Lending Club Loan Analysis CSCIE-81 Final Project 12/16/2016

Ajay Kliyara Philip
kliyaraphilip@g.harvard.edu

Willard Williamson
wwilliamson@g.harvard.edu

ABSTRACT

Lending Club[1] is a web site where individual borrowers can apply for loans, and individual investors can provide capital for loans. The purpose of our project was to investigate lendingclub.com historical data with machine learning algorithms. Specifically, our project had 3 main goals.

Our first goal was to develop machine learning models to predict loan default on Lending Club marketplace notes. The lending club loan marketplace is a note trading platform which allows investors to buy and sell notes from each other. Notes in the marketplace have gone through the loan origination process and and are fully funded by investors.

Our second goal was to develop machine learning models to make loan origination default predictions. Loans in the loan origination process are loans which are currently in the funding process and are available for investment by individual investors.

Our third goal was to explore the Lending Club developer application programming interface (API). Lending Club provides an API which allows investors with sophisticated programming skills to analyze loan data and execute investments via custom software.

1. INTRODUCTION

Lending Club data was analyzed and machine learning models were created using 2 different software packages: Package 1 - H2O and R, package 2 - SciKit Learn and Python.

H2O and R was used to perform data wrangling and create machine learning models for the loan marketplace default prediction and the loan origination default prediction. Specific H2O models included random forest, naive Bayes, generalized linear model, neural network, gradient boosting machine, and the stacking ensemble.

SciKit Learn and Python were used to perform data wrangling, create machine learning models for the loan origination default prediction, and to explore the Lending Club API.

In summary, overlapping exploration work was performed on the loan origination data using H2O and R, and SciKit Learn and Python. Independent exploration on loan marketplace models was performed with H2O and R. And, independent exploration on the Lending Club developer API was performed with SciKit Learn and Python.

The paper is organized as follows. Section 2 describes loan marketplace model development. Section 3 describes loan origination model development. Section 4 describes Lending Club developer API exploration and application development. Section 5 lists future work. Section 6 provides project reference links. Section 7 provides a list of documents and document descriptions. Section 8 describes how to run the code.

2. Loan Marketplace Model Development

The goal of the loan marketplace model is to predict loan default on loans available in the loan marketplace. Loans which are for sale in the loan marketplace have completed the origination process and the borrower started making payments on the loan.

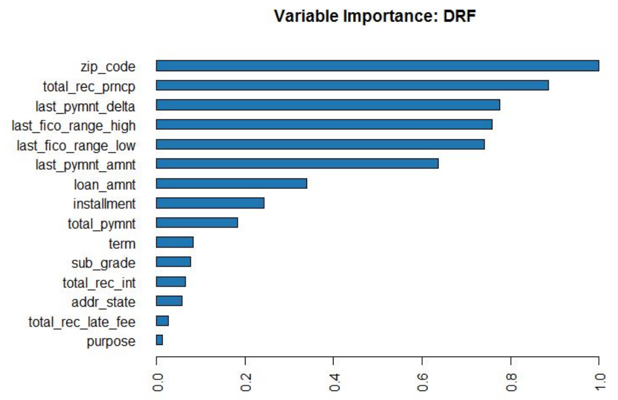
The first step in developing a loan marketplace model was to wrangle the data and perform feature selection. Loan marketplace model data has additional data over and above the data that is available for loan origination models. For example, after the loan is originated and issued to the borrower, Lending Club collects monthly FICO credit score data and makes it available to investors on the loan marketplace. Other examples of additional predictors include total principal received, total interest received, and last payment date.

Data wrangling was accomplished by first downloading the 2007 - 2011 data[2] from the Lending Club web site and creating a data frame with all variables available to investors on the marketplace. General cleanup was performed like removing extraneous characters such as percent signs from numeric columns, removing columns with a percentage of NA values greater than 80%, and removing columns that contained only zeros. A sentiment score column was computed on the borrower's loan descriptions and added to the data frame. The borrower's earliest credit line date was converted to a numeric representing time since epoch. Other time values like months since last delinquency had a large number of NA values which indicated that there was no delinquency information. NA values for situations like this were replaced with zeros. A small number of remaining rows with NA values were removed from the data frame. Lastly, A new "last_pymnt_delta" time feature was engineered to represent the time difference between the borrower's last payment and the loan origination date. Note that H2O automatically standardizes data to have a mean of 0 and a standard deviation of 1 as needed so data standardization was not included in the data wrangling process.

