Hackathon 1 – Group 4

Taiwanese Credit Card Clients

https://github.com/kravigupta/Taiwanese_Credit_Card_Client_Fraud_detection

Team Members:

Aditya Sarode
Calvin Castelino
Ravi Gupta
Shawn Gomes

Agenda

- Problem Statement
- Data Sources and EDA
- Feature Engineering
- Machine Learning Model
- Conclusion and Recommendations.

Problem Statement

- Certain Cases of Customers default on Payments in Taiwan.
- From a Risk Management Perspective a Bank/Credit Card Company is more interested in minimizing their losses towards a particular customer.
- Goal: To compute the predictive accuracy of probability of default for a Taiwanese Credit Card Client.
- Problem Analysis – Classify Probability of default for next month: 1 as “Default” and 0 as “Not Default”.

Data Source

- Data available at - https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients
- Data Consists of -
  - ➢ 23 features
  - ➢ 30000 records
- ID column considered as Index
- 20 Numeric Features -
  - ➢ LIMIT_BAL, AGE,
  - ➢ PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, PAY_6,
  - ➢ BILL_AMT1, BILL_AMT2, BILL_AMT3, BILL_AMT4, BILL_AMT5, BILL_AMT6,
  - ➢ PAY_AMT1, PAY_AMT2, PAY_AMT3, PAY_AMT4, PAY_AMT5, PAY_AMT6
- 3 Categorical Features -
  - ➢ SEX, EDUCATION, MARRIAGE

