CS5830 Project 8 Report

Gavin Murdock and Noah Hammons

Github project, presentation slides

Introduction

Our goal was to predict loan approval and find out what matters the most for getting approved for a loan. This would be useful for people to use before they go in to get a loan approved if they would qualify. It would also help people who got rejected for a loan what they can do to more likely qualify for a loan the next time.

Dataset: https://www.kaggle.com/datasets/devzohaib/eligibility-prediction-for-loan

Dataset

We used an eligibility Prediction for loan database for our models. It is a database from the Dream Housing Finance company that includes details such as Gender, Marital Status, Education, Number of Dependents, Income, etc. as well as whether they were approved for a loan. This dataset fits our models well because it has many categorical variables for our decision trees and neural nets to classify on. The dataset also has a low number of empty values.

Analysis Technique

We will be using Decision trees for classification. This is because of the ease of understanding as well as their ability to handle many categorical and numerical data features. Decision trees will also innately tell us what the most important features are for the dataset. We will then compare our results to a neural network. Neural networks are great at complex data however are very computationally intensive so we would like to see how close decision trees can get.

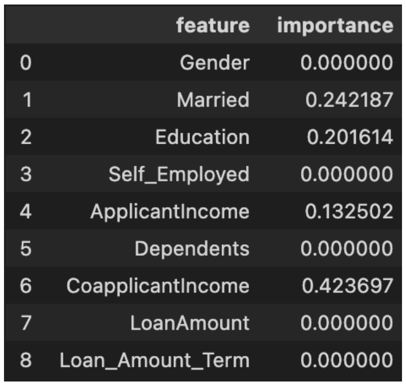
Results

When running the decision trees we ran it at 2 different max depths 3 and 6. This is because different depths of trees can have an effect on the overall accuracy of the model. A more shallow tree will not be as accurate but will be more consistent when adding and removing data (high bias Low variance). The deeper trees will fit this dataset better but will be more susceptible to outliers and will be harder to extrapolate to other data (Low bias high variance).

Both of our trees found different attributes to be the most important, but coapplicant income was top 2 for both our trees and both our trees split on coapplicant income first.

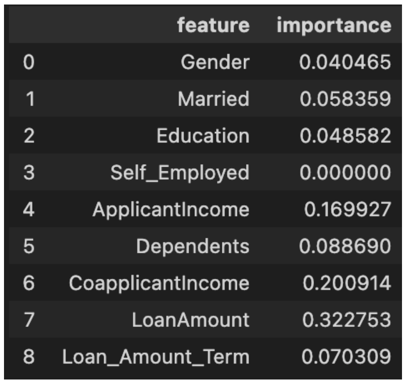
However, the trees do not always split on the same attribute at the same depth. This is because the significance of things can change based on those attributes. For example in our depth 6 tree after splitting on coapplicant income it then splits on education and loan amount. The lower coapplicant income group split on education and the higher coapplicant income group split on loan amount. This could show that people giving loans care less about education if you have a good coapplicant and more about how much money you want to loan if you have a good coapplicant. The feature importance of both decision trees and a visualization of each tree can be seen below.

Decision tree max depth 3:

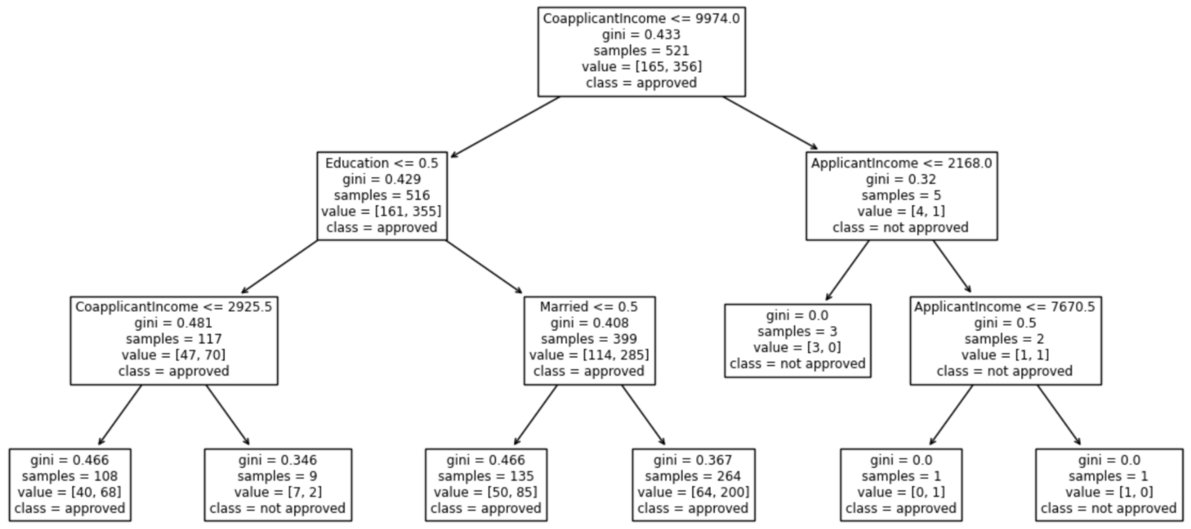


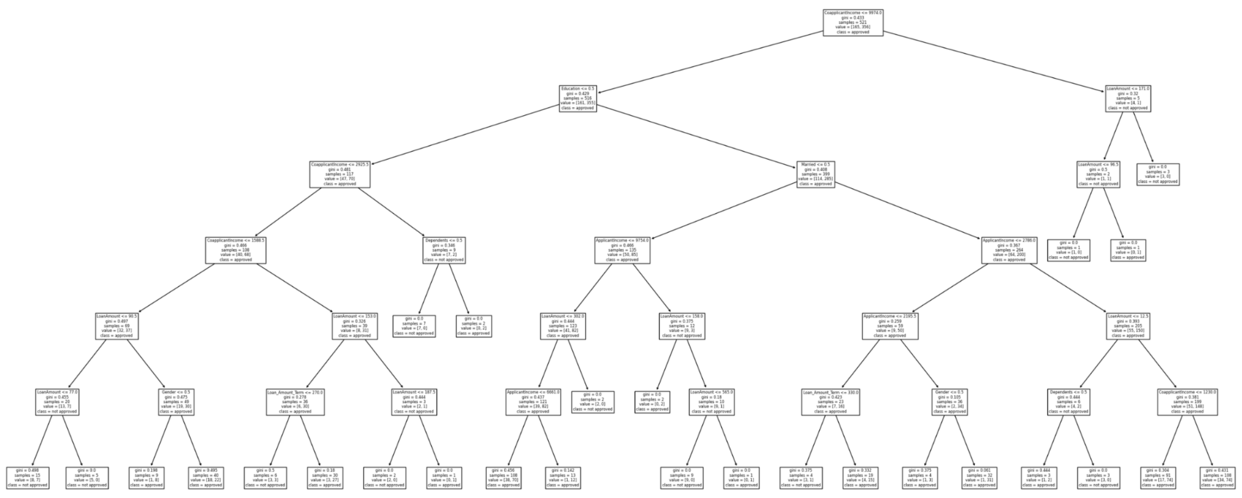
	feature	importance
0	Gender	0.000000
1	Married	0.242187
2	Education	0.201614
3	Self_Employed	0.000000
4	ApplicantIncome	0.132502
5	Dependents	0.000000
6	CoapplicantIncome	0.423697
7	LoanAmount	0.000000
8	Loan_Amount_Term	0.000000

Decision tree max depth 6:



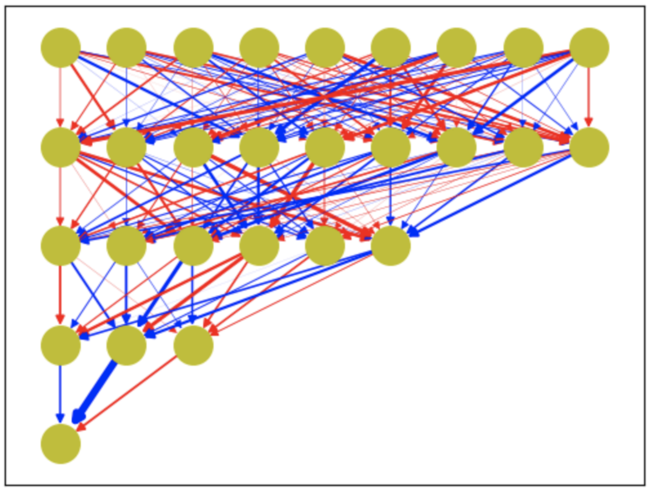
	feature	importance
0	Gender	0.040465
1	Married	0.058359
2	Education	0.048582
3	Self_Employed	0.000000
4	ApplicantIncome	0.169927
5	Dependents	0.088690
6	CoapplicantIncome	0.200914
7	LoanAmount	0.322753
8	Loan_Amount_Term	0.070309

