# LENDING CLUB LOAN DEFAULT PREDICTION BASED ON MACHINE LEARNING TECHNIQUES

**Zian Jiang**
New York University
zj444@nyu.edu

**Yanbing Wang**
New York University
yw2115@nyu.edu

**Denglin Jiang**
New York University
dj1369@nyu.edu

**Zeng Cheng**
New York University
cz1951@nyu.edu

December 9, 2019

**ABSTRACT**

In this work, we will analyze previous personal loan data and try to figure out the relevant features to the default of those loans on the online platform called the Lending Club. The data is from Lending Club website. Based on this analysis, we want to predict whether a future applicant should be accepted or rejected a personal loan. For our initial exploratory data analysis we will use several techniques including Principal Component Analysis (PCA), L1 based linear SVC feature selection, and Random Forest feature selection. Then we will do binary classification task using Logistic Regression, Naive Bayes, Random Forest and XGBoost models. To deal with skewed data here, we will apply SMOTE (Synthetic Minority Over-sampling Technique) here, to receive best model accuracy of 90%. We will find out that loan grade, loan terms and many more factors concerning borrower's financial status are the most important indicators.

## 1 Business Understanding

Giving loans to individuals, compared to larger established businesses, can be a risky decision for financial institutions because of the potential high default rates, thus personal loans can be hard to get for many individual borrowers. Moreover, loan sharks that are willing to lend mostly have outrageously high interest rates that not everyone can afford. Finding a balance, such as a transparent third-party online platform, is therefore important. Lending Club is one of the platforms available. As the world's largest peer-to-peer lending company, it works as the bridge that connects borrowers and investors. Borrowers receive the full amount of the issued loan deducted an origination fee paid to the company, and investors purchase notes backed by the personal loans and pay Lending Club a service fee.

The two main sources of operating revenue of Lending Club, as noted in the company's latest 10-K, are transaction fees, 1% - 6% of the loan amount, paid by the borrowers and 1% service fees paid by the investors. To attract both parties into the business, it is of great significance for the p2p loan platforms like Lending Club to establish an effective model for the prediction of a borrower's likelihood of default and offer competitive risk-adjusted opportunities for the loan investors. As we know, loan application can be biased in many ways, and the best indicator of whether or not to reject a loan should be based on the history of clients with similar backgrounds. We hope to study the data available and be able to utilize the data mining model to predict the loan status given the available information of the loan. If our model works well, it can be deployed to help loan application managers in similar positions with a machine learning based recommendation before they make the final decisions. As a result, It will contribute to the bottom line of the business by optimizing the decision making at the first place.

## 2 Data Understanding

We used the 2018 loan data set retrieved from the Lending Club site. Each borrower of Lending Club would have completed a comprehensive application regarding their past financial history, personal information, the reason for the loan, and more. The information of approved loans will then be shared by the company on the website for the viewing of investors. The 4 quarters combined data set of the year 2018 includes 495,250 borrowers. Each data point is one approved individual or joint loan associated with a unique party of borrower. The approved loan data set we used contains the borrower's personal information including job titles, home ownership status, annual income, and the loan-related information such as term, past payment, and the current status of the loan. Before feature engineering, there are total 150 features in the original data set.

We have three particular goals with this data set. Firstly, we want to use unsupervised techniques to detect any potential trends in the data. This will help us with a general sense of the data set. Then, we want to identify features that affect the loan status. Finally, we want to use these relevant features identified on the previous steps to predict whether or not a new loan application should be approved based on its information.

### 2.1 Deciding on Target Variable

Among the 150 features, the only feature that reflects our main interest of predicting the likelihood of default is the "loan status", thus being our target column. There are 6 different status of the loans: "Current", "Fully paid", "Charged Off", "Late (16-30 days)", "Late (31-120 days)", "In Grace Period", and "Default". The status of "Default" indicates no payment for more than 121 days but still has a small chance of paying. However, the loan labeled as "Charged Off" has no reasonable expectation of future payments. After filtering out the loan status that is still current, we end up with 121,570 data points. Therefore, our data mining problem becomes a binary classification task predicting whether a loan will be "Fully Paid" (the loan has been fully paid within the deadline) as 1 or "Charged Off" (the loan for which there is no longer a reasonable expectation of future payments) as 0. After deciding on our target variable, we simply visualized the percentage of each loan status, and found out that our data set is rather skewed.

