

**Lending Club Dataset**

**Final Report**

**Judah Drelich**

**Abstract**

Since 2000 B.C.E. Loans have been an integral part of society and economics. Lending Club is an online platform that facilitates lending between investors and institutions to borrowers who need short term liquidity. Lending Club publishes data on the borrowers and their loans both current and finished that can be found on Kaggle.com. While the dataset did not have enough data to conduct an audit of how good each loan is, I was able to create a market for the remaining term and debts of current loans based on the results of the finished loans. While I do not have a definitive answer on the precision and accuracy of the market, I use LIME a machine learning explainer, to interpret the valuation of each loan. The best model had a 13% error rate on average but given unaccounted for macroeconomic factors, and a potential skew in the data, with additional datasets and updated models this could be remedied.

**Introduction**

Debt is the backbone of all economies. It allows for non-simultaneous transactions to occur by binding the buyer to the seller until the buyer can repay the seller. Many Americans have entered these pacts without having a clear way out. By Q2 2023, total credit card debt was $1.03 trillion and student loan debt of $1.57 trillion.[[1]](#footnote-2) One way for people to overcome debt is to take on loans that give an infusion of cash so that they can stave off their creditors. This has become increasingly common in the U.S. as “22.7 million Americans owe a collective $232 billion in personal loans, more than double the $117 billion owed in 2017.”[[2]](#footnote-3)

Lending Club is an online lending service that matches borrowers to investors to facilitate loans at lower interest rates. For the borrowers, Lending Club offers a chance to get out of debt or pay for an expense while the investors get a solid return on their investments. Some of Lending Club’s loans are shown to financial institutions but the majority are held by investors. The dataset is taken from Feb 1st 2016 and contains information on each Lending Club loan both current and finished.

**Target Variable**

The first step in data wrangling is to separate the features (X) from the target variable (y). When I initially started this project, I had believed `*loan\_status*` was the target variable because it gave a clear description of the state of the loan and the project description on Kaggle agreed. After much data wrangling and investigation, I realized that it was hard to use many of the features for the target variable `*loan\_status*`. Too many of the features have information that was procured after the loans originated making it not clear if the variable is a part of the features or the target variable. An example of this is `*next\_pymnt\_d`* which is a variable that represents the date that the next loan payment is due. If I wanted to create a model that evaluated potential loans on behalf of investors as an audit of Lending Club, it would be impossible get next payment dates as those only occur during the term of the loan and are closely tied to the status of the loan in a non-causal way.

To use all the features of the dataset, I need to shift the time of the question so that it is concurrent with the dataset. That way `*next\_pymnt\_d`* is a variable that can be recorded and used in the model. Instead of trying to figure out the value of the loan when it is shown to investors and institutions, I can find the value of the remaining loan. This can create a market or evaluation for the value of the existing loans that would be valuable for loan investors or institutions who are thinking of selling off their loans for immediate liquidity.

Now the target variable is the fraction of the loan amount that the borrower will be able to pay back. With this information I separated out every column of the dataset that had information on the amount that the borrower has already paid and the amount that they owe. With those columns I can create the fraction while the remaining columns will become the features.

**Data Wrangling**

One of the best tools for at the start of any data science project is `*ProfileReport`* from `*Y-data Profiling`.* While my computer was unable to perform the function on this dataset, the issues that it highlights are a good guideline of what to make sure is good in the data. The five things that `*ProfileReport`* checks for are:

* Missing values
* Constant features
* Skewed features:
* Large Multicollinearity
* High numbers of zeros:

The simplest of the five to fix are constant values as they add no relational information to the dataset and can be represented in a single cell.

To clean the data, all the features must have only numerical values. Some columns contain numbers but need to cleaning to become fully numeric whereas others are purely categorical and either need to be one-hot encoded or dropped if they have too many unique values.

The imputation of values for this dataset is the most challenging aspect of this project. There are many columns that have large swaths of missing values that need reasonable values filled in to allow the algorithms to capture the existing patterns in the data instead of artificial ones coming from synthetic data.

My assumption was that most missing values were zeroes that people forgot to fill out on forms or didn't bother with. For instance, `open\_il\_12m` refers to the number of opened installment accounts in the last 12 months. If there isn't any information on the number of installment accounts, then there likely aren't any installment accounts.

Values such as `mths\_since\_last\_delinq` or months since last delinquent are different because if someone was never delinquent then the correct answer isn't 0 which would imply that they are currently delinquent but an exceedingly high number. The higher the value, the better the outcome on their loan. I can use 1000 since it is 83 years’ worth of non-delinquency.

