Capstone Final Report – Loan Default Prediction

1. Problem Statement

1.1 Brief Description

A loan company makes profit by first lending money to its customers, and then requesting payment periodically from him/her at certain amount of time. So when a customer, someone who has a purpose but is short of cash, contact the lender company and apply for a loan, the lender will collect much information from the customer (borrower) and evaluate the loan application, and then make a decision to either approve the loan or not, and also how much to lend if the loan is approved.

How does the loan company make the decision? If you don't work in a loan company probably you don't know much about it. But we can imagine. The loan company must have some algorithms to use all the information they collect (as features), and predict that if the borrower might default the loan if it is approved.

Needlessly to say, these algorithms must work well. That means that the algorithm must be able to predict the future status of the loan as accurate as possible. If, for some reason, the algorithm "approves" a loan, but later the borrower is not able to make further payment, or refuse to make further payment, the loan company will be in trouble to try different ways to collect payment from the customer; or if the algorithm denies a qualified loan, the company will lose business opportunities. Therefore, the loan company has to train their model really well, so that the model's prediction ability will reach a high standard, to make sure that the business profit will maintain a high level.

According to this webpage(https://www.spglobal.com/marketintelligence/en/news-insights/latest-news-headlines/us-leveraged-loan-default-rate-expected-to-peak-at-6-6-8211-lcd-survey-60554195), about 6.7% loans are finally defaulted in

2021. And according to this webpage (https://educationdata.org/student-loan-default-rate), 7.8% of all student loan debt is in default.

In this study, we will use a dataset from Lending Club, uploaded to Kaggle.com (https://www.kaggle.com/wordsforthewise/lending-club). This dataset has obtained interests from many users, and you can find 59 different codes which analyze this data.

1.2 Introduction of Data Resource

This dataset is from Lending Club. Her is a short message about this lender company: Lending Club was an American peer-to-peer lending company, headquartered in San Francisco, California. It was the first peer-to-peer lender to register its offerings as securities with the Securities and Exchange Commission (SEC), and to offer loan trading on a secondary market. At its height, Lending Club was the world's largest peer-to-peer lending platform. The company claims that $15.98 billion in loans had been originated through its platform up to December 31, 2015. (from Wikipedia)

2. Data Wrangling

2.1 Background of the Data

The initial dataset has two data files, one for accepted loans and another for rejected loans. Since our target is to predict a customer's future payment behavior, we do not need to study the criteria of loan approval. So in this study we will only use the data file for accepted loans.

In this data file, there are totally 2,260,701 rows and 151 columns. After selecting this dataset, my first reaction is this is a little bit too much for me as a beginner in data science. Actually I had been struggling for quite some time in selecting an appropriate dataset for my Capstone project, and I had tried a few. One dataset

looks great, but after taking a deep study, I feel that the dataset does not look real. For example, in all 230,000 people who rented a house, nobody rented that place less than 10 years. As you can imagine, if you rent a place, most likely you will change your residence every 2 – 3 years. How come in so many people, there is not a single one who rented that place less than 10 years? This really does not make sense. So I lost confidence on this dataset, and I feel not comfortable to work on a made up like data. That's why I selected this data, as it mentioned clearly where it comes from. This also gives me great lessons that as a data scientist, I need to be very cautious to select a reliable data.

2.2 Introduction of Target Variable

After quite a few discussions with my mentor, I decided to select this Lending Club data for my project. For this dataset, the target variable is the column of "Loan Status". There are mainly two values in this variable: fully paid and default (charged off). But also the dataset has 878,317 rows with current loans. For those current loans we don't know if they will be fully paid or defaulted. They are of no use to this study, so they should be dropped. There are also a few other uncommon values in column "loan status", which is also not needed for this study, so rows with those values are also removed. For the purpose of convenience, I have set loan status = 1 for fully paid loans, and loan status = 0 for default loans. So in the rest of this report, when loan status = 0 is mentioned, it means "default loans"; and when loan status = 1, that indicate fully paid loans. To make it more obvious I will clearly write here:

Loan Status = 0 means default loans

Loan Status = 1 means fully paid loans

In this data set, the data points with loan status = 1 is almost 4 times more than the points with loan status = 0, which means that about 80% loans are fully paid and 20% are defaulted.

2.3 Data Size Reduction

Since the dataset is huge (2,260,701 X 151), we need to reduce its size, as we believe not all the data is related to our model prediction. First let's drop unnecessary columns, and then rows.

