CLASSIFICATION ON LENDING CLUB LOAN DATA SET

BY:

Stephen Hong

Dorbens Jean-Pierre

Edward Cheung

GOAL OF ANALYSIS:

2019 is definitely an unforgettable year, when the first COVID-19 case was confirmed many people, including myself, understand that this is a life-threatening virus but would never expect the pandemic will last this long. About more than a year since the coronavirus recession began, there are some signs of improvement in the U.S. labor market, and Americans are feeling somewhat better about their personal finances than they were early in the pandemic. But still, many non-retired adults say the economic impact of the coronavirus outbreak will make it harder for them to achieve their long-term financial goals. Among those who say their financial situation has gotten worse during the pandemic, many believe that it will take them three years or more to get back to where they were a year ago, and some are left wondering if their finances will recover.

From the financial institution perspective, they should be more cautious to approve any future loans to either individual or business in order to avoid the loan becoming default. Therefore, as the final project in our business analysis degree program, we would like to utilize our research and technical skills to analyze the cause of default loans and provide information to the financial institution so that they can have better predictions on scenarios that will most likely have default loans and to avoid it in the future. We categorized our research based on three sections: will the applicant default on their loan or not based on the historic data, how does the income-debt ratio affect the loan and predict default loan based on the geographic location of the applicant.

We’re analyzing the LendingClub dataset due to the dataset having over 2.2 million customers’ data with 167 columns, which will give us a wealth of information to analyze. In addition, over 13% of the customers who received loans defaulted, which is a significant amount of clients and has a significant impact on the company’s revenue. Our ultimate goal is to predict which customers will default, accrue late payments and pay off their loan, which will make LendingClub more profitable.

TOOLS USED:

Jupyter Notebook (ver 3.0.14), part of the Anaconda Navigator bundle

Spyder (ver 4.2.5), part of the Anaconda Navigator bundle

Sagemaker Studio via Amazon Web Services

DATA COLLECTION:

All Lending Club loan data are collected from Kaggle.com (URL: https://www.kaggle.com/wordsforthewise/lending-club). There are 167 columns in the original dataset and we removed columns that are unrelated to our research, which left us with 53 columns. Additionally, we wrote a Python script to create multiple datasets with select variables needed for particular questions and to clean the data.

Variable name	Short Description
funded_amnt	Total loan amount
term	Number of months for the payments on the loan.
int_rate	Interest Rate on the loan
installment	Monthly payment owed by the borrower
grade	LC assigned loan grade
emp_length	Employment length in years.
home_ownership	The home ownership status provided by the borrower. Our values are: RENT, OWN, MORTGAGE, OTHER
annual_inc	The self-reported annual income provided by the borrower
loan_status	Current status of the loan
purpose	A category provided by the borrower for the loan request.
addr_state	The state provided by the borrower in the loan application
dti	A ratio calculated using the borrower’s total monthly debt payments on the total debt obligations, excluding mortgage and the requested LC loan, divided by the borrower’s self-reported monthly income.
delinq_2yrs	The number of 30+ days past-due incidences of delinquency in the borrower's credit file for the past 2 years
earliest_cr_line	The month the borrower's earliest reported credit line was opened
fico_range_low	The lower boundary range the borrower’s FICO at loan origination belongs to.
fico_range_high	The upper boundary range the borrower’s FICO at loan origination belongs to.
mths_since_last_delinq	The number of months since the borrower's last delinquency.
open_acc	The number of open credit lines in the borrower's credit file.
pub_rec	Number of derogatory public records
revol_util	Revolving line utilization rate, or the amount of credit the borrower is using relative to all available revolving credit.
total_acc	The total number of credit lines currently in the borrower's credit file
out_prncp	Remaining outstanding principal for total amount funded
total_pymnt	Payments received to date for total amount funded
total_rec_prncp	Principal received to date
total_rec_int	Interest received to date
total_rec_late_fee	Late fees received to date
application_type	Indicates whether the loan is an individual application or a joint application with two co-borrowers
annual_inc_joint	The combined self-reported annual income provided by the co-borrowers during registration
dti_joint	A ratio calculated using the co-borrowers' total monthly payments on the total debt obligations, excluding mortgages and the requested LC loan, divided by the co-borrowers' combined self-reported monthly income
acc_now_delinq	The number of accounts on which the borrower is now delinquent.
tot_coll_amt	Total collection amounts ever owed

