James W. Yamagata 93265297
Luke Keating 66322089
Greg Dhillon 86127205

Final Project Report

Models	**Score on Training**	**Score on Validation**	**AUC Score**
Naive Bayes	0.634	0.638	0.634
Gradient Boosting	0.759	0.713	0.718
Decision Tree	0.89	0.687	0.67
Decision Forest	0.93	0.729	0.726
Multi-Layered Perceptron	0.675	0.679	0.584
K-Nearest Neighbors	0.967	0.703	0.667
Top 3 Ensemble	*0.79*	*0.73*	*Private: 0.69808 Public: 0.70233*

Model Descriptions

For all Models: Data was split into 80% training and 20% validation. We cross validated using 5 fold validation and took the average error rates and auc scores.

Naive Bayes: This model was trained using the code from mltools. We had also tried writing our own code for this part and created parameters to change the number of times to split each variable and the way the data was split. We tried running splits of ranging from 2 to 10 based on a range (n even groups of data) or by mean (find mean of data, split, find mean of split, repeat...). These changes did not make any large impact in increasing the accuracy of the naive bayes classifier.

Gradient Boosting Algorithm: This model was trained using the Sklearn package. The model hyperparameter settings were chosen by running numerous different learning rates from 0.05 up to 1 and then also running different depth levels from 2 up to 10. What we found was that as we increased the depth the model began to learn better and have a higher AUC score but after we surpassed a depth of 7 we began to see that the model began to overfit and have a reduced AUC score on the validation data due to the model overlearning the training data. We decided on a middle ground depth of 7 which gave the best overall AUC score on the data and a learning rate of 0.5. The features we gave them model were the inclusion of all the training points from the dataset which was included in the training data. This is an effective model in that it takes fits a model and depending on the depth you choose it will then train another model on the error rate. So if you had a depth of one it would just be a linear model, the key is finding the correct depth and learning rate to get the best average performance on validation data. This model was based on the average of 30 estimators in order to create a move stable model.

Decision Tree/Forest: This model was trained using the mltools package. The parameters that we played with were the maxDepth, minParent, and minLeaf. We began by optimizing a decision tree. After running for loops we found that the best parameters seemed to be around maxDepth 20, minLeaf of 5. Our value for minLeaf resulted in larger groupings than what we had for maxParent so maxParent was ignored. These provided us with a reasonable error rate when predicting our validation data. We then decided to run a decision forest to see whether we could improve the accuracy of our classifier. We began with the parameters that we found from the decision trees along with several parameters in the range of our optimal decision tree. We found that the optimal decision tree parameters tended to have the best performance on our decision forest as well. Using a decision forest over a single decision tree improved