Loan Approval Prediction Analysis

1) Problem Description:

My idea is to build a prediction system to analyze the applicant's risk levels for applying for a loan and then make a prediction on whether the applicant should be approved based on the data collected from the applicant and his co-applicant. The dataset's features I have used in implementation are gender, Marriage status, dependency, education, self-employed status, Credit history. The label I am trying to predict is Loan status, either yes or no. I cleaned my dataset by editing all the data as binary inputs, either 0 or 1. 0 stands for negative and 1 stand for positive. I also removed all the rows that contain the missing value.

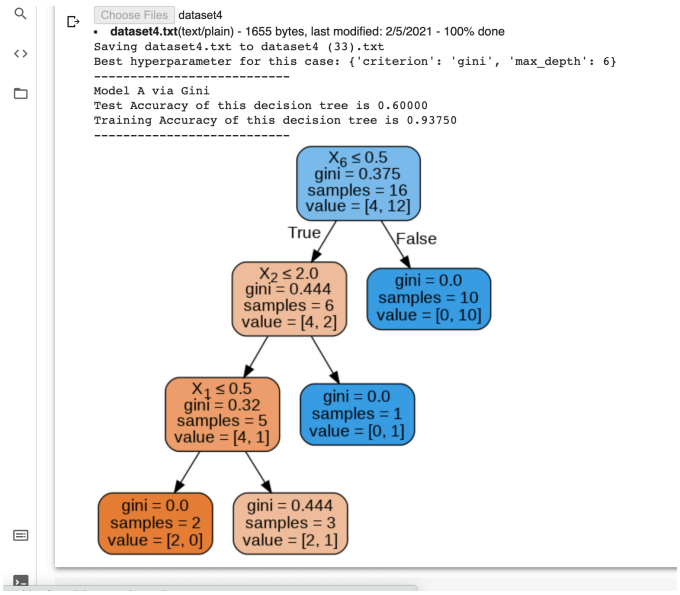
This is classification is a practical implementation for eliminating the risk level from a provider side. The way to improve the performance's accuracy is the key point for this implementation. The higher accuracy the system achieves from the dataset, the safer decision that the system can make will guard the provider's asset in future.

This is classification is a practical implementation for eliminating the risk level from a provider side. The way to improve the performance's accuracy is the key point for this implementation. The higher accuracy the system achieves from dataset, the safer decision that the system can make will guard provider's asset in future.

2) Method Performance:

There are a number of ways to work on how to predict a label. In my case, the label is whether yes or no for the loan approval. I used three algorithms to check the eligibility of each applicant.

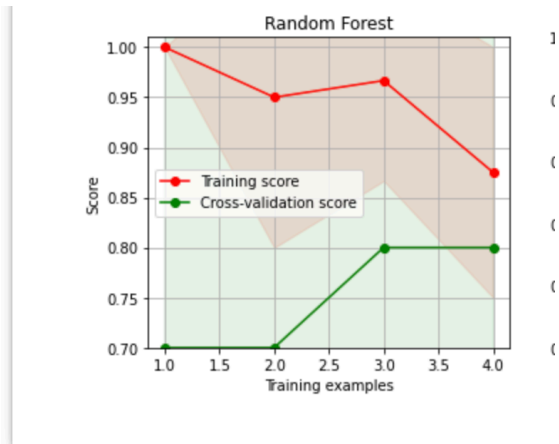
A. Decision Trees



I used the Grid Search Cross Validation method to determine which type of decision tree should be used and this method also gives me the depth of the tree. As a result, The Gini present a 60% of test accuracy.

B. Random Forest

I shuffled random forest with the size of 3, 20, 50, 70, 90, 100 by going up and down. This method gives me a maximum 80% of test accuracy and 87% for training accuracy.

