Brigit Take Home
Richard Wolff

Goal

Produce a model which does a good job of classifying defaulters/non-defaulter

Best Scores Obtained

Logistic Regression (on holdout data)

Area under the ROC: .64
Precision (on holdout data): .18
Recall: .64

Important Features

Features Predicting Default

- Unknown or intermittent pay frequencies
- No bank account activity (The higher the daily rate of less bank account activity, the more of a chance of default)
- Percent of days with a negative bank account balance

Features Predicting Non Default

- Standard pay frequencies, such as weekly or biweekly.
- Loans already repaid to Brigit

Next Steps To Obtain Better Performance

- Brigit may not want to launch this model as is. With a low precision, only 18% of our default predictions actually defaulted.
    - We'd be turning away loans that could be given to a customer.
    - Although we don't charge interest, customers may leave our subscription service
    - They may also perceive this as a negative customer experience and spread this by word of mouth.
- Obtain new data sources, such as occupation, length of employment, or HHI data.
- Better understand current data. Ie Why is dailyIncomeMean negative? Some customers have only loans 3 out of 5 listed. What happened to 1 and 2? (Trying to roll up previous loan data). Possible new features may include:
    - Expected cash on hand after pay day. (ExpectedIncome - (new debits + current balance))
    - Negative balance events per pay period
    - Roll up total loans defaulted in the past
- Try different classifiers such as Support Vector Machine classifier
- Train the model with an imbalanced data set

How I developed these models

In the jupyter notebook file brigit_data_exploration_and_cleaning, I took a look at the data provided, scrubbed it, decided on transforms, and most importantly looked at the ECDFs of defaulters and non defaulters by feature.

Data Cleaning Steps

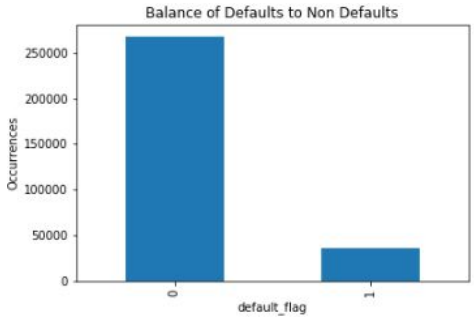
1.) Imputed unknowns for location_state and signup_source. If a user did not have this information, they never have this info on any loan application.
2.) Imputed a user's highest pay frequency from either loans by back filling first then forward filling second. The rest who had no loans with this info were listed as "Unknown"
3.) For numerical missing data, if the loan application had more than 50% 0, missing data, or "Unknowns", I imputed the missing data with 0s.
4.) For the rest of the samples with missing data, I imputed user averages and if they did not have an average, I used the population's average.

Reducing Multicollinearity

- recurringrate & recurrentamountsum had a high correlation. Recurringrate has more information about whether a loan will default or not so I dropped recurrentamountsum.
- balancemean, balancemeanafterpayday0, & balancemeanafterpayday1 also had a high correlation. They all did not seem to have an impact on whether someone would default or not so I resorted to taking the mean of the three.

Imbalanced Data

The target data is highly imbalanced in this dataset (as expected). In fact, only about 11% of loans default which means as we fit our data, we will over predict loans that don't default. To fight that, I will use SMOTE, a synthetic minority over sampling technique, to evenly weight the data.



Train Test Split & Hold Out Data

To help reduce overfitting, I will initially split my data, before oversampling, to have a clean set of data to test my final models on. After oversampling, I will split the data again the better test the oversampled data before testing it on the final holdout data set.

Empirical Cumulative Distribution Functions (ECDF) of The Provided Features

Next I compared the ecdfs of all the numerical columns of loans that default and the loans that are repaid. I also compared the contingency tables of the categorical data. The most interesting features that stood out include:

- **negbalancerate**: More negative balances seems to relate to the odds of defaulting
- **brigitloansrepaid**: The more loans already paid = less default risk

- **accthistorydays**: The longer the account history, the odds of a default increases.
- **noactivityrate**: The less activity can mean more defaults
- **dayswithbrigit**: Users who are new to brigit are more likely to default than those with brigit over a longer period of time
- **recurrentcount**: Those with less recurrent debits have a higher risk of defaulting
- **highestpaymonthage**: Users whose pay is highest the further from underwriting have a risk of default
- **monthswithfeesrate**: the more months a user has months with overdraft fees the more likely they are to default
- **Highestpayfrequency**: It seems the default rate is higher for those that do not have a standard recurring income frequency or if it is unknown.

I thought these would be a good starting point for the model. I withheld any location based discriminants because, at least initially, I don't want my model to predict based off a user's location. There may be merit to using this but for the sake of my initial model I wanted to focus on applicant behaviors.

Models

The initial model I used was a logistic regression model, as we can typically derive the most information back from it's coefficients. My initial training score was pretty good with an Area Under The Curve (AUC) of .63. The AUC on the holdout set was also a .63 which means this model was not overfit.

The model does not perform well when it comes to the precision score but does an ok job with recall. The weighted balanced accuracy is low at a .3.

My second modeling attempt was to use recursive feature elimination (RFE) to see if better features were picked. My initial model performed better on all metrics against the RFE trained model.

My third modeling attempt was the "Kitchen Sink" where I included every feature available into the model. It incrementally improved the model AUC to .64 but took a significant longer time to run. It also had a depressed precision score of .18.

Finally, I tried to run the data through a Random Forests Classifier. This became over fit on the training data but when I ran the holdout data through the classifier I obtained similar scores.