**Multicollinearity**

Multicollinearity is the idea that more than two features share the same line. This is bad because it becomes hard to assign correlation or causation to any given feature since the math doesn't favor either feature for the outcome. Below is the starting point of the multicollinearity for the entire dataset:

A screen shot of a computer screen

Description automatically generated

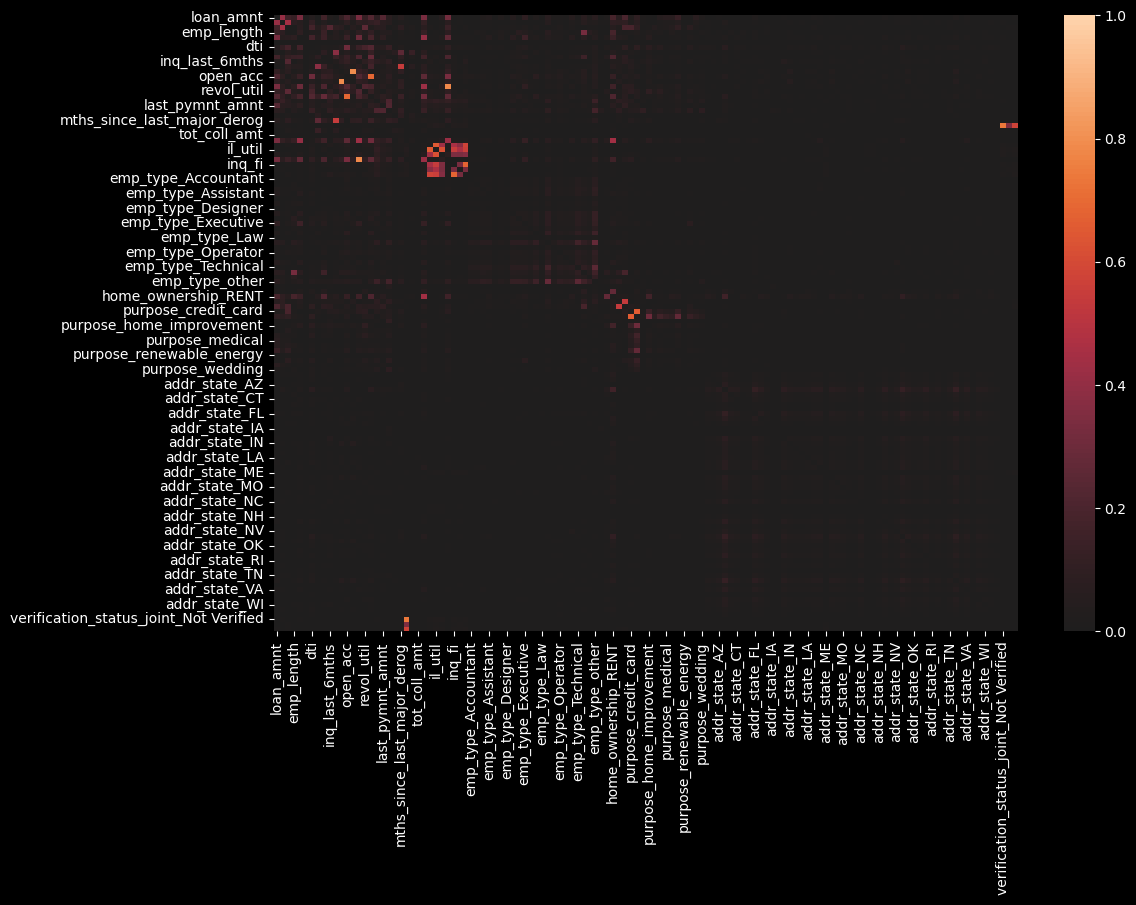
“While it may look like most of the data does not have a multicollinearity problem, the dark part of the graph holds all the dummy variables from the categorical columns. Because I dropped the first value from each of the categorical columns, it is mathematically impossible for those dummy columns to be collinear.

The heatmap shows a concerning square from features 27 to 42. Cleaning up the collinearity is not easy. While it may seem tempting to simply drop all the problematic columns, there is too much information in those columns to discard them.

The first step was to zoom in on bright spot of the heatmap to see if there were any groupings of features that had clear real-world explanations for their high correlations. This produced the following graph:”[[3]](#footnote-4)

A screenshot of a graph

Description automatically generated



“To make sure that I had fully ended the multicollinearity I calculated every remaining feature’s Variance Inflation Factor (VIF). The formula for VIF is:

(1)

Where R2 is the R-squared value that represents correlations.[[4]](#footnote-5) When R2 approaches zero, the VIF approaches one whereas when R2 approaches 1, the VIF approaches infinity. The guidelines for an acceptable VIF score can range from 2.5 to 10 although anything over five is suspicious.[[5]](#footnote-6) With the remaining features that I had, 87 out of the 89 had a VIF of under 2.5 and 2 had a VIF < 3.03.

**Lasso Regularization**

Lasso Regularization is a technique for variable selection that uses regression to evaluate the effect that features have on a target variable. The idea is to add a penalty term to the regression loss function that has the coefficient or slope of the independent variable with respect to the target variable, multiplied by a parameter .