Numerical Features

- BILL_AMT 6, BILL_AMT 5..........BILL_AMT 1

Represents Amount of Bill Statement in each month from April to Sep 2005

- PAY_AMT6, PAY_AMT5.........PAY_AMT1

Represents Amount of Previous Payment in each month from April to Sep 2005

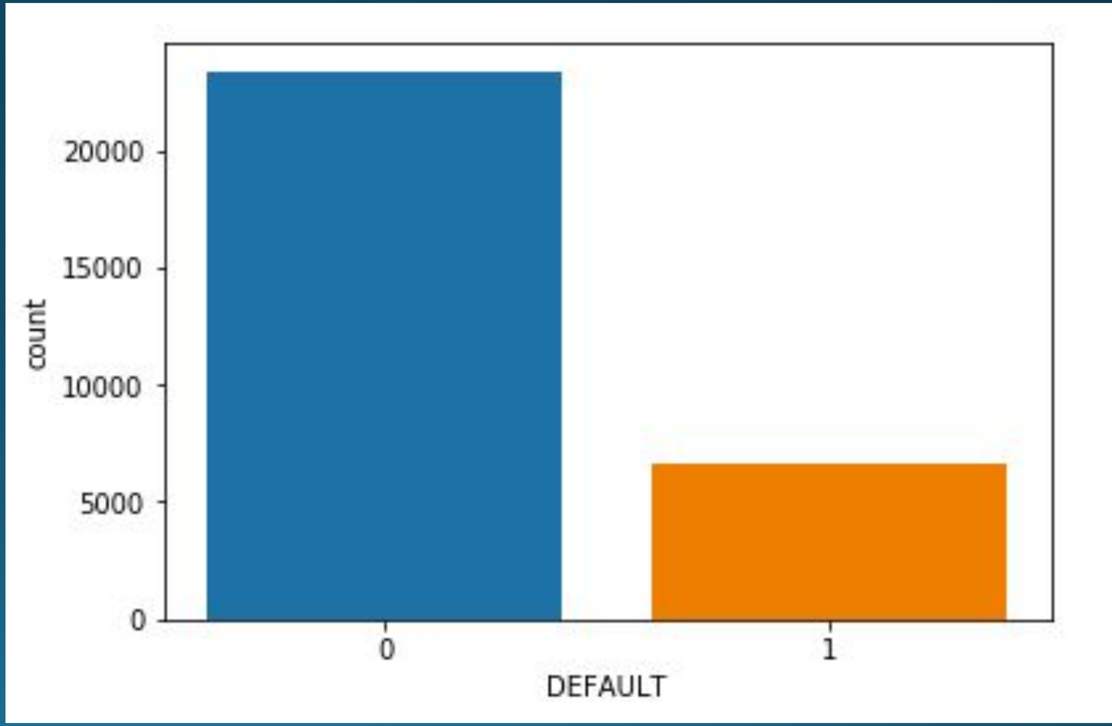
- PAY_2, PAY_3 .... PAY_6

Represents Repayment Status in each month from April to Sep 2005

Target Variable

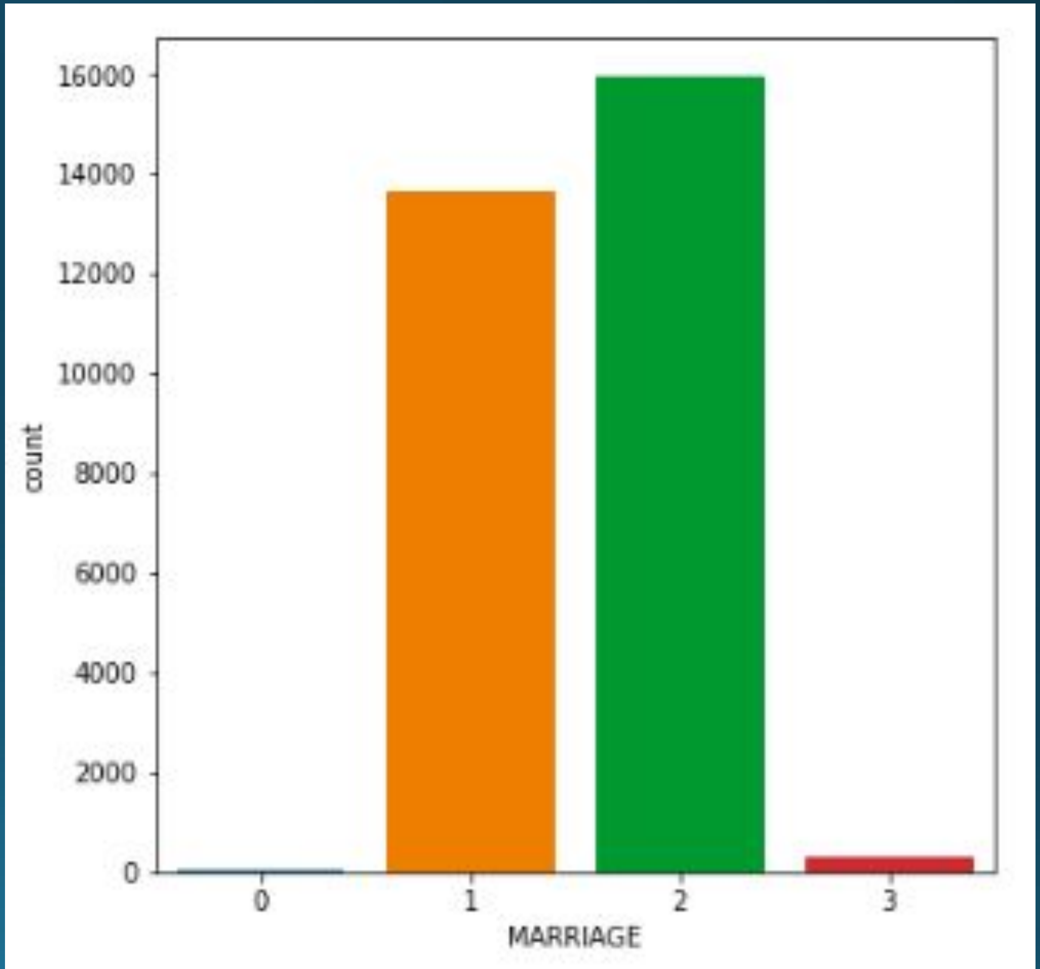
- Whether a person is default or not

- Not default - 0 - 23364
  Default - 1 - 6636



Categorical - Marriage

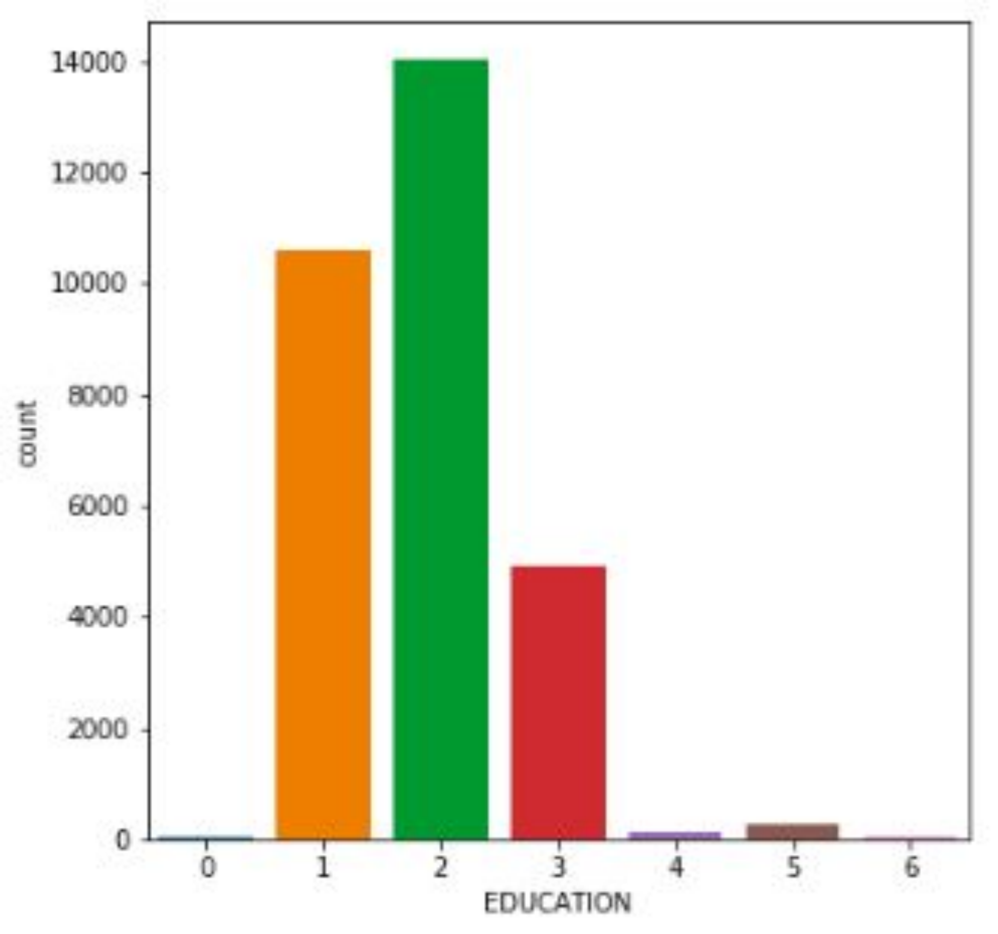
2	15964
1	13659
3	323
0	54 ← Missing Values



Categorical - Education

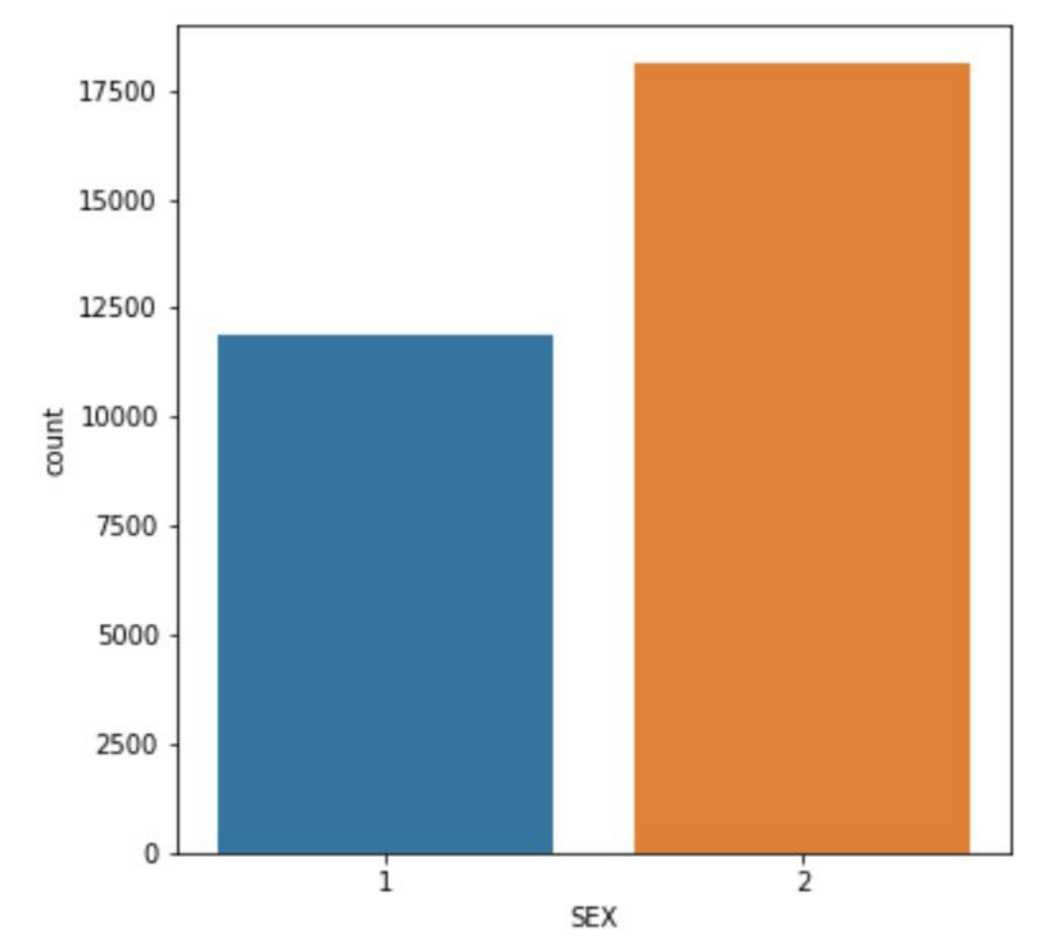
2	14030
1	10585
3	4917
5	280
4	123
6	51
0	14

