UGA Data Science Competition 2021

Ayush Kumar, Chloe Phelps, Faisal Hossain, Nicholas Sung

April 15, 2021

Contents

1 Exploratory Data Analysis and Preprocessing

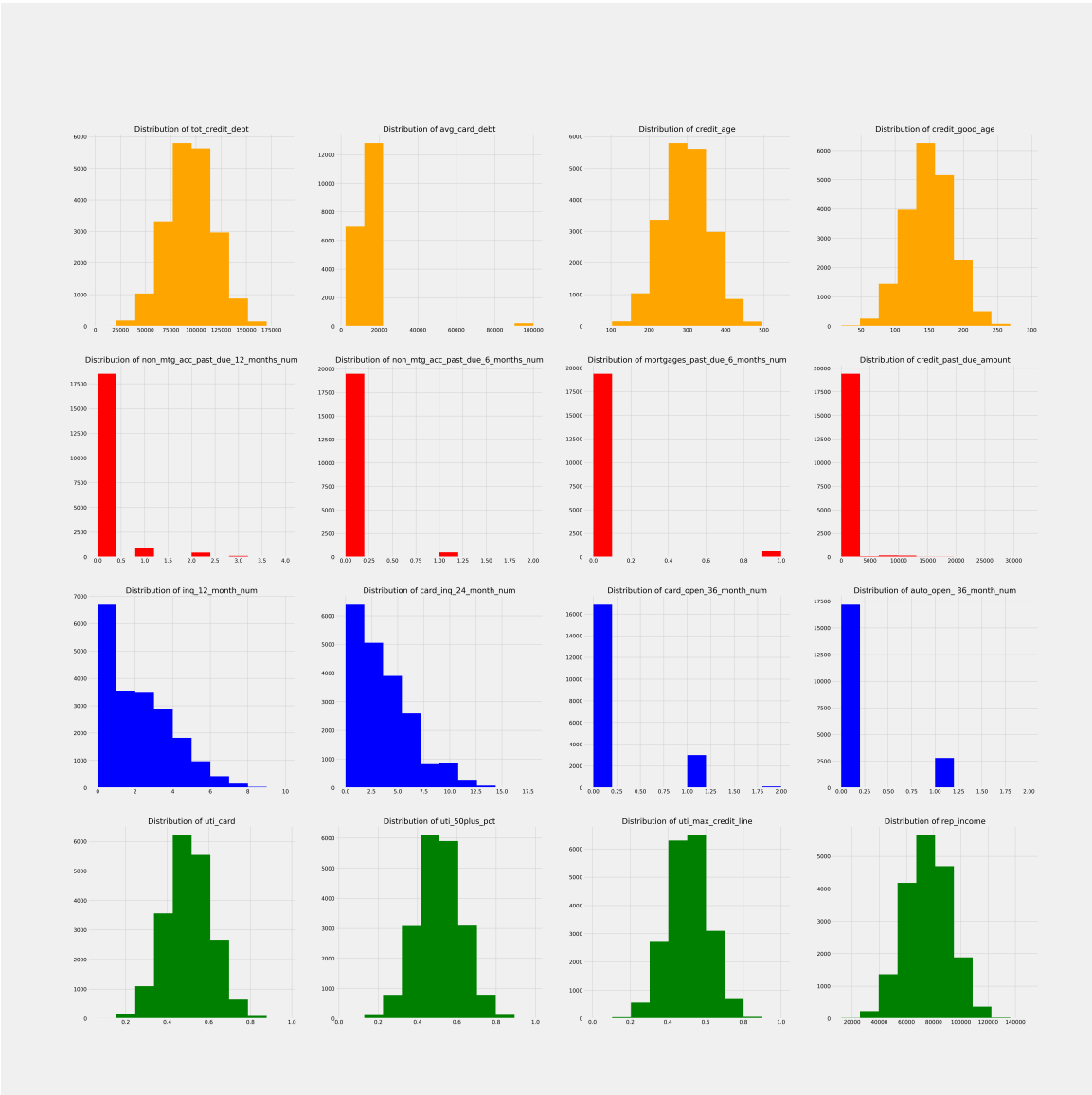
1.1 Summary Statistics for Continuous Variables

We begin our exploratory data analysis by looking into the data, and determining the type of the data for each column. For continuous numerical data, we want to visualize the distribution using histograms as well as taking a look at the following summary statistics: mean, standard deviation, the minimum and the maximum. For the categorical data we want to take a look at the possible categories, and the frequency of each category.

