Predicting Loans Using Deep Learning

Rocco Manzo
Indiana University
rmanzo@iu.edu

Govil Kuma
Indiana University
govkumar@iu.edu

Vasanth Ravikumar
Indiana University
vhassanr@iu.edu

1 Abstract

The business of lending money is a risky one, but there are billions to be made if done properly. For example, in the first quarter of 2019, Wells Fargo earned 12.3 billion dollars in revenue from interest on loans they had outstanding. However simple it may seem, giving loans is not a trivial task. The last recession showed that it is easy to get greedy and give out loans to people who cannot possibly pay them back with the hopes of making a profit. The complexity of such a problem begs the question, how do lenders determine who to loan money to? In this paper, we look to address this question using deep learning, and data from Lending Club.

2 Introduction

For this project, we used data obtained from Kaggle about loans given by Lending Tree. There were a little over two million data points with about 83% being loans that were paid back on time, and 17% being loans that were either paid late or not at all.

3 The Data

An important part of any machine learning problem is the data, and how to deal with problems it may cause.

3.1 Types of Variables

The given dataset is comprised on 145 attributes and mostly the attributes are categorized broadly under 3 different categories:

1. Applicant Centric (Occupation, age etc.)
2. Loan Centric (interest, term etc.)
3. Applicant behavior over loan term Centric (delinquent status, next payment etc.)

All of the behavioral attributes are observed after the loan is given, so for the purpose of this project we ignored those attributes.

3.2 Null Values

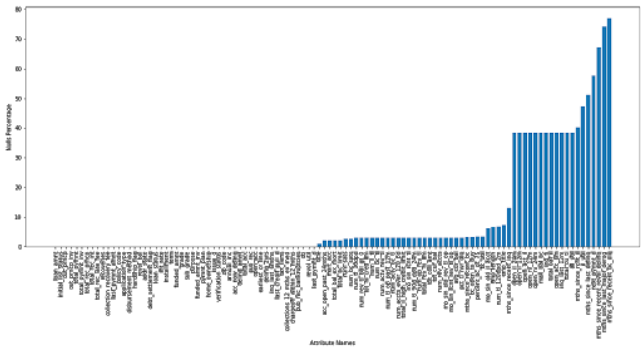
One major issue we faced in this project were null values. As you can see in the graph above, many variables had significant amounts of null values. We decided to ignore these values as we did not believe they would be useful with so many values missing. There were also many attributes that had no null values. We did not use all of these, but we did use a handful of them. There were some variables we decided to use that had some null values. As shown in the graph, there were quite a few variables that had between 5% and 20% null values. For these, we substituted zeros for the null values. This allowed the network to run on these values, and it didn't have a negative impact on performance so it was a simple solution.

3.3 Data Analysis

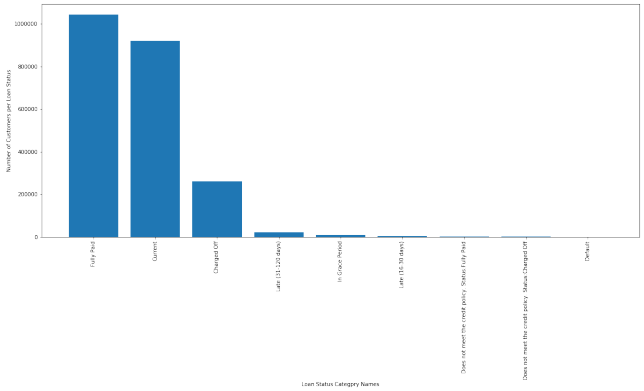
The target variable, which we want to compare across the independent variables, is loan status. The strategy is to figure out compare the average default rates across various independent variables and identify the ones that affect default rate the most.

When the loan status is "Fully Paid" we consider that to be a good loan, so will relabel them as "1" and