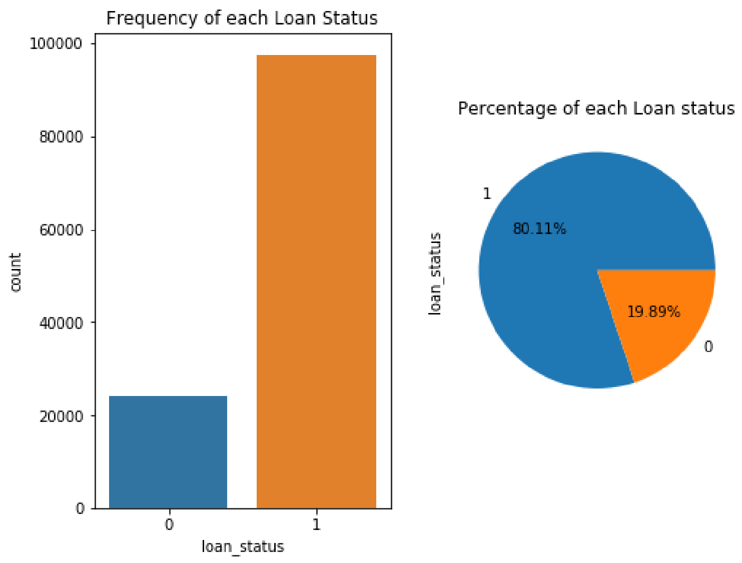


Figure 1: Frequency of Each Loan Status and Percentage of Each Loan Status

### 2.2 Data Leakage

As preparing the features for further modeling, data leakage is a serious problem to be addressed. We noticed that a large number of features in the data set contains information that won't be available at the point of modeling. For example, the funded amount and the outstanding principal are features of loans that are already being funded. These kinds of features will not be available for a new loan applicant. The features that leak information from future are carefully removed during our data cleaning process.

### 2.3 Selection Bias

Another problem to be considered of our dataset is the selection bias. As explained before, the loan data we got from the Lending Club are all approved loans - the loans predicted by the Lending Club's previous evaluation models to be "Fully Paid" and then approved. Therefore, our dataset contains no information of the other applied loans that were rejected by the Lending Club and thus not shared in the website. As the result, our model would only make our predictions based on the loans that previously defined as "more likely to be paid" but ignore the loans that previously defined as "less likely to be paid" and rejected. However, we will never find out the real label (loan status) of those rejected loans anyway.

## 3 Data Preparation

### 3.1 Manual Feature Engineering: Data Leakage, Irrelevant Features, and Co-linearity

First of all, the dataset also includes ongoing loans that we do not know the outcome of, thus we need to remove these data from our training set. Also, data leakage and irrelevant information are factors we need to consider: a data point contains information that will not be available in testing, thus causing the model to overfit. We need to remove these features before proceeding further. Some example features that fall into either of these two categories are Member ID (irrelevant for our purpose) and funded amount (data leakage from the future). Next, notice that the dataset contains features regarding secondary applicants (a family applying for a loan). However, as the plot below shows, these features regarding secondary applicant are highly correlated to those regarding primary applicant. Out of over 120 thousands data points we have, only one sixth of them applied with more than one person, so only one sixth has secondary features. Given these two reasons, we deem it reasonable to remove features regarding secondary applicant that are more than 75% correlated with primary applicant features.

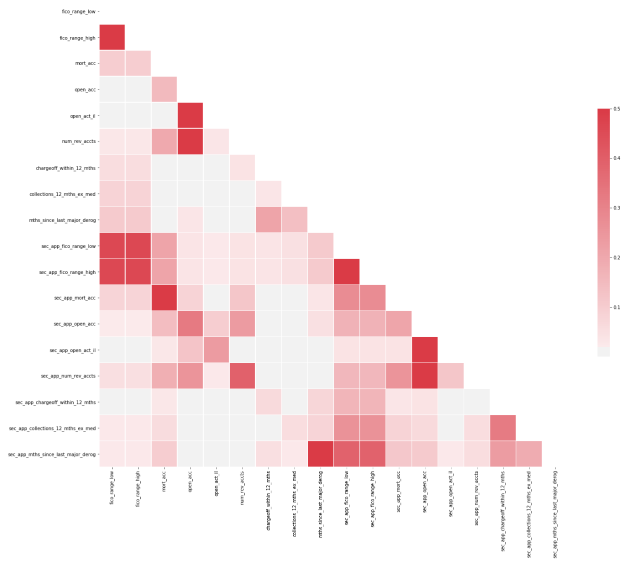


Figure 2: Heatmap of Secondary Applicant Related Features

### 3.2 Manual Feature Engineering: Composite New Features

Also, initially, when applicants applies for a loan, the Lending Club gets their credit score from FICO, and they are given a lower and upper limit that their scores belong to, which are then stored in columns "fico_range_low" and "fico_range_high". After the initial screening, any updates to the FICO score are recorded as "last_fico_range_low", and "last_fico_range_high". Since FICO scores continue to be updated by LendingClub after a loan is funded and the Lending Club may have updated those values after the borrower's application, we can not use them as features for prediction. However, this leaves us with only FICO scores information about the applicants from the beginning: "fico_range_high" and "fico_range_low". To better capture the overall credit score evaluation of a borrower, we create a new feature, "fico_average", which is the average of "fico_range_low" and "fico_range_high".