← Missing Values



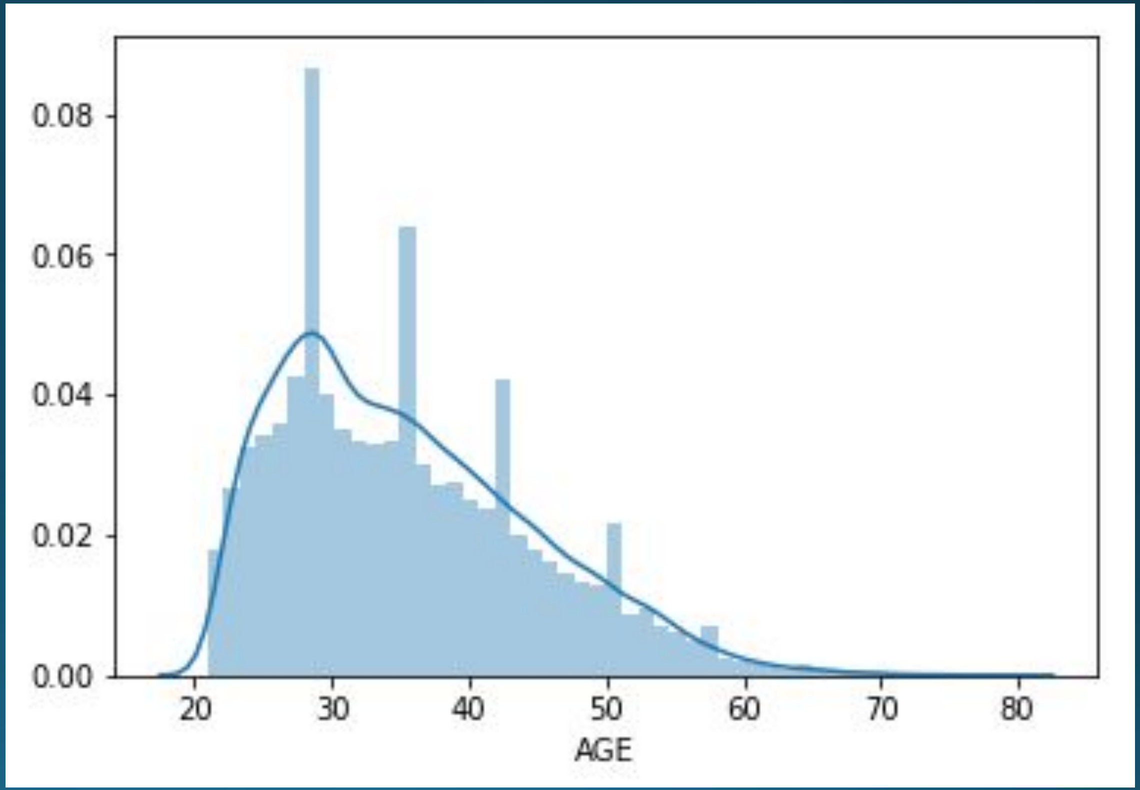
Categorical - Sex

2	18112
1	11888



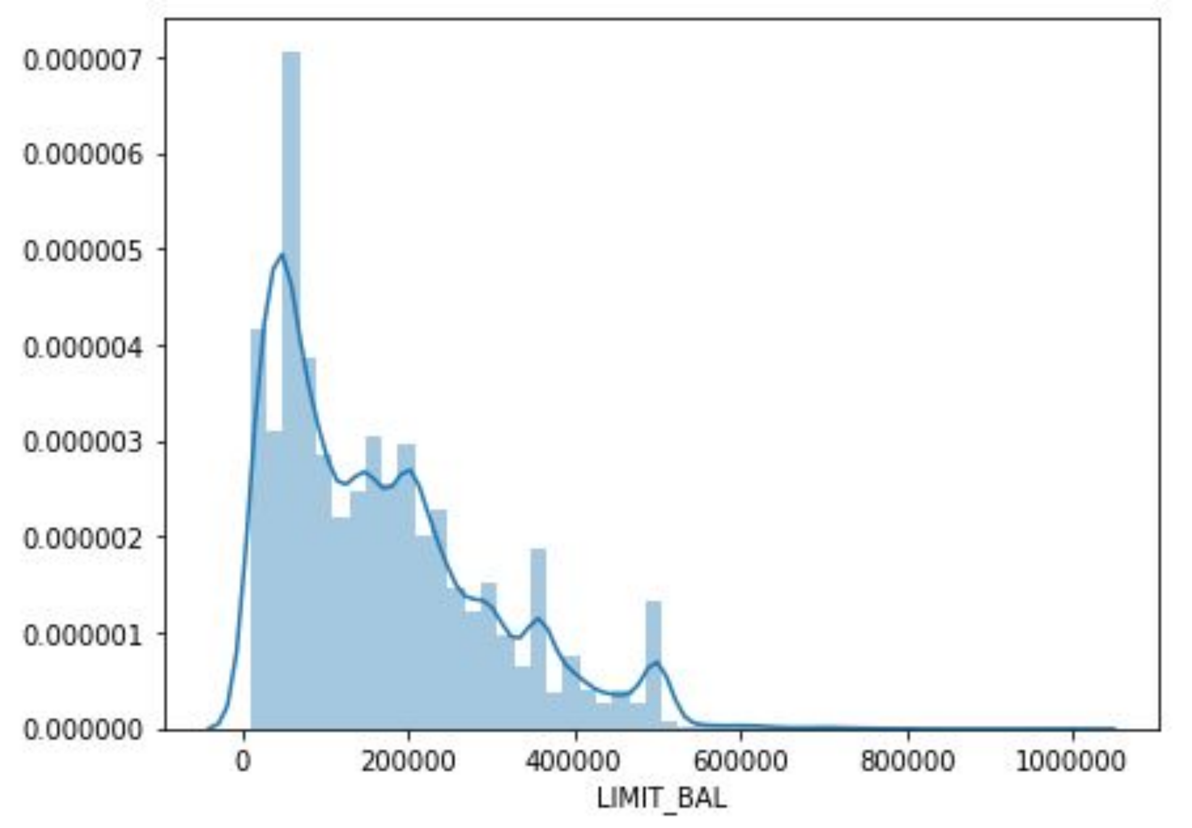
AGE

- Range
  - 21 - 79 years
- Mean
  - 35.485500
- Std
  - 9.217904



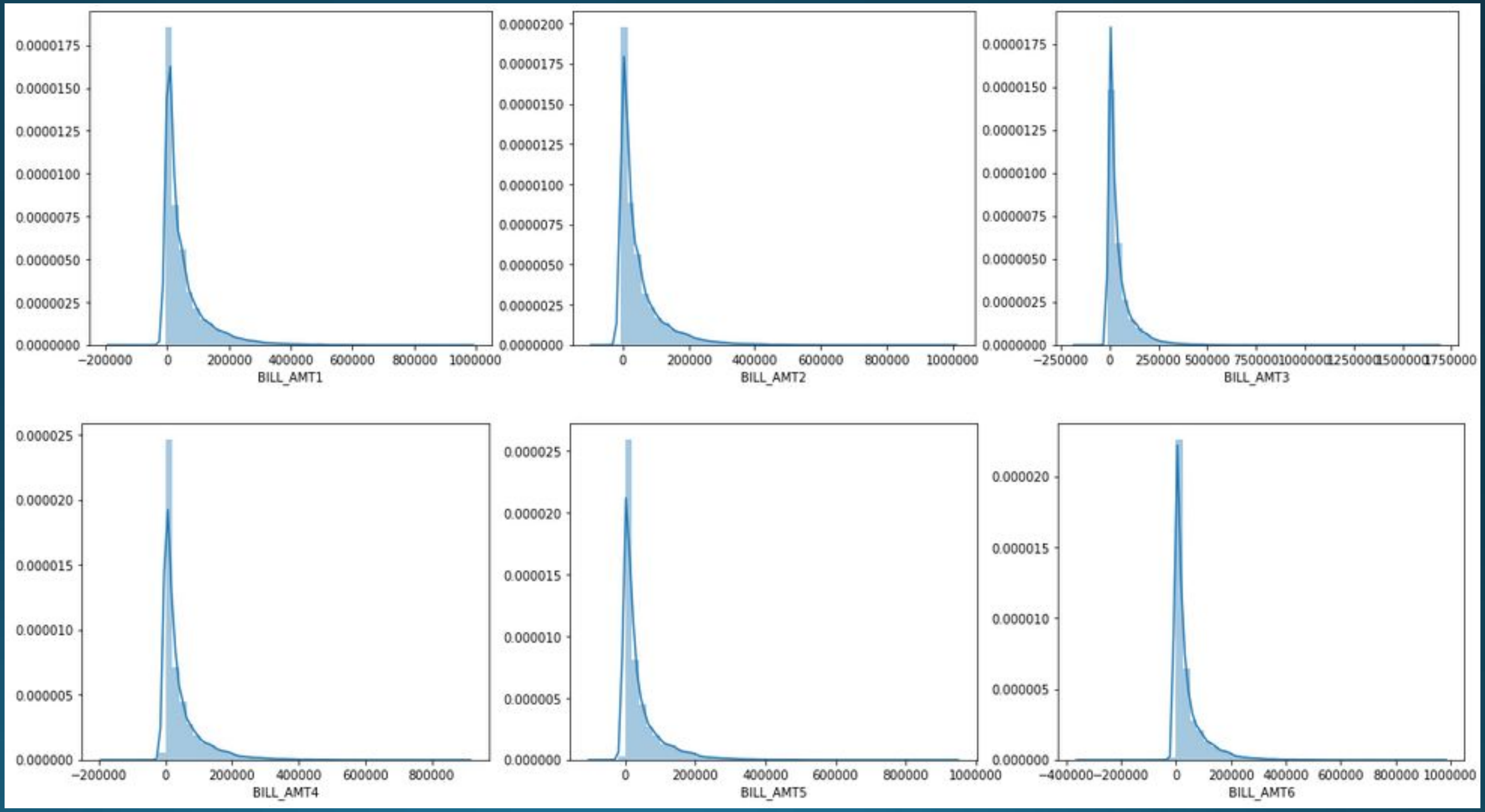
Limit Balance

- Range
  - 10000 - 1000000
- Mean
  - 167484.322
