Ryan Sivoraphonh (ID# 49148239)
Joyce Le (ID# 82549113)

Model Table

	Training AUC	Validation AUC	Public Leaderboard	Private Leaderboard
1. Random Forest	0.913	0.727	0.74267	0.74025
2. Adaboost Decision Tree	0.849	0.735	0.71121	0.70659
3. K-Nearest Neighbors	0.955	0.632	0.72314	0.71980
4. Ensemble (all three above)	0.724		0.73045	0.72347

Analysis
For our Random forest and Adaboost models, we trained and tested our models on folds of data that we split and created from the original data by setting aside 10% of the data as validation for testing and then separating the remaining data into 3 folds. We created the folds by first scaling the data between 0 and 1, and then clustering it with K-means(3 clusters) and adding in 2nd degree polynomial features. For our K-Nearest Neighbors model, we clustered and scaled the data before splitting it 75/25 for training/test data. We decided not to do feature expansion for KNN since we know distance-based models don't perform too well with high dimensions.

Random Forest
For our Random Forest model, we used RandomForestRegressor from the *sklearn.ensemble* library to do the training for our data. The parameters we passed in to our RandomForestRegressor were max_depth=14, max_features=25, min_samples_leaf=7, and min_samples_split=7. We got these parameters by varying the parameters over our XFold_2 and YFold_2 data and finding the best results after testing them on our X_valid and Y_valid. We scored these results by first creating an ROC curve out of our predictions using *sklearn.metrics.roc_curve* and then getting an AUC by using *sklearn.metrics.roc_auc_score*. We would then plot our AUC against the parameter we were testing along with the validation error to find the optimal parameter.

Adaboost Decision Tree
For our second model, we utilized the Adaboost Regressor from the sklearn library with a Decision Tree Regressor as a Base. These parameters were found by sweeping across the same values used for Random Forest on a single decision tree for Fold 1 of the data. We found hyperparameters of a Max Depth of 10, with 7 Features, and 300 iterations of the model with a learning rate of 1 provided the best results. While deeper decision trees and higher training rates seemed to provide higher validation accuracy and auc when entered into the leaderboard we experienced large decreases in our predicted vs actual auc scores.

While it may have been possible and increase the number of iterations and use deeper decision trees, we found that this often led to overfitting the models, so a shallower decision stump was used. Due to the training time required in training a Adaboost classifier, we found it better to instead time optimizing hyper parameters for random forest and KNN instead since they seemed better fit the data.

Ryan Sivoraphonh (ID# 49148239)
Joyce Le (ID# 82549113)

K Nearest Neighbors

For our K-Nearest Neighbors model, we used the KNeighborsClassifier from the sklearn library to classify the data with a parameter of n_neighors=20. Similar to what we did with our Random Forest Regressor, we maximized the n_neighbors parameter by plotting the AUC scores and validation errors of n_neighbors varying from 1-100 and found that 20 was the best option. We kept the default parameter for scoring the distances (Euclidean by default) because we had already scaled the data beforehand.

We then tried to do some feature selection from the *sklearn.feature_selection* library to further improve our model. A tree-based classifier(ExtraTreesClassifier) from the *sklearn.tree* library was used along with SelectFromModel from the *sklearn.feature_selection* library to discard irrelevant features from the training data based on information gain within each feature. This reduced the number of features in our training data from 14 to 10, upon which we ran the KNeighborsClassifier with n_neighbors=20 and found slightly better results.

Ensemble

For our Ensemble we trained a logistic regression using the predictions from our previous 3 models using the validation data set aside before folding the data. We used the default learning rate of the learner set to 1e-3. We chose a logistic regression because it provided a simple method of combining the models using a new learner that was not tree-based. We hoped that the logistic regression model would be able to better find relationships within each of the features that correlated with the target values, and as a result make more accurate predictions. Though we our ensemble had a lower score than our previous models, we were initially not too concerned with the lower AUC due to suspicions that previous models may have overfit the training and validation data.

Conclusion

For this competition, models such as decision trees and Knn saw very much success, likely due to the way features were constructed. As mentioned in class, since the features provided were previously hand crafted and the overall data had low feature count, the models were allowed to be trained relatively quickly. These models may have also been more effective because we were able to be more aggressive when training the models due to the difficulty in overfitting these methods.

We tended to have more issues with the Adaboost and Logistic Regression Ensemble models in particular. It seemed that the Adaboost model suffered from underfitting. We attempted to address this issue by increasing the number of iterations and lowering the learning rate, but that seemed to cause the model to heavily overfit instead. Due to the length of time it took to attempt to train an Adaboost model compared to a KNN or a Random Forest model, we decided that it would be better to focus on optimizing those models as opposed to continuing to try to fine tune the adaboost parameters. While training out, we encountered issues with the Logistic model--it tended to converge to local minima, but altering training rates did not appear to help. The model seemed to want to converge to a small number of predictions ex(0.2754, 0.5794, and 0.8767) instead of creating a linear representation of values. Upon post competition analysis of the data this may have been caused by the model encountering a local minima due to the imbalanced nature of the validation data it was trained upon with an almost 3-1 ratio of no rainfall vs. rainfall. Other potential solutions may be to append the training features to the data as well.