Project Part IV

Noah Keogh (njk6) S01264950

1 Hypertuning Parameters of the Random Forest Classifier Model

After performing a nested cross validation of a logistic regression, support vector machine, K-means clustering and random forest model, it was determined that the random forest model showed the most promise. This was concluded from the results of the nested cross validation that showed that the random forest classifier model tested had an average cross validation (CV) accuracy of around 78%. Thus, it was decided that the other models would no longer be considered and all efforts would be focused on the random forest classifier model.

In order to further optimize the performance of the random forest model, hyperparameter testing was performed in order to determine the optimal parameters for classifying the financial dataset. Hyperparameter testing was performed through the use of validation curves. A validation curve is a hyperparameter tuning method that focuses on measuring the change in a model's performance when one of the values of the parameters of the model is adjusted. In order to perform the calculations needed for performing the validation curve, the validation_curve function from sklearn was utilized. This function takes a dataset of features and corresponding labels, a parameter name, range of parameter values to test, and an integer indicating the number of cross validations to perform. The cross validation is an important aspect of the validation curve since it allows us to get a sense of how the model is performing not only on the training dataset, but on validation data it has not been trained on. A cross validation of size 10 was utilized which means that dataset was split into 10 partitions and 9 of the partitions were used to train the model while the remaining partition was used as a validation dataset. This was repeated 10 times so that every partition was used as a validation dataset and the scores of the models at a given hyperparameter value were then averaged and plotted. This was repeated for each parameter value that was given for testing. The goal was to perform a validation curve on important hyperparameters to determine an optimal value range for each hyperparameter. After this range was determined for each hyperparameter, a randomized CV search was utilized to determine the best performing model within the specified optimal hyperparameter value ranges.

The first hyperparameter that was tested was n_estimators. In the random forest model, the n_estimators parameter indicates the number of random forests that will be generated and used for the making of the final model. Since it specifies the total number of random forest decision trees to generate it is a very important feature for the model. Eleven values were tested ranging from 1 to 100 estimators. This validation curve can be seen in Figure 1 below:

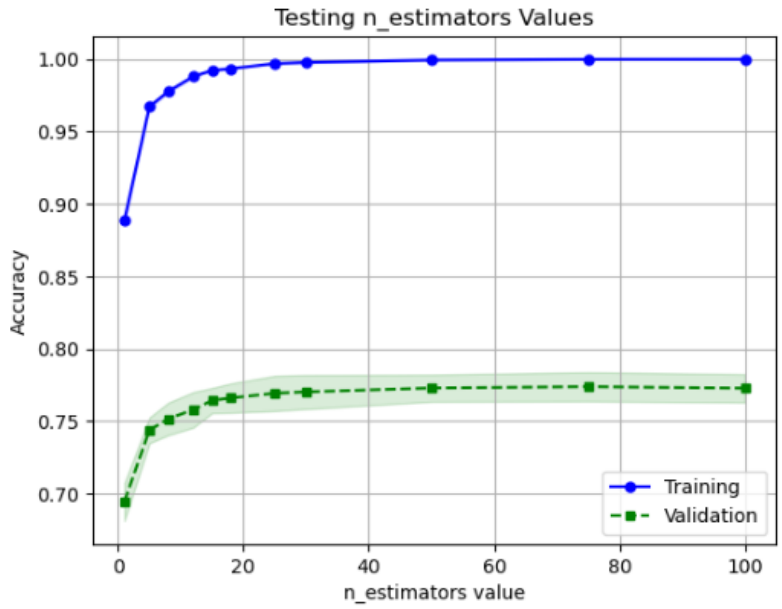

Figure 1: Depicts the results of the n_estimators validation curve.

Based on the results of the validation curve above, it looks like the optimal range for the validation curve may be somewhere between 15 to 25 trees. The reason for this is that the validation curve appears to get close to its highest value starting around 15 estimators and reaches its maximum by 25 estimators. Thus, a range of 15 to 25 trees will be tested in the randomized cross validation search.