tot_cur_bal	Total current balance of all accounts
open_act_il	Number of currently active installment trades at time of application for the secondary applicant
il_util	Ratio of total current balance to high credit/credit limit on all install acct
inq_fi	Number of personal finance inquiries
bc_util	Ratio of total current balance to high credit/credit limit for all bankcard accounts.
chargeoff_within_12_mths	Number of charge-offs within 12 months
delinq_amnt	The past-due amount owed for the accounts on which the borrower is now delinquent.
mort_acc	Number of mortgage accounts.
num_accts_ever_120_pd	Number of accounts ever 120 or more days past due
num_actv_bc_tl	Number of currently active bankcard accounts
num_bc_sats	Number of satisfactory bankcard accounts
num_bc_tl	Number of bankcard accounts
num_il_tl	Number of installment accounts
num_op_rev_tl	Number of open revolving accounts
num_rev_accts	Number of revolving accounts
num_rev_tl_bal_gt_0	Number of revolving trades with balance >0
num_sats	Number of satisfactory accounts
num_tl_120dpd_2m	Number of accounts currently 120 days past due (updated in past 2 months)
num_tl_30dpd	Number of accounts currently 30 days past due (updated in past 2 months)
percent_bc_gt_75	Percentage of all bankcard accounts > 75% of limit.
pub_rec_bankruptcies	Number of public record bankruptcies
tax_liens	Number of tax liens

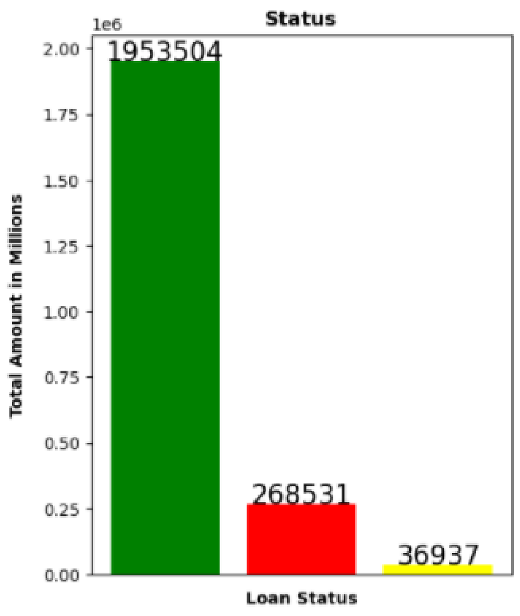
For the first question: Predict who will default based on the approved clients?

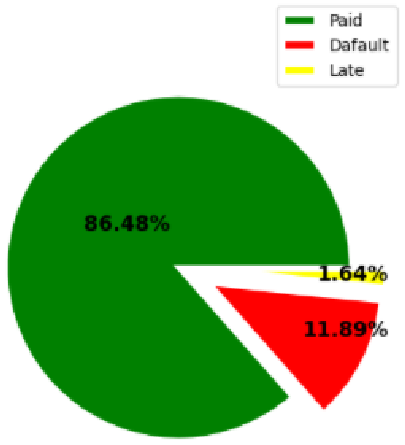
For the second question: Predict default based on Income/Debt Ratio?

For the last question: Predict default based on State?

DATA CLEANUP:

After we found the data that we needed, the first task we needed to do was to select only the columns that are related to our questions and filter out the ones that are irrelevant. After hours of meetings, we decided that the first question requires more data than the other two. Therefore in the cleanup script, we created two separate lists of columns that we need, the first dataset for question one contains 54 columns and the second one for questions two and three contains only 30 columns. We also created several methods to clean up the data as well, like using regular expressions to update the data type, convert the letter grade to an integer, change the date format etc. We also created a new column on the first question named "loan_status_dpi", the purpose of this column is to categorize the dpi into five different bins and concatenate it with the loan status column. There is also a new column named "loan_status_state"on the second dataset, the purpose of this column is to concatenate state and the applicant's loan status. Below is a descriptive summary of the loan status'.





