Richard Lopez [69944917]
Rionel Dmello [15028109]
Allan Wu [45904644]
Team name: Chicky Nuggies (Ensemble model tested under alternate account Rionel Maxim Dmello with professors permission)

Machine Learning Project

Classification Method	Training AUC	Validation AUC	Leaderboard AUC
Random Forest	0.99	0.73	0.75865
AdaBoost w/ Decision Trees	0.75	0.74	0.76195
Extra Trees	0.95	0.78	0.78948
Neural Network	0.53	0.51	.53095
Naive Bayes	0.61	0.60	Did not submit
Combined Ensemble model	0.79	0.77	.78559 (46th)

Random Forest

In using random forests, we wanted to make sure there was as much diversity in each tree so that when averaged together, there would be one classification that was low bias, in addition to the apparent high variance in each classification tree.These ideas heavily influenced the parameters we used to classify the data. We went with Sci-Kit and used their random forest library. This ensured that when given k trees, the training data would randomly be sampled for each tree. We also added a parameter which limited each tree to 9 random features. Random sampling and feature selection, combined, helped to reduce bias since each tree can be vastly different from each other. In addition, we also omitted max depth which ensured a very flexible class of functions. We knew that with random forests, overfitting wouldn't be that much of an issue since the aggregation of multiple diverse trees meant that the overall predictions had reduced complexity, and thus wouldn't overfit and "memorize" the data. This is why k is 1000 trees, which is a fairly large number. We figured with more trees, the predictions would be much stronger.

AdaBoost

We used the AdaBoost library in Sci-Kit and chose a decision tree as our base classifier using a 67/33 train_test_split of the dataset. Picking the parameters for AdaBoost as well as the decision tree was the difficult part about implementing this method. The important parameters were the max_height of the decision tree, the number of iterations and the learning rate. We picked the AUC score as well as the classification error as a measure of how well our classifier was performing and optimized for these values.. After running through a nested loop that tested various combinations of number of iterations(10 to 1000), max_depths(1 to 50), and learning_rate(0.01 to 2), we found that the best values were at 100 iterations with a max_depth of 9 and a learning rate of 0.1. Also setting class weights to “balanced” was good since it gives higher weights to positive samples which accounts for the imbalance in the dataset. Values higher than this led to overfitting and didn’t perform well on test data while values lower than this didn’t accurately predict the classes. We also tried SVM classifiers with AdaBoost, but due to the data size we found it hard to train.

Extra Trees

Finally, we used SciKit Learn’s ensemble library to implement an Extra Trees classifier with 75% training data. The default number of trees was 100, but after testing, we decided that significantly more trees were necessary to give the best result possible. Furthermore, after testing the information gain of the features, we determined that many of the features had similar information gain, so we decided to test max_depth above 30, but below 100. Given the number of trees and max depth of each tree we were going to work with, we also decided that a higher min_samples_split and min_samples_leaf would be beneficial to our model. After careful testing of these parameters, we found that 500 trees, a max depth of 50, a min_samples_split of 9 and a min_samples_leaf of 2 would be ideal for our model.

Neural Networks and Naive Bayes

We tried working with neural networks using Pytorch and started with 2 layers and went upto 7 layers with each having 10 to a 1000 nodes. We used optimizations like SGD, Adam and different loss functions like BCEloss and MSE. We tried using non-linear activation functions like Sigmoid, LogSigmoid, Tanh, Hardtanh, ReLU. For Naive Bayes, we used the Sci-Kit model of multinomial Naive Bayes. We abandoned these methods in our final ensemble since we could not find good ways to improvise the AUC score.

Overall Ensemble

For the overall ensemble, we assigned normalized weights based on the leaderboard auc scores to the predictions made by each of Random Forests, AdaBoost with decision trees, and Extra Tree classifiers. We chose these 3 classifiers since their performance was the best, and on inspection of each of the individual predictions, we found that one classifier may do well where the others dont. The overall ensemble model had a slightly lower score than the Extra Tree classifier which performed the best on the given data.

What worked and what didn’t

Models based on decision trees seemed to work really well with this data and we think it's because rainfall is influenced by different combinations of factors. Since decision trees are able to split data by these factors, grouping these trees allowed us to make an accurate classification by combining several trees’ predictions. It was especially flexible in this case since all the trees had high variation; which we related to the dataset, recognizing that every place experiencing rainfall in the world would have different factors and some of them might not be expressed as features. Neural networks showed promise, but we kept getting stuck at local optimas. If we were able to figure out a way to jump out of a local optima we assume that neural networks could beat decision trees since they are not only really flexible, but different parameters can be finely tuned to suit the data. Naive Bayes was, at best, decent at predicting classes but was bad at estimating probabilities which led to a bad AUC score. We assume this is because a lot of the factors that can influence rainfall are dependent on each other and Naive Bayes assumes that features are independent.