### 3.3 Fill Missing Values and Mark Imputed Missing Values

Further, we remove features that have more than one percent with missing values (1215 rows). Then, it follows naturally that we need to fill in the NaN values for the rest of our dataset. Here, for numerical features, we use the mean values, and most frequent for categorical features with the SimpleImputer class from scikit learn library.

### 3.4 Categorical Feature Encoding: Label encoding, One-hot Encoding and Data and Time Features

Particularly, Date and Time is one group of features that we need to manually engineer based on our domain knowledge of the topic. For any loan, we believe it is very possible that the earliest credit line date, labelled as earliest_cr_line in our dataset, would affect loan status. Moreover, for our dataset, the earliest credit line date could be traced back to January 1950. Thus, we can turn all date features into the number of days since January 1950.

Next, we use either label encoding or one-hot encoding for all of the rest categorical features. These two encoders are parts of the Sk-Learn library in Python, and they are used to convert categorical data, or text data, into numbers, which our predictive models can better understand. Here we adopt label encoding for ”loan grade” and other features that bear inherent ordering, and apply one-hot encoding to the rest.

### 3.5 Feature Reduction: PCA, L1 Linear SVC, Random Forest

After the above feature engineering with our particular domain knowledge, we have successfully selected 49 features from the original 150 features, of which 11 are c

99 features in total, 15 of them being categorical and 84 numerical.

Still, 99 is a large number of features to model with and we would want to further select features using unsupervised techniques.

Since we have adopted one-hot encoding for many of our categorical features, we now have a relatively sparse dataset to work with. Following this sparsity, we want to first try using PCA for feature reduction. It turns out that selecting the first 10 principal components will capture 17% of the variance of our data. What's worse, even the first 20 principal components only capture 37% of the total variance. The figure 3 below shows the first and second principal components. Unsurprisingly, the two classes seem to be densely clustered when we plot the first two principal components. This gives us the conclusion that PCA fails our dataset.

On the other hand, we can also try to use random forests as a feature selection method. More specifically, this is possible since random forests naturally rank the features by how well they improve the purity of the nodes. Also, since the greatest decrease in impurity starts at the beginning of the tree and the lowest at the bottom of the tree, if we prune our trees below a particular node, we end up with a subset of the most important features. Scikit learn has enabled us to do this conveniently with just one line of code. Figure 4 below is the graph of the most 10 important features selected by Random Forest.

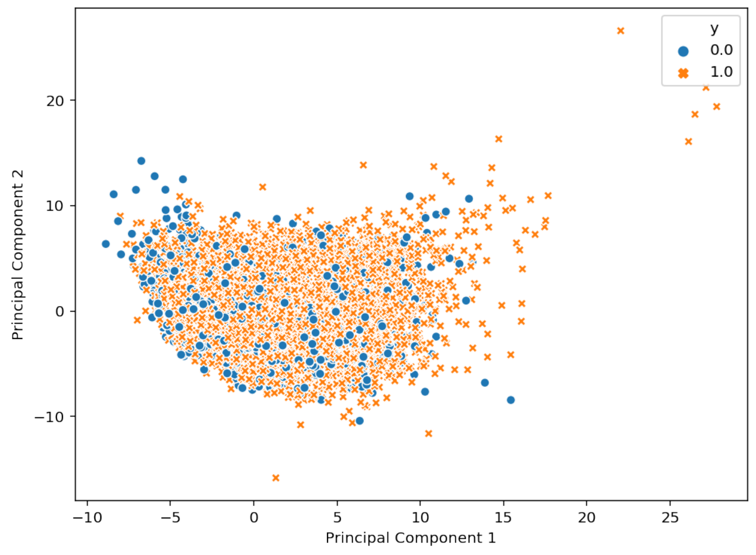


Figure 3: Two Major Principal Components from Cleaned Data Set

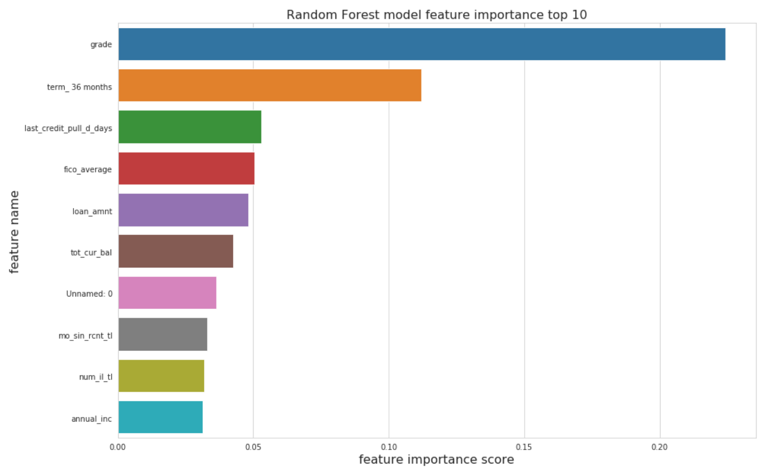


Figure 4: 10 Most Important Features Selected by Random Forest

After feature selection we end up with 49 features, which is much more manageable. Out of the three methods of feature selection: PCA, random forest, and L1-based Linear SVC, given the baseline model, feature selection with random forest gives the best result. Thus for the rest of this work, we will use data transformed by random forest to train our model.