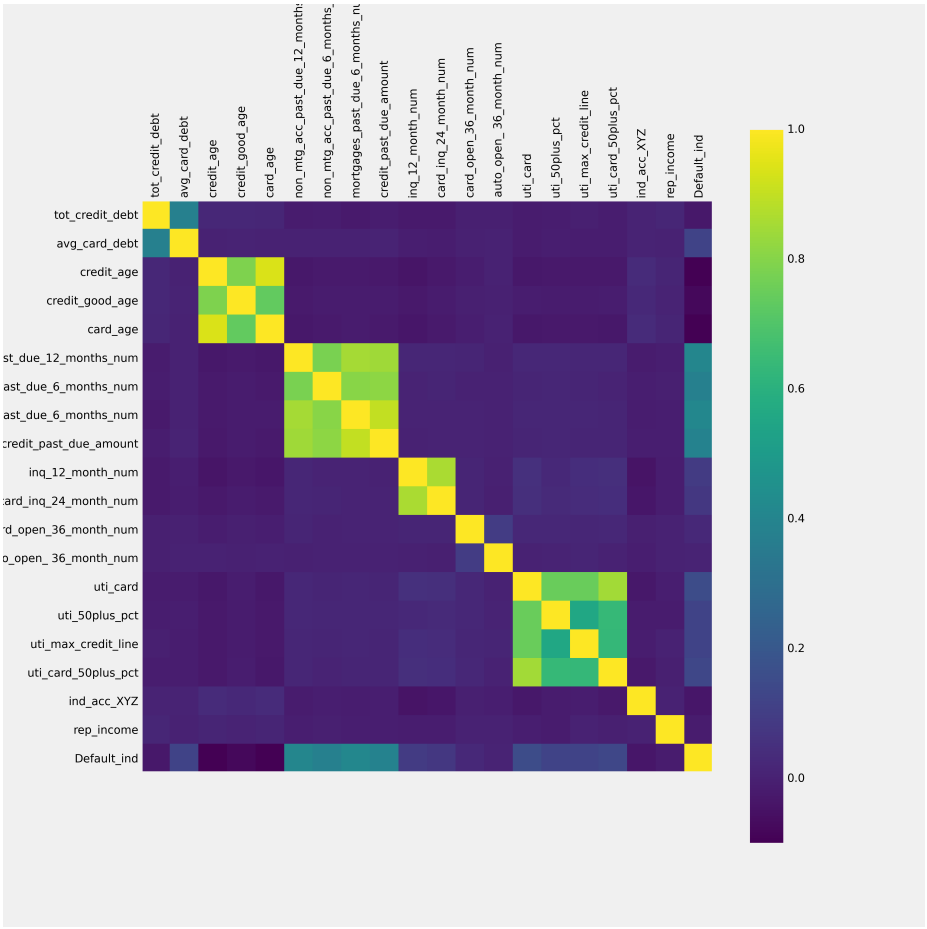
Using pandas and the information sheet here are the numerical variables:

	mean	std	min	max
tot_credit_debt	94563.702530	23546.443862	2367.430000	188890.960000
avg_card_debt	14088.235475	9314.495936	2363.120000	99999.000000
credit_age	296.697000	61.711702	54.000000	545.000000
credit_good_age	149.771750	34.016476	21.000000	296.000000
non_mtg_acc_past_due_12_months_num	0.111350	0.433890	0.000000	4.000000
non_mtg_acc_past_due_6_months_num	0.027400	0.171903	0.000000	2.000000
mortgages_past_due_6_months_num	0.030200	0.171142	0.000000	1.000000
credit_past_due_amount	329.287867	2073.899357	0.000000	32662.980000
inq_12_month_num	1.762700	1.740816	0.000000	10.000000
card_inq_24_month_num	3.409600	2.926697	0.000000	18.000000
card_open_36_month_num	0.163050	0.386099	0.000000	2.000000
auto_open_ 36_month_num	0.141000	0.349607	0.000000	2.000000
uti_card	0.503157	0.109354	0.065120	0.969289
uti_50plus_pct	0.511007	0.113456	0.033749	0.988964
uti_max_credit_line	0.507629	0.108624	0.005174	1.000000
rep_income	75499.511666	16361.955146	12000.000000	150000.000000

