Machine Learning Engineer Nanodegree

Capstone Project

Devin McCormack

October 12, 2017

I. Definition

(approx. 1-2 pages)

Project Overview

Lending Club (LC) is a peer-to-peer (P2P) lending platform, which matches borrowers with one or multiple lenders who “crowd-fund” the loan. P2P lending is touted as a way for everyday people to diversity their total investment profile with loans, and crowd-funding allows people to further mitigate risk by only contributing small amounts to individual loans. Additionally, they operate with low overhead, which allows them to be highly competitive with interest rates for borrowers. Although the returns of direct unsecured lending are potentially higher than either investing in pure-play loan businesses or using a CD or savings account (where the bank does the loaning “for you” and passes on an interest rate), the risk is potentially higher.

However, we know that banks make money through loaning, and there are methods for determining the risk of a loan. Beyond commonly used services such as FICO credit scores, many institutions have their own, proprietary algorithms to determine the riskiness of loaning money to a person as well as a required return interest rate, if they loan is worth investing in at all. P2P services have different criteria for assigning interest rates. Often, they want to beat traditional loans to draw more debt consolidation borrowers (almost 60% of the loans on LC are for debt consolidation). Additionally, as individual lenders hold the risk, LC simply needs to fulfill as many loans as possible, between people searching for the lowest interest rate (borrowers) and potentially inexperienced lenders. They facilitate this by having an Auto-Investor, which automatically invests money in loans based on an individual investor's desired return, using the LC-assigned interest rates of loans.

The problem here is that LCs motivations are not wholly aligned with the people bearing the risk of defaulted loans. Long term, LC needs to have interest rates enticing enough to both lenders and borrowers – as well as meet risk tolerance expectations of lenders – but they are encouraged to apply a much more liberal borrower filter than traditional banks. Smart investors cannot depend on the Auto-Investor, and must be more stringent in the selection of loans. Luckily, LC is open with their previous loan data, giving a large dataset that can be mined by investors: https://www.lendingclub.com/info/download-data.action.

Problem Statement

The problem is that LC – and therefore LC's Auto-Investor – does not screen loans as well as it can, and is potentially passing on more risk than necessary to "naïve" investors. My project intends to create a loan screener that will preferentially select loans with lower than average default rate at the given interest rate I will use machine learning to attempt to regress or classify loans in order to improve potential gains of a LC portfolio. Using the metrics provided by LC, I will develop an algorithm that will be better than random choice (Auto-Investor) at selecting loans at a given interest rate that do not default. I will explore the usage of Naïve Bayes (C.D. Manning, P. Raghavan and H. Schütze (2008). Introduction to Information Retrieval. Cambridge University Press, pp. 234-265), Decision Trees (L. Breiman, et al. Classification and Regression Trees.

Wadsworth, Belmont, CA, 1984), and Random Forest Classifiers (L. Breiman, “Random Forests”, Machine Learning, 45(1), 5-32, 2001.) for this purpose.

Metrics

The metrics used to determine the quality of my improved loan picker needs to be usable for classification of the data. Additionally, I need to be mindful that the dataset is not balanced; most loans are in good standing or were fully paid. In fact, near 79% of loans qualify as “good”, and so my classification metric will have to be very sensitive to false positives. For classification, the simple metric of precision allows us to penalize for false positives and hopefully allow us to have a stricter investment picker. Precision is defined as true positives over the sum of all positives (true and false positives). Additionally, I used ROC AUC as a secondary metric to determine. ROC AUC is defined as the area under the curve defined by true positive rate over false positive rate.

II. Analysis

(approx. 2-4 pages)

Data Exploration

Data exploration takes place within the “Data Cleaning” workbook, this section will serve to describe the data as well as further explain the workbook. The dataset contains 74 columns of information on 887,383 loans that were in funding, currently being paid during the data collection, or have been closed; either through completed payment, or default on the loan. In order to simplify my data, I have removed any loans that were not completely funded, as well as only focused on 36 month term length, narrowing my view to 620,015 loans. Further, I remove any loans with incomplete credit data (with NaNs in the credit data columns), as well as narrowing my loans to only fully paid loans, and loans in various state of disrepute (defaulted, late, etc), leaving me with 209,087 loans for my data. This also served to help limit some earlier problems I had with developing a metric for dealing with current loans that were paid largely in lump sum, but not fully paid off.