(2)

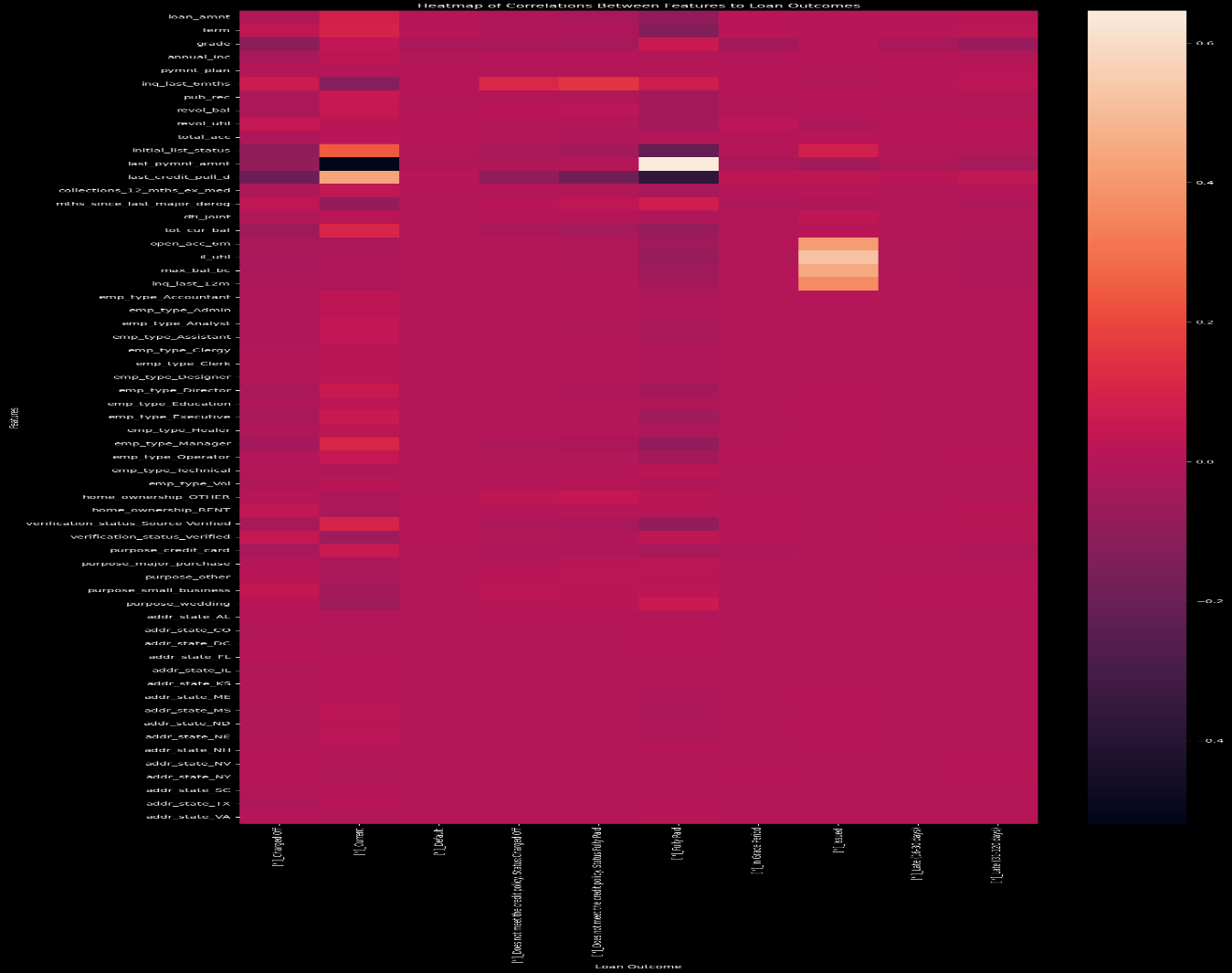
Where is the coefficient of the feature, *SSD* is the sum of squared distances of the observations to the regression line and is the penalty’s parameter. If there are many features, then we generalize equation (2) by summing the coefficients:

(3)

Lasso aims to reduce the loss in equation (3). If a feature is not important, then changing its slope will not move the regression line close enough to the data points to decrease the loss function with a non-zero value. If a feature is important, the regression line will move towards the data and minimize the SSD faster than it increases the penalty term. Since only features with non-zero coefficients are meaningful, I can discard all the features with a coefficient of zero.”[[6]](#footnote-7)

**Exploratory Data Analysis**

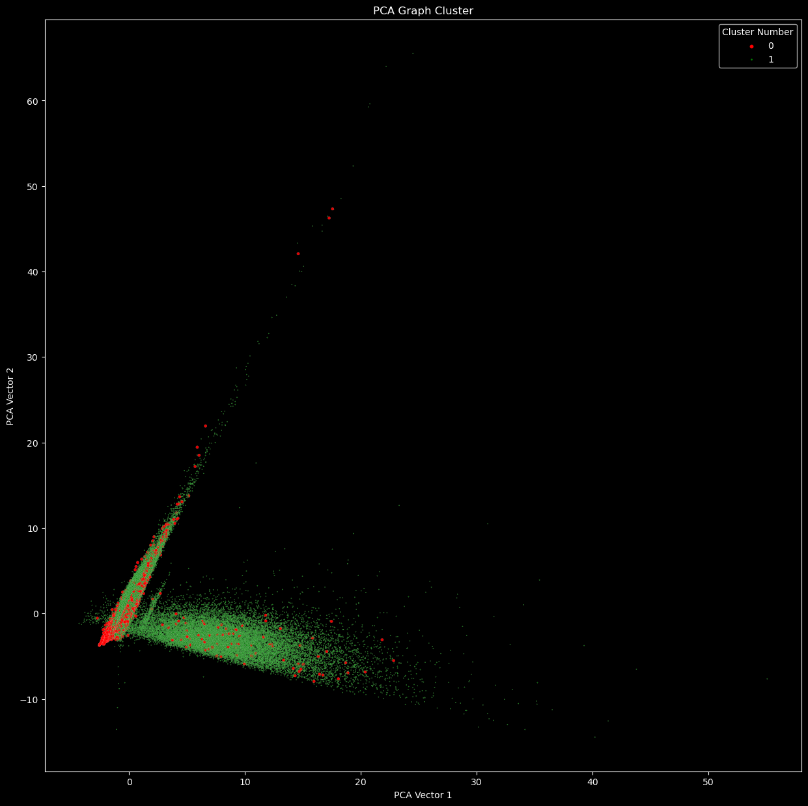
With the dataset fully cleaned and prepped the simplest thing to explore is the correlation between the features and the target variable. While it is true that `*loan\_status`* is no longer the official target variable, it is a really good proxy for the outcome of the loan and easy to analyze.



