Outline:

1. Background and Objective

Building good personal or business credit is becoming increasingly significant in the contemporary financial world. A good credit is required for borrowers to borrow money from financial institutions or government for car loans, mortgage, school loan, investment, and to apply for personal credit card.

Regarding credit card applications, having a reliable credit is highly necessary. Paying bills on time and paying down balances on the credit cards are the most powerful steps one can take to raise a personal credit. Besides, the age of credit products, the number of credit cards held and applied, the utilization of credit cards, personal education level, housing, and salary etc., would altogether contribute to an applicant's credit, which are key factors for financial institutions, such as banks, to decide on the approval of the new credit card application.

Under the circumstance, this study aims to use the historical data containing one binary response and 20 predictor variables from credit card accounts for a hypothetical bank XYZ, to build classification models to decide on which applications should be approved. The model established would help the bank to make better decisions, which is not only effective in performance, but also cost- and time-efficient which reduces the tremendous efforts of the bank. Furthermore, the insights generated on each feature would guide further credit card issuing and management, which can significantly increase customer satisfaction and enhance the long-term revenue.

2. Bullet Points

Table 1. *Bullet Points Overview*

Num	Bullet Points
1.	Both train data and test data are imbalanced (defaulted: non-defaulted = 1:9)
2.	Three features ('pct_card_over_50_uti' etc.) have missing values
3.	Suspicious data detected regarding credit age, numbers of account due, etc.
4.	5 groups of numerical features are highly correlated
5.	7 discrete numerical features are right-skewed, while others are normally distributed
6.	Synthetic Minority Over-Sampling (SMOTE) is applied to balance the data
6.	Logistic Regression, Decision Tree, Random Forest, and Gradient Boost are compared
7.	Hyper-parameter tuning and model evaluation for Logistic Regression and Random Forest
8.	Feature interpretation and importance analysis

9.	Model future application by scoring and ranking

3. Exploratory Data Analysis

3.1 Imbalanced Label Column

The response variable is 'Def_ind', which indicates whether an account was defaulted or not after an account was approved and opened with bank XYZ in the past 18 months. Specifically, 1 represents defaulted, and 0 indicates not defaulted. The pie chart shows that the ratios of defaulted applications to the non-defaulted applications are 1:9, which demonstrates the imbalance of the data.

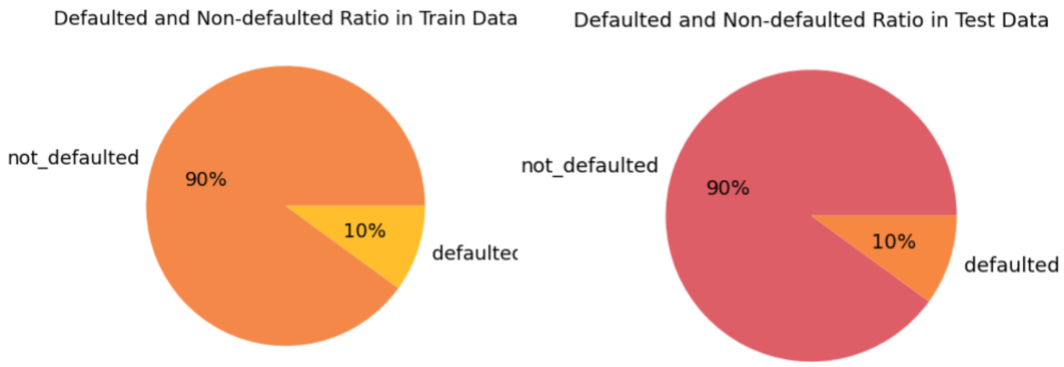


Figure 1. *Pie Chart of the Imbalanced Response Variable*

3.2 General Understanding of the Raw Dataset

To generally understand the data and features, the data size, feature types, missing values and duplicated values are investigated.

Table 2. *General Dataset Information*

Data Size	Feature Type	Missing Values
• Train: 20000 * 21 • Test: 5000 * 21	• 2 categorical features • 7 discrete numerical features • 11 continuous numerical features	• pct_card_over_50_uti: 1958 (9.8%) missing values in train, 489 (9.8%) missing in test • rep_income has 1559 (7.8%) missing values in train, 410 (8.2%) missing in test • rep_education has 1 missing value in train, 4 missing in test
Duplicated		
• No duplicated data		

3.3 Categorical Feature Analysis

There are 2 categorical features: **'ind_XYZ',** and **'rep_education'**. To better understand the effects of features on the response variable, the defaulted rate is proposed as the evaluation metric:

defaulted rate = number of defaulted applications/ number of total applications

1) Effects of Education Level on the Defaulted Rate

In the dataset, there are 4 types of education levels. The college occupies the largest portion (>50%) of all education levels, followed by high_school, graduate, and others.

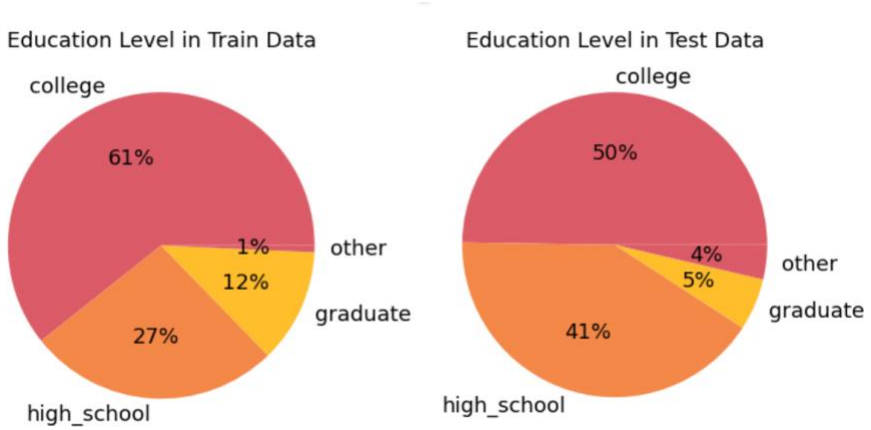


Figure 2. *Pie Chart of the Education Level Ratio*

The defaulted rates for each education level are shown in Figure 3. The defaulted rate decreases following the order of high_school (11.5%) > college (9.7%) > graduate (8.2 %) > others (7.7%) in the train data. Applicants with higher education levels seem to have lower chances of being defaulted. However, the model feature importance analysis (p10-11) shows that the education level isn't statistically significant on predicting the response variable. This answers the regulator's question that there is no discrimination against customers with low education backgrounds. The reason why low education customers having higher defaulted rates should be attributed to other confounding factors such as income, credit age, utilization of open cards etc. (Appendix-S1, p13).