## 4 Modeling and Evaluation

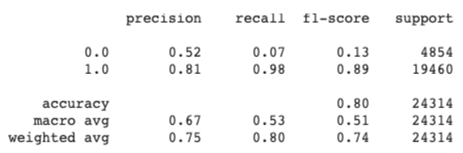
### 4.1 Evaluation Metrics

It is important to note that our classification problem falls into the category of having an imbalanced data set. In our binary classification problem, fully-paid loans take up about 80% of our data, with charged off loans being the minority. Due to the imbalanced nature of our data set, accuracy may not be a good metric. Instead, we have to use other performance metrics including F1 score or precision/recall scores. Moreover, relating to our business problem, we also need to consider evaluation metrics besides accuracy. For example, we need to consider the computational cost of training a model using different methods. For example, in a case where we have trained both random forest and XGBoost model and they have similar performances, we should consider the computational cost involved. In this case, training an XGBoost model is more expensive since XGBoost training involves not only many trees being trained sequentially, which can be memory and time exhaustive, but also the potentially large grid search during hyper-parameter tuning. This type of consideration is generally how the data science, technical goals (achieving best accuracy) can be related to the business goals (trade-off between accuracy and computational cost).

### 4.2 Baseline: Logistic Regression

We first use logistic regression as our baseline model. Advantages of logistic regression include easy implementation and that it outputs conveniently a conditional probability distribution for each observation, so that we can customize a cutoff to classify in the best way for our business problem. However, we do have to keep in mind that logistic regression is a generalized linear model; it assumes the hypothesis space to be linear, which can be a huge assumption. Also, it does not work well when the feature space is large, though we can use L2 regularization to counter this issue. Here, we want to use a generalized linear model such as logistic regression as our baseline model, constraining to a limited hypothesis space, and then explore more complex algorithms from here.

We obtain an accuracy of 0.8016. However, this evaluation metric can be misleading because of the imbalanced data problem. Thus we use classification_report() argument in python to show precision, recall and f1 score of the binary classes in order to further show the accuracy of model. We can see a good prediction on the class of 1 “Fully Paid Off” (precision of 0.81 recall of 0.81; f1 score of 0.89). By the way, if an applicant is judged as “Fully Paid Off”, his loan application will be approved by the lending companies. This class is just what the lending companies focus on, they want each loan they approve can be fully paid off. The prediction of the minority class is not good, we will try to improve it in the latter part.

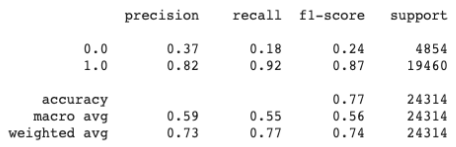


```
              precision    recall  f1-score   support

         0.0       0.52      0.07      0.13      4854
         1.0       0.81      0.98      0.89     19460

    accuracy                           0.80     24314
   macro avg       0.67      0.53      0.51     24314
weighted avg       0.75      0.80      0.74     24314
```

Figure 5: Logistic Regression Model Performance

### 4.3 Naive Bayes

Another algorithm that is simple to implement is naive bayes. Naive bayes poses even a stronger assumption about our problem. Based on bayes theorem, naive bayes assumes that each feature is independent from the other. This independence is highly unlikely for our dataset, but we still decide to test it out given its easy implementation. As our result confirms, naive bayes is not an ideal classifier for our problem, it performs even worse than our baseline model.



```
              precision    recall  f1-score   support

         0.0       0.37      0.18      0.24      4854
         1.0       0.82      0.92      0.87     19460

    accuracy                           0.77     24314
   macro avg       0.59      0.55      0.56     24314
weighted avg       0.73      0.77      0.74     24314
```

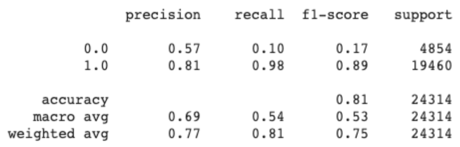
Figure 6: Naive Bayes Model Performance

## 4.4 Random Forest

Random forest is an algorithm that has many great advantages. Apart from its ability to handle curse of dimensionality, it utilizes bagging techniques to improve model performance through reducing model variance effectively.