2.3.1 Size Reduction on Columns

The initial dataset has 150 features (columns), excluding the target variable of "loan status". Our first column drop criteria is the missing data. There are large amount of missing data in some columns. Totally 58 columns have more than 30% missing data. Those columns are dropped as the first step of size reduction.

Then we would like to see the correlations between those columns. We found that some columns are highly correlated to each other. For those high correlated columns, we don't need all of them, as they have the same impact to the target variable with their correlated columns. So here we calculate the correlations between any two numerical columns, and use the Pearson correlation coefficient as the criteria to drop columns. If one column has a coefficient > 0.8 with another column, one of them will be dropped. So in this step, 23 columns are dropped.

Our next step is to use the correlation of features with our target variable. We want to see the correlation between each numerical column and the target variable, "loan status". For those columns with high correlation, we believe they have high impact to our final prediction; and for those with poor correlation, they should have little impact to our model. So in this study this correlation relations are calculated. For columns with < 0.1 correlation with the column "loan status", they are removed in this step. Therefore totally 27 columns are removed.

Since those numerical columns have quite different correlation with the target variable, I am curious to see how is the relation between the loan status with the most correlated column. The highest correlation coefficient is from column "last fico rang low", which means the low range of the customer's fico credit score. It

has a 0.58 correlation coefficient with the loan status. So the plot below shows clearly this relation.

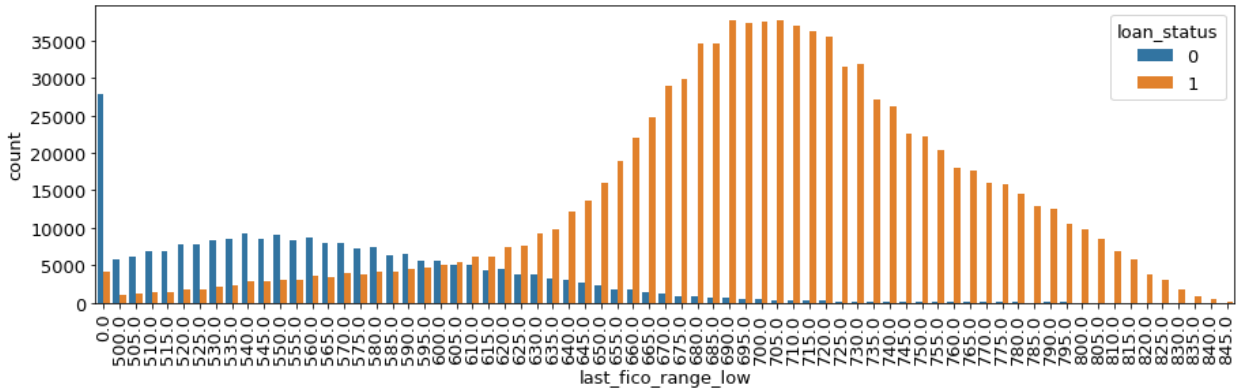

Fig 1. Relation between fico score and the loan status

We can see that if the distribution of the fico score is quite different for the default loans and the fully paid loans. When the score is > 600, there are much more fully paid loans than default loans; and when it is < 600, there are more default loans. That does make sense, because we all know that the lower someone's credit score is, the more likely this person will have default loans.

It is also very interesting to see that some columns have a surprisingly low correlation with the loan status. For example, the annual income's correlation coefficient is 0.04; and the loan amount, with a 0.06 coefficient. This means that weather you will default your loan does NOT largely depend on your annual income or how much you borrow. Default issue is mainly related to how reliable you are.

All the above column removing is regarding the numerical columns. For those categorical features, we want to remove those with large quantity of unique values. For example, the column "title", which stands for the loan title provided by the borrower, has 63,152 unique values. It seems whatever value a data point has in this column, it doesn't matter to our target value. Therefore those columns should be dropped. So in this step, 12 columns are dropped.

By studying the meaning of a few columns, we believe some of them are not related to the target variable, so they are also dropped. For column “earliest_cr_line”, which stands for the time for the customer’s earliest credit line, we removed the month and only kept the year.

As the final result, for those 150 initial columns, 123 have been removed. The new dataset has only 27 predictor variables (columns), excluding the target variable.

2.3.2 Outliners

Outliners are those datapoints which have extremely high or extremely low values. Including those datapoints will bring severe bias to our model. Therefore they should be first analyzed and then removed from the dataset if necessary. A few outliners from columns “revol_util” (revolving line utilization rate), “dti” (ratio of borrower’s total monthly debt payments on total debt obligations), and the “totl_high_cred_limit” (total high credit limit).

