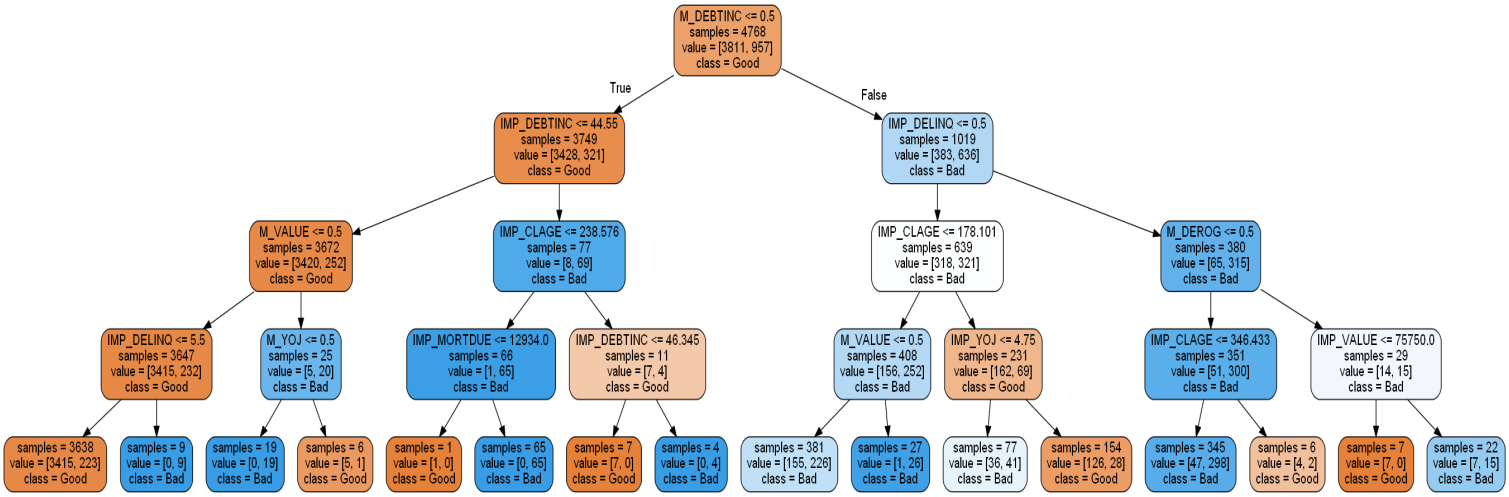
MSDS 422 Machine Learning

Assignment 2

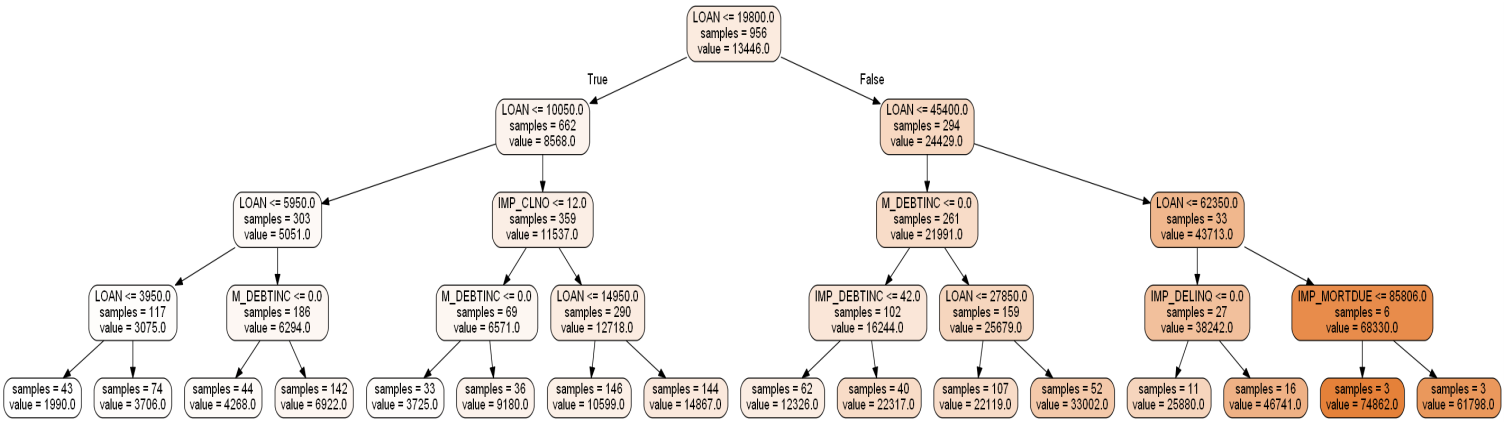
In order to create a model that will accurately predict loan defaults as well as loss amounts, we have implemented three different machine learning-based modeling techniques on a training dataset of 4768 records and a test dataset of 1192 records:

1. **Decision Tree** – this is a flowchart-like structure with nodes and leaves that represent the logic of the predictive model. Here is the decision tree that was produced for predicting loan defaults:



- This model was able to make the correct predictions 87% of the time for the training set and 85% for the test set. It appears that variables such as a customer's debt-to-income ratio, the number of delinquencies, value of their home, and credit line age are key predictors to loan defaults.

  Here is the decision tree that was produced to predict loss amounts:



- The key predictors here are: loan amount, number of credit lines, debt-to-income ratio, and mortgage due.