The next hyperparameter that was tested was max_depth. The depth of a tree is the largest distance or path from the root node to a leaf in the tree. This hyperparameter, therefore, sets a boundary on the maximum depth that any tree in the random forest can have. A smaller depth correlates to a simpler and more interpretable model, thus, it is an important hyperparameter for controlling the complexity of the model. In order to test this hyperparameters values ranging from 1 to 40 at a step size of 2 (1, 3, 5 ...). The results of the validation curve can be seen in Figure 2 below:

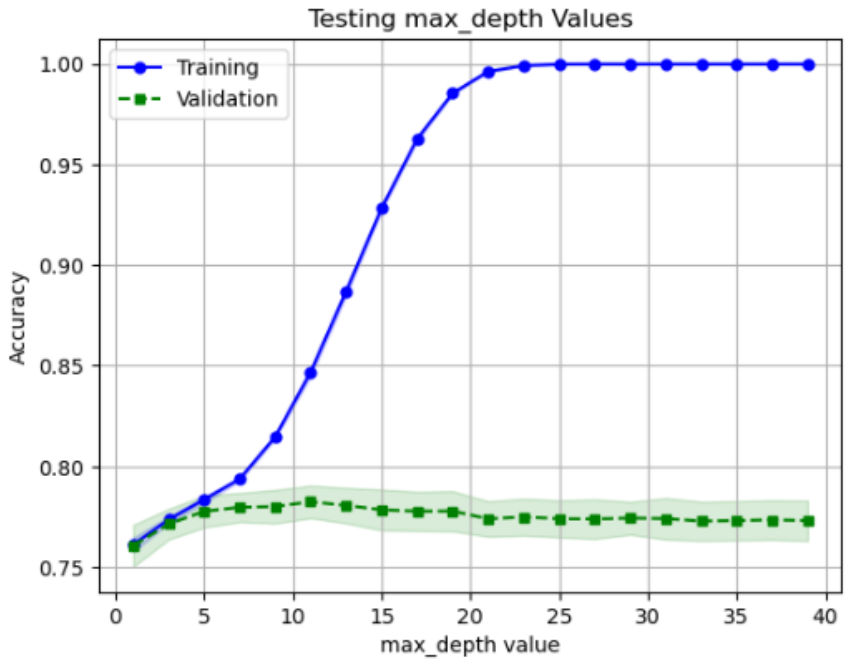

Figure 2: Depicts the results of the max_depth validation curve.

Based on the validation curve above, it appears that the optimal values for max_depth range from values of 3 to 8. Going past a value of 8 appears to lead to a divergence in accuracy between the training and validation curves. Thus, it appears that after a max_depth of 8 the model begins to overfit the training data as the training accuracy increases to a value of 1 and the validation accuracy does not increase, and actually decreases slightly. Thus, max_depth values from 3 to 8 will be tested in the randomized CV search.

The next hyperparameter that was tested was the max_features hyperparameter. The max_features parameter is a parameter that controls the number of features to consider at each split in the tree when deciding upon the next feature to be used to split the tree. This parameter is beneficial for controlling the complexity of the model and ensuring that the model has good generalization. Values ranging from 1 to the maximum number of features present in the dataset (10) were tested. Thus, the whole range of possible values for this parameter can be visualized in the validation curve in Figure 3 below:

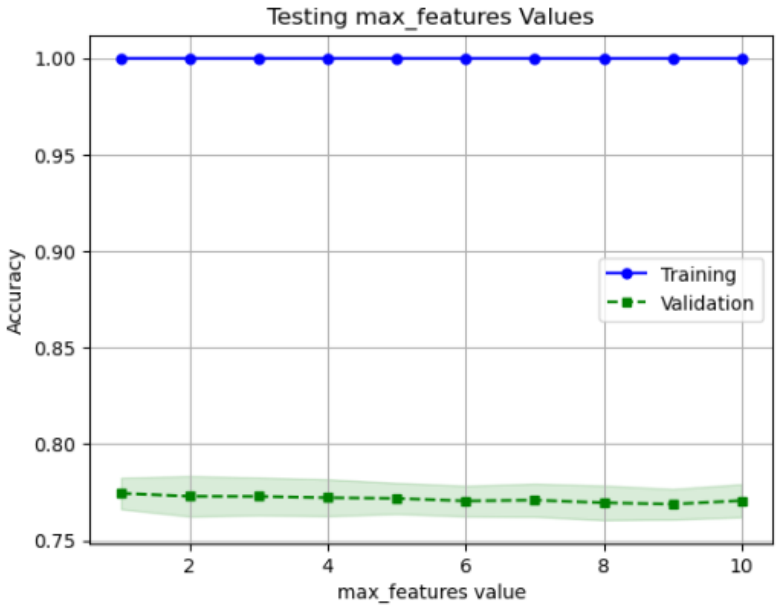

Figure 3: Depicts the results of the max_features validation curve.