2.3.3 Size Reduction on Rows

After we have removed almost 80% of all columns, many missing values are also removed. Now it is safer to remove rows with missing values. By doing this, 912,642 rows are dropped, and 1,348,059 rows have left.

Therefore our new data has 1,348,059 rows and 28 columns in total.

3. Exploratory Data Analysis

3.1 Feature Value Distribution

We would like to have a look on the distribution of numerical features. Histograms are used for plotting these distributions. For those plots with long tails on the right, we use logarithmic axis to show the distribution. All the distributions look good, so we do not need to do anything on them now.

3.2 Principal Components Analysis (PCA)

PCA is a powerful tools to deal with high dimensional feature issue. Let's see how PCA applied to our study. Let's use PCA only for numerical columns. First we draw the PCA accumulative explained variance ratio curve. See Fig.2 below.

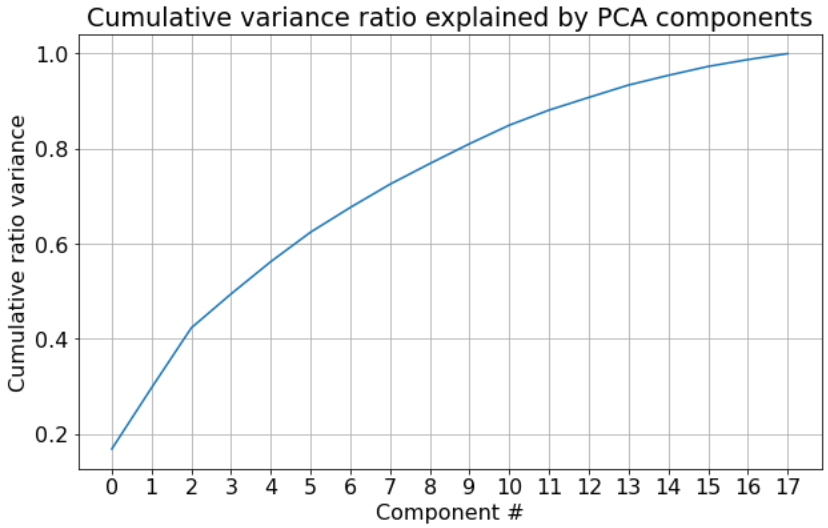

Fig.2 Cumulative explained variance ratio vs components number

Then let's get the value of the first two components and draw those data in the coordinate system formed by the two principal components. Loan status of each point is also shown. See Fig.3 below. It is very interesting to show all those data points visible to us in a 2D coordinate system.

With only two PCA components, we can see that the two target classes have large overlapping to each other. We do see that points of the default loan (loan status = 0, blue points) are more on the top left side, and the points to the fully paid loan (loan status = 1, orange points) are more on the bottom right side. Outliners are also shown in the figure. If we want to make the two classes more separated, we will need more components. From the plot above we can see that with two principal components, the explained variance is only about 42%. That's very low value. Of course with more components, it is not possible to show them in the plot.