DATA ANALYSIS APPROACHES:

1. Dense Neural Network (Question 1)
2. Decision Tree (Question 1)
3. Logistic Regression (Question 2)
4. Random Forest (Question 2)
5. Stochastic Gradient Descent (Question 3)
6. KNN and Naive Bayes (Question 3)

Data Analysis:

For the first question: Predict who will default based on the approved clients?

In the first question we're looking to predict whether the customers will default, accrue late payments or pay off their loan. To analyze this question we performed two different analyses; dense Neural Network and Decision Tree.

For the dense neural network, I developed two models to make a comparison of the prediction rate. Simple neural network consisted of 10 neurons with 1 hidden layer, with the default initializer function, the activation function of relu, with an L1 regularizer and a training rate of 0.1. The output layer was a 3-layer dense neuron with a softmax activation function. Complex Neural Network consisted of 9 layers with default initializer, relu activation function, learning rate of 0.1 and 1000 hidden neurons as the outer layer. Each consecutive layer had half as many hidden neurons, with 500, 250, 125, 63, 32, 16, 8, 3 neurons as the outer layer using the SoftMax activation function.

The rationale for using 2 models from the beginning was to see the performance of a simple model and a complex model was there to determine the max accuracy of the model, or if an extremely complex neural network was necessary due to our model using 53 predictor variables. The choice to use a L1 regularizer model was heavily influenced by our model using that many predictor variables, so the L1 model could remove any unnecessary predictor variables to predict our outcome.

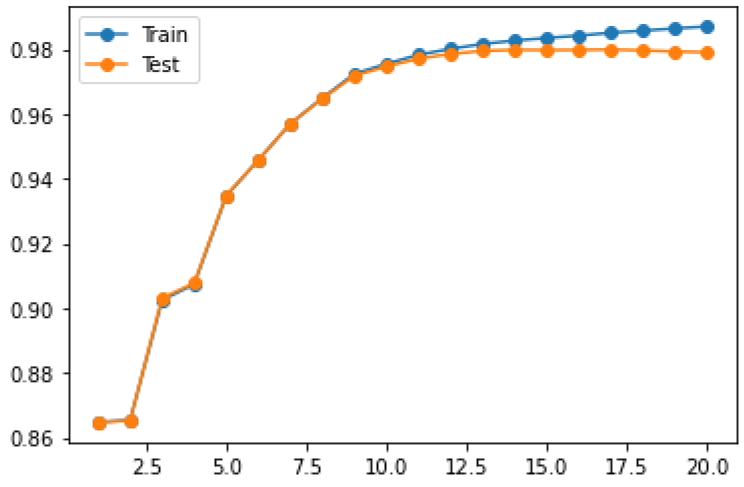
When I performed the 1st analysis with both networks it was learning at a very slow rate. Once I added in batch normalization to the neural network the model trained extremely fast with an accuracy of 93% after the 1st epoch. I ran each model with 50 epochs and received a

98% accuracy rate for both models. The only issue is when examining the training and validation loss for the complex model, there were large spikes in the validation accuracy due to the model overfitting the data. Bear in mind that the baseline accuracy rate of the model is 86%, because of the data being heavily skewed with 86% of the customers paying off their loan. Examining the validation loss had several epochs at 10-20% accuracy and one epoch had an accuracy of 1%. This was obviously extreme overfitting, so I stopped the model after less than 15 epochs.