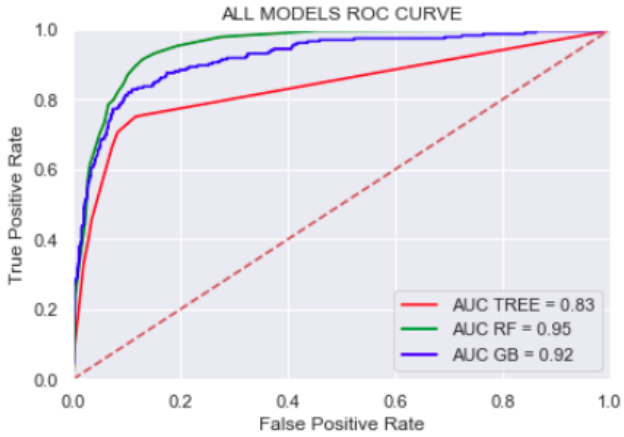
2. **Random Forest** – this technique uses random data to create many different decision trees, then takes the average of the tree findings. Our random forest model was shown to accurately predict loan default on 99% of the training data, and 90% of the test data. Key predictors were found to be the debt-to-income ratio, credit line age, number of delinquencies, value of home, and loan amount.

For predicting loss amount, our random forest model was found to have a Root Mean Square Error (RMSE) of 1347 for the training set and 2798 for the test set. Key predictors for loss amount included loan amount, number of lines of credit, and debt-to-income ratio.

3. **Gradient Boosting** – this technique builds very shallow trees and puts more weight on values that the tree incorrectly classified into the next tree. It will average out the classifications of the small trees. Our gradient boosting model was shown to accurately predict loan default on 92% of the training set and 89% of the test set. Key predictors were debt-to-income ratio, number of delinquencies, and credit line age.

   For predicting loss amount, our gradient boosting model was found to have a RMSE of 1286 for the training set and 2256 for the test set. Key predictors included the loan amount, number of credit lines, and debt-to-income ratio.

   To better judge which of the 3 models was most effective at predicting losses, we plotted an Receiver Operating Characteristic (ROC) curve to find the model with the largest area under the curve:




```
Root Mean Square Average For Losses
TREE 5993.590136935047
RF 2798.169528051947
GB 2256.658099228771
```

   Based on this graph, we can see that the random forest has the largest area under the curve. However, it must be noted that the gradient boosting method had a lower RMSE. Based on this, it could be a toss-up between random forest and gradient boosting being the most effective model.

   All 3 models shared common key predictors for loan default: number of delinquencies, debt-to-income ratio, credit line age. As for predicting loss amounts, all 3 models shared these predictors: loan amount, debt-to-income ratio, and number of credit lines. It makes sense that all of these are predictors in their respective scenario.

Bingo Bonus

1. Decision Tree attribute observations for predicting loan default:
   - Observed a decent jump in accuracy for both data sets when going from max_depth of 3 to 4
   - Implementing and raising the 'max_leaf_nodes' seemed to raise the prediction accuracy of the training set, but not so much with the test dataset

- Implementing min_samples_split with a larger number seems to decrease accuracy. This makes sense as it would limit the amount of node splits in the tree based on sample numbers that satisfy the node. The same can be said for min_samples_leaf.

2. Decision Tree attribute observations for predicting loss amounts:
    - In general noticed lower RMSE for both datasets when raising the max_depth
    - Using a high 'max_leaf_nodes' seemed to lower the training set RMSE more than the test set.
    - Implementing min_samples_split can have different effects on the RMSE for both sets. For example, a value of 40 will result in an RMSE of 4415 (training) and 4527 (test). Dropping to 30 will result in lower RMSE in 3543 and 4291 respectively. However, dropping even more to 20 will lower the RMSE for the training set to 2871 yet raise the RMSE for the test set to 4768.
    - For min_samples_leaf, a lower number resulting in lower RMSE's for both datasets.

3. Random Forest attribute observations for predicting loan default:
    - Changing the random_state had minimal impact to the accuracy for both datasets
    - Same can be said for n_estimators (number of trees). It's not until a very low value is passed for this attribute (ex: 3) that any notable difference is made.
    - Using verbose will actually show some additional verbiage such as the program building out the trees in the forest. Doesn't impact accuracy.
    - Setting bootstrap = 'false' had no impact
    - Setting class_weight = 'balanced' or 'balanced_subsample' had minimal impact

4. Random Forest attribute observations for predicting loss amount:
    - Changing the random_state had some impact on the RMSE for both datasets. For the most part the same predictive variables showed up for each random_state
    - Raising the n_estimators had various impacts (rising and lowering) for the RMSE of both
    - Setting bootstrap = 'false' had no impact

5. Gradient Boosting attribute observations for predicting loan default:
    - Changing the random_state had no impact to the accuracy for both datasets
    - A higher number of n_estimators (> 100) results in accuracy gains for both datasets
    - A learning_rate > .1 will increase the accuracy

6. Gradient Boosting attribute observations for predicting loan default:
    - Changing the random_state had very small impacts on the RMSE of the test dataset. It did not seem to impact the RMSE of the training dataset. Predictive variables seemed to not change as well
    - A high number of n_estimators (>100) results in lower RMSE for both
    - Raising the learning_rate results in lower RMSE for the training set but usually a higher RMSE for the test set