Based on the validation curve above, it appears that there is no significant difference in the accuracy of the model on the training or validation sets when the max_features value is varied. Thus, since there does not appear to be a significant difference in model performance when this parameter value is varied, all values ranging from 1 to 10 (total number of features in the dataset) will be used in the randomized CV search.

The final hyperparameter that was tested was the min_samples_leaf parameter. This parameter specifies the minimum number of samples required for a leaf node. The split of a node will only be considered if the left and right branches contain the minimum number of samples set by the min_samples_leaf parameter. Values ranging from 1 to 100 in a step size of 4 were tested. The results of these parameters are shown in the validation curve in Figure 4:

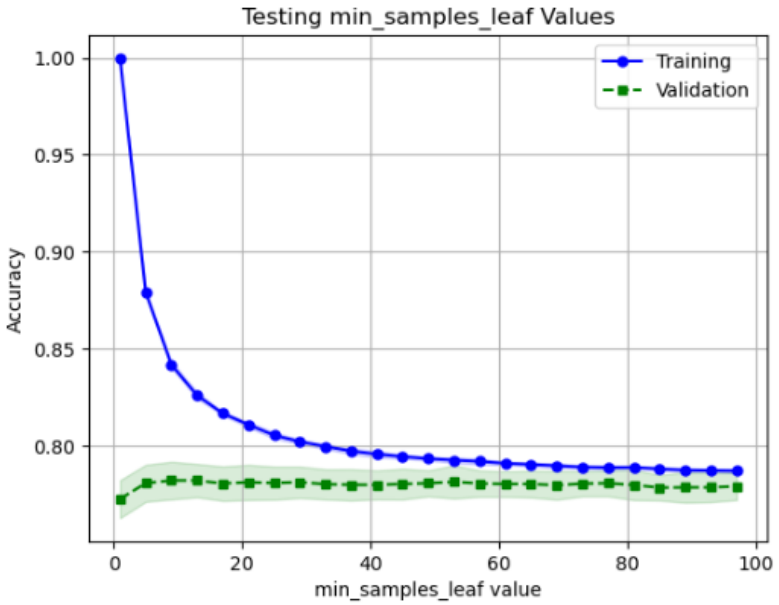

Figure 4: Depicts the results of the min_samples_leaf validation curve.

Based on the validation curve we can see that at low min_samples_leaf values the model appears to be overfitting the training data. This is shown from the model having an accuracy close to 1 on the training data and the model having a much lower accuracy on the validation data. As the min_samples_leaf parameter increases in value we can see that the training and validation accuracies become closer together and are more closely aligned with one another. This pattern can be seen at values starting at 45 and continuing until the highest value tested (98). Based on these results, min_samples_leaf values ranging from 45 to 100 will be used in the randomized CV search for the final model.

2 Randomized CV Search for Final Model

At this point, all the validation curves for each of the hyperparameters have been tested. In addition, optimal ranges for the values for each of the hyperparameters have been identified. These ranges will be used when defining the randomized CV search. Thus, we have the following values and ranges for each hyperparameter to be tested:

1. **n_estimators:** values of 15 to 25
2. **max_depth:** values of 3 to 8
3. **max_features:** values of 1 to 10
4. **min_samples_leaf:** values of 45 to 100

A randomized search CV was performed using these parameter ranges described above. The randomized search CV was performing 50 different times with each iteration having a CV size of 5 (5 splits). After performing this randomized search, the optimal parameters were found to be a model with the following accuracy and parameter values:

```
Best Model Accuracy: 0.7811989968985296
Best Model Parameters: {'max_depth': 7, 'max_features': 5, 'min_samples_leaf': 48, 'n_estimators': 19}
```

The optimal model was determined by taking the model with the highest average CV score. The model described above had the highest accuracy which was 78.12%. This model was then used to make predictions on the test dataset.