- Std
  - 129747.66



BILL_AMTx

- Continuous
- Contains both positive and negative values

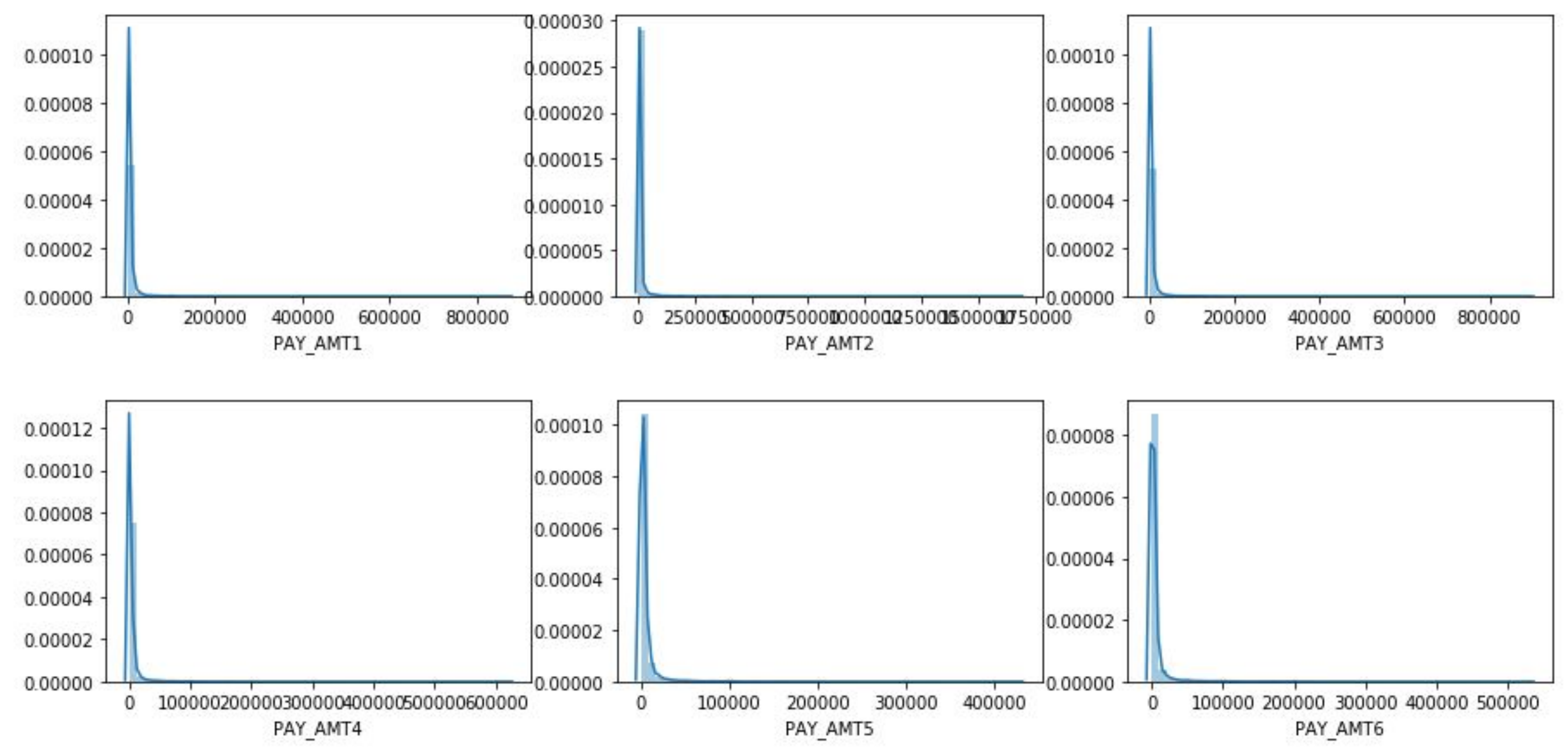


PAY_AMTx

Continuous

Positive values

Min - 0 (zero)



PAY_x

Values from

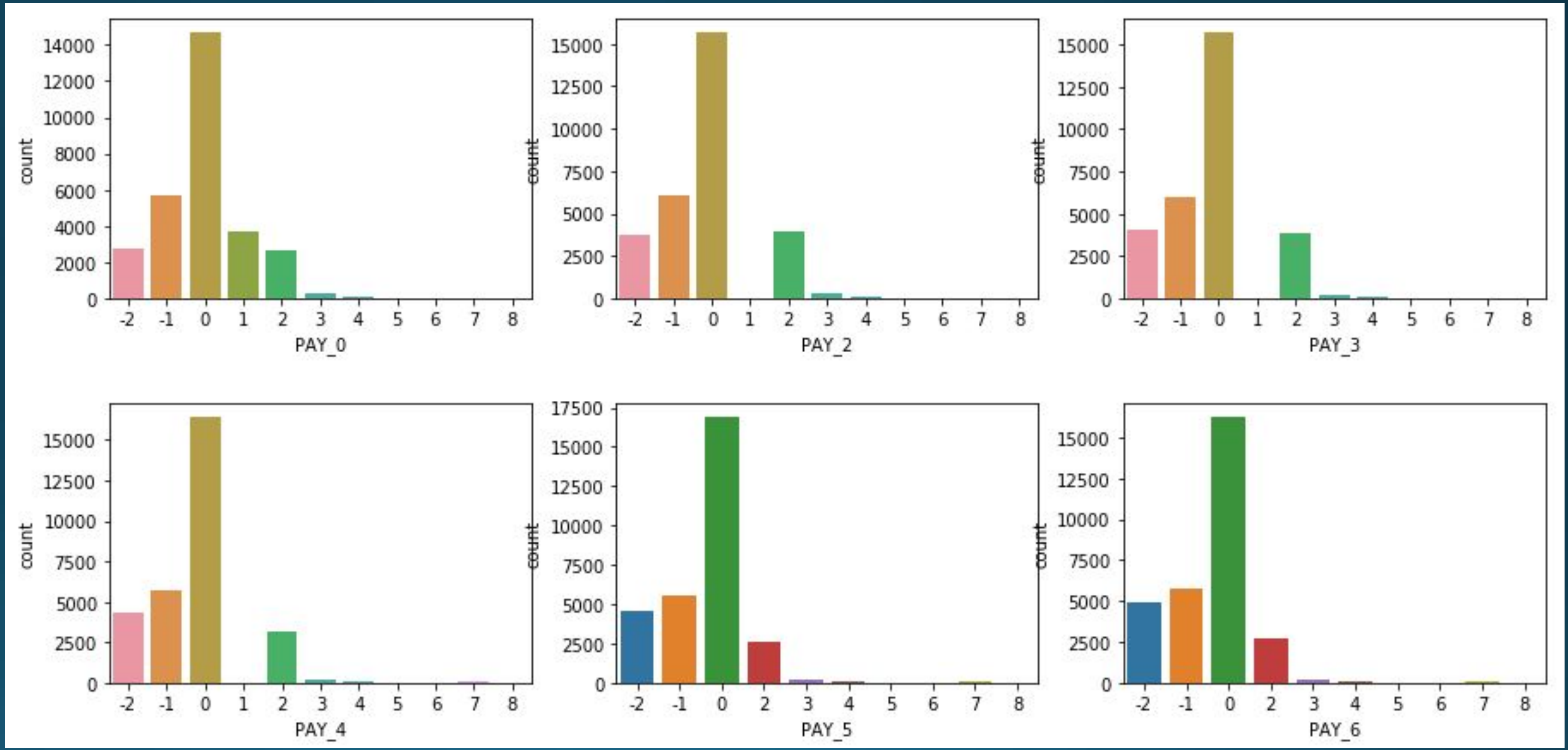
-2 to 8

Value - 0

(People paid

in same month

after due date)



