Capstone 2 Final Report: Lending Club Loan Default Detection

Problem Statement

From Kaggle – Lending Club is a US 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.

Lending Club enables borrowers to create unsecured personal loans between $1,000 and $40,000. The standard loan period is three years. Investors can search and browse the loan listings on Lending Club website and select loans that they want to invest in based on the information supplied about the borrower, amount of loan, loan grade, and loan purpose. Investors make money from interest. Lending Club makes money by charging borrowers and origination fee and investors a service fee.

There is inherent risk in lending money. Will the lender be paid back or will the borrower default on the loan? It would be useful to have a model that can predict loan defaults. In this project I will create a model to predict which loans will default.

Data Wrangling

The Lending Club data that I am using to train my model started out with 74 columns and 887,379 rows. The holdout data consisted of 72 columns and 759,338 rows. Information about each column in the data set can be found here. With 74 columns, feature selection was a part of every step of building the model.

I started data wrangling by removing the two columns that were in the training data, that were not in the holdout data. Next, I removed the 17 columns that had over 85% missing values and another 17 columns that would not be available before a loan is issued. If I had kept the columns that are not available to lenders, that would be a form of data leakage during machine learning.

When deciding how to deal with missing values, I chose to zero fill some missing data and I used the mean of the training data in other columns. I then dropped the remaining rows with missing values.

Using the 'loan_status' column, I created a new column called 'Defaults.' Where loan status had the value of "Default" or "Charged Off," the Defaults column was filled with the value of 1. For training a model, I chose to keep just the "Fully Paid" loans as 0. The loan status values of "Current" and "Issued" were dropped. This is the column that I will use as labels for machine learning.

Then I cleaned categorical columns to remove unnecessary strings and align all values with the definitions of the columns. For example, the definition of 'home_ownership' says the values are 'RENT', 'OWN', 'MORTGAGE', and 'OTHER'. However, 'home_ownership' contained two additional values 'NONE' and 'ANY' which got replaced with 'OTHER'.

I culminated the data wrangling step by doing initial outlier removal. At the end of the data wrangling notebook, the data contained 256,640 rows and 27 columns.

The Jupyter notebook – 2_Data_Wrangling – can be found here.

Exploratory Data Analysis (EDA)

During EDA, I first looked at the PearsonR correlation between each numerical column in the data set. I also looked at how each column correlates to defaults. I then graphed each column of the data grouped by defaults. By viewing the graphs, it became clear that many features in the data were skewed, with long right tails. I used the mean and standard deviation of each feature to detect outliers.

I used the Kruskal-Wallis statistic to test whether loan amount and annual income impact whether a loan will default. For both, the p-value was remarkably close to zero, meaning that each feature does impact defaults. I also used the chi-square statistic on the employment length and home ownership features. For each of these columns, the p-value left me unable to reject the null-hypothesis and I chose to drop both columns.

At the end of the EDA step, the data set had 256640 rows and 23 columns. The full Jupyter notebook – 3 _Exploratory_Data_Analysis – can be found here.

Data Preprocessing and Training

Before modeling the data, I first separated it into features and labels (the Defaults column). For the categorical features I used a Label Encoder and then split the training data into test and train splits. Since the data is imbalanced, with only 20% defaults, I used a combination of SMOTE and random under sampling to bring the number of defaults up to 30%. I finished preprocessing the data by using RFE with a Random Forest Classifier for feature selection. RFE determined the best number of features to be 21.

I used GridSearchCV to tune hyperparameters for both RandomForestClassifier and XGBoost models. I optimized the models on precision to try to minimize the number of false positives (defaults) predicted. To visualize model performance, I used a classification matrix, confusion matrix, and precision-recall curves, like the one below.