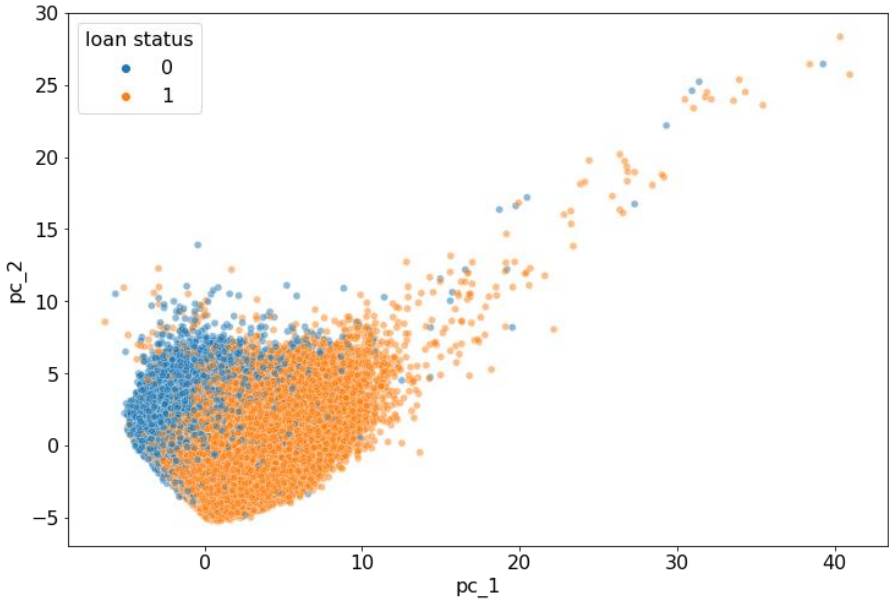

Fig.3 Data points in the PC coordinate system

3.3 Analysis on Numerical Features

For numerical features, we want to know how they will contribute to the final loan status. So first the average value of each column under two loan status classes is calculated. The value difference above 20% is believed to have noticeable effect on loan status; and also just like before, the correlation between those columns with the loan status is computed, and columns with a Pearson correlation coefficient greater than 0.3 is treated as having high impact to the loan status. By trying those two different methods, we got two lists which are much overlapping to each other. This gives us confidence that we have found the right columns, and we will select columns based on these lists.

3.4 Analysis on Categorical Features

Studying how the numerical features affect the target variable seems easy, but for categorical features, it is a little bit challenging. So here the question is, for a certain categorical feature, do some values appear more often for a specific class of the target variable? For example, the column “term” has two values, “36 months” and “60 months”. If, for those data points with term = 36 months, the loan status always

equals 0; and when term = 60 months, loan status always equals 1, then we can see that this column has big impact to our target variable.

For the column “term”, for those loans whose term is 60 months, about half of them are defaulted. And for loans with 36 months as term, only about 1/6 are defaulted. This means that column “term” has impact on the target variable.

By doing the same analysis, we can see that column “grade” and “debt_ settlement_flag” also have significant impact to the target variable. Other categorical features have minor influence to the dependent variable.

4. Pre-processing & Training Data Development

4.1 Balance the Data

In this part of the study, our main task is to make the dataset balanced. After reducing the data size, there are about 80% of rows with loan status = 1 (fully paid), and only about 20% are with loan status = 0 (default). So the ratio is about 4:1, which indicates that the dataset is not balanced. For an imbalanced dataset, the prediction of some certain classes is typically very weak. To make the data more balanced, you can either increase the amount of datapoint with the weak class (in this case it is loan status = 0), or you can drop some datapoints within the strong class. We take the latter one and drop about 60% of datapoints with loan status = 1. After dropping those rows, there are 234,900 rows with loan status = 0, and 381,398 rows with loan status = 1. So the ratio is about 1.6 : 1, which means the dataset is much balanced now.

4.2 Selecting Features for Modeling

Remember that we have 28 predictor features after the data size reduction. Theoretically we should use all of them for the prediction of loan status. But my computer does not have a large memory and robust processor, so the computer

will crush when I use the whole data set for model fitting and predicting. Therefore I have to select only part of them. Thinking the correlation with the target variable really matters, I selected six most correlated numerical variables and two categorical variables. They are listed below:

- last_fico_range_low: The lower boundary range the borrower's last FICO pulled belongs to;
- recoveries: Post charge off gross recovery;
- last_pymnt_amnt: Last total payment amount received;
- int_rate: Interest rate on the loan;
- fico_range_high: The upper boundary of the range the borrower's FICO at loan origination belongs to;
- total_rec_late_fee: Late fees received to date;
- term: The number of payments on the loan;
- debt_settlement_flag: Flags whether or not the borrower, who has charged-off, is working with a debt-settlement company.

4.3 Effect of Scaling Data

Data scaling is always recommended for many models. But when I try it in the KNN (K Nearest Neighbors) model and use f1 score as metrics, I found actually the performance drops a little. So in this study I did not scale the data.

4.4 Performance Evaluation of Different Classifiers

In this part of the study I tried a few different classifiers, including KNN, random forest, and logistic regression. Basically the random forest classifier gives me the best performance.

4.5 Effect of Principal Components Analysis (PCA)

PCA is a powerful tool to reduce feature dimensions and still get a satisfying result. But in this case, I found that it does not work well, especially in the prediction of default loans. So it is not used in the final modeling.