However, random forest may be severely affected by data skewness. The reason is that in random forest, we create a tree from a bootstrap sample of the data that is sampled with replacement. Thus, for a highly imbalanced data set, there is a probability that each bootstrap sample does not contain any minority class at all. Since random forest is constructed to minimize the overall error, it is possible that the model will perform poorly on predicting the minority class for an imbalanced data set. To address this problem, we can use oversampling techniques such as SMOTE, or variations of random forest such as balanced random forest and weighted random forest (we will try SMOTE in 4.6).

We tune the hyper-parameter n_estimators and find the best n_estimators of 120. Then we apply the same evaluation algorithms and get the above metrics. The Random Forest Model perform slightly better than our baseline model (f1 score is as the same as that of our baseline model, but the accuracy of Random Forest Model is higher).



```
              precision    recall  f1-score   support

         0.0       0.57      0.10      0.17      4854
         1.0       0.81      0.98      0.89     19460

    accuracy                           0.81     24314
   macro avg       0.69      0.54      0.53     24314
weighted avg       0.77      0.81      0.75     24314
```

Figure 7: Random Forest Model Performance

## 4.5 XGBoost

XGBoost is one of the most popular machine learning algorithms for its speed and consistent accuracy. In contrast to bagging methods such as random forest, XGBoost starts with a weak learner with high bias and uses boosting techniques to produce sequentially a strong learner with low bias and low variance, where each tree helps to correct errors made by the previous trained tree. However, XGBoost, or any gradient boosting machines, will continue improving to minimize all errors, which can result in overemphasizing outliers and overfitting. We should use cross-validation to address this issue. However due to the limitations of our computer, we fail to use cross-validation to deal with a mass of data.

The n_estimators argument represents the number of trees in XGBoost whose the default value is 100 while max_depth argument represents the maximum depth of each tree whose default value is 3. There is a relationship between two parameters. Here we create a grid of 4 different max_depth values (2, 4, 6, 8) and 6 different n_estimators values (50, 100, 150, 200, 250, 300), and then compare the log loss of each combination to get the optimal combination. Finally we choose a combination of (max_depth = 6, n_estimators = 100) and get the following metrics.

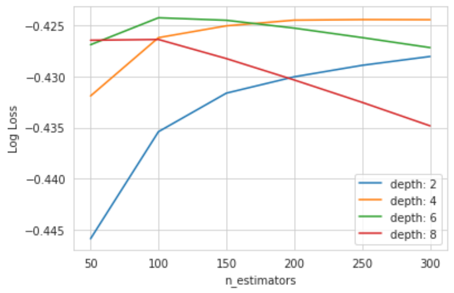
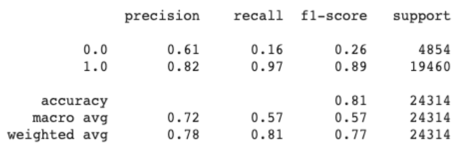


Figure 8: XGBoost Model Fine-tuning

The f1 score of XGBoost model is also similar to that of our baseline model, but XGBoost has an accuracy of 0.8119, slightly higher than 0.8016 of Logistic Model.



```
              precision    recall  f1-score   support

         0.0       0.61      0.16      0.26      4854
         1.0       0.82      0.97      0.89     19460

    accuracy                           0.81     24314
   macro avg       0.72      0.57      0.57     24314
weighted avg       0.78      0.81      0.77     24314
```

Figure 9: XGBoost Model Performance

### 4.6 Model Comparison, Undersampling, Oversampling

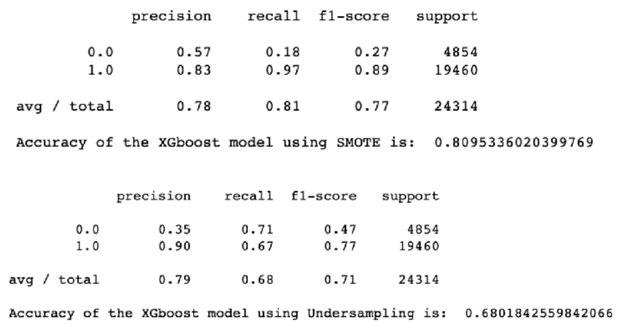
By comparing the performance of these 4 models, we find that the tree-based models perform better and XGBoost becomes the best model through the comparison of f1 score and accuracy. Other evaluation metrics are presented below.

Table 1: Comparisons between Four Different Models

| Model/Evaluation Metrics | Precision | Recall | F1 score | Accuracy |
|---|---|---|---|---|
| Logistic Regression | 0.81 | 0.98 | 0.89 | 0.8016 |
| Naïve Bayes | 0.82 | 0.92 | 0.87 | 0.7742 |
| Random Forest | 0.81 | 0.98 | 0.89 | 0.8068 |
| XGBoost | 0.82 | 0.97 | 0.89 | 0.8118 |

However, there is an imbalanced data problem. There is a vast majority in ”Fully Paid Off” class while a minority in ”Charged Off” class (80% vs. 20%). We will apply algorithms on the imbalanced data set first and see if there is an improvement.

