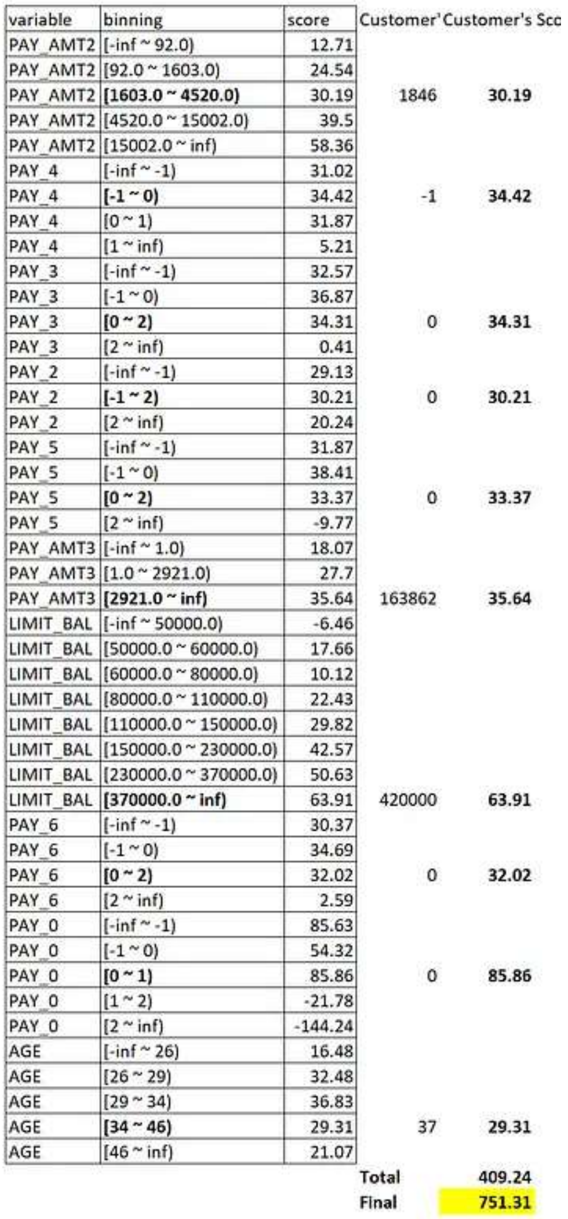
```
test.iloc[0,:]
```

```
ID                      22500.000000
LIMIT_BAL              420000.000000
SEX                         2.000000
EDUCATION                   2.000000
MARRIAGE                    1.000000
AGE                        37.000000
PAY_0                       0.000000
PAY_2                       0.000000
PAY_3                       0.000000
PAY_4                      -1.000000
PAY_5                       0.000000
PAY_6                       0.000000
BILL_AMT1               36032.000000
BILL_AMT2               41932.000000
BILL_AMT3                9778.000000
BILL_AMT4              158901.000000
BILL_AMT5              161876.000000
BILL_AMT6              165378.000000
PAY_AMT1                 7022.000000
PAY_AMT2                 1846.000000
PAY_AMT3               163862.000000
PAY_AMT4                 6000.000000
PAY_AMT5                 6000.000000
PAY_AMT6                 6000.000000
label                       0.000000
CreditScore               751.330191
CreditScore_level           8.000000
```

Let's manually check the scorecard.