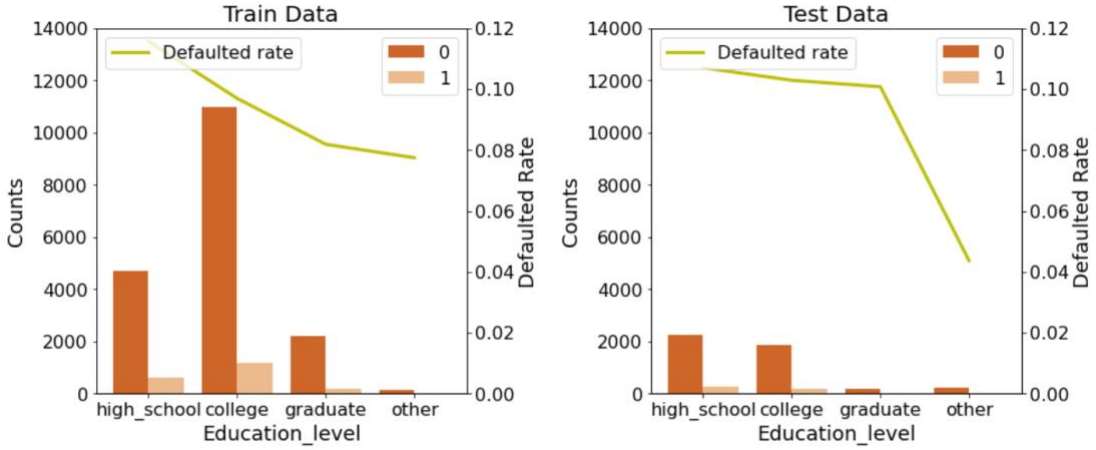


Figure 3. *Education Level and Defaulted Rate*

2) Effects of Having Accounts or Not on the Defaulted Rate

The data 'ind_XYZ' shows that 25% of applicants already have accounts. It is interesting to note that applicants who already have accounts in the bank experienced lower defaulted rates. However, the model feature important analysis shows whether having accounts or not isn't statistically significant on predicting the response variable. Therefore, the customers with accounts in the bank don't receive favourable treatment. The reason for the lower defaulted rates for those customers is probably attributed to their good credit record. The approval of their accounts previously indicates that they have already established good credit and background, which passed the selection of the bank at the first time.

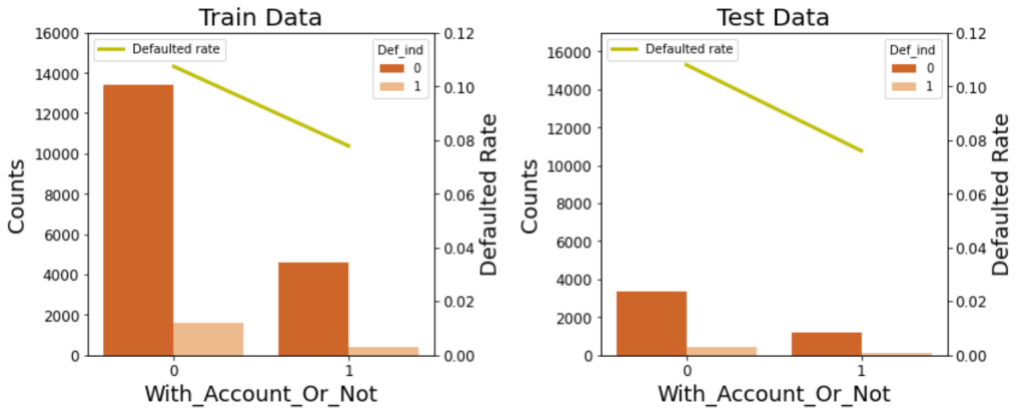


Figure 4. *With Account or Not and Defaulted Rate*

3.4 Numerical Feature Analysis

There are 18 numerical features, including 7 discrete numerical features (integer data type) and 11 continuous numerical features (float data type). Please see the details in Appendix-S3, p13.

1) Distribution of Discrete Numerical Features

A great number of the 7 discrete numerical features are zeros.

- num_mortgage_currently_past_due: 97.0% zeros
- num_acc_30d_past_due_6_months: 97.1% zeros

This is as expected because most applicants perform well in the credit history without delinquent accounts mortgage. Besides, the opening of new accounts and loans should not be very frequent at month-level for normal applicants. Therefore, those features should be 0 for most applicants with good credit records. The ratios of zeros accord with the 90% non-defaulted applications. The distribution indicates that those features are right-skewedly distributed.

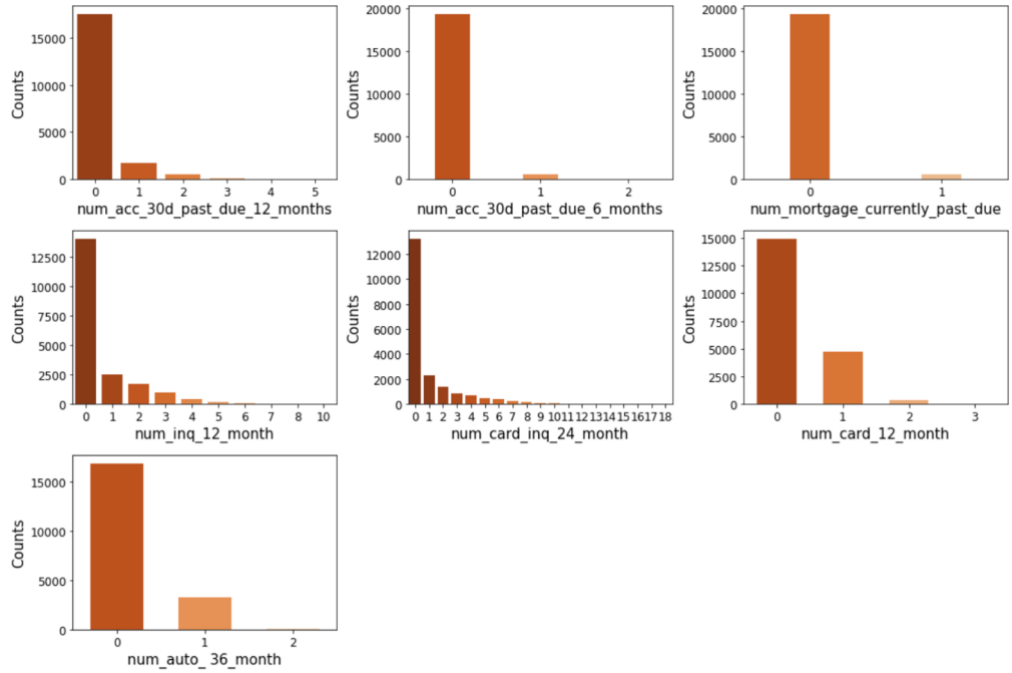


Figure 5. *Distribution of the Discrete Numerical Features*

To understand the effects of each discrete numerical feature on the response variable, the defaulted rates are calculated respectively. The defaulted rates increase with the increasing values of

each feature, although the higher values of each feature are minorities when zeros are the majority. This indicates that higher values of those features would negatively contribute to the approval of the credit cards, due to potential unreliability and difficulties in paying the bills.