Given the evidence I decided to abandon the complex neural network and continue with the simple neural network. When looking at the graph of the training and validation loss there were large spikes or changes, this pointed to a potential issue that I needed to examine. I decided to change the learning rate from 0.1 to 0.01, because the learning rate might be too high, so the model might be taking too large of a step. This might be causing the large spikes in training and validation loss. Once this took place the 1st epoch produced an accuracy of 97%, and each epoch after that only had minimal changes in the training/validation loss and accuracy. This was a pretty drastic change in the initial accuracy of the model. Although the initial accuracy of the model was much higher, the overall accuracy wasn't able to improve past 98%. This might be due to our model predicting people who were late, with very little accuracy. 2% of our data are customers who were currently late on their payments.

The highest validation accuracy of the model after 50 epochs was 98.07%. To further improve the model the decision was made to try different initializers to examine the impact on the simple model. We tried 4 different initializers; truncated normal, zeros, he normal, and constant. For comparison our simple model with the default initializer produced a validation loss of 0.1385, and a validation accuracy of 98.07%. When we ran truncated normal, which produced the best out of the 4 initializers, except our default initializer, our validation loss was 0.1425 with an accuracy of 97.55%. The 2nd best initializer turned out to be constant with a 0.1759 loss and 97.24% accuracy. He normal produced a slightly less optimized model with 0.2013 loss and 97% accuracy. Our worst model by far was the zeros model with 0.4456 loss and 86.48% accuracy. Our baseline accuracy due to an extremely imbalanced dataset is 86.47% accuracy, so the zeros model barely beat the baseline.

Then we used a Decision tree. During the cleaning process for this model we utilized dummy coding on the target variable turning it into three columns: Loan_Status_Paid, Loan_Status_Late and Loan_Status_Approved. The model performed very well in predicting the individuals who should be approved and rejected(based on default variables) with both having a recall/f1 score of 99 % . This means the model was 99% of the time guessed True Positive against the test data. The model also had an accuracy of 98% at 10 epochs beating the baseline by about 12%.



The model best for this problem would be the decision tree.

Model	Accuracy	Approved Recall	Default Recall	F1 Score
1Layer NN (default)	98.07%	99%	98	
Decision Tree	98%	99%	99%	99%

For the second question: Predict default based on Income/Debt Ratio?

For question 2 we're trying to examine the impact of debt-to-income ratio on loan status. To answer this question we decided to combine the debt-to-income variable with the loan status (default, late payment, pay off loan). To accomplish this we changed the debt-to-income ratio to 5 bin levels (0-20%, 21-40%, 41-60%, 61-80%, 81-100%)CHANGE BIN LEVELS. Then we combined the variables, which produced 15 outcome categories.

To Answer this question our first analysis was using a random forest classifier. For this analysis I didn't normalize the dataset due to normalizing doesn't have a significant impact on the analysis. The random forest classifier was done with default settings, which produced an accuracy rate of 86.56%. Based on a previous analysis for our first research question we used a decision tree with a max of 9 limbs, it produced a result of 97%, I decided to try the random forest with a max limb of 9. When I ran the random forest with the new parameters, I received the same 86% accuracy.

To try to improve the accuracy of the model I decided to use k-fold cross-validation and then use a grid search decision tree to see all the possible combinations. My rationale is to exhaust all my options with the random forest model to find a better accuracy than 86%. The only problem is that the current AWS server I was utilizing would run out of memory when I would try to run the model. I increased the computing power of my AWS server to 64 gig of ram and 32 vCPU, but it was still unable to handle the amount of ram necessary to run the model. Due to the intense computing power and ram requirements, I decided to abandon the k-fold cross validation with grid search random forest classifier analysis.

Another approach that we use for question two is Logistic regression. Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, even if more complex extensions could exist. The advantage of logistic regression is that it allows the evaluation of multiple explanatory variables by extension of the basic principle. In question 2, what we are determining is will the loan be default based on the applicant's debt – income ratio. Therefore, we think Logistic regression is one of the best methods to use for this binary prediction.

Since the input for this regression may not need as many fields as in Question one, we simplified the dataset by using the cleanup.py script to use only 30 fields instead of 54 fields as in Question one. We also created a new field in the data named "loan_stat_dpi", which concatenates the loan status and the dpi field as our prediction variable. First, there are several fields that contain data type as String, we converted those into dummy variables for regression. After that step is completed, we split the data into input, which is all columns in the dataset except for the prediction column, and the prediction column "loan_status_dpi". After the prediction is completed, we have around 23.40% accuracy.