SMOTE algorithm is one of the most commonly used over-sampling methods. It uses a K-nearest neighbors method to increase minority class examples and make it equal to majority class in order to solve the imbalanced problem. RandomUnderSampler() under-samples the majority class by randomly picking samples. Unfortunately, both SMOTE and RandomUnderSampler() fail to improve the precision and recall of the minority class very noticeable, meanwhile, they lower the accuracy of the original models. Thus we decide not to use oversampling or sampling algorithms and choose XGBoost as our best selection.



```
             precision    recall  f1-score   support

        0.0       0.57      0.18      0.27      4854
        1.0       0.83      0.97      0.89     19460

avg / total       0.78      0.81      0.77     24314

Accuracy of the XGboost model using SMOTE is:  0.8095336020399769

             precision    recall  f1-score   support

        0.0       0.35      0.71      0.47      4854
        1.0       0.90      0.67      0.77     19460

avg / total       0.79      0.68      0.71     24314

Accuracy of the XGboost model using Undersampling is:  0.6801842559842066
```

Figure 10: Comparison of XGBoosting Models with Undersampling and SMOTE

## 5 Deployment

The result of the data mining process can be framed as an online learning problem. When we are deploying the trained model, we use the model's prediction and wait for a period of time for the ground truth data for each new applicants. Then this will be fed forward to the model as a new training data point and a learning step will be taken. In this way, given our company's huge transaction volume, our model in its deployment stage will be trained and updated on a daily basis.

However, there are two issues that we need to be aware of. Firstly, in the above online learning setup, knowing whether an applicant will default may take a long time. Thus, even though at a technical level online learning sounds like a good idea, at a production level we still need to make sure in details that our online learning setup aligns with our business goals. Also, though our model is good at predicting the fully paid class, on the other hand it is performing badly, in terms of precision and recall, at predicting the default class. In other words, applicants whom the model predicts will fully pay their loans usually end up fully paying their loans, so the lenders should be relatively confident in our model's prediction and the safety of their funds. However, many applicants who will fully pay their loans are predicted to default by our model. Thus, if fully following the model's prediction, a lot of people in desperate need of money will not get their loans successfully, which raises an ethical issue.

We currently have two solutions to mitigate these issues. Firstly, training our model to be more accurate should always be our goal. Also, given the poor performance on the negative class, temporarily we can involve human intelligence, for example hiring loan officers, to examine the cases that are particularly rejected by our model, and let the loan officers have the final decisions on whether or not to repeal the model's predictions. In other words, we do not fully automate the loan application process; we hire experienced loan officers to supervise the model and make the final decision.

## References

[1] Lending Club official website. https://www.lendingclub.com/info/statistics.action In *LendingClub Loan Dataset 2018.*

[2] Everett, C. R. In *Group Membership, Relationship Banking and Loan Default Risk: The Case of Online Social Lending. SSRN Electronic Journal. doi: 10.2139/ssrn.1114428.*

[3] https://towardsdatascience.com/methods-for-dealing-with-imbalanced-data-5b761be45a18. In *Dealing with Imbalanced Data - Towards Data Science. (n.d.)..*

[4] https://rstudio-pubs-static.s3.amazonaws.com/203258_d20c1a34bc094151a0a1e4f4180c5f6f.html. In *Predict LendingClub's Loan Data -AWS.*

## 6 Appendix A: Contributions

**Denglin Jiang**
Data Cleaning
Data Visualization
Feature Engineering: PCA, L1 SVC and Random Forest
Logistic Regression and XGBoost
Write-up for Data Preparation section, Appendix, proof Reading and Editing

**Yanbing Wang**
Data Cleaning
Data Visualization
Exploratory Data Analysis
Naive Bayes and Random Forest
Write-up for Business and Data Understanding section

**Cheng Zeng**
Related work research
Data Preparation
Logistic Regression and XGBoost
Compare and analyze model performances
Write-up for Model and Evaluation section

**Zian Jiang**

Data Cleaning

Exploratory Data Analysis

Naive Bayes and Random Forest

Model Fine-tuning

Write-up for Abstract, Data Preparation, and Deployment sections

## 7 Appendix B: Manual Feature Engineering: Remove Irrelevant Features or that Leak Future Info

Recall a column is considered to leak information when the data it contains won't be available when we use our model to make predictions. Data leakage would cause the model to overfit.