As far as columns, 74 features per loan is a lot, but most of it is highly correlated. Things like loan amount, funded amount, and installment amount; or total account and open accounts, are directly tied to each other. Additionally, many of the columns are descriptive of current loans and would not be available/predictive for new loans, such as: total payments, payments towards principle, etc. By looking at correlation heatmaps and using some domain knowledge, I was able to comfortably reduce my complexity to 12 features per loan. Many features that were removed are due to high levels of correlation with other features. For example, months since last delinquency is highly inversely correlated with delinquencies in last 2 years. Additionally, other features were defined with the "months since…" designation. This is a moderately informative metric, but it is a continuous feature that is poorly defined: the "best" designation is infinite or "never", and never is infinitely better any other number. Although it could be useful, only few loans even had values for these features, and the highly skewed feature would add a lot of complexity to the model. Additionally, most of these features had a complimentary feature that measured over a timeframe instead of counting from last incident. My final feature space has 12 features, including employment length, loan amount, annual income, and credit information like number of accounts and debt-to-income (DTI) ratio.

Earlier iterations cast a wider net where I had to deal with a lot of outliers. Many outliers were taken care of in the data, as shown in the notebook, including people/organizations that took out giant personal loans while having a few orders of magnitude larger income than average, and people who made large portions of their payments as lump sums. By restricting to completed loans, I did not have to deal with these in later iterations.

The final data has 162,960 loans, with 79.8% of loans being fully paid, and 20.2% of loans defaulting or late. On average, loans were for $12,567, and an interest rate of 12.73%, leading to an average monthly installment payment of $422.17. Average income was $69,416, with a DTI of ~17%. Large grain analysis of fully paid vs defaulted loans show there are few differences between them, although defaulted loans tend to have higher interest rates, higher DTI, higher credit utilization, as well as nearly $10k less income per year on average, with a larger spread of incomes than fully paid loans.

Exploratory Visualization

Here (Figure 1) I have a correlation plot of all the data categories for this dataset, which I use to help narrow down my dimensions, cutting out some of the highly correlated dimensions. Highly correlated features (either positively – red, or negatively – blue) means that a change in one of the features will cause a highly predictable change in the other feature, and keeping two highly correlated features in our algorithm increases complexity with minimal predictive gain. For example, the loan amount and installment amount are directly related by an amortization equation, so using both in our equation would add unnecessary dimensions without adding any predictive power. Similarly, there are several dimensions that are elaborations of each other; e.g. opened installment loans in the last 12 months, or last 24 months. Although these are certainly different and both could affect their ability to pay off a new installment loan, they include duplicate data, and we can use one or the other instead of both. It is important to note that highly correlated features are often very important, but it requires some domain knowledge to pick the best representative feature out of a list of highly correlated features in order to reduce algorithm complexity with retaining as much predictive power as possible. [note: added labels to figure]