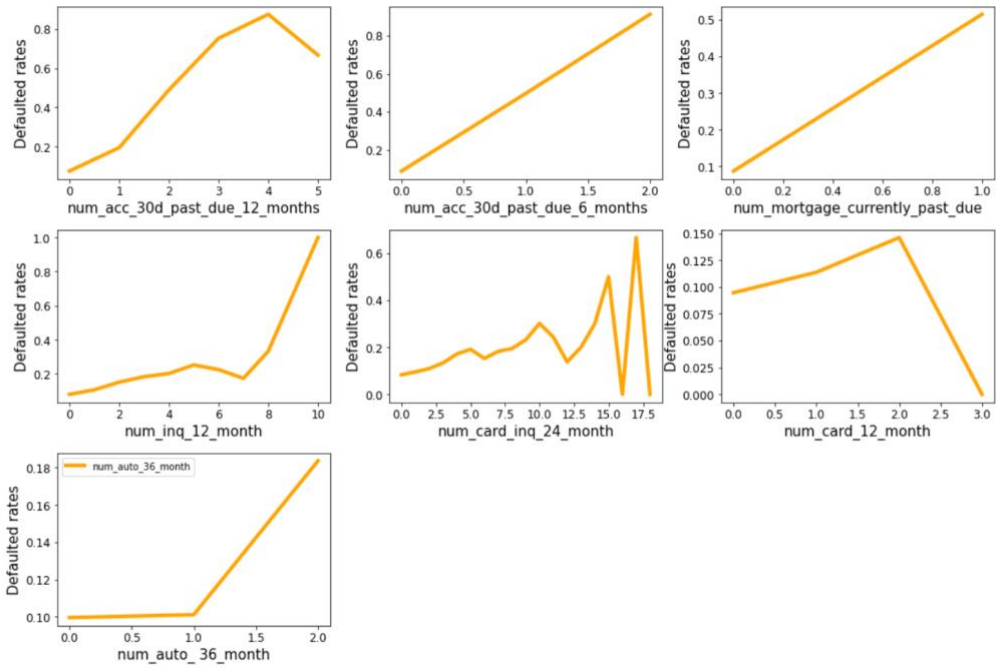


Figure 6. *Discrete Numerical Features and Defaulted Rate*

2) Distribution of Continuous Numerical Features

There are 11 continuous numerical features. No outliers are detected (Appendix-S4, p14). Except for 'tot_amount_currently_past_due', the others are normally distributed including the income, the utilization of cards and accounts, age of credit or credit card, etc. This accords with the normal credit behaviour of most good applicants.