3 Measuring Accuracy of Final Model

In order to assess the accuracy of the final model derived from the randomized CV search, the test training dataset that had initially been split from the original unprocessed dataset was used. The test dataset represents 20% of the overall data that was present on the Kaggle website. Before testing the model against the test dataset, the test data needed to be processed in the same manner as the training data. To do this, the imputer that was used to fill the missing values of the training dataset was used to fill the missing values in the test dataset. This means that the median value for each of the features of the training dataset was used to fill the missing values of the test dataset. After performing this operation, the dataset was ready to be tested using the final random forest classifier model.

The test dataset consisted of a total of 30,000 samples. Of those samples, 2,005 samples were positive (experience delinquency) and 27,995 samples were negative (did not experience delinquency). As a preliminary analysis of the accuracy of the model, a confusion matrix was made from the predictions formulated on the test data by the final model. The results of the confusion matrix can be seen below written out in terms of the true positives, true negatives, false positives and false negatives.

```
Number of Samples: 30000
Number Positive: 2005
Number Negative: 27995
------------------
True Positive (TN): 1579
True Negative (TN): 21815
False Positive (TN): 6180
False Negative (TN): 426
```

To further analyze how the model was performing metrics such as the accuracy, precision, recall and f1 score of the model were calculated. The mathematical definitions for each of these metrics can be seen below:

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$
$$precision = \frac{TP}{TP+FP}$$
$$recall = \frac{TP}{FP+FN}$$
$$F1\_score = \frac{2 \cdot precision \cdot recall}{precision+recall}$$

The results of these calculations can be seen in the figure below:

```
Model Accuracy: 0.7798
Precision: 0.2035
Recall: 0.7875
F1 Score: 0.3234
```

Looking at the accuracy of the model, we can see that it is making relatively good predictions since it has an accuracy of 77.98%. This is not too different from the performance on the training dataset which had an accuracy of 78.12%. Thus, based on the performance of the model on the training dataset and how similar it is to accuracy of the model on the testing dataset, it appears that the model is performing quite well. The recall score also appeared to be relatively good having a value of 0.7875. The recall score is the proportion of positive samples the model correctly identified to all the positive samples in the dataset. Thus, since this model had a score of 0.7875 which is relatively high (since a perfect score is 1), the model appears to have good recall. The precision score, however, appears to be quite low with a value of 0.2305. The precision score is a measure of the proportion of correctly identified positive samples to all the samples that were identified to be positive by the model. Since the model has a score of 0.2305 which is not close to 1, this means that the model does not have a good precision score. The F1 score is a combination of the precision and recall scores and is used to capture the overall performance of the model. The F1 score of the model was calculated to be 0.3234, which is quite low, as a value of 1 is considered to be a perfect model. Thus, it would seem that the F1 score indicates that the model is not performing well. This, however, is not entirely true since the balance of the labels in the test dataset needs to be considered. Looking at the number of samples in each class it is clear they are imbalanced as there are only 2,005 positive samples and 27,995 negative samples in the dataset. The large imbalance, and greater number of negative samples can lead to issues when calculating the precision and consequently F1 score. Since there are many more negative samples than positive ones, it means that the model is likely to have a much larger FP sample size. Even though the percentage of negative samples it may be identifying incorrectly as positive may be small (22% of negative samples in this case), the total number is going to be very large compared to the number of correctly identified positive samples (TP), since there are very few positive samples to begin with. This would, therefore, result in a low precision score, and consequently a low F1 score. Thus, even though the precision and F1 score indicated the model may not be performing well, they are not a good metric to be used when the labels of the dataset are imbalanced. Thus, these results should be taken lightly.

To better analyze the performance of the model, it was determined that a metric that is more robust to imbalanced labels should be used. Thus, it was determined that a receiver operating characteristic (ROC) curve should be used. The ROC curve is a graphical method of identifying the diagnostic ability of the binary classifier as the discrimination threshold of the model is varied. The reason that this method is more robust to label imbalance in the testing dataset is because it is calculated using the true positive rate (TPR) and false positive rate (FPR). These metrics are more robust to label imbalance because each of these metrics only depend on the predictions within each class and not the relative size of each class (such as in the case of precision). The TPR calculates the proportion of correctly identified positive samples out of all the positive samples in the dataset. The FPR calculates the proportion of incorrectly identified negative samples out of all the negative samples in the dataset. Thus, since these metrics only compare the classification metrics within each class they are more robust to label imbalance. The formula for calculating the TPR and FPR can be seen below:

$$TPR = \frac{TP}{TP+FN}$$
$$FPR = \frac{FP}{FP+TN}$$

The ROC curve that was generated from calculating the TPR and FPR metrics at each discrimination threshold of the model is shown in Figure 5 below:

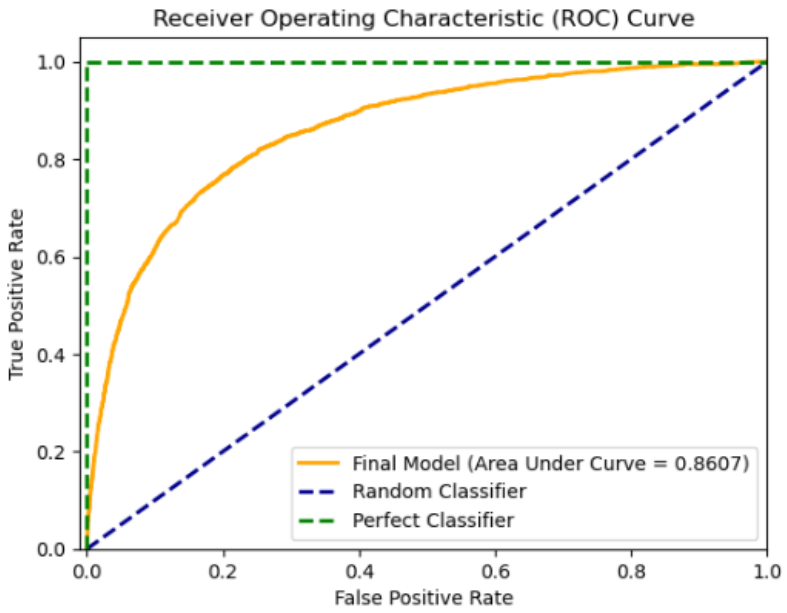

Figure 5: The ROC curve of the final random forest classification model.

Based on the results of the ROC curve above it is clear that the model appears to be performing relatively well. This is shown from the area under the curve (AUC) captured by the model. The model has an AUC of 0.86, which is much better than a random guessing model would have an AUC of 0.5. Furthermore, a perfect classifier would have an AUC value of 1.0. Thus, based on the results of the ROC curve, the model appears to have good performance, which more closely aligns with the conclusion that can be drawn with from the accuracy score of the model. Thus, based on these results it can be determined that the model is performing well, but there is room for improvement.

4 Future Work

Based on the accuracy scoring metric used on the model, it appeared that a relatively good performing model was obtained (based on an AUC value of 0.86). Though this is a relatively good performing model, this model can still be greatly improved upon to get it closer to an AUC value of 1.0. There are many possibilities and different steps at this point that could be taken to further improve the performance of the model. It is possible that there needs to be more processing occurring in the data cleaning steps. For example, there may be some nonsensical data points that should have been removed but were missed in the original data cleaning steps. The model could also be further optimized by trying to adjust the value of other hyperparameters. In addition, the collection of more training data could be performed so that the model may have more information to learn from and, therefore, may be able to make more accurate predictions. There are many different possibilities and avenues to be explored for optimizing the current model.

In addition to trying to further optimize the current random forest classifier model, it may be beneficial to explore other types of models that had not been previously tested. One possible model that could be considered is a neural network. The only problem with using a neural network is that the interpretability of the model may not be very good. Thus, it may be worthwhile to investigate what techniques could be used to improve the interpretability of a neural network model. In addition, going back to some of the other models investigated earlier, such as logistic regression, or support vector machines may be a worthwhile endeavour since further optimization of the parameters pertaining to those models could further enhance their predictive capabilities.

Thus, this work has produced a working random forest classifier model that has generally good performance on the test dataset. This model could be used in conjunction with a number of other metrics or models on a person's financial data to give them a credit worthiness score. Thus, the business objective of making a model that has the ability to predict whether someone is likely to experience serious delinquency has been obtained; however, there is much room for improving the performance of the model and exploring alternative models to obtain more accurate results.