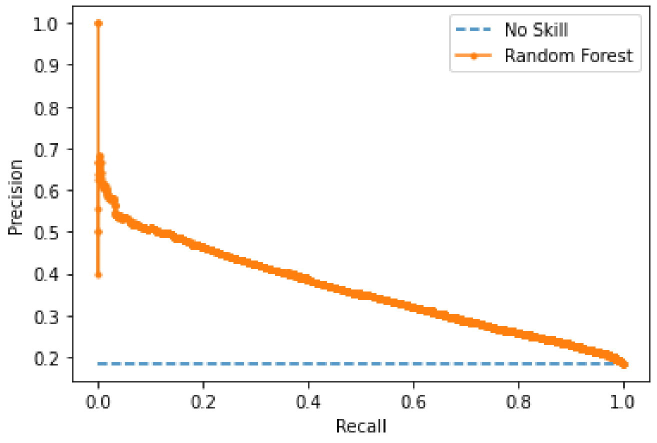

Figure 1: Random Forest Precision-Recall Curve

The Jupyter notebook – 4_Preprocessing_and_Training – can be found here.

Model Selection and Prediction on Holdout Data

The RandomForestClassifier, with the tuned hyperparameters, was chosen as the final model since it had a slightly higher area under the precision-recall curve than the XGBoost model.

I fit the RandomForestClassifer on all the available training data and then used it to predict on the holdout set. The model predicted 94% of the loans accurately. However, the results were underwhelming for the default class. It was only able to predict less than 3% of the actual defaults.

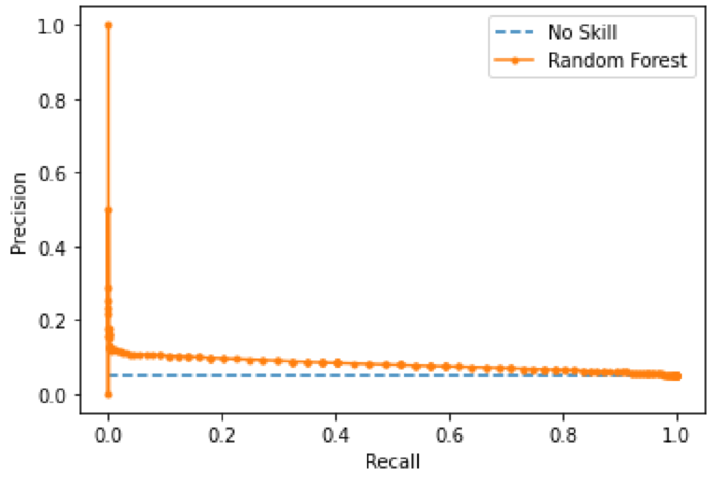

Figure 2: Random Forest Predictions on the Holdout Data

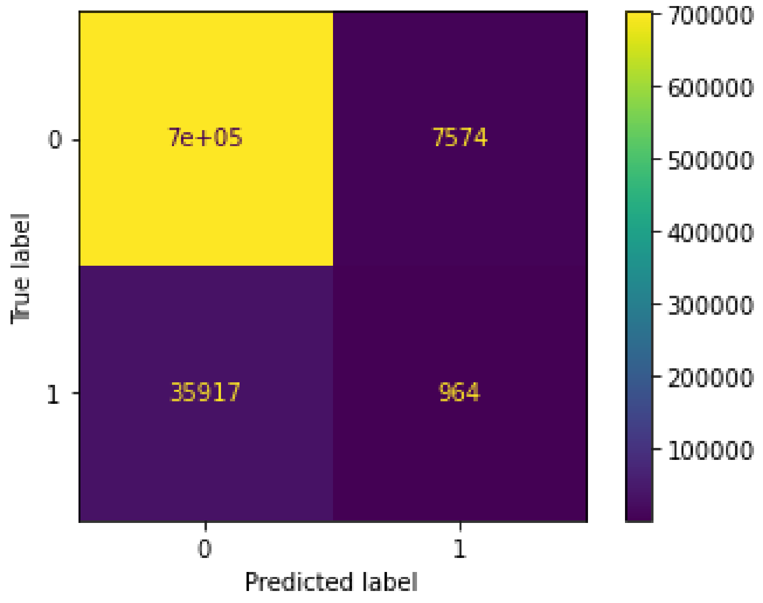

Figure 3: Confusion Matrix for Random Forest Predictions on Holdout Data

The Jupyter notebook – 5_Modeling – can be found here.

Future Work

For future work, I would like to see if additional feature engineering could increase the model's predictions.