After data wrangling, the data frame was split into 70% train, 15% validation, and 15% holdout test sets. Feature selection was accomplished by training an H2O random forest model on all available features within the training data set, and extracting the most important features from the resulting model. The variable importance plot shown below lists the top 15 predictors.

Loan Marketplace Random Forest Variable Importance Plot

Top 15 predictors

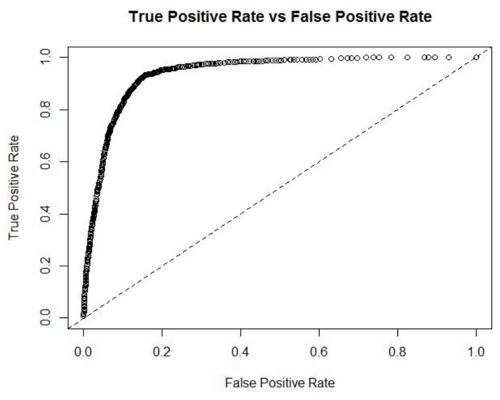


It's interesting to note that the most important predictor is the zip code where the borrower lived. It's also interesting to note that the last_pymnt_delta feature introduced in the data wrangling stage was the 4th most important predictor. It turned out that the top 5 features in the variable importance plot produced an AUC score on hold out test data greater than 0.99. The high AUC score indicated a high degree of model complexity so we selected new features by intuition to reduce model complexity. The last_pymnt_delta feature was selected to introduce a time element indicating the amount of time that loan payments were made. The motivation for using last_pymnt_delta was that it added a time element to the model which was representative of the amount of time that the borrower made payments.

Next, we chose the borrower's initial FICO score at loan origination time, and the borrower's last recorded FICO score. The intuition behind this model is that the time element is used to tie the initial and ending FICO scores together. The borrower's last FICO score is a strong indicator of loan default and the model tries to use the borrower's initial FICO, last FICO, and a time element to tie them together. Using these 3 features, the following ROC curve with an AUC score of 0.9383 was generated on the hold out test data.

Loan Marketplace Default Prediction

Random Forest Model, AUC = 0.938



3. Loan Origination Model Development

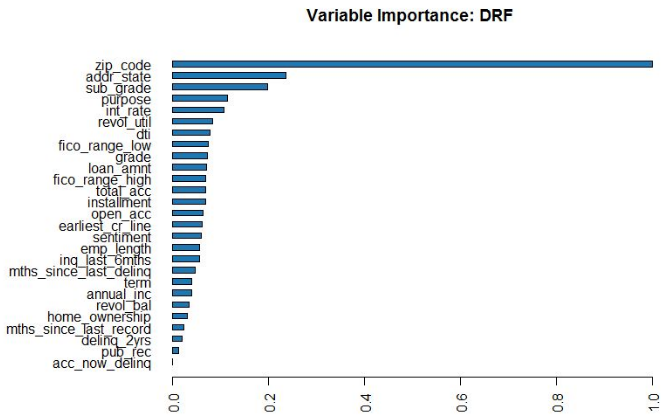
3.1 H2O Loan Origination Models

The goal of the loan origination model is to predict loan default on loans based solely on data available when a borrower applies for a loan.

Loan origination model data wrangling was accomplished similar to data wrangling for the loan marketplace model except that only the features available at loan origination time were selected. For example, only the borrower's initial FICO score is available and the last payment date is not available because the borrower has not made any payments on the loan. Data wrangling produced 27 predictors and a variable importance plot was created from a random forest model.

Loan Origination Random Forest Variable Importance Plot

Top 27 Predictors (All Predictors)



After data wrangling was complete, random forest, naive Bayes, generalized linear model, neural network, and gradient boosting machine models were constructed. A grid search was used for random forest, neural network, and gradient boosting machine models. Each grid constructed 25 models and performed 5 fold cross validation on each model. The wrangled data was split into 70% train, 15% validation, and 15% test splits. A generalized linear model stacking ensemble was used to tie all the models together and produce a final AUC score. The table below summarizes a sample run using 25 models per grid and 5 fold cross validation.