Here, we hope to drop some columns are not useful for our purposes. Here, we found the following columns that: 1) contain redundant information, and 2) are likely to leak data from the future:

1. desc - contains text explanation for a loan.
2. url - contains a link to each loan from Lending Club
3. id — randomly generated field by LendingClub for unique identification purposes only.
4. member_id — also randomly generated field by LendingClub for identification purposes only.
5. funded_amnt — leaks information from the future (after the loan is already started to be funded).
6. funded_amnt_inv — also leaks data from the future.
7. sub_grade — contains redundant information that is already in the grade column (more below).
8. int_rate — also included within the grade column.
9. emp_title — requires other data and a lot of processing to become potentially useful
10. issued_d — leaks data from the future.
11. zip_code – mostly redundant with the addr_state column since only the first 3 digits of the 5 digit zip code are visible. If all digits are available, we could probably use this feature.
12. out_prncp – leaks data from the future.
13. out_prncp_inv – also leaks data from the future.
14. total_pymnt – also leaks data from the future.
15. total_pymnt_inv – also leaks data from the future.
16. total_rec_prncp - also leaks data from the future.
17. total_rec_int - also leaks data from the future.
18. total_rec_late_fee - also leaks data from the future.
19. recoveries - also leaks data from the future.

20. collection_recovery_fee - also leaks data from the future.
21. last_pymnt_d - also leaks data from the future.
22. last_pymnt_amnt - also leaks data from the future.
23. last_fico_range_low - also leaks data from the future.
24. last_fico_range_high - also leaks data from the future.

The following features concerning hardship or settlement all leak data from the future.

25. hardship_flag
26. hardship_type
27. hardship_reason
28. hardship_status
29. deferral_term
30. hardship_amount
31. hardship_start_date
32. hardship_end_date
33. payment_plan_start_date
34. hardship_length
35. hardship_dpd
36. hardship_loan_status
37. orig_projected_additional_accrued_interest
38. hardship_payoff_balance_amount
39. hardship_last_payment_amount
40. disbursement_method
41. debt_settlement_flag
42. debt_settlement_flag_date
43. settlement_status
44. settlement_date
45. settlement_amount
46. settlement_percentage
47. settlement_term

## 8 Appendix C: Data Cleaning and Explanatory Data Analysis

In this appendix, we attached some visualization that we won't be able to show in the thesis.