By looking at the descriptive statistics we can see if there are any major outliers, and determine how we can use these variables in our data. We see that minimum and maximum values for many of the continuous variable are very extreme for Total Credit Debt, Average Credit Debt, and Credit past due amount. We also see that many variables seem to have very similar distributions, which may be a problem in regards to multi-collinearity. To investigate this we can create a correlation matrix and re-scale certain variables to try to preserve information, while combating multi-collinearity.

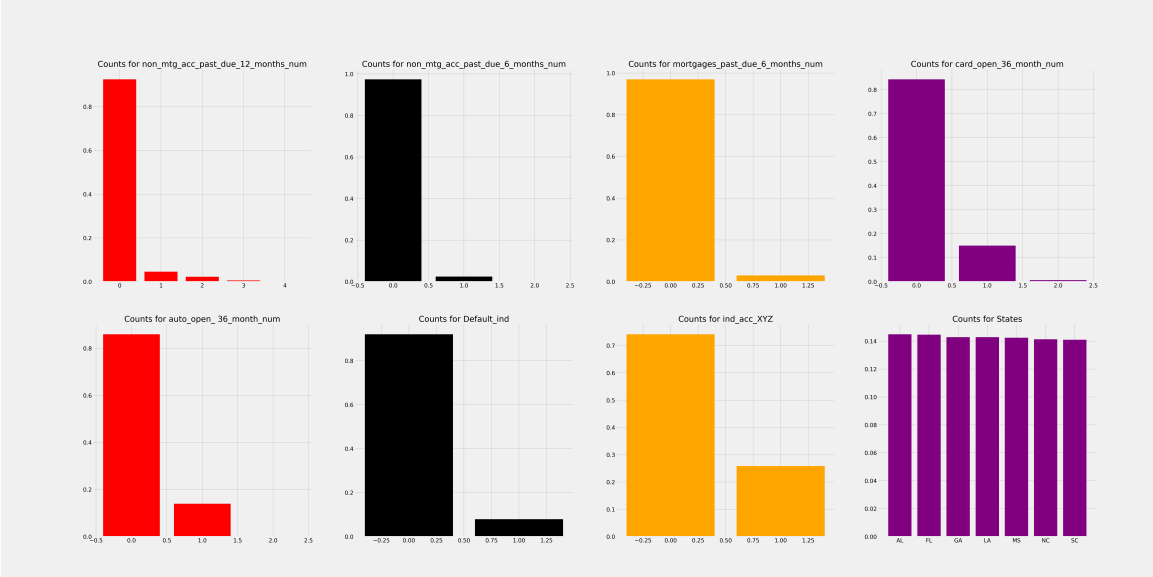


Looking at the histogram gives us a better picture of the distributions of the variables because it allows us to visualize the shapes of the variables. We can see that the outliers for Average Credit Debt are seriously impacting the distribution in comparison to Total Credit Debt. This visualization also allows us to look at some variables which by their description seem to be numerical, but display behavior that is more characteristic of categorical variables. Variables like the number of mortgages past due or non-mortgages past due would be better suited to being dummy variables rather than continuous ones.



Taking a look at the correlation matrix we can see multiple points of high collinearity between variables that are similar in nature due to the time that they are recorded or other inherent similarities. Based on this correlation matrix we can see that some of the variables may need to be rescaled to preserve as much information as possible, but bring down the high collinearity. Another approach would be to carry on with all the variables and not put too much stock into the individual variable t-tests and instead focus on joint F-tests and ensure that the variables are being properly captured.

1.2 Summary Statistics for Categorical Variables



We can take a look at the counts and the relative frequencies for the variables and make some observations. For many of these variables it would be wise to make multiple dummy variables to allow for better predictive models. States and card_open_36_month_num should be split into multiple variables for the models. The other variables are either in a form that already mimics a dummy variable, or they can easily be converted into 0 or 1 dummy variables because given the small frequency numbers higher than one appear it would be superfluous to have multiple dummy variables for the different variables. Further analysis is required for the State dummy variables because there need to be substantial differences between the different states to add them as dummy variables.