The graph above is a heatmap of the correlation between the features and each potential outcome of the loan. Most of the columns have little correlation because they are very infrequent. Overall there are only a few amount of features that seam to have an impact on loan status. The feature with the highest correlation is `*last\_pyment\_amnt`* to *Fully Paid.* This makes sense as loans that are fully paid will have consistently higher payment amounts than non-paid loans.

**Clustering**

One way to try and see deeper patterns in the data is clustering where instead of looking at singular features that would pop with correlation, now there can be multiple features that would set the data apart. Given the different outcomes of the loans, it would seem evident that there should be at least 3 groups of borrowers. Those that can Fully pay on time, those that will be late but will pay everything in full and possibly with interest, and those that will be unable to pay.

Before creating the clusters I visualized the dataset with Principal Component Analysis (PCA). PCA is method that reduces the number of variables by prioritizing variables with the highest variances and then combining them into a set number of vectors. This allows us to visualize high dimensional data to see if there are clear visual patterns. While it is possible for valuable information to be lost in this process if a low variance variable carries especially important information, PCA is still a good starting point. The graph above shows that while the bad loan outcomes are in red seem to congregate, there is enough dispersion without a clear structure that it is hard to continue with clustering in good faith.

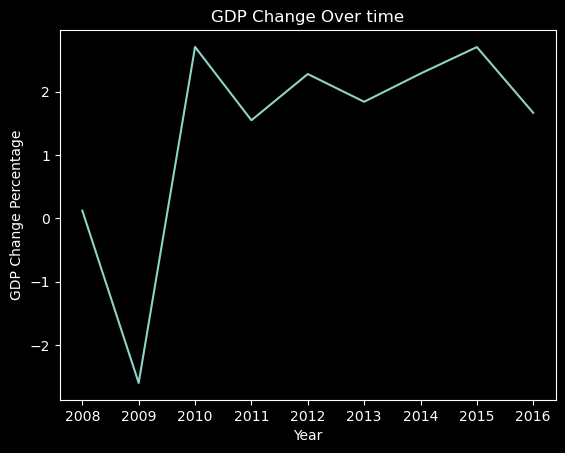
**External Factors**

Any financial system that occurs over many years is going to be subject to market forces. For borrowers, it can determine the amount of liquidity that they have left to pay off their existing loans. To check this I have graphed the last payment amounts against the time that they were made. A graph showing a line graph

Description automatically generated with medium confidence

There is a sharp falloff in the last two months of last payments which I’m not exactly sure why. Many people decide to make larger payments then the allotted amount which can drive up the mean payment amounts. It seems as though the last two months people are keeping up to date with their payments and aren’t trying to pay off the loans.

Now that I have data over a period of time I can also look at macroeconomic data from that time period to see if it matches up.



This graph is eerily similar to the previous graph. This makes sense as the economic winds will dictate if people feel that they are able to pay off their loans, if they need to pay off their loans and if they have the available capital to pay off their loans. The problem is that none of this information is in the dataset and to try and use macroeconomic data for the future means having access to a reliable forecast which is impossible to find.

**Preprocessing**

Only a part of the dataset has loans that have finished. While the end goal of the project is to create a market with active loans, the only way to evaluate the strength of the model is through data points that already have a result. Therefore, I can separate out the finished loans from the unfinished loans by the `*loan\_status*` column that describes the current state of the loan.

Originally I had thought that `loan\_status` was the target variable but the active loans had too much information that came after the time of origination. I pivotted to trying find the amount that the loans are worth for active loans. To do this I can calculate the expected return on each loan from lending club. Lending club does not compound interest for its loans so the formula is:

I can then calculate the amount that has been paid back by this formula:

Where `Total Payment` is the amount that has been paid back from the loan, `Total Late fees` are the late fees that the borrower would have incurred if they were late with a fee. `Recoveries` is everything that that the company was able to recover after the loan was charged off.