Logistic regression is relatively fast compared to other supervised classification techniques such as kernel SVM or ensemble methods but suffers to some degree in its accuracy. It also has the same problems as linear regression as both techniques are far too simplistic for complex relationships between variables. Finally, logistic regression tends to underperform when the decision boundary is nonlinear.

Utilizing the Random Forest model for this hypothesis would provide the best outcome for Lending Club.

Model	Accuracy	Approved Recall	Default Recall	F1 Score
Random Forest	32.33%	32	32	29
Logistic Regression	23.40%	23	23	20

For the third question: Predict default based on various states

Stochastic gradient descent is a very popular and common algorithm used in various Machine Learning algorithms, most importantly forming the basis of Neural Networks. What is Gradient Descent? Gradient descent is an iterative algorithm that starts from a random point on a function and travels down its slope in steps until it reaches the lowest point of that function. A problem with gradient descent is that it can bounce around the search space on optimization problems that have large amounts of curvature or noisy gradients, and it can get stuck in flat spots in the search space that have no gradient. How do we solve the problem? Stochastic, means random, is the key for the solution. This approach will randomly pick one data point from the whole data set at each iteration to reduce the computations enormously.

Just like the previous analysis, after we read the data after the data cleaning process, we dropped the unnecessary fields that are irrelevant for the prediction. Since question 3 is focused on the default loan for each state, we added a field named "loan_status_state" which concatenated the loan status and the states as our prediction variables. We also dummied several fields that contained String variables such as "home_ownership", "application_type" and "purpose" before we ran the regression. We also normalized the data to minimize any redundancy and to ensure only related data is stored in the table. After the prediction is completed, we have around 34% accuracy, which is far better than the baseline of 12%.

There are pros and cons for every type of regression. Some of the advantages of Stochastic Gradient Descent is it can computationally fast as only one sample is processed at a time, it can also coverage faster on larger dataset as it causes updates to the parameters more frequently. But there are also disadvantages, such as the steps taken towards the minima are very noisy. This can often lean the gradient descent into other directions. The frequent updates are also computationally expensive because of using all resources for processing one training sample at a time.

After we ran SGD we wanted to compare it to a different machine learning model but ran into issues. At first ran a KNN model which classifies a new data point based on the similarity from training data. You are able to change the amount of neighbors the model uses in the training. At first I used 7 but received an error because there was a target variable that only contained a sample size of 5. So then we reduced it to 3 and ran it on the target test data and feature data. Unfortunately, it ran for 4 hours but ended up erroring out due to having a huge sample size. After decreasing the sample size to 200,000 the model gave us a prediction of only .4% accuracy which is well below the baseline of 12%. Clearly, this model is not cost efficient for large datasets, so we tried the Naive Bayes classification model. This model did significantly worse with a .03% accuracy.

Overall, for predicting which state customers are most likely to default the best model would be SGD. This model beats the baseline by about three fold and is cost efficient.

The best model for this hypothesis is Stochastic Gradient Descent.

Model	Accuracy	Approved Recall	Default Recall	F1 Score
SGD	34%	34	34	24
KNN	.4 %	N/A	N/A	N/A
Naive Bayes	.03%	N/A	N/A	N/A

Impact:

Our main question stems from seeing if we can build a second model that will predict whether the customer will default, accrue late payments or pay off the loan. Our model for both decision tree and dense neural network provided 99% accuracy for decision tree and 98%

accuracy for a neural network. The impact of this will help LendingClub improve their loan application system allowing them to only give out loans to customers with a 98% chance of them paying the loan off versus their current model of 11-13% of their customers defaulting on their payments. The problem with our analysis was we classified people who were late on their payments, which we weren't able to predict with a high degree of accuracy. If we were to rerun our analysis without including customers who were late on payments and only try to classify people who defaulted or paid off our loan, our models might be slightly more accurate.