Dew Point and Pitching Take Home Project
Sam Ehrich

This project seeks to answer which pitches are thrown when the dew point is greater than 65 degrees. All the information provided is pitch level data about the game state, pitcher, and ball flight path. To detect dew point, I was told that a high dew point can lead to mugginess in the air and an uncomfortable pitcher. This leads me to believe that high dew points can be classified as pitches that do not follow the normal path that the same kind of pitch would tend to follow.

Getting into the data, I first want to orient the pitches coming from both the left and right handed pitchers. This involves inverting the right handed pitchers to match the left handed pitchers on all horizontally tracked data. At this point, I also decided to add full pitch names for clarity for me to refer to in the future.

I needed to do some data exploration and cleaning to better understand the dataset. There were 4 unknown pitch types thrown in the entire dataset. I did my best to impute the missing classification by looking at the movement and release speed of the ball. One of the pitches had a release speed of 102 with a spin of 2400 and a small amount of IVB and Horizontal break, so I classified that as a fastball. The next had a release speed of 82 with not much left/right and up/down movement that I would consider closest to a slider. The final 2 pitches are pitches thrown back to back with low speed, low spin and a similar release height. I would classify those pitches as split finger fastball pitches.

With all the data classified, I decided to manually attempt to classify outlier pitches by selecting features that represent the pitch. For example, the lower end of speed for fastballs are typically ones where the pitcher is either fatigued, or had a bad grip on the ball. That could be due to higher dew point and a rise in density in the air, so classifying potential outliers could be an option. Breaking balls with poor spin could be potential outliers for the same reason fastballs with poor speed are outliers. Once I did the manual outlier detection, I visualized the data by speed and spin rate, and also by plate location. The visuals lead me to believe a better system for detecting outliers would probably be a machine learning algorithm like an isolation forest.

Isolation forest is an outlier detection method that assigns scores to each observation by how many splits can be made between the given point and all the other points. The lower the score, the fewer splits need to be made to isolate the observation. To test this method, I used only the plate location to detect the outliers. In this situation, the lowest scores should be every point on the outside of the plate location. Visualizing the plate location data, I confirmed that the isolation forest algorithm works as I expected, and I was ready to test the algorithm on all the features that define the ball flight path and plate location. This model with all the features included is the one I used for the rest of the project. To define potential high dew point pitches, I took the bottom 5% of scores for each pitch type and labeled them as high dew point pitches.

After detecting the pitches and assigning my target variable, I was ready to get to modeling. This is a binary classification problem, so I used a generalized linear model (logistic regression) to fit my input variables to my target class. I also decided to test my linear model against a non linear model in a random forest. Random forests are a tree algorithm that uses splits in the dataset to identify the classification while working down each node of the tree. A random forest algorithm could be a better fit for this problem as it can identify non linearities in the features to classify the target variable.

Starting with the linear model, I split the data up by pitch type and used cross fold validation to train and test the subsets of data. Cross fold validation splits the data up into separate subsets of data to train and test which is useful for situations like this where the dataset size is limited. Once every part of the data has been run through the model, I saved the predictions from the folds that were used as test sets. Evaluating the results of the is as easy as looking at the confusion matrix. The glm performed at a 95% accuracy with a specificity of 22%. This means that the model was largely correct, but since the data is not balanced, I am focused on the imbalance and underrepresented class. Specificity indicated how many times the model was able to predict a high dew point game (True Negative), since the model is already easily detecting the regular dew point pitches at a 99% sensitivity.

Modeling the random forest was very similar to the glm model aside from the additional tuning options of a random forest. A few ways forests can be tuned is the amount of features each tree uses to make a decision, and the amount of trees they grow. For simplicity I kept these hyper parameters set to their default amounts. After extracting the predictions from the random forest model, I compared the results of the linear model to the forest. They both performed at a high overall accuracy, but this is to be expected when the overwhelming amount of pitches are classified as being thrown in regular conditions. However, the specificity of the forest resulted in 35% or an increase of 13% from the logistic regression model. The forest is able to predict the pitches that are thrown in higher dew points at a better rate than the initial model. For that reason, I decided to use the predictions from the random forest as my submission for the project.