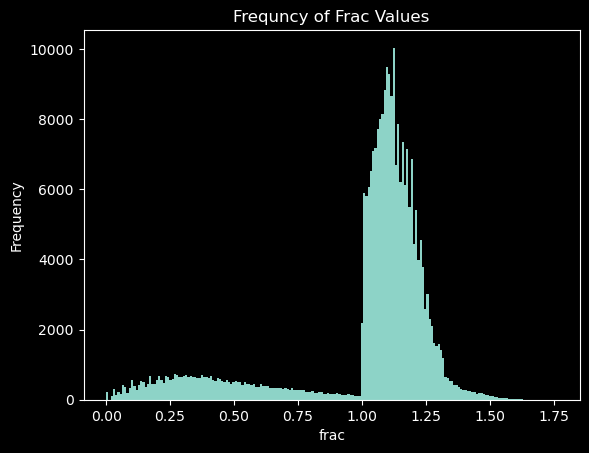
`frac` is the fraction in which

The active loans are slightly different in that the expected payment is what is remaining on the loan instead of the entire loan. To calculate this I did:

Where `Actual Payment' is the same as the cell above. The idea is that the expected payment is the entire loan's expected value minus the money that the borrower has already paid.

**Modelling**

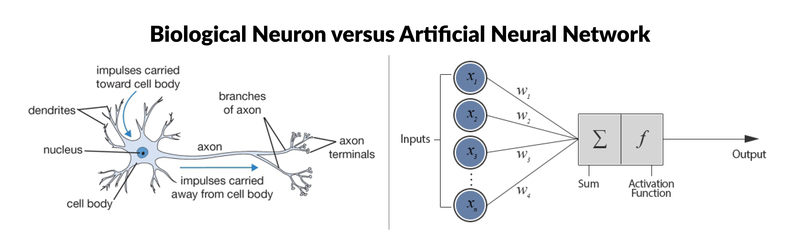
**EDA**

Before I start the modelling sections, it is important to know how a little bit more about the `*frac`* variable. Below I created a histogram of the distribution of frac to gain an idea of what the model is trying to predict.

Most of the borrowers paid off their loans so the frequency around 1 jumps. There is also a decline the closer `frac` gets to 1 because it is less likely that a loan that is close to being paid off won't be fully paid off rather than a loan that the borrower has no hope of paying off will not be paid off.

Neural Networks

“The term neural networks is inspired by the structure and function of biological neurons. The diagram below shows both a diagram of a neuron, and a node of a neural network.



[This Photo](https://devopedia.org/artificial-neural-network) by Unknown Author is licensed under [CC BY-SA](https://creativecommons.org/licenses/by-sa/3.0/)

In both, a node receives and processes many inputs, then produces an output. When many nodes are connected, they form a network that can emulate non-linear complex systems. The formula for a node in a neural network is:

(7)

Where is the activation function of the node, is the weight of a given input into the node, is the value of a connected node, is all the inputs into the node, and b is the bias of the node. The output of one node can become one of the inputs of another node or it can be the final node in the network that bears its prediction.

When a neural network is training, the network updates its model weights to minimize the difference between its predictions and the target values in the training set. This is achieved by using the derivative of the function representation of the entire model. The network moves in the direction with the lowest derivative value to find the minima until there are no more directions with negative derivative values. When the model finishes, it can be used to predict a test set that it has not been trained on but has the same type of data as the model to make predictions.”[[7]](#footnote-8)

The smaller neural net that I created has an input layer of 61, 12 hidden layers of varying numbers of nodes and a final output layer of a single node. The activation function that I used for every layer except the last was Relu. which stands for rectified linear unit. The Relu function is:

(8)

Where the idea is to discard the negative nodes that are not helpful while allowing the useful nodes to equally contribute to the model.

The final layer does not have an activation function and outputs the sum of all of the inputs from the previous layer.

**Metrics**

The main metric that I’m using to measure the success of the model is mean absolute error (MAE) which is:

(9)

Where represents the difference between the estimated `*frac`* and the actual frac for any loan *i,* and *n* is the number of loans in the data. This gives a sense of the total amount of error that is occurring without the errors cancelling out due to some being negative and some being positive. I decided not to use mean square error (MSE) nor root mean square error (RMSE) which are:

(10)

(11)

Because the results are significantly harder to interpret. While adding a square does increase the impact that outliers or larger errors have, to judge the effectiveness of the model I need to find the percentage error. With RMSE its not clear what that percentage means whereas with MAE on average the model will be off by the percentage error.

**Training**

A graph with a line

Description automatically generatedThe model was able to train well as shown in the graph below.