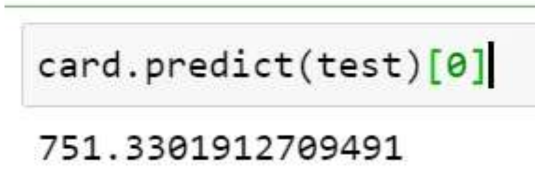


variable	binning	score	Customer's Value	Customer's Score
EDUCATION	**[-inf ~ 2)**	29.66	2	
EDUCATION	[2 ~ inf)	27.2		**27.2**
BILL_AMT4	[-inf ~ 2501.0)	28.21		
BILL_AMT4	**[2501.0 ~ inf)**	27.97	158901	**27.97**
BILL_AMT6	[-inf ~ 19195.0)	27.33		
BILL_AMT6	[19195.0 ~ 50442.0)	31.37		
BILL_AMT6	**[50442.0 ~ inf)**	26.23	165378	**26.23**
BILL_AMT5	[-inf ~ 2395.0)	28.16		
BILL_AMT5	[2395.0 ~ 8110.0)	32.85		
BILL_AMT5	[8110.0 ~ 35288.0)	25.2		
BILL_AMT5	**[35288.0 ~ inf)**	29.41	161876	**29.41**
BILL_AMT1	[-inf ~ 2501.0)	25.09		
BILL_AMT1	**[2501.0 ~ inf)**	28.87	36032	**28.87**
PAY_AMT5	[-inf ~ 1.0)	23.01		
PAY_AMT5	[1.0 ~ 1880.0)	26.88		
PAY_AMT5	**[1880.0 ~ 10000.0)**	30.6	6000	**30.6**
PAY_AMT5	[10000.0 ~ inf)	37.59		
PAY_AMT1	[-inf ~ 6.0)	11.44		
PAY_AMT1	[6.0 ~ 4902.0)	28.26		
PAY_AMT1	**[4902.0 ~ inf)**	42.7	7022	**42.7**
PAY_AMT4	[-inf ~ 1.0)	22.23		
PAY_AMT4	[1.0 ~ 1506.0)	27.31		
PAY_AMT4	[1506.0 ~ 3946.0)	29.4		
PAY_AMT4	**[3946.0 ~ inf)**	33.87	6000	**33.87**
BILL_AMT2	[-inf ~ 2400.0)	27.65		
BILL_AMT2	[2400.0 ~ 8287.0)	24.8		
BILL_AMT2	[8287.0 ~ 35213.0)	35.81		
BILL_AMT2	**[35213.0 ~ inf)**	23.66	41932	**23.66**
PAY_AMT6	[-inf ~ 28.0)	20.84		
PAY_AMT6	[28.0 ~ 480.0)	27.67		
PAY_AMT6	[480.0 ~ 926.0)	22.64		
PAY_AMT6	[926.0 ~ 1078.0)	30.54		
PAY_AMT6	[1078.0 ~ 1501.0)	23.17		
PAY_AMT6	[1501.0 ~ 4060.0)	29.76		
PAY_AMT6	[4060.0 ~ 9567.0)	36.22	6000	**36.22**
PAY_AMT6	[9567.0 ~ inf)	42.58		
BILL_AMT3	[-inf ~ 2400.0)	27.6		
BILL_AMT3	[2400.0 ~ 7458.0)	15.3		
BILL_AMT3	**[7458.0 ~ 81890.0)**	35.34	9778	**35.34**
BILL_AMT3	[81890.0 ~ inf)	14		
			Total	**342.07**



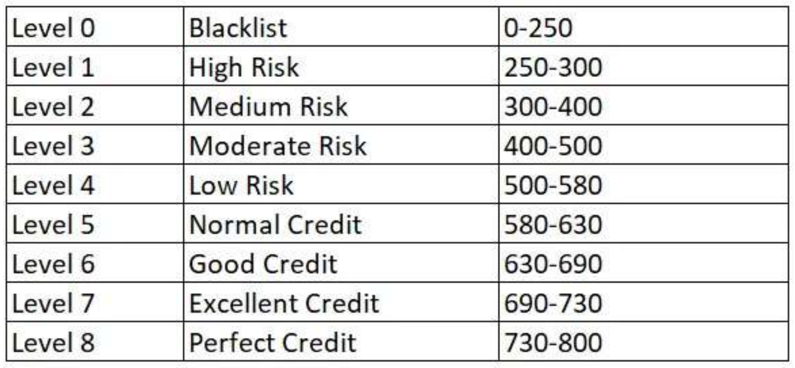
variable	binning	score	Customer'	Customer's Scc
PAY_AMT2	[-inf ~ 92.0)	12.71		
PAY_AMT2	[92.0 ~ 1603.0)	24.54		
PAY_AMT2	**[1603.0 ~ 4520.0)**	30.19	1846	**30.19**
PAY_AMT2	[4520.0 ~ 15002.0)	39.5		
PAY_AMT2	[15002.0 ~ inf)	58.36		
PAY_4	[-inf ~ -1)	31.02		
PAY_4	**[-1 ~ 0)**	34.42	-1	**34.42**
PAY_4	[0 ~ 1)	31.87		
PAY_4	[1 ~ inf)	5.21		
PAY_3	[-inf ~ -1)	32.57		
PAY_3	[-1 ~ 0)	36.87		
PAY_3	**[0 ~ 2)**	34.31	0	**34.31**
PAY_3	[2 ~ inf)	0.41		
PAY_2	[-inf ~ -1)	29.13		
PAY_2	**[-1 ~ 2)**	30.21	0	**30.21**
PAY_2	[2 ~ inf)	20.24		
PAY_5	[-inf ~ -1)	31.87		
PAY_5	[-1 ~ 0)	38.41		
PAY_5	**[0 ~ 2)**	33.37	0	**33.37**
PAY_5	[2 ~ inf)	-9.77		
PAY_AMT3	[-inf ~ 1.0)	18.07		
PAY_AMT3	[1.0 ~ 2921.0)	27.7		
PAY_AMT3	**[2921.0 ~ inf)**	35.64	163862	**35.64**
LIMIT_BAL	[-inf ~ 50000.0)	-6.46		
LIMIT_BAL	[50000.0 ~ 60000.0)	17.66		
LIMIT_BAL	[60000.0 ~ 80000.0)	10.12		
LIMIT_BAL	[80000.0 ~ 110000.0)	22.43		
LIMIT_BAL	[110000.0 ~ 150000.0)	29.82		
LIMIT_BAL	[150000.0 ~ 230000.0)	42.57		
LIMIT_BAL	[230000.0 ~ 370000.0)	50.63		
LIMIT_BAL	**[370000.0 ~ inf)**	63.91	420000	**63.91**
PAY_6	[-inf ~ -1)	30.37		
PAY_6	[-1 ~ 0)	34.69		
PAY_6	**[0 ~ 2)**	32.02	0	**32.02**
PAY_6	[2 ~ inf)	2.59		
PAY_0	[-inf ~ -1)	85.63		
PAY_0	[-1 ~ 0)	54.32		
PAY_0	**[0 ~ 1)**	85.86	0	**85.86**
PAY_0	[1 ~ 2)	-21.78		
PAY_0	[2 ~ inf)	-144.24		
AGE	[-inf ~ 26)	16.48		
AGE	[26 ~ 29)	32.48		
AGE	[29 ~ 34)	36.83		
AGE	**[34 ~ 46)**	29.31	37	**29.31**
AGE	[46 ~ inf)	21.07		
			Total	**409.24**
			Final	**751.31**

