Kody Cheung 85737824
Aaron Ching 28162665
Team Name: IronKhode

CS178 W17 Project

Table of Models:

Model	Training Performance	Validation Performance	Leaderboard Performance
KNN	0.686660	0.68627	0.72989
Random Forest	0.770376	0.77133	0.75716
Adaptive Boost	0.989050	0.69393	0.69507
Neural Net	0.647753	0.64864	0.63359
SVM	0.72065	0.6995	0.66570
Gradient Boost	0.9662875	0.69955	0.72598
Combined Ensemble			**0.76352**

KNN

The KNN model had to overcome the barrier of a lot of features and a lot of data points, so what we did for this was find the best K value per kfold in the cross validation. It was implemented using sklearn's KNeighborsClassifier, and we used 10 folds that transformed the training, validation, and test data to be predicted. Surprisingly, this model gave the second best performance on the leaderboard, but we hope that it does account for overfitting from using 10 cross validations.

Random Forest

This model was an ensemble of 25 decision tree classifiers, and evaluated at maxDepth = 20, minLeaf = 4, and nFeatures = 8. It resulted in an AUC of around 0.8, which we determined was not underfitting or overfitting too much. The biggest jumps of fitting resulted from increasing the depth of the tree classifiers in the ensemble, and increasing the subset of features. As we increased the features to n = 12, 13, 14 and depth to 30 or 40, the model did return higher AUC values, but that did not mean it was necessarily better, because it would mean we were overfitting to the leaderboard data. The final soft predictions were simply averaged out because the trees randomly selected the nFeatures, not best features to be weighed heavier.

Adaptive Boost

This model utilized the sklearn library for AdaBoostClassifier and a base DecisionTreeClassifier. The Gaussian naïve Bayes and perceptron base classifiers had much worse results, so after testing them we did not use them. The decision tree was given a depth of 15, chosen based on our previous result in the random forest model, and was trained on an 80/20 training/validation split of the data. All the depths below 15 showed underfitting, and anything above showed overfitting, so we settled at a depth of 15. The predicted probabilities were just averaged over the number of bags, and it gave an acceptable performance on the public data.

Neural Network

Our neural network model utilized the MLPClassifier from the sklearn library, and we used the 'adam' solver to account for the large dataset, and a random_state of 0, which, when combined with a 10-fold cross- validation, gave a subpar performance on the leaderboard, but that may be because of the stochastic gradient descent the library uses, meaning the error adjustments may have undershot or overshot the target value.

Gradient boosting

Gradient boosting combines a set of weak learners using a gradient descent-like procedure where outcomes are weighed on previous instances of weak learners. These weak learners are typically regression trees, where their outputs are combined to correct the error residuals in the previous predictions. The gradient descent procedure minimizes loss when adding the trees together. We will change parameters to put constraints on the trees as well as affect the impact each tree has. The learning rate determines the impact of each tree on the outcome, also known as shrinkage; the larger the learning rate, the faster each tree can reduce the loss of the objective function, but the less robust the model will be to different characteristics of each tree. n_estimators is the number of weak learners to use in the model and is typically good to have at a high value with the trade off of taking longer to compute and possible overfitting. For a large data set like X_test, having a lot of trees that were able to have an impact on the outcome lead us to having solid 3000 estimators with a learning rate of 1. Max_depth limits the max depth of each tree, and allows us to fit more complex models as well as control over-fitting. The high max_depth value will help us fit the large data set that includes a multitude of features.

Support vector machines

SVMs work by finding the plane that gives the largest minimum distance from the closest data points to the margin separating the data. The C value represents how willing we are to misclassify data in order to get a more accurate model. Here we found that a higher C value fit the data well, implying a more destinct separation of points. We also picked a small gamma value to increase the radius of influence the support vectors had, using more points to determine a decision boundary and therefore making it more linear since points near the boundary had less influence. The combination of a high C value and a low gamma value with good prediction accuracy imply that our data is relatively separable, as we are able to make a stingier (high C value), more linear (low gamma) decision boundary that predicts well.

Combined Ensemble

After getting the individual results from each type of model, we simply took the average of each prediction probability for the X_test data set. The combined example did better than any of the other models by itself, as expected. We chose not to weigh any over the other because we were unsure of which ones were actually overfitting and underfitting compared to the private data points, so we choose to take a flat average. The average also gave better results in the random forests in HW4, compared to other ways of combining the predicted probabilities.