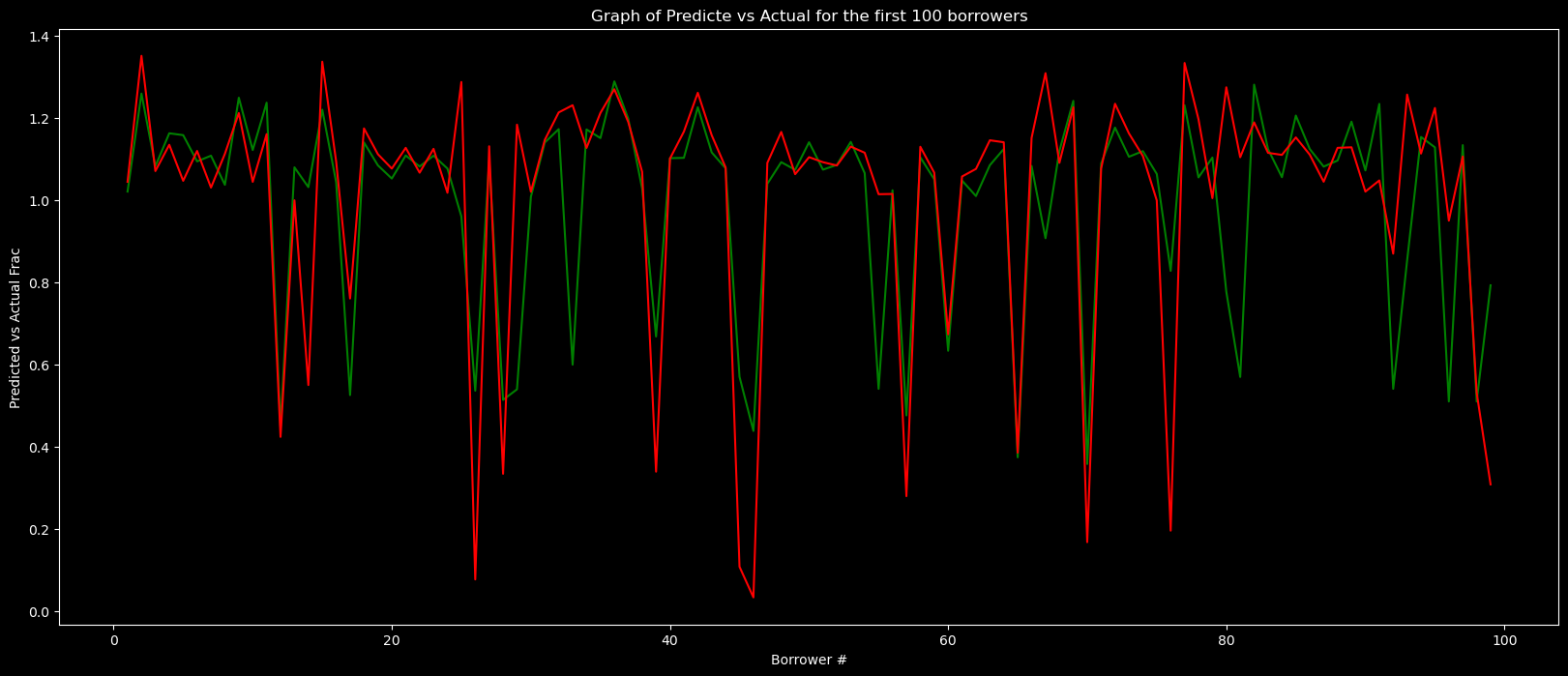
An epoch occurs each time the model runs through the entire dataset. The more epochs a model gets the more chances it has to train the weights of the model to optimize the outcome. The cost is the difference between the model’s prediction and the correct answer. The sharp decline in the first epoch and gradual decline onward shows that there are clear patterns that warrant deep learning and that it does make sense to proceed. The training produced an MAE of 0.13 which given that the average value of `*frac`* is around 1 should mean that the percent error is close to 13 percent.

**Cross Validation**

To show that model is generalizable it is important to use cross validation as a way of testing the model without using the test set. This way I can see how well the model performs when it is able to use new data but it will not have the biases created by training on test data.

I ran the cross validation 5 times and each of them had an MAE in the 0.17 to 0.2 which is slightly higher than the MAE of the training set but not egregious enough to stop using model.

**Testing**

 The model had an MSE = 0.0490, RMSE = 0.2213, MAE = 0.1386, and Mean Percent Error 14.6275%. While it is easy to understand MAE and percent error the root mean square error is more concerning. Root mean square is used to highlight potential outliers in the dataset since higher squared values increase faster while taking the square root equalizes it with MAE. This shows that the model is most likely accurate for a lot of guesses but is really off on others. The graph above shows the first 100 loans with predictions in green and actual values in red. What sticks out is that while the predictions are able to follow the same shape as the actual values, there are some values that it either significantly under or over estimated. Further research would reveal if these cases were easily diagnosable by a financial professional and don’t need a better model or if the model is not performing well enough.

A graph of a computer generated data

Description automatically generated with medium confidence Below is a histogram of the error for the neural network. This gives a much better visualization to how the model performed and the number of outliers that there are.

It is reassuring that there is no heteroscedasticity as that means that there isn’t an intrinsic flaw with the model. There appears to be a lot of values outside of -0.5 and 0.5 which is concerning given that the largest values of `*frac`* is 1.76.

A more direct way of looking at the results from the model is to overlay a histogram of the predictions onto the actual values.

A graph with a black background

Description automatically generated

Interestingly the model had an awfully hard time predicting loans that did not recoup at least all of the expected investment. Given an outcome like this it is possible to consider a second level of models that could transform the underperforming loans into the distribution of the actual loans.