Sample Model Run Data Summary 25 models per grid, 5 fold cross validation	
Model	**AUC (sorted high to low)**
Stacking Ensemble (GLM)	0.7089
GLM	0.7045
Neural Network	0.7028
Random Forest	0.6878
GBM	0.6824
Naive Bayes	0.6657

3.1.1 Random Forest Grid Search Summary

The following 2 tables summarize the random forest random grid search parameters and provide a sample of the best model results returned by the random grid search.

Random Forest Random Grid Search Summary Stop the Grid Search on Max Number of Models	
Hyper Parameter	**Parameter Values**
Num Trees	100 to 1000 by 100
Num variables on which to split a node	2 to number of predictors - 1 by 1
Max Depth	2 to 20 by 1

Random Forest Sample Grid Best Model Result	
Hyper Parameter	**Best Grid Value**
Num Trees	300
Num variables on which to split a node	2
Max Depth	3

3.1.2 GBM Grid Search Summary

The following 2 tables summarize the GBM random grid search parameters and provide a sample of the best model results returned by the random grid search.

GBM Random Grid Search Summary Stop the Grid Search on Max Number of Models	
Hyper Parameter	**Parameter Values**
Learn Rate	0.01, 0.1
Max Depth	3 to 15 by 2
Row Sample Rate	0.5 to 1 by 0.1
Column Sample Rate	0.2 to 1 by 0.1
Num Trees	100 to 500 by 50

GBM Random Grid Search Summary Stop the Grid Search on Max Number of Models	
Hyper Parameter	**Best Grid Value**
Learn Rate	0.1
Max Depth	3
Row Sample Rate	1
Column Sample Rate	0.2
Num Trees	100

3.1.3 Deep Learning Grid Search Summary

The following 2 tables summarize the deep learning random grid search parameters and provide a sample of the best model results returned by the random grid search. Note that the number of hidden layer nodes was based on 27 predictors but in hindsight 14 nodes is likely too small. On the last day of the project, we discovered that H2O converts factor columns to a number of features similar to the cardinality of the column. For example, there are 836 unique zip codes so H2O converts the zip code column to at least 836 features. The deep learning model generally worked as good as the other models but we probably could have gotten better performance out of the deep learning model had we learned about how H2O treats factors earlier.

Deep Learning Random Grid Search Summary Stop the Grid Search on Max Number of Models Optionally Stop on max time or max epochs	
Hyper Parameter	**Best Grid Value**
Activation	Rectifier, Tanh, Tanh with dropout, rectifier with dropout, maxout, maxout with dropout

Hidden Layers	2, 3
hidden layer nodes (constant across all layers)	8, 10, 12, 14
L1	0.00001, 0.0001, 0.001, 0.01, 0.1
L2	0.00001, 0.0001, 0.001, 0.01, 0.1

Deep Learning Random Grid Search Best Model Summary	
Hyper Parameter	**Best Grid Value**
Activation	maxout with dropout
Hidden Layers	2
hidden layer nodes (constant across all layers)	14
L1	0.00001
L2	0.00001

3.1.4 GLM Summary

The GLM model consistently produced the best overall AUC scores. A lambda regularization search was attempted but the model results were no better than the results without lambda regularization. There is a feature in the H2O GLM model to tell H2O to automatically remove collinear columns from the data; however, this feature reduced the AUC score.

3.1.5 Naive Bayes Summary

No hyper parameters are available for tuning in the naive Bayes model.

3.2 SciKit Learn Loan Origination Models

Up to this point, all models were optimized on AUC. As a next step, we wanted to build a model which is optimized from an investor's perspective.

Why optimize for the investor?

Per the 3rd Quarter 2016[3] results released by Lending Club, its borrower base is about 1.7 million individuals and investor base is at about 142,000. This implies that for every 10 borrowers there is only 1 investor. This is the reason why we wanted to optimize it from an investor perspective. There is a need to protect the interest of the investor and thereby help sustain this 250M dollar industry.