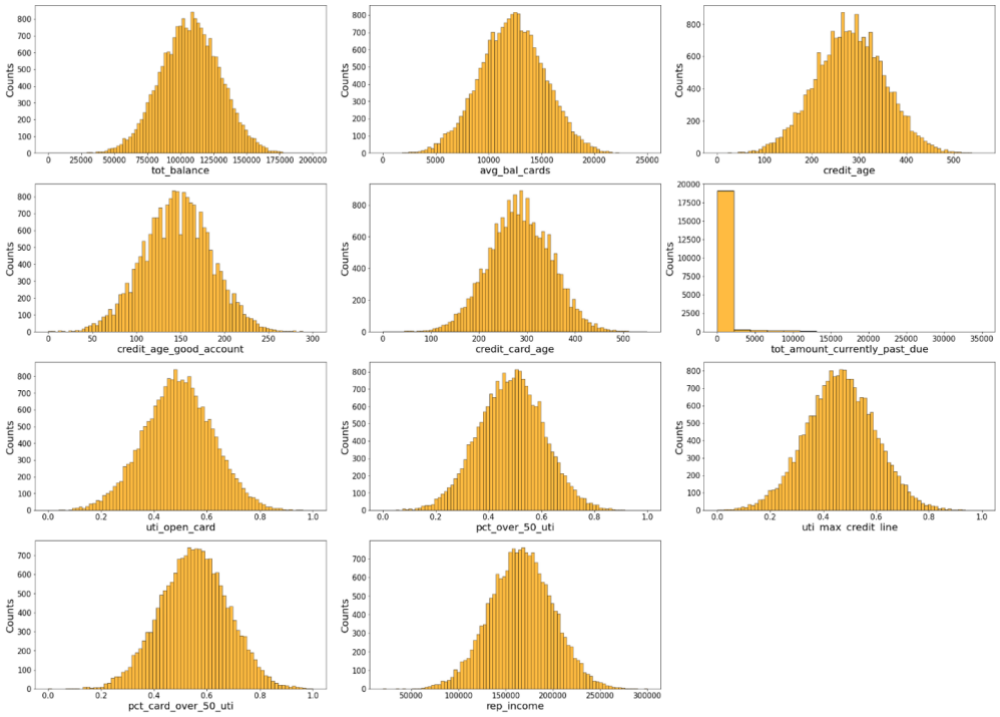


Figure 7. *Distribution of the Continuous Numerical Features*

3.5 Feature Correlation

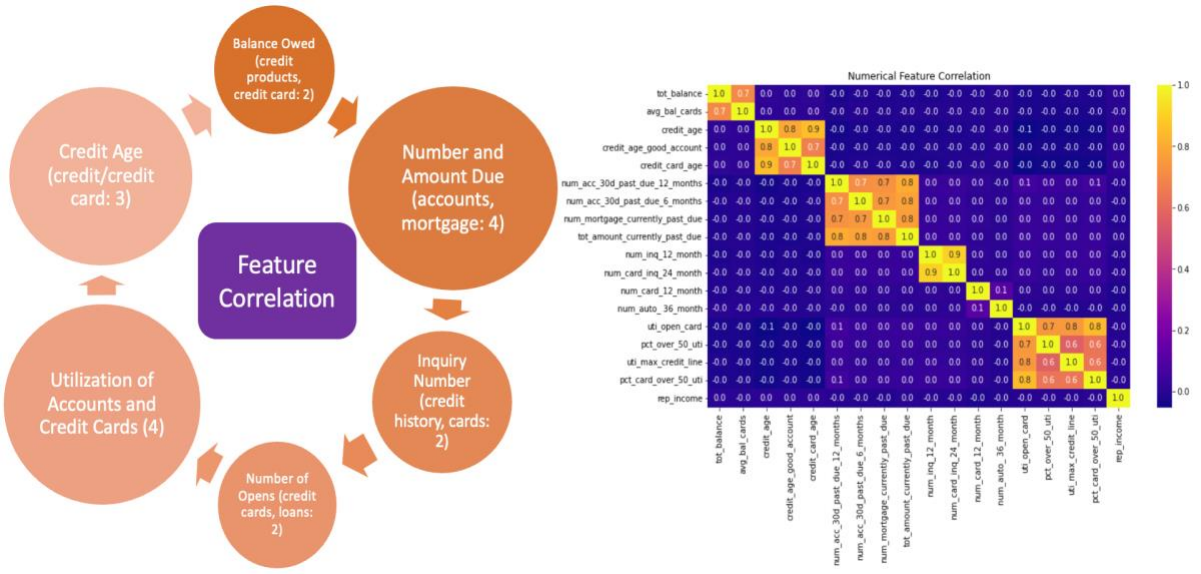


Figure 8. *Feature Correlation (Appendix-S5)*

Feature correlation based on the Pearson Correlation Coefficient is analysed to find the potential correlation and multicollinearity between variables. There are 6 groups of correlated features, i.e., balance owed, credit age related features, number and amounts due including accounts and mortgage, inquiry numbers, number of credit card and accounts open, and utilization of accounts and credit cards.

Correlated features can increase the complexity of models, cause overfitting problems, and increase the computation expense etc. Besides, highly correlated features and multicollinearity can reduce the stability of models, which induces more severe problems. Therefore, feature selection or other methods to avoid overfitting are required (Figure 9). In this study, to avoid information loss by dropping features, regularization and ensemble learning models are applied.

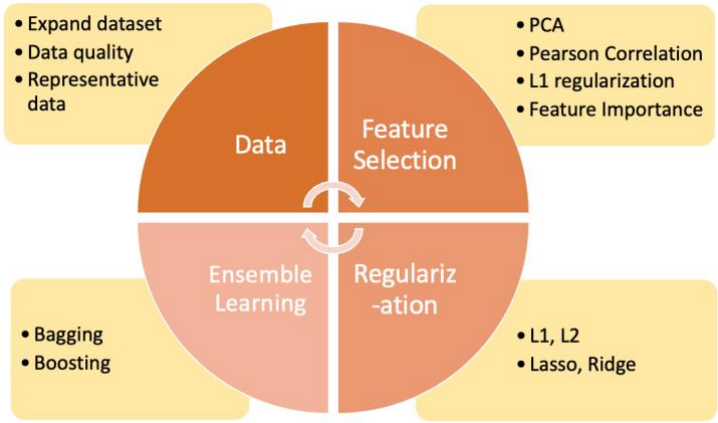


Figure 9. *Methods to Prevent Overfitting*

4. Data Pre-processing

4.1 Filling Missing Values

- pct_card_over_50_uti: 1958 (9.8%) missing values in train, 489 (9.8%) missing in test
- rep_income: 1559 (7.8%) missing values in train, 410 (8.2%) missing in test
- rep_education: 1 missing value in train, 4 missing in test

Since ‘pct_card_over_50_uti’ and ‘rep_income’ are normally distributed, mean values are used to fill the missing values. For ‘rep_education ‘, ‘other’ is used to fill the missing values.

4.2 Replacing Potential Incorrect Data

1) Credit age data: 'credit_age’ vs ' credit_card_age’ vs ‘credit_age_good_account'

It shows that 10809 (54.0%) ‘credit_age’ data are smaller than the ‘credit_card_age’. The ‘credit_age’ as delineated in the dataset description is ‘age in months of first credit product obtained by the applicant’. The credit product should include the credit card, thus the ‘credit_age’ should be equal or larger than the ‘credit_card_age’. Three potential reasons for this are proposed:

- Incomplete or incorrect data collection of the credit products
- The closure of some oldest credit cards, and ‘credit_age’ data collection can’t access the inactive credit product
- No utilization of the oldest credit cards, and ‘credit_age’ data collection doesn’t record it

Table 3. *Credit Age Data Check*

Scenarios	Count	Reason
credit_age < credit_card_age	10809 (54.0%)	Incorrect Data/System Issue
credit_age = credit_card_age	213 (1.1%)	Credit card is the oldest credit product
credit_age > credit_card_age	8978 (44.9%)	Loans, mortgages are older

The exact scenarios need further verification. Regardless, the credit card age should be incorporated into the credit age. Therefore, 54.0% of ‘credit_age’ data that are smaller than the ‘credit_card_age’ data are replaced with the ‘credit_card_age’ values.

2) Balance owed data: 'tot_balance' & 'avg_bal_cards'

3) Due account: 'num_acc_30d_past_due_12_months' vs 'num_acc_30d_past_due_6_months'

4) Credit inquiries: 'num_inq_12_month' vs 'num_card_12_month' vs 'num_card_inq_24_month'

Table 4. *Overall Data Check and Replacement*

Scenarios	Count (train/test)	Replacement
credit_age < credit_card_age	10809/3336	credit_card_age
tot_balance < avg_bal_cards	1/2	avg_bal_cards
num_acc_30d_past_due_12_months < 'num_acc_30d_past_due_6_months	0 /5	num_acc_30d_past_due_6_months
num_inq_12_month < num_card_12_month num_card_inq_24_month < num_card_12_month	3576/811 3384/744	num_card_12_month num_card_12_month

4.3 Numerical Feature Scaling

Scaling the features makes the flow of gradient descent smooth and helps algorithms quickly reach the minimized cost function. Without scaling features, the algorithm may be biased toward the feature which has values higher in magnitude.

- Min-Max Scaling is applied to the numerical features in the train dataset to standardize all numerical features to the range of 0-1.
- The scaler established is applied to the test data.