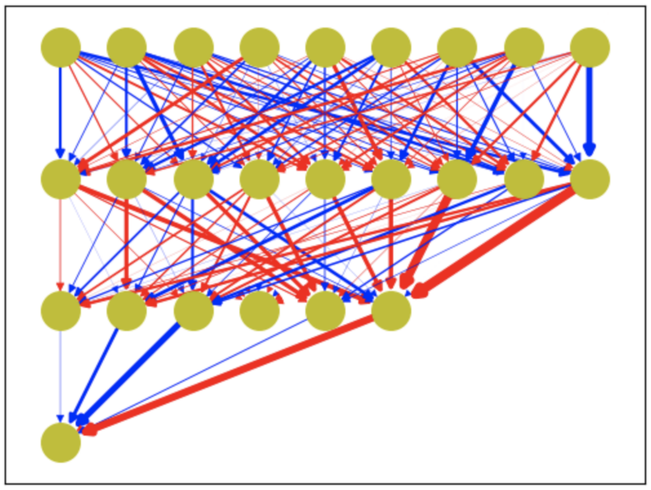

CoapplicantIncome <= 9974.0 gini = 0.433 samples = 521 value = [165, 356] class = approved
Education <= 0.5 gini = 0.429 samples = 516 value = [161, 355] class = approved
ApplicantIncome <= 2168.0 gini = 0.32 samples = 5 value = [4, 1] class = not approved
CoapplicantIncome <= 2925.5 gini = 0.481 samples = 117 value = [47, 70] class = approved
Married <= 0.5 gini = 0.408 samples = 399 value = [114, 285] class = approved
gini = 0.0 samples = 3 value = [3, 0] class = not approved
ApplicantIncome <= 7670.5 gini = 0.5 samples = 2 value = [1, 1] class = not approved
gini = 0.466 samples = 108 value = [40, 68] class = approved
gini = 0.346 samples = 9 value = [7, 2] class = not approved
gini = 0.466 samples = 135 value = [50, 85] class = approved
gini = 0.367 samples = 264 value = [64, 200] class = approved
gini = 0.0 samples = 1 value = [0, 1] class = approved
gini = 0.0 samples = 1 value = [1, 0] class = not approved

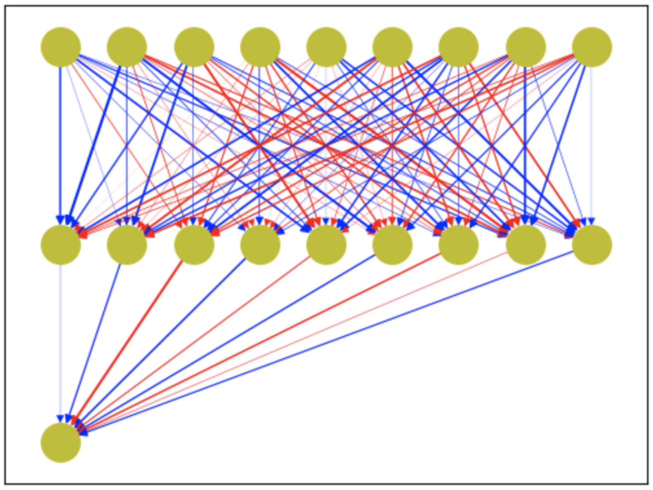


Our decision tree with a max depth of 3 had a better f1 score of 0.833 compared to 0.8 for the decision tree with a max depth of 6. The max depth 6 tree may have overfit the training data and led to a worse f1 score.

When running our neural networks we used three different combinations of hidden layers and nodes in each layer. Our first neural net had 3 hidden layers with 9 nodes in the first layer, 6 nodes in the second layer, and 3 nodes in the third layer. Our second neural net had two hidden layers with 9 nodes in the first layer and 6 nodes in the second layer. Our third neural net had a single hidden layer with 9 nodes. We started with nine nodes in the first layer of each neural net because we have nine input features so it seemed like a good number to start with. We tried visualizing these neural nets with code from class, but for some reason only one node was appearing in the output layer instead of two. I made sure our model was predicting both classes and not assigning the same class to every input by printing a confusion matrix which verified that our model was making predictions for both classes. I wasn't sure how to fix the visualizations, but the ones we were able to make can be seen below.







Since neural nets can be affected by the random initial weights assigned to each feature, we trained and tested each neural net 10 times and took the average f1 score. Our first neural net with three hidden layers had an average f1 score of 0.753, our second neural net with two hidden layers had an average f1 score of 0.774, and our third neural net with a single hidden layer had an average f1 score of 0.809. We were surprised to see that the simplest neural net with only one hidden layer of 9 nodes had the best f1 score, so we made one more neural net with a single hidden layer of only 6 nodes to see what would happen. This new neural net had an average f1 score of 0.806. This last attempt produced about the same score as our previous best neural net and was less complex.

In conclusion, our best neural net and our best decision tree performed very similarly with f1 scores only 0.024 apart. Our best decision tree had a slightly higher f1 score than our best neural net, but both decision trees and neural nets could be considered good options for our case of predicting loan approval.