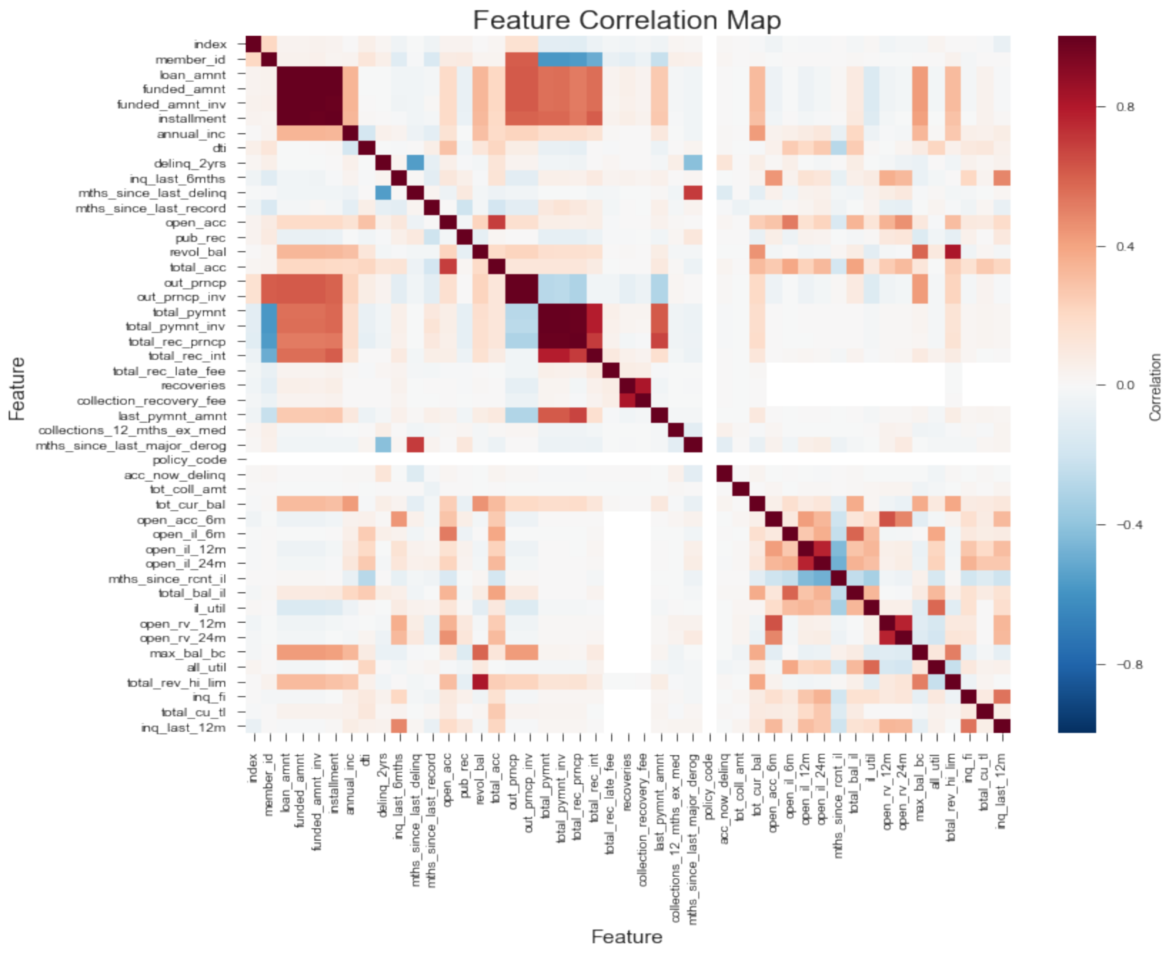

Figure 1: Feature correlation map of full feature space. Red values indicate high positive correlation, while Blue values indicate high negative correlation. Features are always 100% correlated with themselves (diagonal line).

Algorithms and Techniques

Since there are many more fully paid loans than bad ones, classification may be “good enough” for individual investor loan picking. I use three common techniques to attempt to classify the data: Gaussian Naïve Bayes, a simple decision tree classifier, and the popular random forests classifier.

Gaussian naïve Bayes assumes independence between dimensions. Although I have taken steps to remove correlations in the dimensions, this is certainly not a "true" assumption, but often naïve Bayes (C.D. Manning, P. Raghavan and H. Schütze (2008). Introduction to Information Retrieval. Cambridge University Press, pp. 234-265) can still be a pretty good predictor with dependencies. The positive is that Naïve Bayes is a simple application of Bayes theorem:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

Where A is the class (good or bad loan),B is a specific predictor, and P(A I B) is the probability of a class given a specific predictor. Sklearn has only one input for GaussianNB, of priors, with it defaulting to having no initialized prior values.