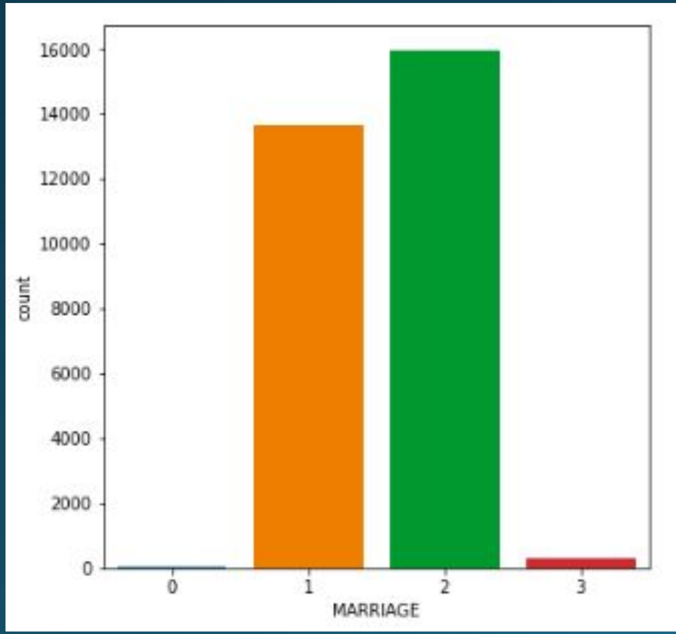
Findings

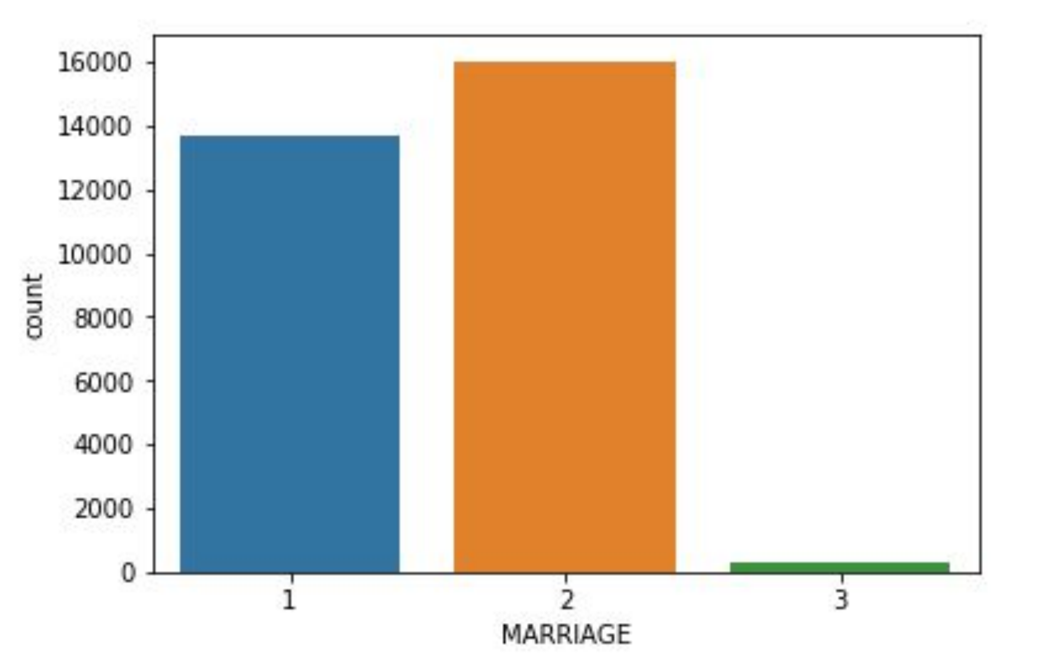
- No direct Missing Values
  - Zero NaN, missing values
  - For Features - Marriage, Education
    - Missing values are marked as 0 (zero)
- Relation between
  - Bill amounts of previous month <--> Pay amount of current month
  - If a previous month bill amount was high
    - Next month can be default

Feature Engineering

Imputation - MARRIAGE

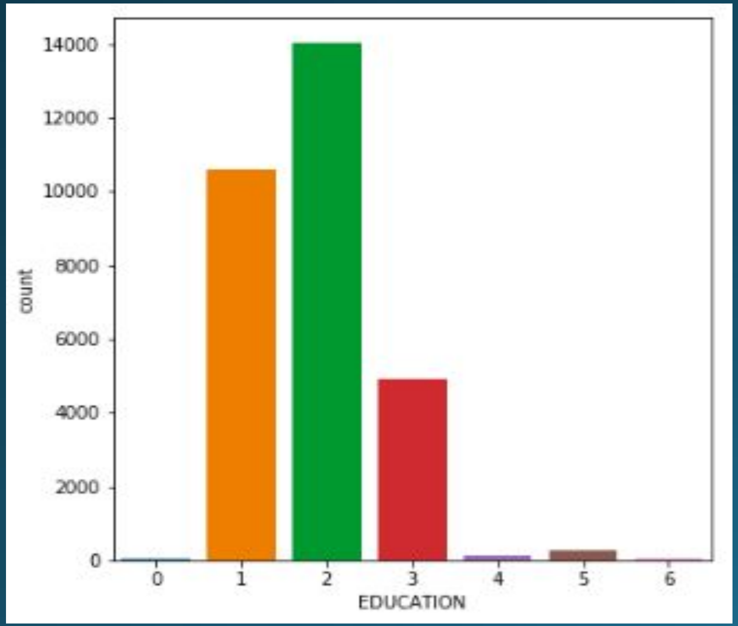
- MARRIAGE
  - 54 values are zero
  - We can impute this with mode