4.6 Great Lesson about PCA

It is stated very often that you should only use “unseen” data for your model testing. But what is “unseen” data? It seems not clear. When the first time I apply PCA in this study, I first use PCA to fit and transform the whole data, and then split the transformed data for training and testing. And then I got a very high f1 score, as high as 99.5%. Wow! This is amazing! I thought PCA is really powerful to result a high predicting model. Then I remembered someone said that “if you get a score higher than 99%, most likely you have data leakage”. So did I have data leakage? I searched the internet, and I found that a common suggestion is that you should not do the train test split after PCA. Instead, you should first split your data, fit and transform your training data, and then transform your test data. So I followed this procedure and apply PCA again. By using 9 principal components, I only get a f1 score of 94%. This is not as good as models with no PCA. So I did not use PCA in this study. But I have learned a lesson which I will remember for ever: If you want to use PCA, you need to do your train test split first.

5. Modeling

5.1 Comparison on Different Models

In the final modeling, 4 different classifiers are tried: KNN (K Nearest Neighbors), Decision Tree, Logistic Regression, and Random Forest. Grid search cross validation is also done on KNN and Random Forest. By doing the Grid search CV, we found that the best n_neighbors = 11 for KNN, and the best n_estimators = 63 for Random Forest. We used these hyperparameter values for the final modeling.

Fig.4 shows the prediction accuracy from the four models, and Fig.5 shows the f1 score from those models. It is very clear that the Random Forest gave the best performance. The prediction accuracy from Random Forest is 96.7%, and the f1 score is 95.7% for loan status = 0 and 97.3% for loan status = 1. The confusion matrix is shown in Table 1.

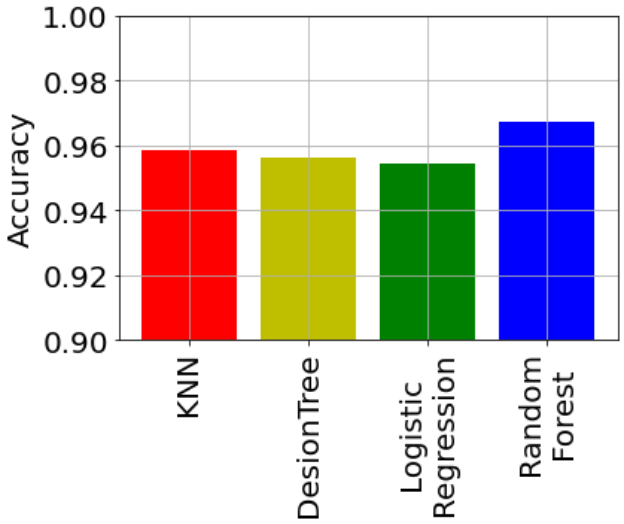

Fig.4 Comparison on prediction accuracy

Fig.5 Comparison on f1 score

	Predicted Default loans	Predicted fully paid loans
True default loans	56165	2515
True fully paid loans	2548	92847

Table 1 Confusion matrix (Random Forest)

5.2 Which Class is More Important?

In the discussion above, we can see that the Random Forest model predicts the default loans to be 95.7% accurate, and predicts the fully paid loans to be 97.3% accurate. So the model is more successful in predicting fully paid loans. Let's think about this question: does the lender want a model to predict default loans better, or another model to predict fully paid loans better? This question could only be answered by that loan company itself. Suppose they want a model which predicts the default loans better (considering the loan company might be in big trouble collecting payments if the borrower is not able to or does not want to make further payment). So let's reduce the amount of the data points for fully paid loans. After dropping half of those data points, we have 234,900 rows for default loans and only 190,699 rows for fully paid loans. Now there are more data from default loans than from fully paid loans.

Fig.6 shows the f1 score of this new dataset comparing the old dataset. It is amazing to see that the f1 score for loan status = 0 (default loan) has been increased largely after using the new dataset. Therefore we can conclude that the proportion of data in different target variable class determines that which class the model predicts better. So if the lender company wants a model which is more successful in predicting default loans, we can include more data points for default loans and use this dataset to train our model.

Fig.6 Comparison on f1 score with different dataset

6. Summaries

Here are summaries of this study on loan status prediction.

1. **Random Forest is the most successful model in predicting loan status**. It shows that random forest performs better than KNN, decision tree, and the logistic regression models.

2. It is not necessary to have all features in our model. With only 8 features we can reach accuracy to be 97%. If more features are included, very minor accuracy improvement could be reached but the calculation time will be increased largely.

3. **If you want the model to be more successful in predicting one certain class, you should use a dataset with more data points from that class**. This way you can control your model to meet your requirement based on your concerns.

4. **If you need to use PCA and test your model performance, you must first do train test data split and then fit PCA with training data**. Otherwise there will be data leakage.