Decision tree classifiers (L. Breiman, et al. Classification and Regression Trees. Wadsworth, Belmont, CA, 1984) are maybe one of the most commonly used machine learning algorithms for classification – if not for their power, certainly for how easy it is to explain what they do. Simply, they use dimensions and set thresholds, attempting to split the data into appropriate classes in a flow-chartable way. There are many inputs into the algorithm, including the criterion for a branch split, as well as many characteristics of the tree: maximum depth, maximum leaves, and minimum samples per leaf. The default values sets split criterion to gini impurity, which is a measure of how often a randomly chosen element would be incorrectly labeled if chosen from the distribution of labels in the data. It takes the form of a probability squared, which is calculated faster than entropy (which uses a logarithmic function). The default values that restrict tree size are all set to 'None' or essentially allowing unrestricted tree growth.

Finally, random forest (L. Breiman, "Random Forests", Machine Learning, 45(1), 5-32, 2001.) is an ensemble method that can be thought of as simply a collection of multiple decision trees that only see a portion of the features, which help prevent overfitting. This algorithm has similar inputs as decision trees with two notable additions. First, you can change the size of the "forest" by adjusting n_estimators, with a default being 10 decision trees. Secondly, you can determine whether bootstrapping (aka random sampling with replacement) is used, with it defaulting to true.

All algorithms were initially tested under default sklearn settings as described above. The algorithm that performed best, decision trees, was then put through a grid search optimization to further tweak and improve the model.

Benchmark

For classification, as a majority of the loans are "good", they benchmark is simply assuming all loans will be paid in full. This achieves a precision of 79.83%, equivalent to the percentage of loans that are classified as "good". The ROC AUC of this benchmark would be 50%.

III. Methodology

(approx. 3-5 pages)

Data Preprocessing

The majority of my preprocessing involved converting object data to the correct numeric values. For example, the interest rate included the percent sign (e.g. 8%), which needed to be removed so interest rate could be taken in as an actual number. Similarly, term included the string "months", and employment included "years". After removing these strings, they were numeric values. Also, the dates required some processing to convert them to a form that allowed mathematical manipulation.

Further preprocessing was required to input data into the models. I needed to log transform some of the skewed data categories, specifically, annual income, delinquencies, open accounts, total accounts, public records, revolving balance, and collections. These categories tended to have long tails that would affect classification. Next, I rescaled all the numerical data, as things like annual income is on a much different scale than number of accounts. I simply used MinMaxScaler from scikitlearn to do this. Finally, I one-hot encoded the categories of home ownership (rent, mortgage, own), and loan purpose (credit card and debt consolidation).

Implementation

I compared three classification algorithms for this data: Gaussian Naïve Bayes, Decision tree classifier, and random forest classifier. These were all implemented within the scikitlearn toolbox framework. I compared the default algorithms against each other with portions of the dataset and tested for speed and two accuracy metrics, ROC AUC (Receiver operator curve Area Under Curve) and precision.

For all algorithms, I attempted to classify using two classes: “good” loans that were paid in full, vs. “bad” loans, which were either defaulted or were late at the time of the data collection. Data was randomly split into training and test groups (80/20) and each algorithm was trained and tested on the same data splits for comparison to each other and to baseline.

First, the algorithms were tested for speed, ROC AUC, and precision with 1%, 10% and 100% of the dataset with default parameters, and then the best algorithm was further refined using a grid search method.

There were no major complications with implementation for classification, as I paid a lot of attention towards proper data preprocessing, and I was fairly conservative on what loans were included in the dataset. Performing the fits using default settings for the models allows for quicker comparisons between them, however it could potentially rule out some techniques that actually would be better, but require extensive tuning.

Refinement

After testing the three algorithms, I came to the conclusion that Decision Trees had the best chance of beating the benchmarks. Most of the algorithms were able to modestly match or beat the benchmark with 100% of the training data, and both decision trees and the random forest classifier showing signs of overfitting: 100% precision on training set, benchmark precision on the test set. As overfitting can be attenuated with pruning and model tuning, decision trees looked promising. And since both decision trees and random forest had similar results, and decision trees is much simpler, I decided to look into pruning the decision tree .