4.4 Categorical Feature Encoding

There are 2 categorical features, i.e., 'ind_XYZ' (0, 1), and 'rep_education' (college, high school, graduate, others). One hot encoding using ‘get_dummies’ is applied to both features. After encoding, there are 25 feature columns instead of 21 feature columns in the dataset.

5. Modelling

5.1 Model Selection with Cross Validation on Train Data (no hyperparameter tuning)

Based on the above analysis, binary classification models should be established. Four classification models, i.e., LogisticRegression, DecisionTreeClassifier, RandomForestClassifier, and GradientBoostingClassifier are compared through cross validation using the original imbalanced data. F1 score is used to evaluate the performance, because other scores ('accuracy', 'precision', or 'recall') can't comprehensively represent the model performance on the imbalanced data. Additionally, the time consumed for model establishment is also considered for model selection.

Table 5. *Comparing Models Based on F1 Score (Original Imbalanced Data)*

Model	Time Consumption	F1 score	Evaluation
Logistic Regression (LR)	3.1 µs	0.26	Mid speed, Low performance
Decision Tree (DT)	8.34 µs	0.33	Slow, Average performance
Random Forest (RF)	2.86 µs	0.36	Fast, Relative high performance
Gradient Boosting (GB)	3.81 µs	0.43	Mid speed, High performance

- Model performance: LR < DT < RF < GB
- Time efficiency: DT < GB < LR < RF
- Gradient Boosting performs the best on the original imbalanced data.

5.2 Model Selection with Cross Validation on SMOTED Data (no hyperparameter tuning)

To avoid the modelling issue using the imbalanced data, Synthetic Minority Over-Sampling (SMOTE) is applied to balance the data. SMOTE is generating new data points on the train dataset by interpolating new data on the line connecting every two adjacent data points. The interpolations are based on distance calculation. After SMOTE sampling, both defaulted and non-defaulted data have 18000 data points. Four classification models, i.e., LogisticRegression, DecisionTreeClassifier, RandomForestClassifier, and GradientBoostingClassifier are compared through cross validation using the SMOTED data.

Table 6. *Comparing Models Based on F1 Score (SMOTED Data)*

Model	Time Consumption	F1 score	Evaluation
Logistic Regression (LR)	1.91 µs	0.72	Fats, Low performance
Decision Tree (DT)	10 µs	0.86	Slow, Average performance
Random Forest (RF)	1.91 µs	0.94	Fast, High performance
Gradient Boosting (GB)	3.1 µs	0.83	Mid speed, Average performance

After SMOTE sampling, the models' performance is highly increased. This is attributed to the balanced dataset that allows higher precision and recall of the models. Regarding the time efficiency, all models witness faster speed except for the Decision Tree model. This model is slow because of the complicated tree structures built through feature prioritization, feature splitting, with relatively high computing depth (tree layers). Decision Tree model usually performs poorer than Random Forest and Gradient Boosting model, resulting in high variance or bias.

- Random Forest is selected as the machine learning algorithm due to the highest performance and the fastest speed.
- Random Forest and the Logistic Regression models will be compared in the next section.
- However, the extremely high F1 scores may indicate overfitting problems, which will be inhibited using regularization and hyper-parameter tuning in ensemble learning.

5.3 User Defined Function (UDF)

A helper function for printing out the gird search results is built, including using the Grid Search Cross Validation to select the model structure and hyper-parameters, the best parameters and model, making prediction on the test dataset, and presenting a variety of evaluation metrics, i.e., confusion matrix, roc_auc_score, F1 score, accuracy, recall and precision (Appendix-S6).

5.4 LogisticRegression Optimization

1) Regularization Term

There are two types of regularization penalty terms, L1 and L2. Both can help inhibit model overfitting, while there are differences between their application.

- L1 can generate very sparse results, which can be used for feature selection. However, the results are not stable.
- L2 tends to provide stable solution, which tackles the correlation problem well. However, they cannot be used for feature selection.

2) Hyper-parameter

The hyper-parameter C associated with the penalty term is selected from 0.001, 0.01, 0.1, 1, 10, 100, 1000. Both 'penalty' (L1 or L2) and C are provided as parameters for Logistic Regression Model optimization.

3) Model Evaluation

The results showed that the bests parameters are L2 regularization with C = 1.

Table 7. *Evaluation Metrics for the Logistic Regression Model*

Metric	Score
Roc_auc_score	0.79
F1_score	0.33
Accuracy	0.69
Recall	0.75
Precision	0.21

4) Results Interpretation

The statistical analysis shows that 7 features are statistically significant ($p < 0.05$) to predict the response variable, given the significance level is 0.05.

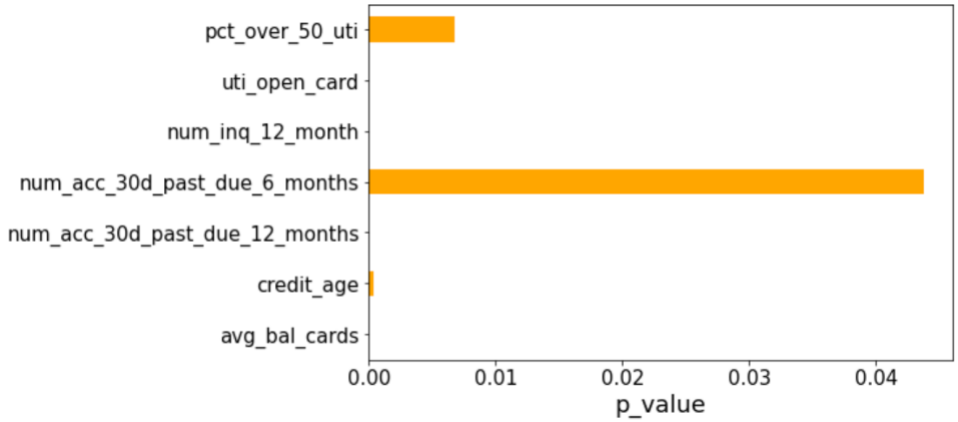


Figure 10. *Logistic Regression Feature Selection*

Table 8. *Logistic Regression Feature Interpretation*

Features	Correlation	Interpretation
avg_bal_cards	negative (-0.1)	Higher values, lower chance of default
uti_open_card	positive (0.2)	Higher values, higher chance of default
credit_age	negatively (-0.1)	Older age, lower chance of default
num_acc_30d_past_due_12_months	positive (0.3)	More due accounts, higher chance
num_inq_12_months	positive (0.1)	More inquiries, higher chance
num_acc_30d_past_due_6_months	positive (0.2)	More due accounts, higher chance
pct_over_50_uti	positive (0.2)	More frequent accounts, higher chance

5.5 RandomForestClassifier Optimization

1) Hyper-parameter

There are two hyper-parameters for the model:

- n_estimators: the number of weak learners, i.e., 10,50,100,500
- max_depth: the depth of each weak learner, i.e., 1,3,6,9,12

2) Model Evaluation

The results shows that the bests parameters are 100 trees, and the max_depth is 12.