**LIME**

LIME stands for Locally Interpretable Model Explanation. It is an attempt to explain deep learning models in a way that humans can understand. While the full process of a model’s prediction may never be known, at the simplest levels there is a way to have a basic understanding of the variables that the model cares about. LIME does this by evaluating the model’s prediction compared to a hypothetical linear instance of the model at the point of the observation in question.

To create the local instance, LIME creates a binary representation of an observation x.

This binary representation may have significantly more dimensions depending on the type of data that is in x. The reason that it becomes a binary representation is that lime is trying to find the impact of each of the variables. Representing them in binary will show whether they have a clear effect or not. Then LIME creates an arbitrary number of random samples through perturbations by changing some of the values in to 0. Then we use an inverse mapping function to convert all the binary perturbed samples back into nonbinary samples. The samples weighted by the following kernel:

Where is a weighting function with respect to the observation x. is a distance function between the original observation and the sampled point. This ensures that the closer the sample is, the more impact it will have on the explanation since the original model is complex and nonlinear but the explanation needs to be simple and linear.

To find a model that can interpret other models there needs to be two things, fidelity, and interpretability. Fidelity is represented by how close the new local linear model from the sample is to the original model. In equation form it is

Where is the result that would have happened if the sample point had been in the original model and is the result of the binary perturbation in the simpler explainable model. This equation sums the squares of the difference between the models and then weighs each point on how far it is from the original observations. If it is a small number, then it is a faithful adaptation.

Interpretability can also be measured by the complexity of a model. For example, a random forest could have 10 layers and would have a depth of 10, Neural networks could have a 500 nodes and have a complexity of 500. The more complex the model, the harder it is to find reasonable explanations for its conclusions. As such the LIME is represented by the following equation.

Where the represents a function that is the closest to the original model while also being the simplest. To find the variables of interest from this construction, we can use Ridge regression. The way to represent the complexity of a linear model is the number of coefficients,

Where represents the coefficient of the ith variables in the local linear model. When combined the entire equation becomes.

This equation bears a shocking resemblance to eq. (3) which is the equation for lasso regularization. The middle term is a Sum of Square Distances or SSD, and the other term is a penalty term with all of the coefficients of a linear model. While it isn’t Lasso, since the coefficients are squared, It is perfect for Ridge regularization which uses a squared penalty term. By performing ridge regularization, we can attain the similar results as Lasso and understand not just which features are the most important, but why they are the most important.

 Below is a graphical representation of how LIME works, the white vs. blue parts represent a complex non-linear model whereas LIME is creating the simple Dashed black line as a much easier model to interpret. The observation in question is the bolded red plus sign that the model is trying to explain. When we examine close to the point of interest a linear model appears to be effective but as the model curves and turns it is no longer helpful. [[8]](#footnote-9)

With the LIME python package I can quickly use LIME to explain any given loan and determine the most impactful features. Below is an example of the output of the LIME Package

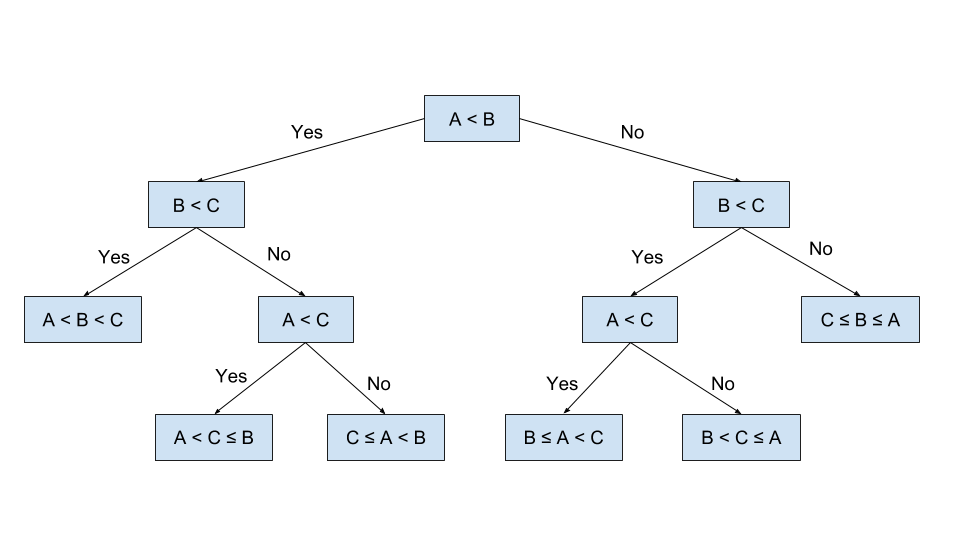
A screenshot of a computer screen

Description automatically generated

The predicted value on the left is created by the possible values that the inverse mapping function creates when transitioning from the binary vectors to numerical vectors. The graphical representation shows of the importance of the features from the ridge regression. Finally the last column shows the actual values that were used in the LIME analysis. With numerical data is less meaningful than categorical data or binary data but it is still interesting.

**Random Forests**

“Random forests are a collection of decision trees that predict a target variable based on the features of the dataset. Below is an example of a decision tree.



