CS 178: Machine Learning W21 - Project Report

Osvaldo Vallejo Zamora
Bao Tran
Adarsh B Shankar

Classifier Model	**Training Error**	**Validation Error**	Kaggle Score
k-Nearest-Neighbor	0.2253	0.3615	0.7159
Random Forest	0.1661	0.2567	0.79256
Multi-layer perceptron	0.3781	0.4226	0.63788
Overall Ensemble	0.2403	0.2889	0.77019

Random Forest Classifier:

Hyperparameters

A. Number trees:
While testing the classifier, we found that the performance of the classifier improved significantly up until 1000 trees. After this threshold the classifier showed little to no improvement by increasing the amount of trees.

B. Features per split:
When testing the number of features, the classifier performed the best when it used 1 feature per split. For max features = 2+ the classifier showed significant drops in performance and therefore were not optimal.

C. Max depth:
To test the optimal depth, the classifier was run multiple times for a max depth = 50,100,200,300,400,500. On average a max depth of 300 performed the best out of all the tests and therefore was determined to be the optimal choice for this data set.

D. Min leaf samples:
To determine the minimal samples needed to create a leaf node, the classifier was tested on values from 2 - 6. On average a minimum of 4 samples performed the best on this data set and therefore was the optimal choice.

Neural Network/Multilayer Perceptron Classifier:

Hyperparameters:

A. Hidden Layer Size:
The tuple consists of the number of hidden units per layer. When it comes to the number of layers, 2 or 3 layers give the optimal solution where as anything more gave more inaccurate results

B. Solver:
The three different solvers available were ‘adam, ‘lbfgs’ and ‘sgd’ and amongst all three ‘adam’ worked the best since it is a more optimal gradient descent algorithm. ‘lbfgs’ a quasi-newton optimization method and regular stochastic gradient descent also gave similar results for hidden layer sizes less than 3 layers and 10 units per layer but we prefer ‘adam’ for bigger hidden layer sizes so we use that algorithm throughout.

C. Max Iterations:
Between 300-500 gave the most accurate results Anything beyond 500 gave the same result for less than 3 hidden layers and since anything more than 3 hidden layers gave us inaccurate results we keep it at 500.

K-Nearest-Neighbor Classifier:

Issues: Since training data has many features and each feature consists of 200,000 data points, choosing the most important features is a critical process before using KNN classifier to test on the massive test data.

Hyperparameters:

a. We have used the Gaussian mixture method to select the top three features by deciding which three features can be nearly separable by assigning each feature a cluster center. It is extremely helpful to plot the corresponding features and their assigned clusters to judge which features we should pick.
b. After knowing which features produce the least error rates, we then plotted the error rates for training data and validation data using different K's, the numbers of nearest neighbors, such as 2,3,4,6,8,10,15,50. To avoid overfitting while minimizing computational cost, we decided to use **K = 3** instead of K=2.

Overall Prediction Ensemble:

The overall prediction ensemble was built using the three classifiers mentioned above using the optimal Hyperparameters found for each. Each classifier was trained individually on the same data set and made predictions on the same validation data set. The final ensemble prediction was obtained by averaging the prediction scores of the individual classifiers to simulate voting. The final auc score for the overall prediction ensemble given in the kaggle competition was 0.77019.

Conclusion:

After experimenting various classifiers based on K-Nearest-Neighbor, Random Forest and Multi-layer Perceptron techniques, we agree that Random Forest which is based on applying decision trees gives the best performance with highest accuracy in predicting a massive data competition like Kaggle. The neural network classifier gave the worst score amongst the three because this classifier tends to overfit when learning from the training data especially with larger hidden unit sizes, and even worse with more hidden layers. The reason the Random Forest Classifier performed the best is because it is the most flexible classifier and the easiest to train on random data sets such as Kaggle competitions.