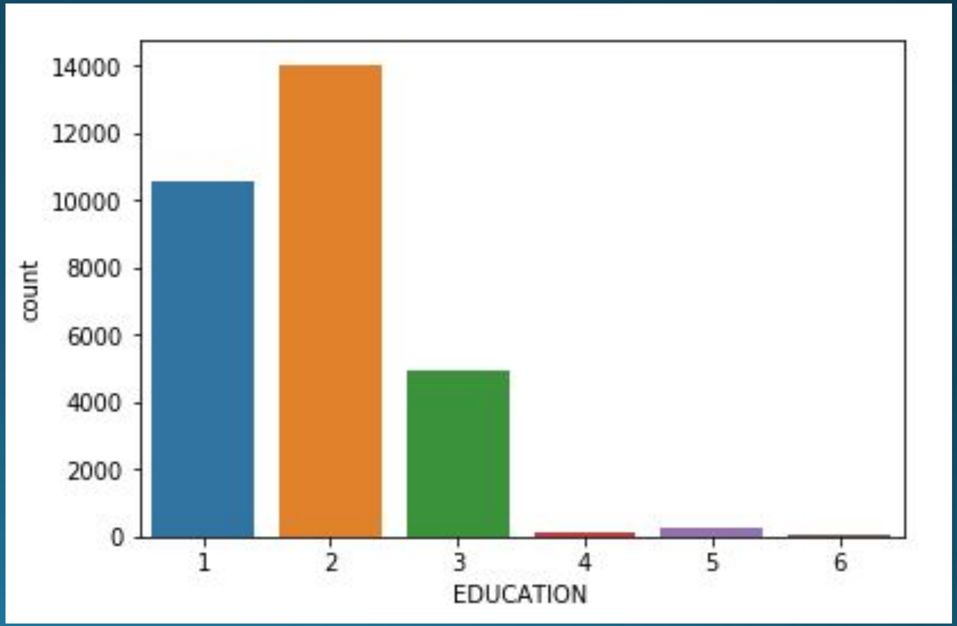




Imputation - EDUCATION

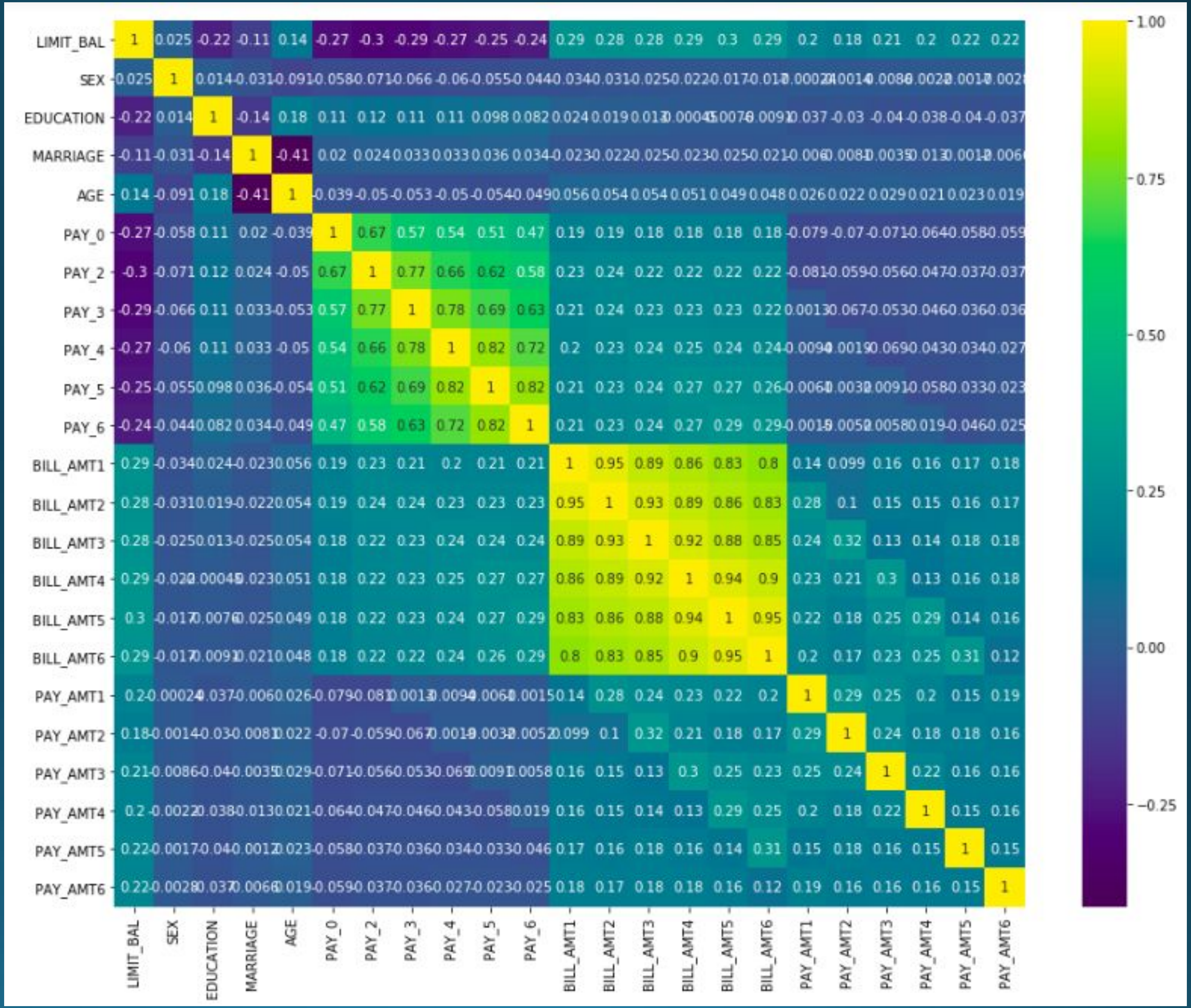
- EDUCATION
  - 14 values are zero
  - We can impute this with mode





Correlation

- High correlation among BILL_AMTx features
- High correlation among PAY_x features

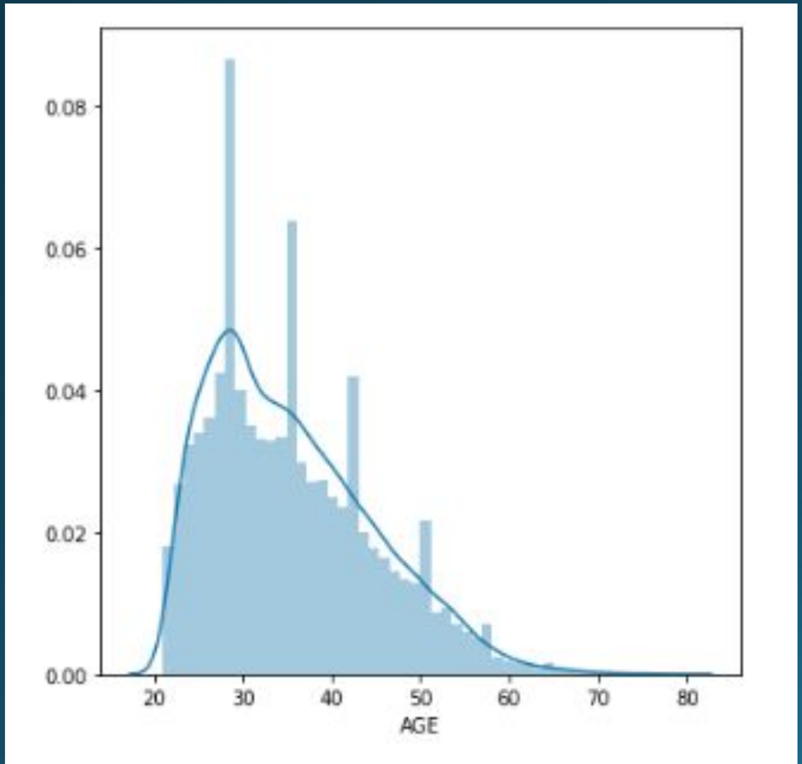


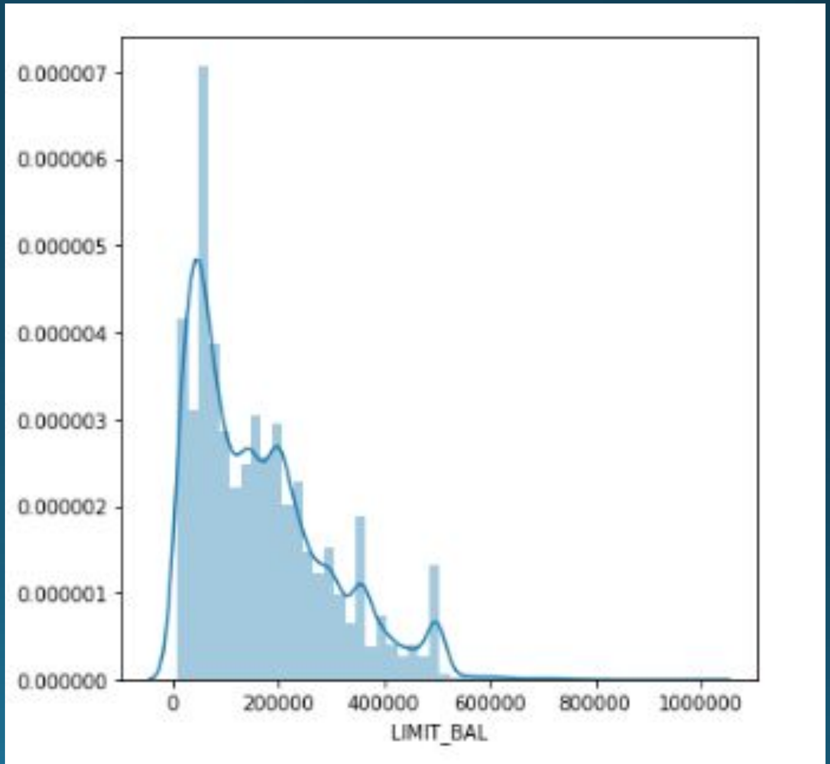
One Hot Encoding

- MARRIAGE, SEX, and EDUCATION are numeric values but these are really categorical.
- These can be one hot encoded.
- First converted to category features
- Next, pd.get_dummies to encode.