The analysis we choose for adding state dummy variables is to simply take a look a random sample of observations and plot a subset of two variables that are representative of the data available and plot them by state. Based on the graphs on page 5 we can see that there are some patterns however weak in the data and we should take them into account when building the model. The significance of the state dummy variables can be assessed at a later time after the models have been built.

1.3 Model Selection

In addition to the Logistic Regression we were asked to choose between a Feed-Forward Neural Network, a Random Forest Classifier, or XGBoost. Due to the high number of possible interaction effects as well as the high number of continuous variables our team has opted to use the Feed-Forward Neural Network. These networks can be useful because they allow for a very high number of interaction effects without extensive testing. We will use a sigmoid function as our last response function to ensure the model output is similar to the output of the logistic regression.

Another challenge in model building will be the relatively low number of observations that do default on their credit debt. This makes it extraordinarily difficult to make sure our models are genuine improvements over the baseline model of predicting 0 for every observations. This baseline model has an accuracy of 92% so it will be important that we do not over-depend on accuracy as a measure of model fit. Precision, recall, and f1-scores will be much better indicator of model fit, and should be used when tuning hyper-parameters.

1.4 Preprocessing Transforms

We will be using data in numpy arrays, but first we must overcome some of the challenges of this approach. We must have numerical data for every observation, and there can be no missing values or the logistic regression model will not function properly. Our `load_data(path: String): X, y` function will conduct the following transforms on the data to prepare it for model use.

1. Import data from a csv into a pandas dataframe

2. Impute missing numeric values using the expected value of a column

3. Turn the following rows into dummy variables based on various levels: States, non_mtg_*, card_open_*, auto_open_*

4. Split the dataset into X - the data matrix and y - the response column

2 Logistic Regression Model

2.1 Baseline Model

Using the preprocessed data as discussed above we trained a logistic regression model from sklearn without tuning any hyper parameters. The following were the model coefficients and performance statistics on the testing dataset.

		precision	recall	f1-score	support
	0.0	0.93	0.99	0.96	4599
	1.0	0.72	0.20	0.31	401
accuracy				0.93	5000
macro avg		0.83	0.60	0.64	5000
weighted avg		0.92	0.93	0.91	5000

The coefficients and interpretation of them is available in Appendix A. The scores can help us assess model performance. The initial news seems to be good with an overall accuracy of 93%, but this is only marginal better than baseline model. The precision of the model is also ok with a score of 0.92. This means that of all the observations identified by the model as clients who will default we are correct around 72% of the time. The worrisome part of this model is the recall. A recall of 0.20 implies that we are missing nearly 4/5 of all clients who will end up defaulting. In other words this model is extremely good at identifying that a client will NOT default, but in the process misses a lot of clients who will default. This is all assuming a decision boundary of 0.5, which we will be optimizing. If the firm wants to minimize default risk it will have to increase the decision boundary, and if a firm wants to maximize the amount of credit they are lending the decision boundary should be decreased. Our team believes that defaults are much more costly, and we should try to tune the decision boundary in way that is beneficial to the end goal of reducing defaults.

It should also be noted that for a logistic regression we can directly interpret the variables and the effect they will have on the overall model. This will be discussed in greater detail in the explanation section of the report.