[This Photo](https://elf11.github.io/2018/07/01/python-decision-trees-acm.html) by Unknown Author is licensed under [CC BY](https://creativecommons.org/licenses/by/3.0/)

**Decision Tree**

When decision trees are applied to datasets, they have specific ways of creating decisions. If the data is binary, then the decision tree will create two branches for each of the binary values. If the data is categorical then the decision tree can choose any number of the categories to become another branch. If the data is numerical, then the decision tree creates a cutoff value where all observations with values below the cutoff go to one branch, and the rest go to another branch.

The decision tree then partitions the least homogeneous feature of the dataset as measured by the Gini index which is:

(12)

Where is the probability of each of the classes and is all the classes. The decision tree will continue to iterate through this process until it reaches 100% purity and has nothing left to partition. When all the observations in each group are the same, then y-class that is the most popular for that group becomes the prediction of the model.

This can lead to overfitting as the decision tree has learned all the patterns of the training data that may or may not be helpful. To fix this problem we can create a forest of decision trees with bootstrapped datasets. Bootstrapping is when the data is randomly sampled with replacement to create another dataset that has the same parameters as the original dataset but does not carry the same noise. When all the trees in the forest have finished their processes, they vote on the correct prediction. The prediction that the greatest number of trees support is the prediction of the forest.

There are many different hyper-parameters for a random forest model. Some examples are:

* Number of decision trees
* Homogeneity metric
* Maximum depth of an individual tree

The only hyper parameter that I had time to test was the number of decision trees because the calculations took significantly more time as the number estimators increased. It is possible that there are some hyper-parameters that would have produced significantly better results, but I did not have the time to find them.”[[9]](#footnote-10)

When trying to find the number of trees for the forest, I tested 5, 50, 100, 150, 200, 300, 600 estimators. After 600 estimators the additional estimator does not provide significant improvement to the model. Given the amount of time that it takes for my computer to evaluate the different numbers of estimators, 600 is the maximum amount.

**Testing**

The Random Forest model had a MSE = 0.0309, RMSE = 0.1758, MAE = 0.0872 and a Mean Percent Error of 13.6849%. The discrepancy between the RMSE and the MAE shows that there are a lot of outliers. Below is a representation of the first 100 loans with the predictions in green and the actual values in red. A screenshot of a computer screen

Description automatically generatedWhat is interesting is that for every single observation the model has the correct pattern but gets the amplitude wrong. This matches the RMSE discrepancy where many of the predictions are correct but some of the wrong estimates are severely wrong. Below is a graph of the distribution of error for the random forest.

A graph of a error

Description automatically generated with medium confidence

It is stark how much more concentrated all the values are around zero but the outliers are still there.

Finally, I can compare the histograms of the actual values to the predictions from the random forest.

A graph of a forest prediction

Description automatically generated with medium confidence

This is better than the neural network, but the model again is struggling to predict values near one. The distribution of values over one is interesting. If this had been a classification model where it was trying to Identify which loans will be paid back or not, it would score very highly.

Lastly I have the LIME output of a random loan using the random forest model.

A screen shot of a computer

Description automatically generated

Interestingly the random forest model used the `last\_pymnt\_amnt` more than the neural network and was able to get a better score.

**Conclusion**

This was not an easy dataset to analyze. The simplest question: “Is investing in a given loan a prudent use of money?” did not have enough features that would be available at loan origination to create a viable model. By thinking deeply about the type of analysis the dataset could support I was creating a market that could be helpful to current and prospective investors.

The most important methods from this paper are:

* Locally Interepretable Modelling Explanation (LIME)
* Random Forest Modelling

With most of the dataset comprised of current loans, it is impossible to evaluate the accuracy and precision of supervised learning methods. As such it is crucial to have an algorithm like LIME that can give an explanation of the model so that If someone is curious as to why the current valuation of a loan is a given number, they will be able to look behind the curtain.

The best model was Random Forest. I’m surprised that it beat out the neural network but there was no depth limit for the Random Forest and it had 600 estimators. It is possible that the Neural Network could have been improved with a lot more layers.

**References**

# Bibliography

Marco Tulio Ribeiro, S. S. (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *Arxiv*, 3-4.

1. [Household Debt and Credit Report - FEDERAL RESERVE BANK of NEW YORK (newyorkfed.org)](https://www.newyorkfed.org/microeconomics/hhdc.html) [↑](#footnote-ref-2)
2. [Personal Loan Statistics: 2023 | LendingTree](https://www.lendingtree.com/personal/personal-loans-statistics/) [↑](#footnote-ref-3)
3. Taken from the Final Report on Network Intrusions [↑](#footnote-ref-4)
4. [How to Calculate VIF in Excel - Sheetaki](https://sheetaki.com/how-to-calculate-vif-in-excel/) [↑](#footnote-ref-5)
5. [Variance Inflation Factors (VIFs) - Statistics By Jim](https://statisticsbyjim.com/regression/variance-inflation-factors/) [↑](#footnote-ref-6)
6. Taken from the Final Report on Network Intrusions [↑](#footnote-ref-7)
7. Taken from the Final Report on Network Intrusions [↑](#footnote-ref-8)
8. (Marco Tulio Ribeiro, 2016) [↑](#footnote-ref-9)
9. Taken from the Final Report on Network Intrusions [↑](#footnote-ref-10)