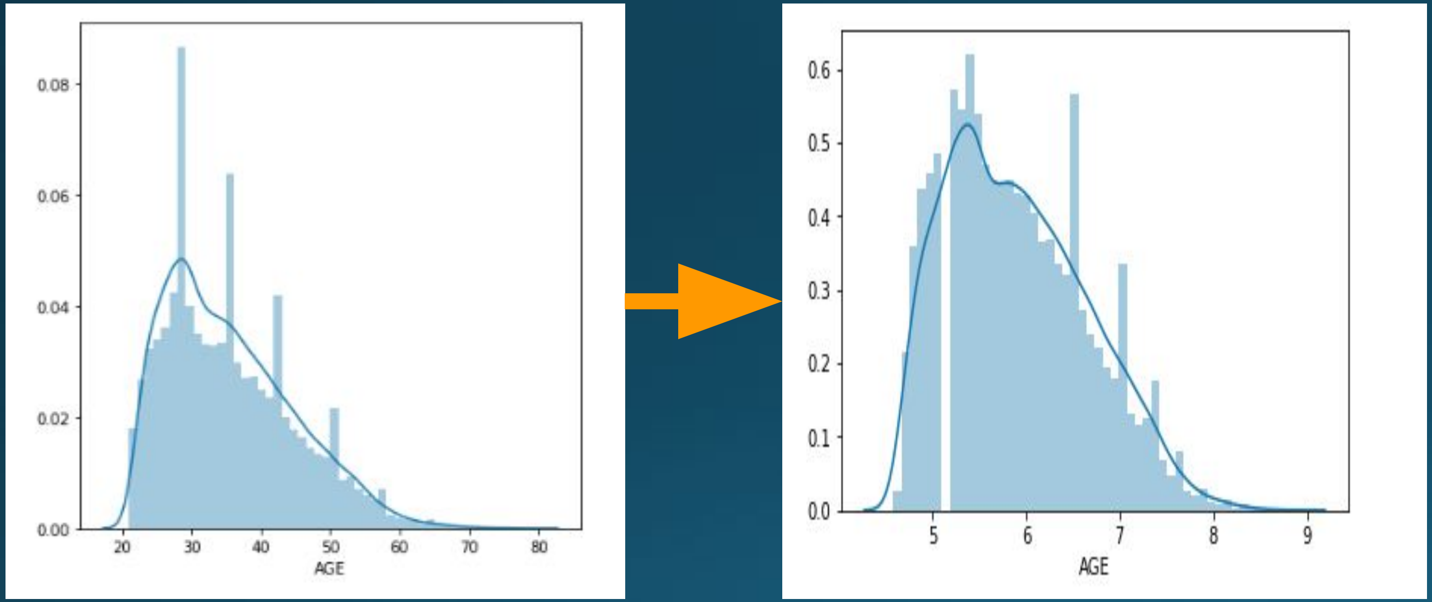
Skewness Handling

- For AGE and LIMIT_BAL



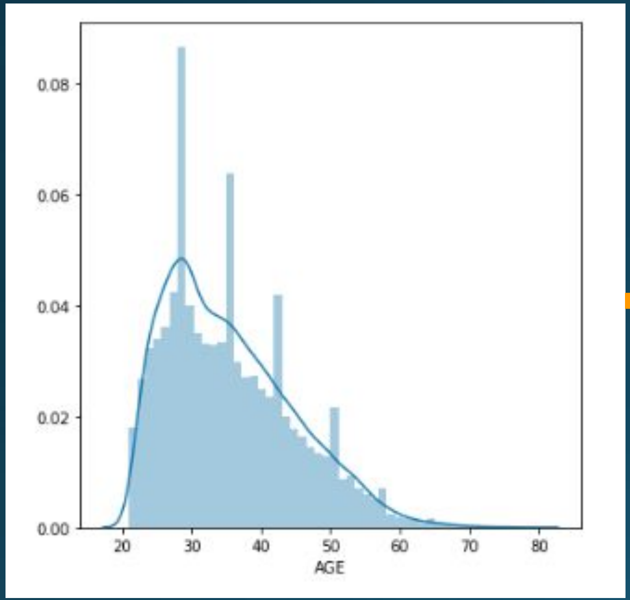


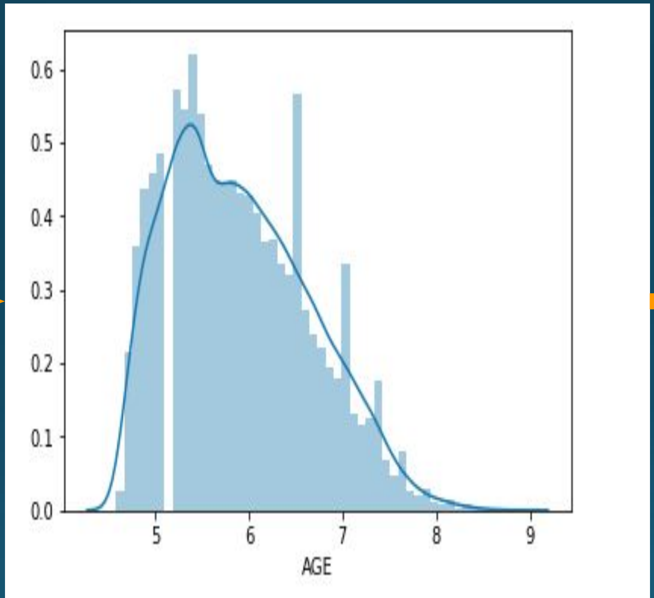
Transformation for AGE



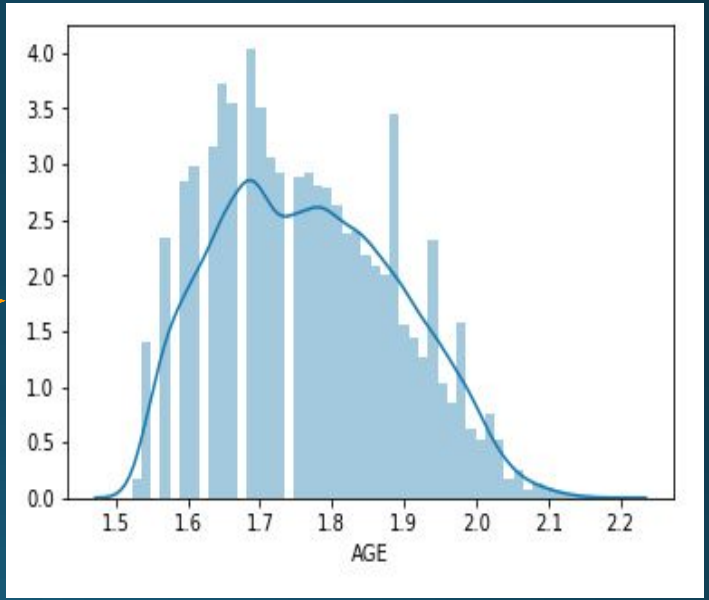
Square Root Transform

Transformation for AGE



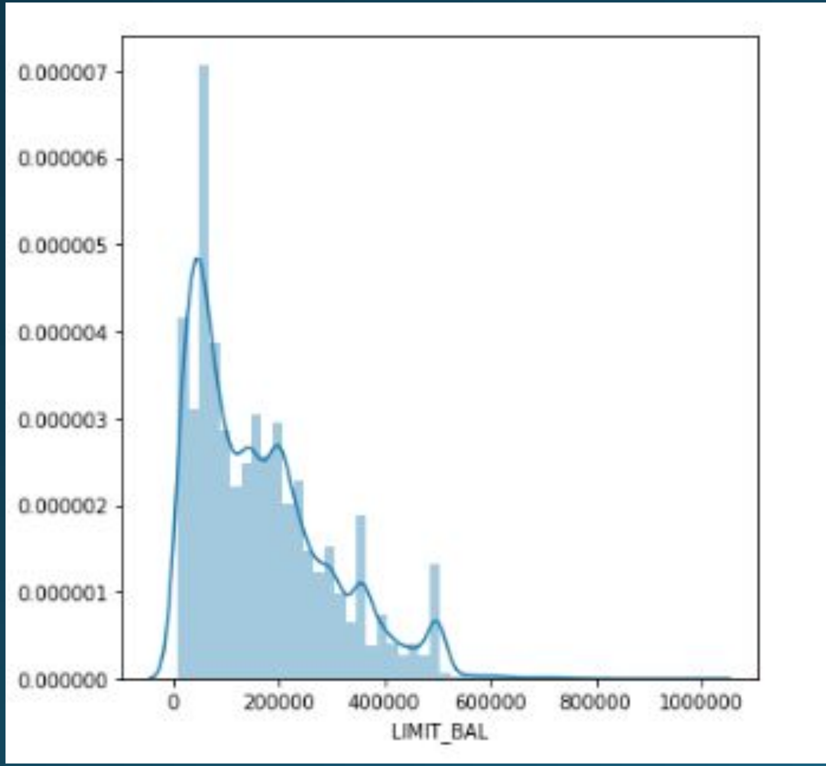


Square Root Transform

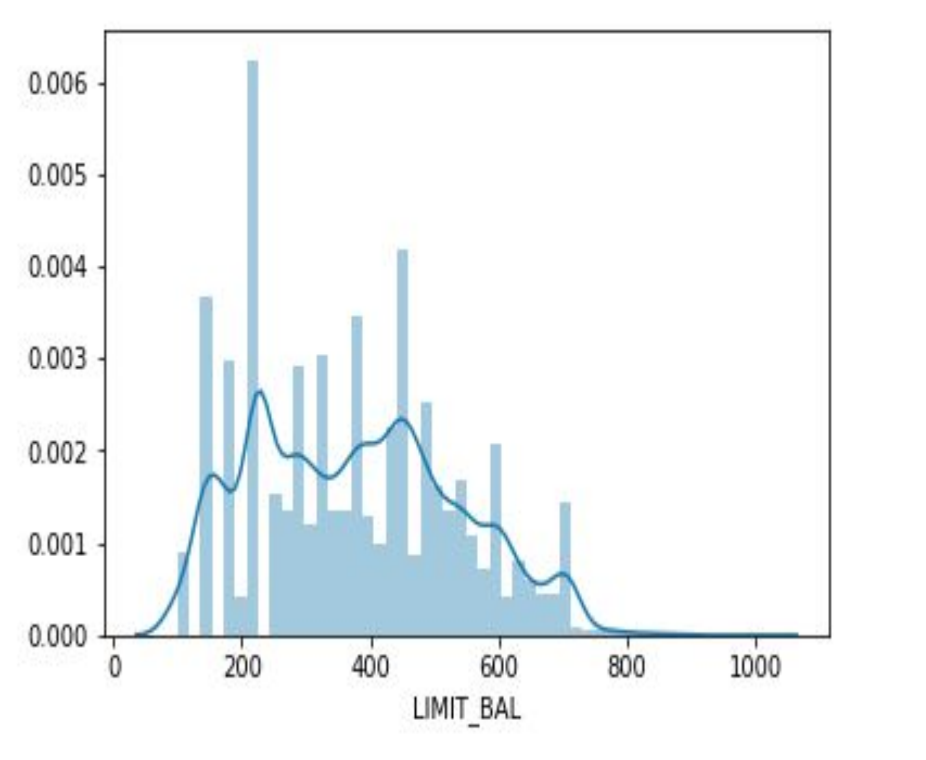


Log Transform

Transformation for LIMIT_BAL



Square Root Transform



New Features - Ratio

$$\text{Ratio} = \frac{\text{Pay Amount}}{\text{Bill Amount}}$$

(Higher the ratio, lesser the chance of Default)

Negative/NaN number handling:

1. If Bill Amount <= 0 : Then convert Ratio to positive
2. Impute NaN to 1

PCA

Apply PCA on highly correlated features:

1. Pay Amt
2. Bill Amt
3. Ratio

Variance Threshold: 95 %

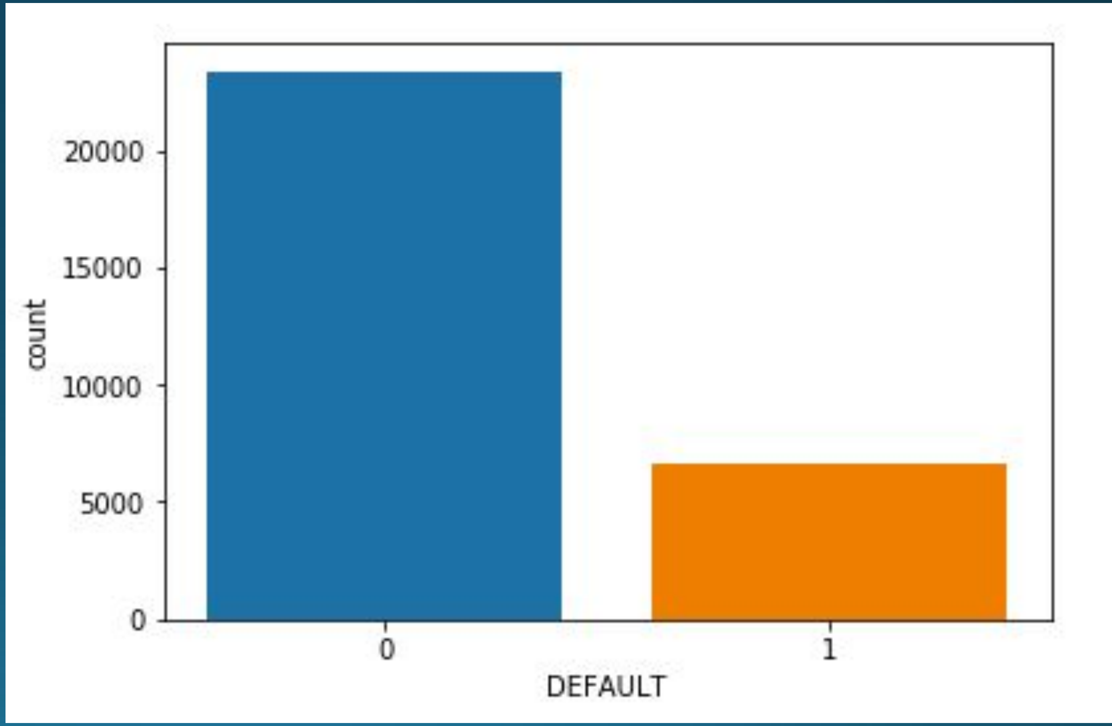
```
pca_df = doPCA(encoded_df, pca_cols, 95)

0 84.85143505128202
1 89.7216345283982
2 92.42966604981976
3 94.03553887387395
4 95.47703150574846
```

Model

Re-Sampling

- The target variable is not balanced.

- Possible ways of dealing:
  a. Under Sampling
  b. Over Sampling
  c. No Sampling



Winner: Undersampling

Model - Decision Tree Classifier

```
[130] dec_param_grid = {"min_samples_leaf" : [10, 100,500],
                  "splitter" :   ["best", "random"],
                  "criterion": ["gini","entropy"]
                }


      DEC = DecisionTreeClassifier()

      # run grid search
      grid_search_DEC = GridSearchCV(DEC, param_grid=dec_param_grid, scoring = 'roc_auc')
      grid_search_DEC.fit(X_under_sample, y_under_sample)
      print_model_params(grid_search_DEC)
```

```
Best Params: {'criterion': 'gini', 'min_samples_leaf': 100, 'splitter': 'best'}
Best Model Train Scores: {'accuracy': 0.7176233635448137, 'recall': 0.705337361530715, 'roc_auc': 0.7176233635448137}
Best Model Test Scores: {'accuracy': 0.7038666666666666, 'recall': 0.7079593058049073, 'roc_auc': 0.705326367604804}
Train Confidence Matrix:
[3624 1341]
[1463 3502]
Test Confidence Matrix:
[4096 1733]
[ 488 1183]
```