<u>Inference by toad:</u>



```
card.predict(test)[0]

751.3301912709491
```

By checking our Credit Level table:



Level 0	Blacklist	0-250
Level 1	High Risk	250-300
Level 2	Medium Risk	300-400
Level 3	Moderate Risk	400-500
Level 4	Low Risk	500-580
Level 5	Normal Credit	580-630
Level 6	Good Credit	630-690
Level 7	Excellent Credit	690-730
Level 8	Perfect Credit	730-800

We can see that this customer is a Level 8 customer with Perfect Credit. So we can issue him/her a balance transfer offer.

Conclusion:

This project walks through the end-to-end process of building a credit scorecard based on the open-source ML tool Toad. The following ML buzzwords are discussed in this blog: Information Value (IV), Weight of evidence (WOE), Population Stability Index (PSI), AUC (Area under the ROC Curve), KS (Kolmogorov-Smirnov), Logistic Regression (LR), GBDT (Gradient Boosting Decision Tree)

For future work, as a data scientist, anyone could try using other ML models to build the scorecard, such as Deep Neural Networks, to increase the accuracy further and lower the false positive rate to improve customer satisfaction.

Project Repository-

https://github.com/akash2262/End-To-End-Project-On-Building-A-Credit-Scorecard-Using-Statistical-Models

References:

- Thomas, L. C. (2002). Credit scoring and its applications. Philadelphia: SIAM- *https://epubs.siam.org/doi/pdf/10.1137/1.9780898718317.fm*

- Baesens, B., Roesch, D., Scheule, H., & Smaers, P. (2015). Credit risk analytics: Measurement techniques, applications, and examples in SAS. John Wiley & Sons- *https://www.scirp.org/(S(351jmbntvnsjt1aadkozje))/reference/referencespapers.aspx?referenceid=2983025*

- Hand, D. J., & Henley, W. E. (1997). Statistical classification methods in consumer credit scoring: A review. Journal of the Royal Statistical Society: Series A (Statistics in Society), 160(3), 523-541- *https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-985X.1997.00078.x*

- Gerasymchuk, S., & de Leeuw, J. (2013). Predicting credit card customer attrition using data mining techniques. Expert Systems with Applications, 40(17), 6822-6828- *https://www.researchgate.net/publication/23646357_Predicting_credit_card_customer_churn_in_banks_using_data_mining*

- Thomas, L. C., & Edelman, D. B. (2002). The credit scoring toolkit: Theory and practice for retail credit risk management and decision automation. Oxford University Press.- *https://epubs.siam.org/doi/pdf/10.1137/1.9781611974560.bm*

- Bartlett, J. E., Kotrlik, J. W., & Higgins, C. C. (2001). Organizational research: Determining appropriate sample size in survey research. Information Technology, Learning, and Performance Journal, 19(1), 43-50.- *https://www.scirp.org/(S(351jmbntvnsjt1aadkposzje))/reference/ReferencesPapers.aspx?ReferenceID=1588649*

- Baesens, B., Setiono, R., & Mues, C. (2003). Using neural network rule extraction and decision tables for credit-risk evaluation. Management Science, 49(3), 312-329. – *https://www.jstor.org/stable/4133928*

- Crook, J., Banasik, J., & Thomas, L. (2007). Neural networks and genetic algorithms for predicting bankruptcy. Neural Computing & Applications, 16(2), 149-159. - *https://www.researchgate.net/publication/262159616_H_A_ABDOU_AND_J_POINTON*

- Lim, A. (2014). Predicting creditworthiness using logistic regression and decision trees. International Journal of Business and Management, 9(9), 65-72.-

  https://www.diva-portal.org/smash/get/diva2:1375762/FULLTEXT01.pdf

- Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: Data mining, inference, and prediction (2nd ed.). Springer.- https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1751-5823.2009.00095_18.x