Attribute vs Null Percent

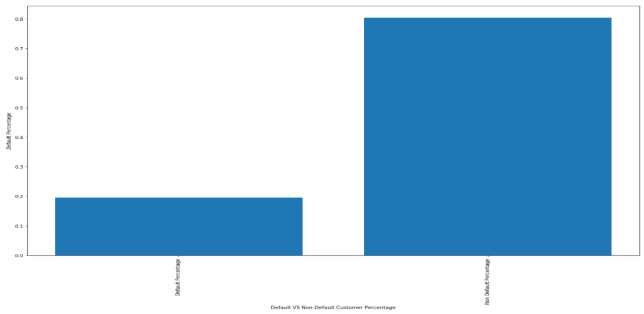


Customers vs Loan Status



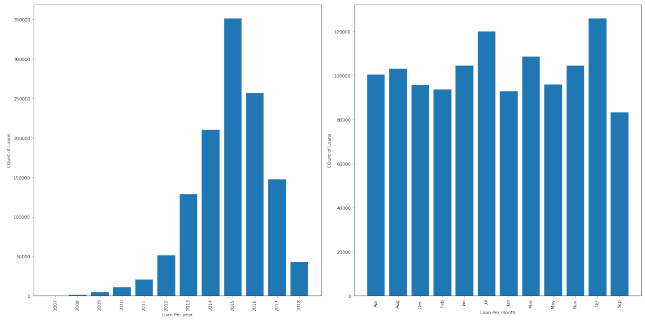
when the status is "Charged Off" it mean the debtor did not pay the loan back, so will relabel them as "0". The rest of the status may or may not be considered. For the analysis part we wont consider them.

Repayment Status vs Percent



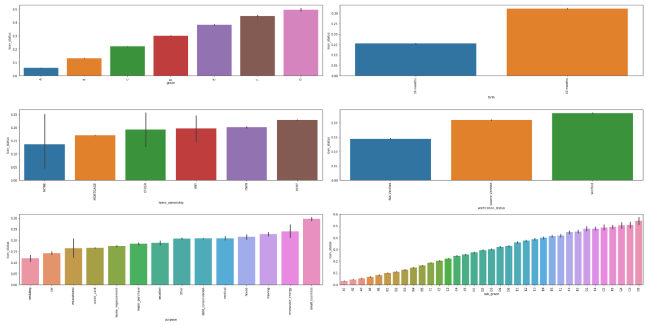
This next graphs shows a significant increase in loans from 2007 to 2015. This can likely be explained by the housing bubble popping and leaving financial institutions hesitant to give out new loans as billions of dollars had just been lost, but over time they got past their fear and started giving more loans as the economy recovered. The proceeding drop off can likely be explained by rising interest rates pricing people out of the market.

Loans Per Year



The following graph shows the relationship across a few major categories and how those variables impact the default rate.

Default Customer Distribution Across Variables



Based on the above graphs say the following things about these variables:

Grade: As the grade of loan goes from A to G, the default rate increases. The Grade is defined by the lender and it reflects the risk associated with the loan.

Term: 60 months loans default more than 36 months loans.

Subgrade: As expected - A1 is better than A2 better than A3 and so on.

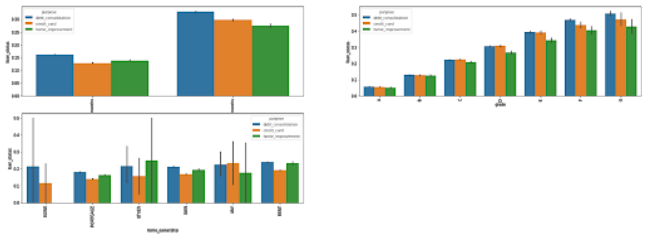
Home Ownership: This is not a good factor for determining default rates.

Verification Status: Surprisingly, verified loans default more than not verified.

Purpose: Small business loans default the most, then renewable energy, and education.

The next set of graphs show default rates across a few major bivariate categories.

Bivariate Default Data



From the above three categories variable analysis, we found that loans for the "debt_consolidation" purpose had the highest tendency to default.

4 Modeling

A machine learning paper would not be complete without discussing the model. In this section, we will explain our methods for preprocessing to making predictions.

4.1 Preprocessing

Before we can begin training a network, there is some work that needs to be done. The first step is to load the data from the CSV file into an array. For this we used the pandas package. Once we got the data loaded, we needed to filter out the categories we didn't want to consider, and changing strings into integers. We created a function that went through each row of data and selected the variables of interest, converted them to integers when necessary, and replaced null values with zero. One this was done, each variable was normalized by dividing every variable by the maximum value observed for that variable. This helped the reduce the amount of time it took our model to learn.

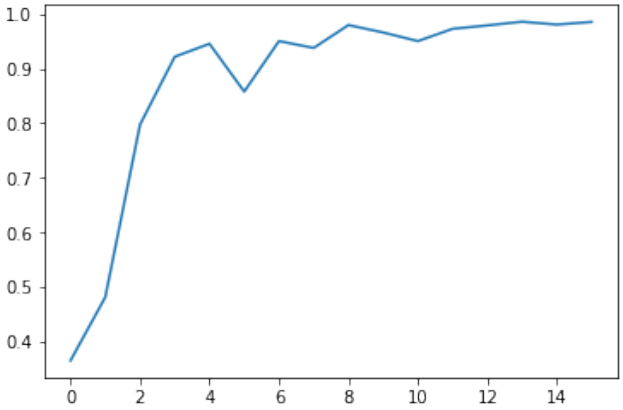
4.2 The Network

Our model was a fully connected model with an input layer that had 39 nodes. The model also has six hidden layers each consisting of 1024 hidden units and each one using the tanh activation function. The output layer is a two dimensional output with one node corresponding to a good loan and the other to a bad loan.