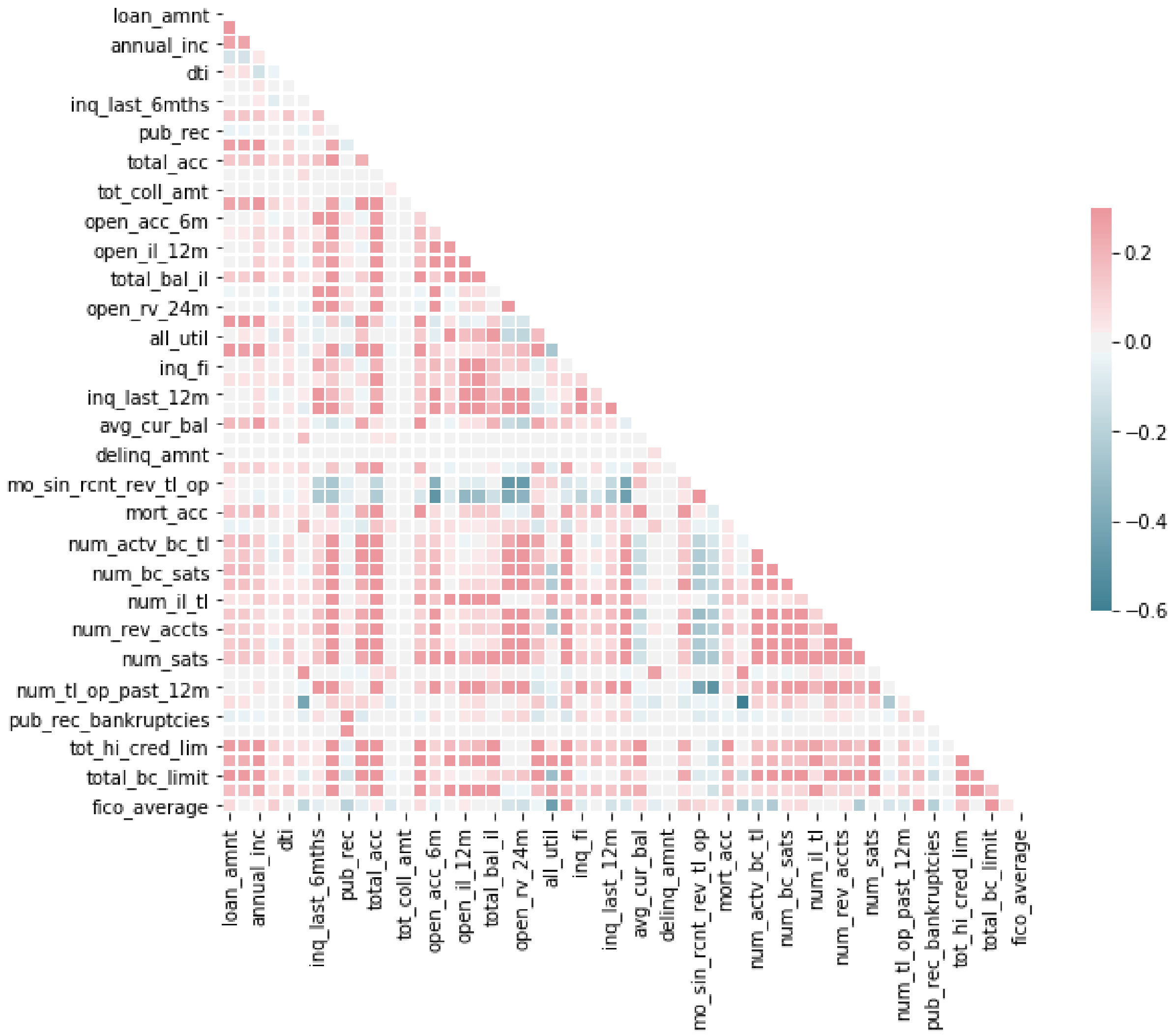


Figure 11: Heap-map of Highly Correlated Features Before Feature Reduction

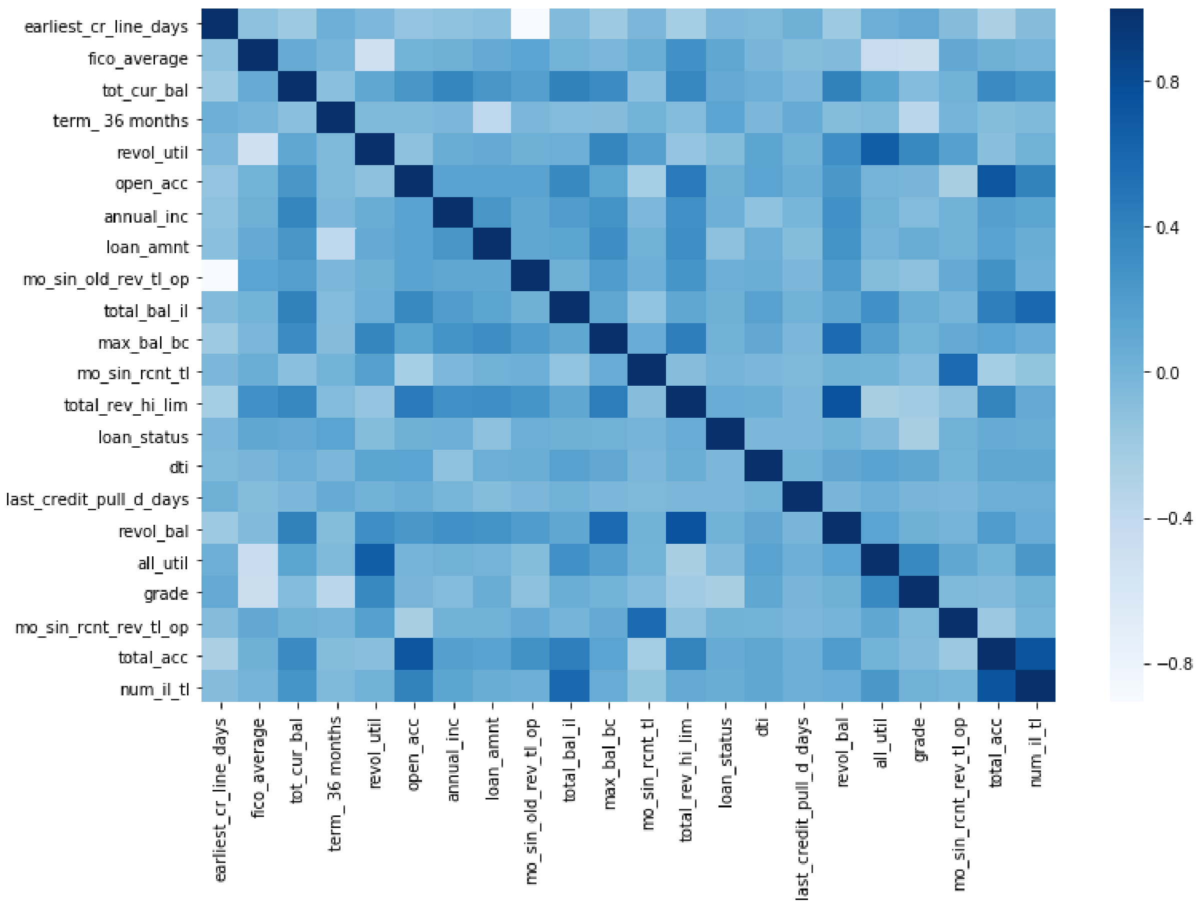


Figure 12: Heap-map of Highly Correlated Features After Naive Random Forest Feature Reduction