Toolset

The toolset used for this portion of the project was Python and SciKit Learn.

Pre-Processing

Categorical features like Grade of Loan, Purpose, Home Ownership etc. were binarized to support the SK learn models.

Feature Engineering

Many features were synthesized to help augment the model using our understanding of the data.

Earliest Credit Line: Number of years since the first credit line was computed.

Average FICO: Average of Low and High FICO scores at loan origination.

% of Open Account: Derived from Open Account by Total Accounts.

Sentiment Score: A sentiment score was calculated using the Afinn package from the borrower's loan description entered on the loan application. A net sentiment score was assigned based on the sentiment of the words used in the description.

Zip Code, income & Population: Using Zip Code directly in SCiKit, as binarizing it would result in ~800 features. Instead, zip code was substituted with median income and population [4] from the corresponding zip code and used as predictors.

Standardization

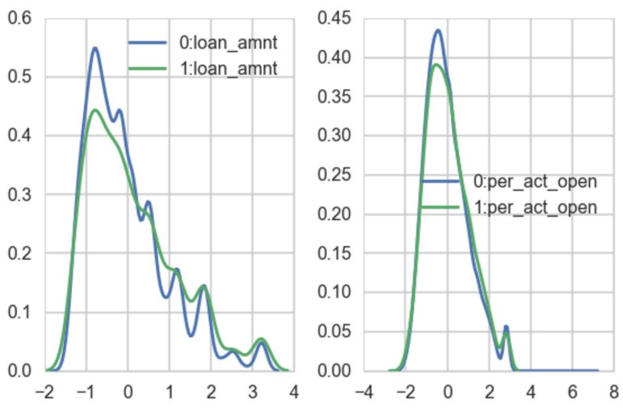
The numeric features were standardized by removing the mean and scaling to unit variance.

Exploratory Analysis

Data was visualized using Kernel Density Estimate (KDE) plots and histograms to identify if there were any clear predictors of good and bad loans.

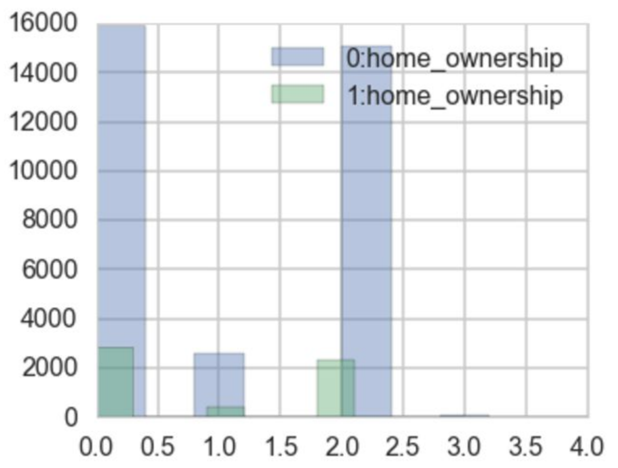
KDE plots were used to visualize continuous predictors like loan amount, interest rate, etc, grouped by good and bad loans. Unfortunately, all continuous features overlapped and no clear predictors emerged.

Kernel Density Estimate



Histograms were used to analyze discrete predictors like home ownership, loan purpose, etc. Homeownership and loan terms of 36 months seemed to be relatively strong predictors of bad loans. The blue line/bar stands for good loan and green line/bar stands for a bad loan.

Home Ownership Distribution Plot



X Axis Represents Mortgage Factors:

{ 'NONE': 4, 'OTHER': 3, 'MORTGAGE': 2, 'OWN': 1, 'RENT': 0}

Color Key:

Blue : 0 : Good Loan; Green : 1 : Bad Loan

Modeling

Establishing Baseline

The data was split into an 80 / 20 train/test set ratio. To get the feel of the data, a simple classifier - Linear SVM (SKlearn: LinearSVC) was used. The SVM model produced an accuracy of 85% on the blind test set where accuracy = (TP + TN) / number of loans.

It turns out that 85% of the loans in our dataset are good loans, so an accuracy of 85% is same as predicting all loans as good. Based on this fact, we set out to develop a better model.