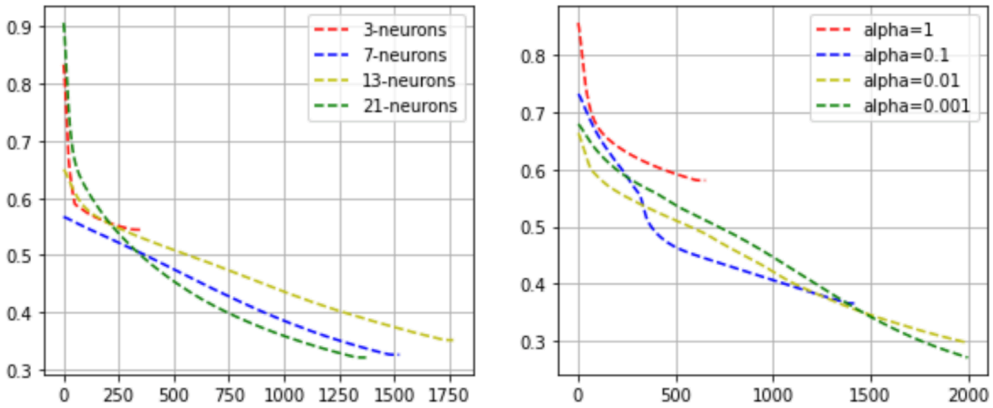


```
----------------------------
Model B via Random Forest
Test Accuracy of this decision tree with size 20 is 0.80000. Training Accuracy is 0.93750.
Test Accuracy of this decision tree with size 3 is 0.80000. Training Accuracy is 0.87500.
Test Accuracy of this decision tree with size 20 is 0.80000. Training Accuracy is 1.00000.
Test Accuracy of this decision tree with size 50 is 0.80000. Training Accuracy is 1.00000.
Test Accuracy of this decision tree with size 70 is 0.80000. Training Accuracy is 1.00000.
Test Accuracy of this decision tree with size 90 is 0.80000. Training Accuracy is 1.00000.
Test Accuracy of this decision tree with size 100 is 0.80000. Training Accuracy is 1.00000.
Test Accuracy of this decision tree with size 90 is 0.80000. Training Accuracy is 1.00000.
Test Accuracy of this decision tree with size 70 is 0.80000. Training Accuracy is 1.00000.
Test Accuracy of this decision tree with size 50 is 0.80000. Training Accuracy is 1.00000.
Test Accuracy of this decision tree with size 20 is 1.00000. Training Accuracy is 1.00000.
Test Accuracy of this decision tree with size 3 is 0.80000. Training Accuracy is 0.93750.
----------------------------
```

C. Neural Network

By experimenting with the SGD algorithm, I get at most 80% of test accuracy and 87.5% for the training. I have altered the number of neutrons and the values of alpha and realized the smaller size of Neutrons with a larger value of alpha can give us a more accurate prediction. And the larger number of neutrons help me get more accurate training data.

```
<matplotlib.legend.Legend at 0x7f87c208a2b0>
```




```
----------------------------
Model C is via Neural Network
Test accuracy for Neural Network with 3 Neurons alpha=0.1 is 0.80000. Training accuracy is 0.75000.
Test accuracy for Neural Network with 7 Neurons alpha=0.1 is 0.60000. Training accuracy is 0.81250.
Test accuracy for Neural Network with 13 Neurons alpha=0.1 is 0.60000. Training accuracy is 0.81250.
Test accuracy for Neural Network with 21 Neurons alpha=0.1 is 0.60000. Training accuracy is 0.93750.
----------------------------
Test accuracy for Neural Network with 3 Neurons alpha=1 is 0.60000. Training accuracy is 0.87500.
Test accuracy for Neural Network with 3 Neurons alpha=0.1 is 0.80000. Training accuracy is 0.75000.
Test accuracy for Neural Network with 3 Neurons alpha=0.01 is 0.80000. Training accuracy is 0.87500.
Test accuracy for Neural Network with 3 Neurons alpha=0.001 is 0.80000. Training accuracy is 0.75000.
```

3) Detailed Analysis

A. Decision Tree

Before running the decision tree algorithm, there are two options available for me to measure the impurity. One is the Gini index and another is entropy. I ran the GridSearch method for me to decide which one should I choose and the optimal depth that I need for this decision tree. It turns out the best hyperparameter is the use of the Gini Index impurity measure and the depth of the tree is three.

This Gini index went through four levels (depth of three) and it split the tree based on the information gain from each feature. I used DecisionTreeClassifier() from the sklearn library helping me complete the implementation. This gives me a 60% test accuracy and corresponding high training accuracy.

B. Random Forest

The random forest consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction. In my problem, It generates many classifiers and aggregates their result. The most common label is our prediction.

For this part, I used the sklearn library to help me implement. Random forest is able to handle high-dimensional data without having to do the feature selection. It did a great job when we implemented it for the first time. The size of the estimator initially starts from root d, and increments its size until 100 then decreases its size back to root d.

Since d refers to the number of features in the dataset. Therefore, root d is about 3 here.

The overall performance of the random forest is better than the Gini decision tree. Both testing and training accuracy increased. The testing accuracy is largely improved by the use of Random Forest. Specifically, among the estimator size of 20 - 50, the dataset is able to have the best performance under random forest.

C.Neural Network

Neural networks perform a better way to predict labels than Decision Tree but the accuracy for both testing and training is worse than Random Forest.

I experimented with the Neural Network by altering the size of the neutron and alpha's value (learning rate). I found out the small number of neutrons and a low level of alpha used results in more accurate testing results. However, a large number of neutrons and alpha can create more accurate training data. In order to improve the accuracy of this Neural Network, I would try to add more capacity to the model. One of the constrain in my data is when I cleaned the dataset I deleted some columns that are difficult to fit in a binary CSV file. And I change all the data to either 0 or 1. After this series of operations, the model loses some of the capacities.

4) Conclusion

I spent quite a bit of time cleaning and editing the old dataset. The cleaning and edition might make some of the models less powerful. However, the overall performance is good as we expected. The average testing accuracy for Random Forest is 80%.

The constraint of this problem comes from the dataset. If I could have figured out a relationship that classifies between applicant's salaries and how much they are required. The models will be more powerful and the prediction will be more accurate.