Table 9. *Evaluation Metrics for the Random Forest Model*

Metric	Score
Roc_auc_score	0.84
F1_score	0.42
Accuracy	0.82
Recall	0.66
Precision	0.31

3) Results Interpretation

Feature importance is calculated to show the significant features on predicting the response variable. The higher the importance value and the ranking, the more important the feature.

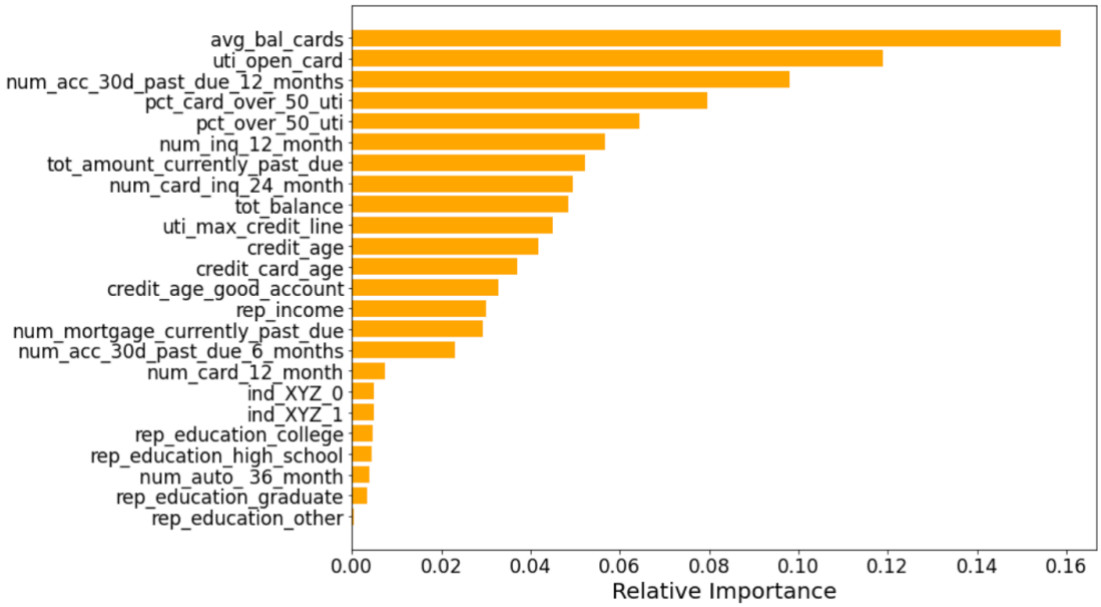


Figure 11. *Random Forest Feature Importance*

The demonstrated importance of the features is consistent with the Logistic Regression feature analysis on: 'avg_bal_cards', 'uti_open_card', 'num_acc_30d_past_due_12_months', 'pct_over_50_uti', 'num_inq_12_months', etc.

5.6 Comparing Logistic Regression Model and Random Forest Model

- Running time:

  Random Forest model has similar running time with Logistic Regression model at the current dataset size. However, as the dataset size increases, the computation cost may exceed that of the Logistic Regression model.

- Performance:

  Random Forest performs better than the Logistic Regression model. The roc_auc-score is increased by 6.3%, F1_score by 27%, accuracy by 18.9%, and precision by 47.7%. However, recall is decreased by 12 %. The trade-off between precision and recall needs further discussion with the institution. In general, higher recall is desired to reduce the money loss by identifying more defaulted applications. Meanwhile, manual monitoring should be included to control the false positive cases and ensure the customers' satisfaction. To increase the recall of Random Forest model, the power of the test can be increased by expanding the sample size.

- Flexibility:

  Random Forest model is more flexible than Logistic Regression model. The tree-based models don't require feature scaling; they are non-linear model, which requires less assumptions compared to the Logistic Regression model; the ensemble learning can automatically inhibit overfitting problems.

- Interpretability:

  Random Forest model has feature importance analysis, which enhances its interpretability. In comparison, Logistic Regression models can be explained by the coefficient analysis.

Overall, I select the Random Forest model for credit approval, due to the relatively fast speed, the high performance, the model flexibility, and interpretability.

6. Model Application

As discussed, the Random Forest model is selected for future credit approval. There are 2 steps to use it to make decisions on future credit card applications:

1) Comparing values of the significant features with provided standards and trends:

Features	0	1
avg_bal_cards	12346	11192
uti_open_card	0.48	0.57
credit_age	300	279
pct_over_50_uti	0.47	0.55

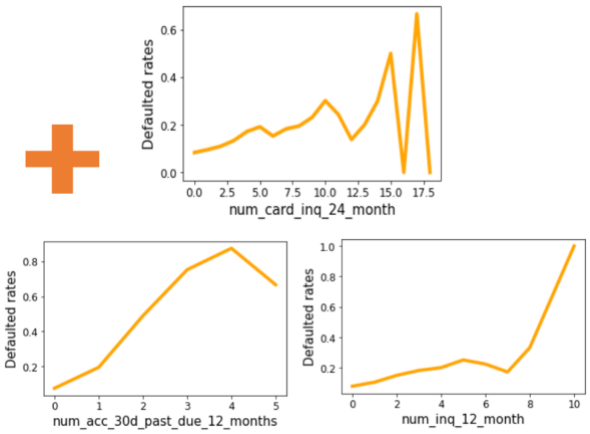


Figure 12. *Significant Feature Comparison*

For instance, if an application has large values of average balance owed for all credit cards (avg_bal_cards), frequent utilization of open credit card accounts (uti_open_card), small credit_age, great percentage of accounts with over 50% utilization, and large numbers of card inquiry

(num_card_inq_24_month), due accounts (num_acc_30d_past_due_12_month) and credit inquires, it is highly possible that the application will be defaulted therefore the model should reject it. This is a quick but approximate way to make estimations.

2) Modelling and Ranking:

Feed the new application data into the model and predict the probability of being defaulted (Def_ind= 1). The probability can be converted to a ranking score ranging from 1 – 10, which can be categorized into 3 types. The decision is based on the score obtained.