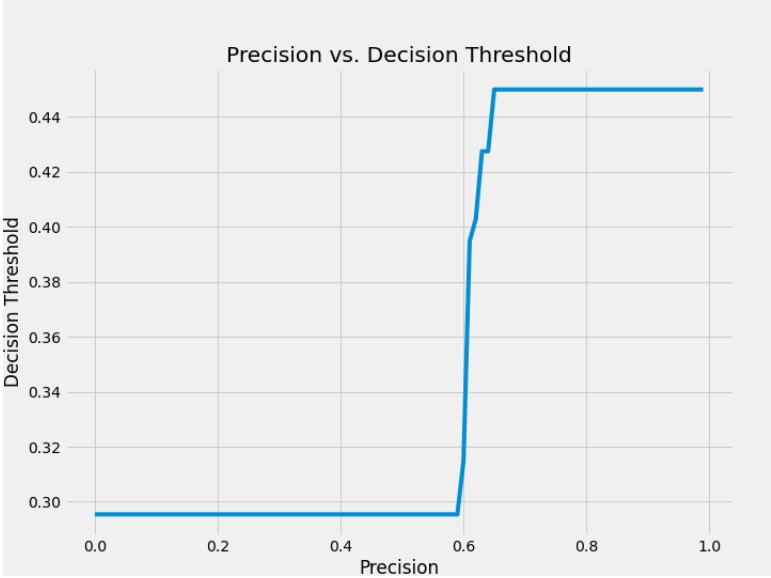
2.2 Tuning Decision Boundary

We will be using an algorithm that tries to maximize recall while ensuring that precision does not fall below a certain level. We will be able to test multiple values of the level we want to let the precision fall down to. We will be using a variable learning rate that decreases in a reciprocal fashion every iteration. We will be updating the decision boundary by subtracting the previous boundary times the learning rate for that iteration.

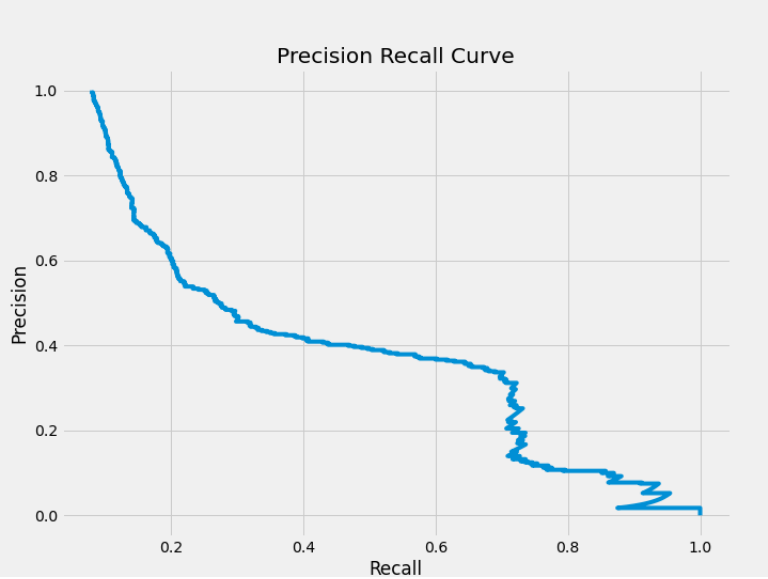
$$\eta_{t+1} = \eta_0/t$$

$$threshold_{t+1} = threshold_t * eta_t$$

This is to ensure that we arrive at a decision boundary that converges. The method signature is as follows: `tune_threshold(y, yhat, eta, plev): Double`. The plev is the level of precision we are willing to go down to and by generating many thresholds at different levels of precision we will be able to generate graphs of where the ideal decision boundary may lie. We assume an initial threshold of 0.5 and we will tune from there. Using this algorithm our group tested every precision between 0 and 1 with an increment of 0.1 and plotted the results.



Based on the results it seems that precision is not very sensitive for most values between 0.6 and 1.0 and between 0.0 and 0.6. This behavior is indicative of the precision recall trade-off in our model. To have an substantial gains of recall we would have to expand the number of applications the model will reject be quite a heavy amount. We can see that in the following precision and recall curve for our logistic regression model.



We see much of the same behavior we expected from our initial analysis of the precision-threshold curve. Around a precision of 0.5 we begin to see a substantial gain of recall at the cost of precision. Based on this our team recommends using a decision boundary based on preserving a precision of 0.5 and attempting to get much recall as we can. This will reduce our overall accuracy, but ensure that we catch more people who are likely to default on their credit loans minimizing the risk to the firm. This decision boundary for precision level of 0.5 happens to be 0.1514. Here is a summary of model fit after using this decision boundary on the test dataset. The decision boundary was tuned using the validation dataset.

		precision	recall	f1-score	support
	0.0	0.95	0.98	0.96	4599
	1.0	0.61	0.37	0.46	401
accuracy				0.93	5000
macro avg		0.78	0.67	0.71	5000
weighted avg		0.92	0.93	0.92	5000

When using the testing dataset we have greatly improved the recall of our model, while sacrificing a small amount of precision. This means the firm can still be extremely confident in knowing the model will rarely classify a non-defaulter as someone who will default, and will be assuming less risk in terms of trying to catch as many defaulters as possible.