A primary objective was to optimize a model from an investor's perspective which meant looking for something other than accuracy/AUC. We instead came up with a custom scoring function to optimize the investor's profit.

A profit matrix was developed which was designed to penalize false negatives (a false negative is falsely predicting a good loan). Falsely predicting a good loan means that investors lose money. The profit matrix is calculated as follows:

True negatives are good loans that do not default. The true negative cell of the profit matrix was calculated as number of true negatives in the confusion matrix times the mean of the interest accrued from the good loans.

False negatives are bad loans that are predicted to be good loans. The profit value for false negatives in the profit matrix was computed as the negative of the number of false negatives times the mean of the principal lost plus the mean of the interest lost.

False positives are good loans classified as bad. The profit value for the false positives in the profit matrix was set to 0 as there was no loss or gain.

True positives are bad loans classified as bad loans. The profit value for the true positive cell in the profit matrix was set to 0 because the model does not make recommendations on true positives.

The following table summarizes the profit matrix described above.

Profit Confusion Matrix

TN * mean(interest accrued from good loans) TN * 1333	FP * 0
FN * [mean(principal lost due to loan default) + mean(interest lost due to loan defaults)] FN * -6108	TP * 0

The profit matrix is used to compute an overall profit score with the following equation:

Custom Profit Score Equation

custom profit score = 1333 * TN - 6108 * FN

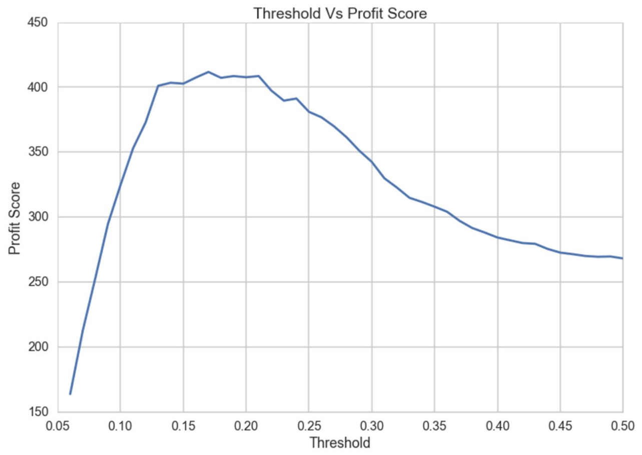
After the profit matrix and custom profit score equation were established, a baseline calculation was made by predicting all loans as good. The goal was to establish a baseline profit value based on the 85% good loan skew of the dataset. The resulting baseline confusion matrix true negative and false negative values were plugged into the custom profit score equation defined above and a baseline profit score of $260 was obtained.

Classification

All classification models were created using 5 fold cross validation and the custom profit score function described above was used as the scoring function in all the models.

The first model to be implemented was Logistic Regression with L1 Lasso Feature selection on the unbalanced dataset. There was a small increase in the profit score from the baseline profit calculation of $260 but nothing significant.

Since our objective was to protect the investor, the default threshold was changed from 0.5 to 0.16, implying only loans with a probability of default less than 0.2 were considered good.



This increased the profit score from \$260 to \$407, but the sensitivity was only 0.5.

For the next step, the Pearson correlation factor was used to determined features most correlated with the outcome. The top 25 features were passed through a linear SVM model. Unfortunately, the profit score dropped to \$260 so this model did not suit our needs.

In the next step, we balanced the data by picking all bad loans and then randomly picking an equal number of good loans to balance the training dataset. The balanced dataset was passed through a Linear SVM model and the profit score was higher than the previous SVM model but still lower than the logistic regression on the whole training set.

Next we made a prediction using a logistic regression (L1) model on the balanced dataset. This gave the highest overall profit score of \$412 and the highest sensitivity of 0.64.

This was followed by decision tree and random forest models. Both models did not beat the score obtained by logistic regression on the balanced dataset.