Table 10. *Ranking Score for Credit Card Approval*

Color	Ranking Score	Decision
Green	1-3	Approve
Yellow	4-7	Manual Investigation
Red	8-9	Decline

7. Rejection Explanation

If a credit card application is rejected using the model, I would like to take several steps to explain to the customer:

1) Before conversation:
   - Double check if the model works right, whether the data regarding this customer is correctly collected and fed into the model
   - Check manually and compare the feature values of the customer's data with the provided standards and trends of those defaulted applications.
   - If everything is correct, then find out the potential reasons for the rejection, such as the customer may have blemishes on his/her credit report. The customer may have unpaid balance, or due accounts, or too high credit card utilization ratio, or too short credit age, or too many accounts and credit cards applied, etc. Or it could be something else.
2) During conversation:
   - I would talk to the customer patiently and explain the reasons in easy-understanding language. For example, if the customers' credit age is too short, I would suggest him/her to wait until being qualified or applying for other more suitable card. If the rejection is due to unpaid bills, I would suggest working paying off the bills and set-up securer and more reliable payment methods to avoid forgetting about paying the bill again.
3) After conversation:
   - I would organize a copy of the credit report of the customer to help him/her better understand the situation.
   - I would summarize the detailed suggestions and send them to the customer later.

Appendix

S1. Whether model discriminates against customers with low education backgrounds

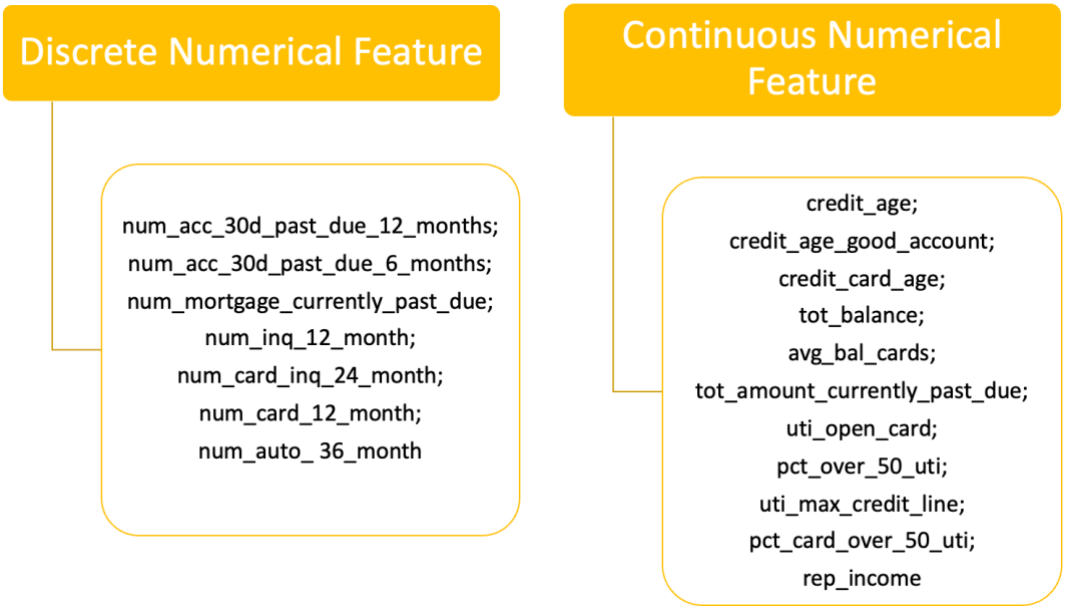
Please see the detailed answer highlighted in Page 3. There is no discrimination against customers with low education backgrounds. The reason why low education customers having higher defaulted rates should be attributed to other confounding factors such as income, credit age, utilization of open cards etc.

Education Level	Mean Income	Mean Credit Age	Card Utilization
High school	165747	296	0.50
College	166581	298	0.48
Graduate	166885	301	0.48
Others	163428	294	0.49

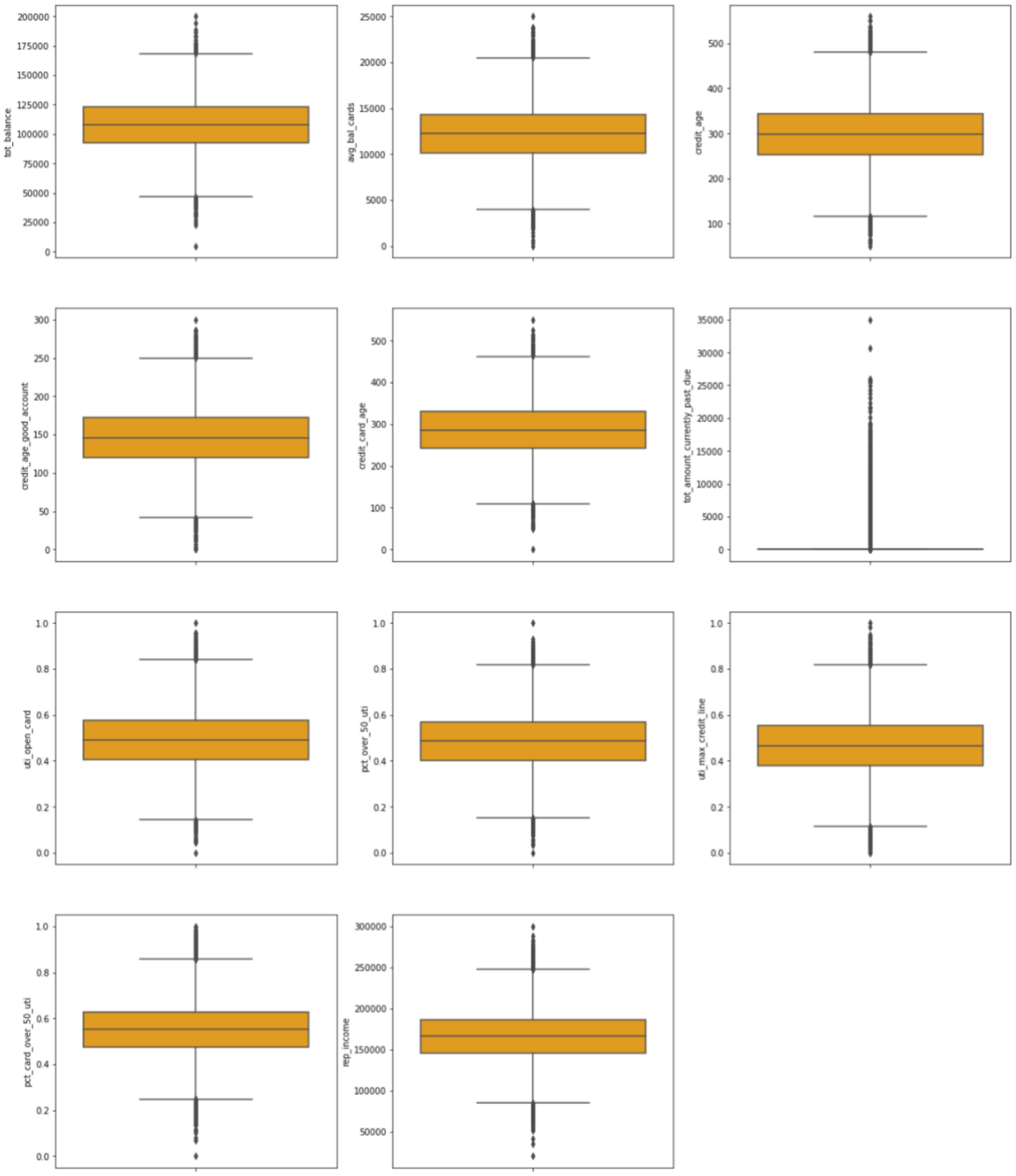
S2. Are customers who already have an account with the financial institution receive favourable treatment in the model?