3 Feed Forward Neural Network Model

3.1 Initial Model

The feed-forward neural network was chosen because of its ability to capture nonlinear interaction effects that may exist between variables. Some downsides of this type of network include its black-box nature, the cost of training, and the possibility of arriving at a local optimum rather than a global optimum. We must take extreme care to ensure that we do not accidentally converge at an incorrect minimum. Our model will be using a single hidden layer with a relu activation function

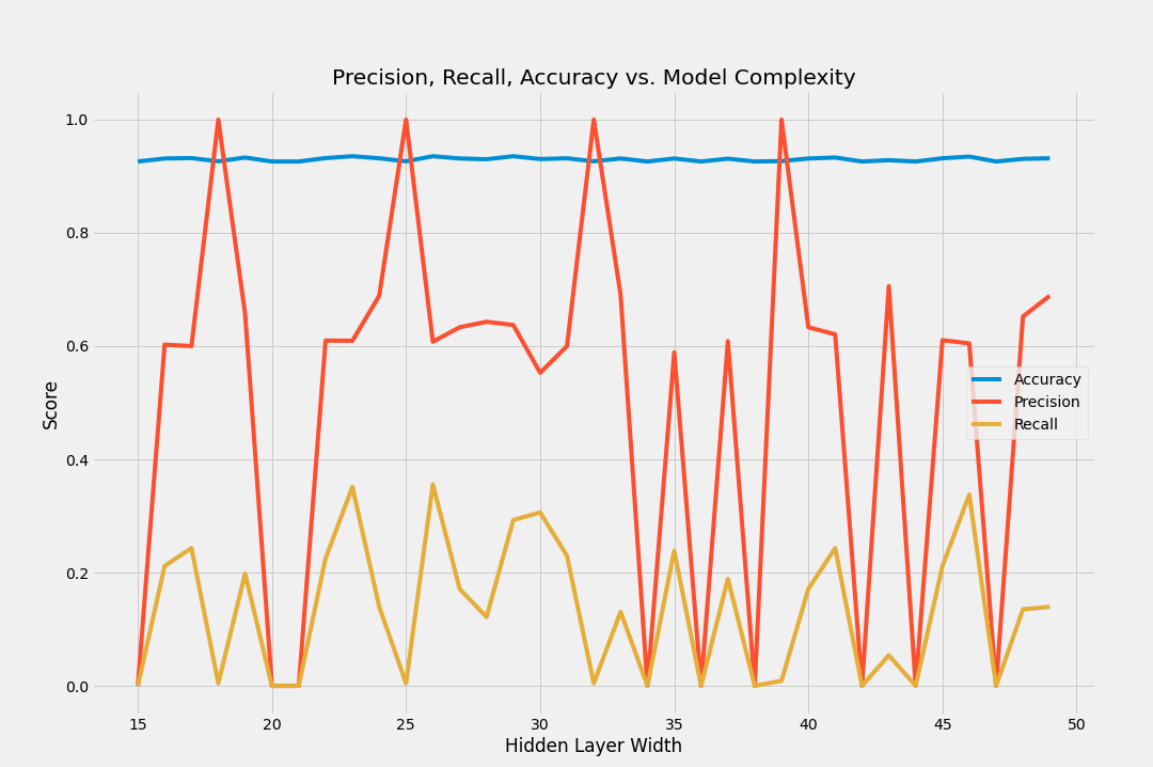
and a single output node of with a sigmoid activation function. We will start with a model with a number of hidden nodes equal to the number of input features and a loss based on binary cross entropy.

		precision	recall	f1-score	support
	0.0	0.93	0.99	0.96	4599
	1.0	0.76	0.19	0.31	401
accuracy				0.93	5000
macro avg		0.85	0.59	0.64	5000
weighted avg		0.92	0.93	0.91	5000

Our initial model has a similar problem to the original logistic regression, but before be begin tuning the decision curve, we can make adjustments to the model architecture.