I tuned the decision tree on three key features: the criterion, max depth, and the max nodes per leaf. Practically all tuning of decision trees is designed to prevent overfitting, but there was no improvement over default parameters for max depth or max nodes per leaf. Interestingly, there was an improvement from using entropy instead of Gini impurity as the criterion. The final features of my model were as follows: max depth, none; max nodes per leaf, none, and criterion, entropy. The final tuned model achieves a precision of 81.16%, a modest improvement over the default model with 81.03% precision.

IV. Results

(approx. 2-3 pages)

Model Evaluation and Validation

My final model is set in the end of the LC_MLtrain.ipynb notebook, and is implemented on a new set of data in the Predicting new loans.ipynb notebook. As noted above, grid searching some common constraints on the decision tree algorithm did not provide any additional precision over the base, unconstrained version. The final decision tree has a precision of 81.16%, which is an improvement over the benchmark of 79.83%. The final decision tree shows the five most important features are: dti, revolving balance, revolving utilization, annual income, and loan amount; accounting for over 50% of the predictive power of the features. Using cross validation scoring of the data, with 20 folds, the precision stays highly stable, with a standard deviation of 0.19%.

Applying the trained model to new data was simple using the Prediction new loans.ipynb notebook. The new data is simply pre-processed, deskewed, one hot encoded, and normalized like the test data. Unfortunately, I cannot access the veracity of the classification of the new set of loans, but there do not seem to be any outstanding issues with applying it to a new set of loans.

Model stability is a common concern with decision trees, as small changes in the input data can have large repercussions in the tree structure. In order to test the robustness of my model, I used a 10-fold test on the training data. For each fold, I re-fit the data using a decision tree with the same parameters as the final tuned model (default except that criterion is set to ‘entropy’), and output the precision and feature importances. Precision on the 10-fold test was in line with previous precisions, indicating the model still performed adequately well. Using the top ten most predictive features from the full test set model (see Figure 2 in free form visualization), I calculated the mean and standard deviation of the feature weights of these top-ten features. For all top ten features, the standard deviation is at least one order of magnitude lower (or one-tenth the size) of the mean value, indicating relatively stable importances, even with different training data.

Justification

Using the standard deviation values from the cross validation data (0.19%), I find that the improvement of 1.33% (81.16% vs 79.83% benchmark) over benchmark is small but significant using standard one-sided t-test and p-values from the cross validation test. Essentially, there is over 5 standard deviations between the means, which indicates a extremely small p-value, allowing me to reject the null hypothesis that the classification is as good as a blind sampling.

V. Conclusion

(approx. 1-2 pages)