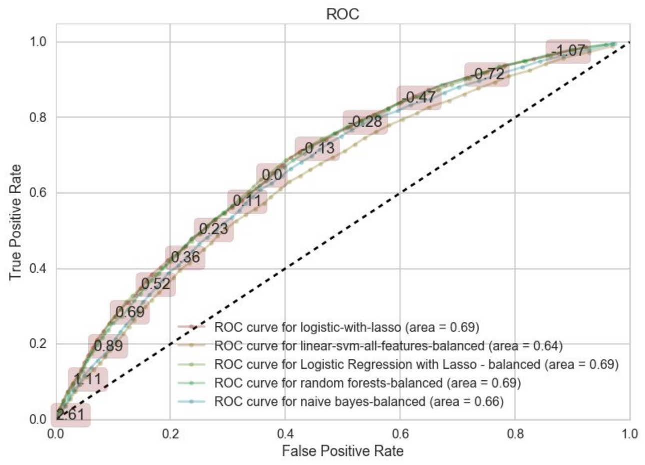
The logistic regression model trained on the balanced dataset was our best overall model and was retained to be used in the application using the lending club API.

The following table summarizes the classifier results:

Classifier Results Summary Table

	classifier	Profit Score	AUC	Sensitivity
0	Base Classifier:Predicting all loans = good	260.804342	NaN	0.00
1	Logistic Regression with Lasso	268.093800	0.69	0.01
2	Logistic with Lasso modified Threshold = 0.16	407.410483	NaN	0.55
3	Linear SVM	260.804342	0.59	0.00
4	Linear SVM - Balanced	347.380544	0.64	0.63
5	Logistic Regression with Lasso - balanced	412.554406	0.69	0.64
6	Random Forests	400.557544	0.69	0.65
7	Naive Bayes	385.217878	0.66	0.50

Classifier Receiver Operating Curves (ROC)



Classifier Receiver Operating Curve Notes:

1. The best logistic regression classifier thresholds are shown in boxes at regular intervals on the curve for that classifier.

4. Lending Club Developer API

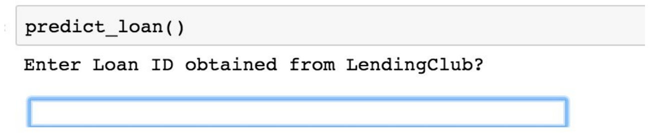
Now that we have our best model, we wanted to build an application to allow the investors to interact with our model and make informed decisions on their loan investments.

Lending Club makes available REST[5] API's to access its data. One of the API's is the listing API, which allows pulling data for loans which are available for investment. Essentially, the listing API lists loans which are not completely funded. Using Python code, we used the listing API to programmatically extract all loan listings available for investment. The extracted loans were preprocessed and feature engineered exactly same way it was done for the training dataset. Once processed, the loan data was passed through our best model. The probability of default for each loan was determined using the model and the probability of default was added to each loan.

The dataset was then encapsulated in a python function thus allowing an investor to interact with the model. The following steps outline how to interact with the loan data:

Step1 - Execute the loan prediction function and key in the identifier of a loan from the lending club website which you would like to analyze.

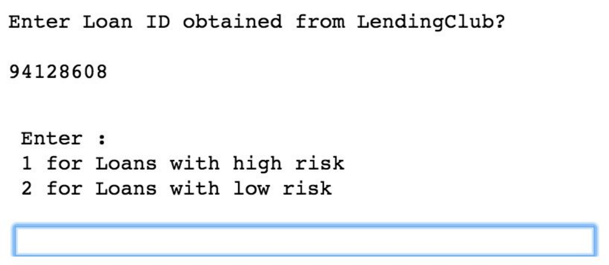
Graphic Showing The Python Function Waiting for Input



```
predict_loan()
Enter Loan ID obtained from LendingClub?
```

Step2 - After entering a loan ID, the next step is to choose a risk path. High Risk stands for loans with probability of default less than 0.5 and low risk path stands for loans with probability of default less than 0.3.