4.3 Training

To train the network, we used a batch size of 128 and in every batch we balanced the data by making sure 64 of the data points were good loans, and 64 were bad loans. We did this to avoid any bias there could be caused by the fact that 83% of the loans were good. For the loss function, we used softmax cross entropy with logits that is built in to tensorflow and tried to reduce the mean. For the optimizer, we used Adam with a learning rate of 0.00001. We allowed the network to run for up to 100,000 iterations or until the accuracy was better than 98% which often took only a few thousand loops. The following graph shows the iteration number (in hundreds) vs the accuracy.

Iteration vs Accuracy



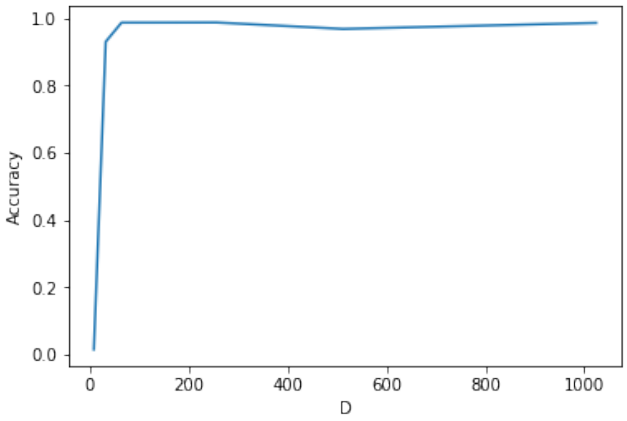
4.4 Compression

After training, we decided compressing the network would be a good idea. This would speed up the classification process which would allow financial institutions to process more loan applications per day which can increase profits, and make the application process less frustrating for the applicant. The method we used for compression was to take the singular value decomposition of each layer and take the N most important nodes. The following graph shows how the accuracy changes as N increases. Not surprisingly, the more hidden units, the higher the accuracy. The only exception to this rule is when compressing to 512 hidden units per layer, the accuracy drops, but recovers when using all 1024 nodes. From the graph, it is easy to see not all 1024 hidden units are needed to make good predictions.

4.5 Compressed Network Training

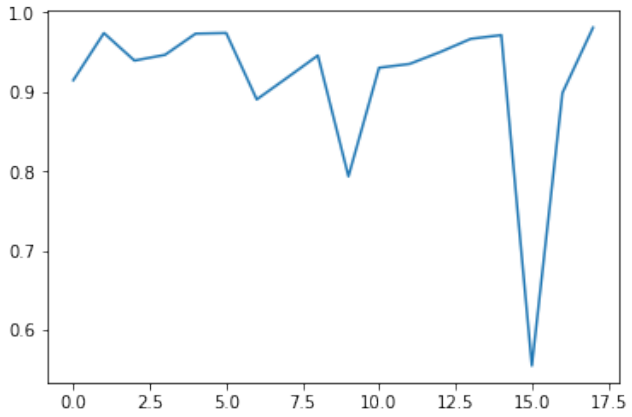
After viewing this data, we decided to compress the network to twenty hidden units per layer. This yielded an accuracy of 91.4% which isn't bad, but we knew it

Hidden Units vs Accuracy



could be better. We decided to use the same methods mentioned in section 4.3 of this paper to train the compressed network. The learning wasn't as smooth as it was for the 1024 unit network, and it took slightly more iterations, but it did get the accuracy we were looking for. in the end, it had 98% accuracy. The following graph shows the accuracy at each iteration (in hundreds) of the network during training. The accuracy is 0.5% lower than full network but is uses 1004 less hidden units per hidden layer.

Iteration vs Accuracy



5 Conclusion

Overall, we were satisfied with our results. 98% accuracy for a problem like this is a great result. Predicting if someone will pay a loan back or not is not an easy task, and being able to do so with 98% certainty is a great achievement. We believe this method for prediction could increase profits for any financial institution that use it, and can help automate the loan process making life easier for borrowers and saving financial institutions money on labor and losses from bad loans.

6 References

1. "Analysis and Modlling of Lending Club Loan Data." Kaggle, www.kaggle.com/adityasheth/analysis-and-modelling-of-lending-club-loan-data
2. Kan, Wendy. Lending Club Loan Data. Kaggle, 18 Mar. 2019, www.kaggle.com/wendykan/lending-club-loan-data.
3. "Lending Club Risk Analysis and Metrics." Kaggle, www.kaggle.com/janiobachmann/lending-club-risk-analysis-and-metrics.