3.2 Tuning Model Width

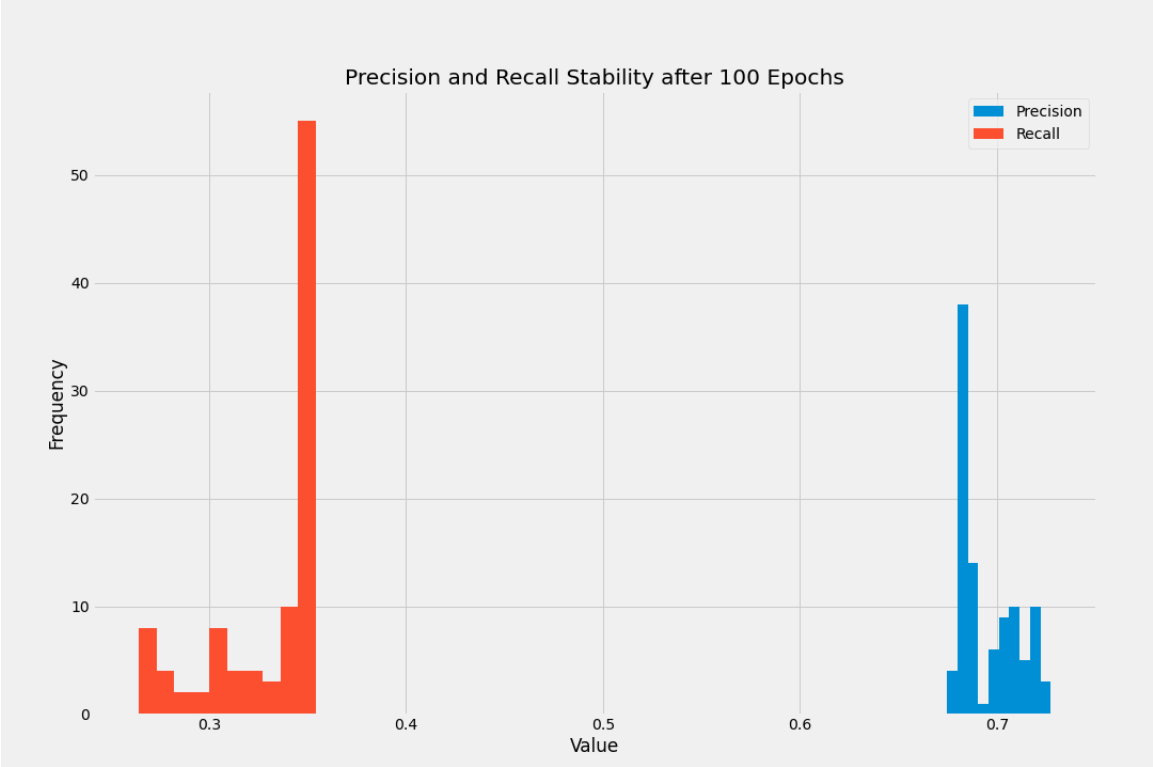
The first item to consider is the complexity of the model. We may be over or under parameterize, and this can be tuned by simply changing the number of elements in the hidden node. Here are graphs of precision, recall, and accuracy, as we change the number of nodes in the hidden layer.



The testing of model with different widths shows that the model is unstable, and that there are no real benefits to increasing model complexity. The most stable models were between 25-30 and so we will be sticking with 32 as the width for our model. The models were only trained for 25 epochs. We will be increasing the training time to ensure that our algorithms are generating stable models.

3.3 Model Stability

To ensure model stability we are training our feed forward neural network 100 times over 100 epochs and then checking the distribution of precision and recall scores to ensure that our model is stable.