Model - Bagging with DTs

```
bag_param_grid2 = {"base_estimator__min_samples_leaf" : [10, 100,500],
               "base_estimator__splitter" :   ["best", "random"],
               "base_estimator__criterion": ["gini","entropy"],
               "n_estimators": [50,100,150]
              }


DTC3 = DecisionTreeClassifier()

BAC = BaggingClassifier(base_estimator = DTC3, oob_score=True)

# run grid search
grid_search_BAC = GridSearchCV(BAC, param_grid=bag_param_grid2, scoring = 'roc_auc')
grid_search_BAC.fit(X_under_sample, y_under_sample)
```

```
print_model_params(grid_search_BAC)
```

```
Best Params: {'base_estimator__criterion': 'gini', 'base_estimator__min_samples_leaf': 10, 'base_estimator__splitte
r': 'random', 'n_estimators': 100}
Best Model Train Scores: {'accuracy': 0.7609561752988048, 'recall': 0.704183266932271, 'roc_auc': 0.7609561752988047}
Best Model Test Scores: {'accuracy': 0.7394666666666667, 'recall': 0.655940594059406, 'roc_auc': 0.7091735601160388}
Train Confidence Matrix:
[4105  915]
[1485 3535]
Test Confidence Matrix:
[4486 1398]
[ 556 1060]
```

Model - Logistic Regression

```
lrg_param_grid = {"penalty" : ['l1','l2'],
                  "C": [0.2,0.4,0.6,0.8,1.0]}


LGR = LogisticRegression()

# run grid search
grid_search_LGR = GridSearchCV(LGR, param_grid=lrg_param_grid, scoring = 'roc_auc')
grid_search_LGR.fit(X_under_sample, y_under_sample)
print_model_params(grid_search_LGR)
```

```
Best Params: {'C': 0.6, 'penalty': 'l1'}
Best Model Train Scores: {'accuracy': 0.678147029204431, 'recall': 0.6561933534743203, 'roc_auc': 0.6781470292044312}
Best Model Test Scores: {'accuracy': 0.6797333333333333, 'recall': 0.6546977857570317, 'roc_auc': 0.6708040309811064}
Train Confidence Matrix:
[3476 1489]
[1707 3258]
Test Confidence Matrix:
[4004 1825]
[ 577 1094]
```

Conclusion

Tried without feature engineering - Almost same results

Possible Improvements:

1. Remove outliers
2. Extend Grid Search Parameters
3. Ensembling the three model from previous slides

Github Profile

https://github.com/kravigupta/Taiwanese_Credit_Card_Client_Fraud_detection