Graphic Showing The Python Function Asking for Risk Path



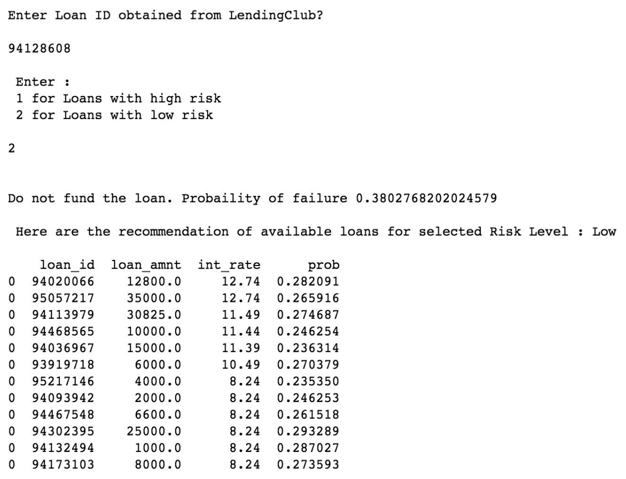
```
Enter Loan ID obtained from LendingClub?

94128608

 Enter :
 1 for Loans with high risk
 2 for Loans with low risk
```

Step3 - If the keyed in loan meets the risk criteria, the application makes a recommendation to proceed with the investment. If the keyed in loan does not meet the risk criteria, the application advises not to invest and alternative loans are provided to the user which better align with the user's investment objectives.

Graphic Showing The Python Function Suggesting Alternative Loans



```
Enter Loan ID obtained from LendingClub?

94128608

 Enter :
 1 for Loans with high risk
 2 for Loans with low risk

2


Do not fund the loan. Probaility of failure 0.3802768202024579

 Here are the recommendation of available loans for selected Risk Level : Low

    loan_id  loan_amnt  int_rate      prob
0  94020066    12800.0     12.74  0.282091
0  95057217    35000.0     12.74  0.265916
0  94113979    30825.0     11.49  0.274687
0  94468565    10000.0     11.44  0.246254
0  94036967    15000.0     11.39  0.236314
0  93919718     6000.0     10.49  0.270379
0  95217146     4000.0      8.24  0.235350
0  94093942     2000.0      8.24  0.246253
0  94467548     6600.0      8.24  0.261518
0  94302395    25000.0      8.24  0.293289
0  94132494     1000.0      8.24  0.287027
0  94173103     8000.0      8.24  0.273593
```

5. Future Work

- Build a web based application to better allow investors to interact with the predictor application.
- The marketplace loan data provides monthly FICO score data since loan origination but the historic data available for download only provides the final FICO score. Try to increase model performance by using the the monthly FICO score data.
- Further investigate deep learning by better correlating the number of hidden layer neurons to the the number of features including hidden features generated by factors.
- Build a models using more recent lending club data extracted using the API.

6. References

1. Lending Club Website

   https://www.lendingclub.com/

2. Lending Club Data

https://www.lendingclub.com/info/download-data.action

3. Lending Club Reports Third Quarter 2016 Results

   http://ir.lendingclub.com/corporateprofile.aspx?iid=4213397#PRdialog

4. Zip Code Median Income and Population

   http://www.psc.isr.umich.edu/dis/census/Features/tract2zip/

5. Lending Club Loan Listing API

   https://api.lendingclub.com/api/investor/v1/loans/listing

7. Document Descriptions

File Name	Description
intial_processing.ipynb	This file has the code for reading in data, preprocessing it, creating few features such as sentiment score and imputing zip code with median income and population.
Initial Classification with Feature Engineering.ipynb	This file contains bulk of the feature engineering, eliminating multi collinear features and initial classifier runs to get the feel for the data.
Initial Classification with Feature Engineering.ipynb	This file has the code which creates the custom scoring function and optimizes various classifiers using this function, thus helping the classifier which maxmizes profit.
lending_club.Rmd	The lending_club.Rmd file is the main file used to explore H2O models for loan origination and the loan marketplace.
lending_club_num_models_25_xvalidate_5.html	Sample lending_club.Rmd run for 25 models and 5 fold cross validation.
lending_club_num_models_50_xvalidate_5.html	Sample lending_club.Rmd run for 50 models and 5 fold cross validation.
lending_club_num_models_2_xvalidate_5.html	Sample lending_club.Rmd run for 2 models and 5 fold cross validation. made with the final version of the Rmd.

8. How to run the code

- To run lending_club.Rmd, open lending_club.Rmd and knit the file to html.