Please see the detailed answer highlighted in Page 3. Therefore, the customers with accounts in the bank don't receive favourable treatment. The reason for the lower defaulted rates for those customers is probably attributed to their good credit record. The approval of their accounts previously indicates that they have already established good credit and background, which passed the selection of the bank at the first time.

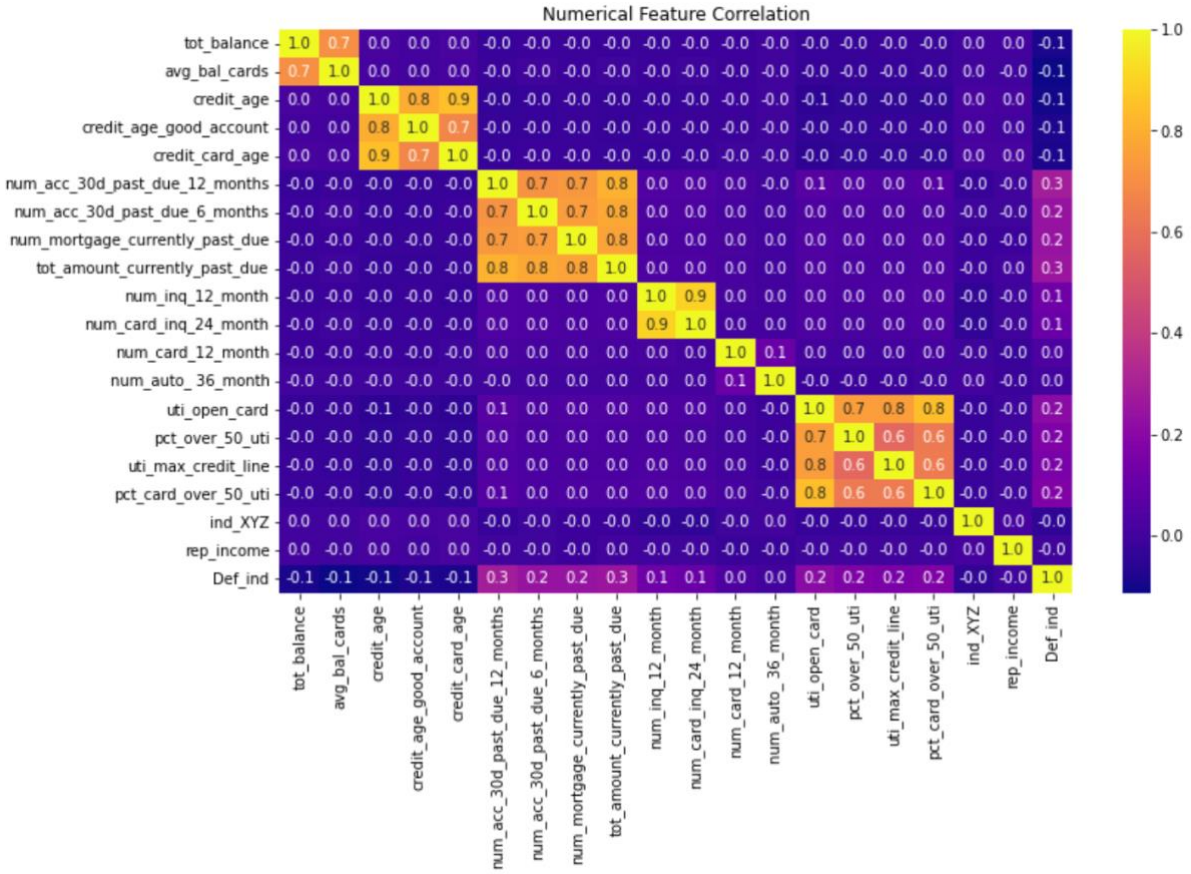
S3. Table *Numerical Feature*



S4. Figure *Numerical Feature Box Plot*



S5. Figure *Feature Correlation Heatmap*



S6. UDF: Helper function for printing out grid search and test data prediction results

```
# Helper function for printing out grid search results
def grid_search_wrapper(model, parameters, refit_score='f1_score'):
    """
    fits a GridSearchCV classifier using refit_score for optimization(refit on the best model a
    prints classifier performance metrics
    """

    grid_search = GridSearchCV(model, parameters, refit=refit_score,
                               cv=4, return_train_score=True)
    grid_search.fit(X_train_sm, y_sm)

    print('Best params for {}'.format(refit_score))
    print(grid_search.best_params_)

    # make the predictions
    y_pred = grid_search.predict(X_test)
    y_prob = grid_search.predict_proba(X_test)[:, 1]

    # confusion matrix on the test data.
    print('\nConfusion matrix optimized for {} on the test data:'.format(refit_score))
    cm = confusion_matrix(y_test, y_pred)
    cmDF = pd.DataFrame(cm, columns=['pred_0', 'pred_1'], index=['true_0', 'true_1'])
    print(cmDF)

    # scores
    print("roc_auc_score is: ", roc_auc_score(y_test, y_prob))
    print("f1_score is: ", f1_score(y_test, y_pred))
    print("Accuracy: ", accuracy_score(y_test, y_pred))
    print("recall = ", recall_score(y_test, y_pred))
    print("precision = ", precision_score(y_test, y_pred))

    return grid_search
```