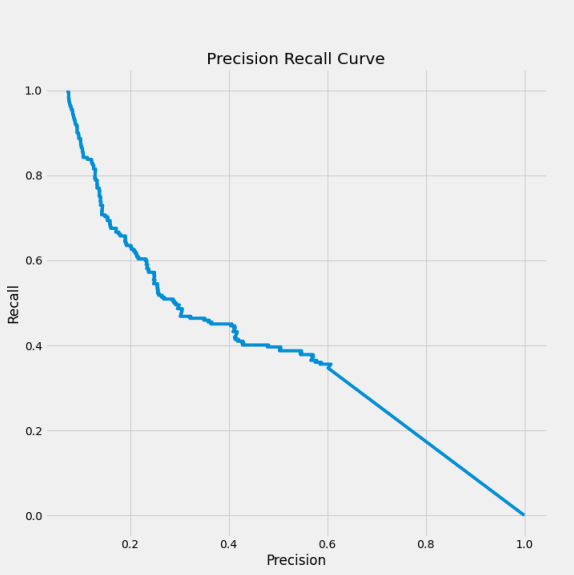


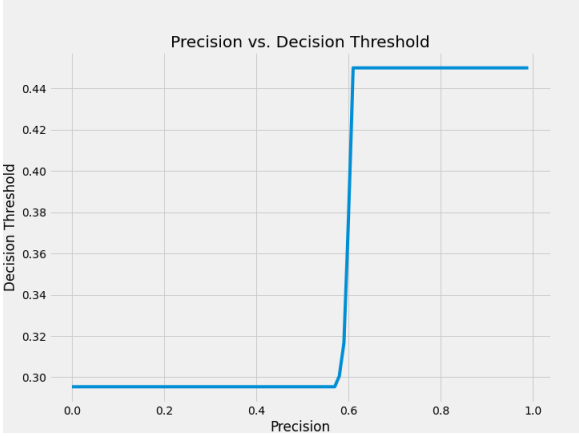
After running 100 networks for 100 epochs we find that we have indeed reached a stable network. Furthermore there is even more good news as we can see that recall scores are on average much higher than the recall scores from the logistic regression. Here is the performance report for a network trained for 100 epochs.

		precision	recall	f1-score	support
	0.0	0.95	0.99	0.97	4599
	1.0	0.69	0.34	0.46	401
accuracy				0.93	5000
macro avg		0.82	0.67	0.71	5000
weighted avg		0.92	0.93	0.92	5000

This is extremely promising and is almost outperforming our tuned logistic regression model without taking a look at the precision recall curve. The next step in analysis for the feed forward network is to of course try to tune the decision boundary to maximize recall while making minimum sacrifices to our precision.

3.4 Tuning Decision Boundary





Taking a look at the two graphs we can see very similar behavior when it comes to the Precision vs. Threshold graph in comparison to the logistic regression model, but the precision recall curve seems to be much more promising when it comes to our neural network. Looking at the curve the tuning precision of 0.6 seems to be a good sacrifice as as it was for the logistic regression. This yields a decision boundary of 0.2345 and the following evaluation:

3.5 Comparison with Logistic Regression

4 Future Decision Making

4.1 Previous Customers & Bias

4.2 Explaining Model Decisions

A Logistic Regression Interpretation of Coefficients

	coefficient
tot_credit_debt	-1.232827e-05
avg_card_debt	3.454299e-05
credit_age	-3.204479e-03
credit_good_age	-1.468200e-03
card_age	-3.005929e-03
mortgages_past_due_6_months_num	3.354028e-06
credit_past_due_amount	3.354364e-04
inq_12_month_num	4.684227e-05
card_inq_24_month_num	6.345529e-05
uti_card	3.871631e-06
uti_50plus_pct	2.662602e-06
uti_max_credit_line	2.447248e-06
uti_card_50plus_pct	3.056732e-06
ind_acc_XYZ	-5.878229e-06
rep_income	-2.520097e-06
Default_ind	-1.361982e-06
AL	-1.165361e-07
MS	-8.170879e-09
GA	-1.773207e-06
LA	6.228547e-07
SC	-9.937016e-07
NC	4.239908e-06
non_mtg_acc_past_due_12_months_num==1.0	6.256940e-06
non_mtg_acc_past_due_12_months_num==2.0	-1.022812e-06
non_mtg_acc_past_due_12_months_num==3.0	7.319980e-09
non_mtg_acc_past_due_12_months_num==4.0	4.094181e-06
non_mtg_acc_past_due_6_months_num==1.0	9.777651e-08
non_mtg_acc_past_due_6_months_num==2.0	1.230158e-06
card_open_36_month_num==1.0	3.868057e-07
card_open_36_month_num==2.0	2.899424e-07
auto_open_ 36_month_num==0.0	-3.625797e-08