Free-Form Visualization

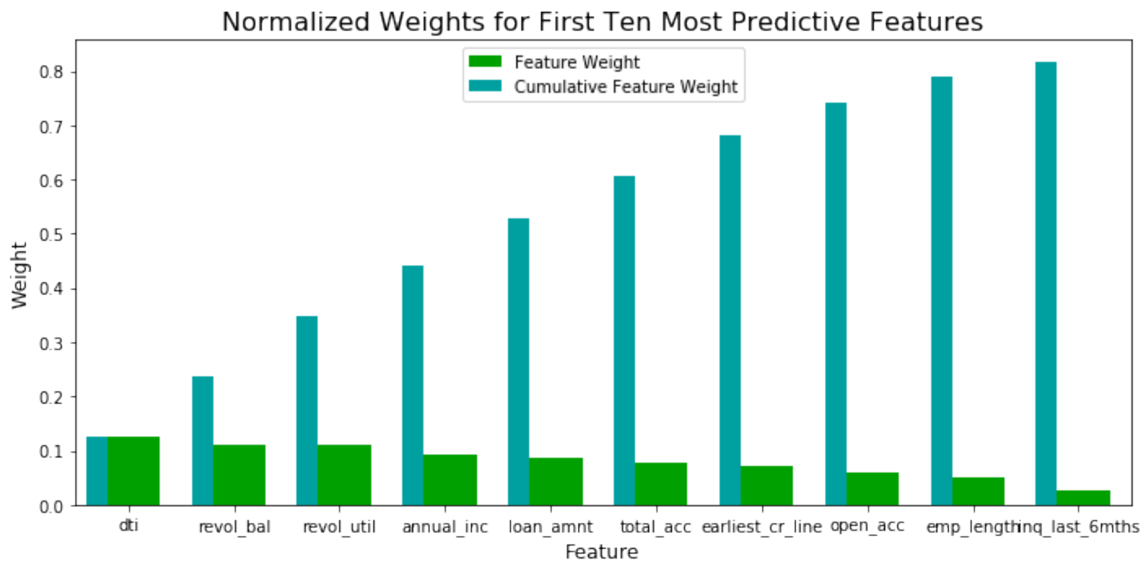

Figure 2: Ten most predictive features of final tuned model. Green bars are individual weights, blue bars indicate cumulative weight.

Here is a plot of the 10 most predictive features, and their cumulative weight in the tree. This shows that 10 features account for 80% of the predictive power of the tree. The most powerful feature, unsurprisingly, is debt-to-income ratio. This will essentially measure how much of their paycheck they need to spend to stay on top of their debt. Another feature that is interesting to point out is employment length, which can be thought of as income stability.

Reflection

This project has been an interest to me for a long time, and I have done previous iterations using simple filtering algorithms. The most difficult aspect of this is that defaults are plain hard to model, and have been a challenge to multiple gigantic industries. There are too many life altering event variables that cannot be captured by previous numbers and loans. Clearly, there are obvious extreme risk cases – but those don't even make it through LC's initial screening process. This project wasn't just modeling defaults: it was, in an ideal world, modeling a slippage in LC's own proprietary interest rate algorithms. Secondly, the fact is, that with enough investment in many different loans, the risk of default is ameliorated greatly. A large majority of loans are paid in full. LC itself dedicates much of it's technical talk on the website towards the idea of risk management through diversification of loan investment. I think this was an interesting exercise in machine learning, and through using decision trees, I have in theory created a more complex version of my previous simple filters. In fact, a few of the most predictive features were used in my simple filter. I think having any incremental improvement over just random loan picking is fantastic, and I plan to use, test, and iterate this model as I use the LC platform.

The act of cleaning the data and selecting features really allowed me to build a better domain knowledge of finance and credit risk. I feel I may have oversimplified things at certain points, but I wanted to start simple and allow for further elaboration, such as adding more of the features that I cut. Understanding the skew and determining whether features were truly continuous or discrete was an interesting challenge especially for some more ambiguous features. I tried to be agnostic about preconceived notions of relative worth of non-obvious numerical features, so as to not insert a bias that would muddy the model of a truer underlying pattern. I initially was very excited to try regression on this data, but quickly realized that it was too complex and not fully continuous enough to make such a task easy. The fact that two credit-report identical people can have vastly different loan outcomes due to something like a death, loss of job, or other unpredictable life event added a large amount of noise for regression. I had the forethought to preprocess my data for potential classification, and looking back I think I will always start with classification for this kind of high feature-space data. My end result was satisfying, but was a bit dampened by how high of a benchmark bar I was required to pass. I hope to learn more about ROC implementation in python to improve my error function. The model can only perform as well as an error function describes your goal. Overall, this was a great experience that has engendered a deep respect for very large datasets, and how important data cleaning is to developing a model.

Improvement

The biggest improvement could be made on gathering more data, and making better use of the data that is there. In order to narrow down the problem, I turned a dataset of 600,000+ loans into a workable set of 160,000. If I had more understanding of the moderately cryptic datapoints, and potentially if I was cleverer about removing data, I potentially could've had more information to base my model on.

Additionally, It would be very interesting to use neural networks on this dataset, but it would be it's own capstone to do so. It would be nice to have a risk factor metric that comes out the end of this model, but I'd have to really think hard about the implementation on this very complex and non-trivial feature space.