Brief:

Tasked with finding the probability that a pitch was affected by a dew point greater than 65 degrees F, I treated this as an unsupervised clustering problem as the data provided did not contain any labels that denoted whether pitches were or were not affected. I debated using KMeans initially, ultimately I decided to use DBSCAN as I would not have to specify the number of clusters to look for. Instead, I ran DBSCAN for each separate pitch type.

Explanation:

To start, I run through some quick analysis to take a look at the data as well as identify any potential cleaning/imputing that may be necessary. I also generate simple plots for each pitch type using the PLATE\_X/Z values to try to visualize if there are any patterns for each pitch type. Most of them are bunched towards the center, as pitchers try to pitch into the strike zone, but a few, like sliders, show a pattern indicative of the type of path they are expected to take and may be more apparent after clustering.

Following that, I go through the features to select which ones will be most useful in the model. Below I highlight some notable features and my logic behind their use or omission; all other features not explicitly omitted will be used in the model. The numerical features are the primary features I want to include as those values are what factor into how the pitch will cross home plate.

* PID: The pitch identifier; won’t be included in the model
* BATTER\_IN\_INNING\_KEY: We wouldn’t expect the weather conditions to change dramatically between batters in an inning so this adds noise to our data, but it is reasonable to expect changes in over the course of a game which is why I left in INNING\_KEY
* PITCH\_NUMBER, OUT\_KEY, BALLS, STRIKES, IS\_RUNNER\_ON\_\*: I am treating these data points as independent of the pitch and each other because while previous pitches could affect a future pitch due to something like fatigue as the game goes on, they do not affect the dew point during the game
* THROW\_SIDE\_KEY, RELEASE\_SIDE, HORIZONTAL\_BREAK, HORIZONTAL\_APPROACH\_ANGLE: I debated treating pitchers as equal regardless of their throwing side and using the absolute value of these horizontal values, but figured I would also have to use the absolute value of the PLATE\_X values to account for the change which would lead to an inaccurate representation of the way pitches are meant to break (Looking at sliders for example and seeing how they are meant to travel laterally and down)
* EVENT\_RESULT\_KEY, PITCH\_RESULT\_KEY: While the temperature and humidity during the game can influence the pitch result, like hot, humid air allowing hits to travel farther, we are only looking at the pitch and not necessarily the result; plus, EVENT\_RESULT\_KEY is also mainly NaN values

After selecting which features to exclude, I divide the features I use into categorical and numerical feature lists. Then I use .get\_dummies on the categorical columns in order to pass numerical categorical values into the clustering algorithm, while dropping the first column produced for each categorical feature to avoid adding unnecessary noise to the dataset.

The exception to this is the ‘PITCH\_TYPE\_TRACKED\_KEY’ column. Originally I was going to pass the encoded values into DBSCAN, but I decided to use .get\_dummies but not drop the first column since I planned to run DBSCAN for each individual pitch type instead. I updated the categorical feature list to include the dummy columns, as well as creating a separate list for pitch types.

After, I needed to wrangle the numerical values in the data as there are several ranges of values within the numerical values in the data, both positive and negative. I used MinMaxScaler to scale them accordingly within a range of [-1,1] to maintain the negative values in the data rather than scaling them to a range of [0,1]. The new dataframe, scaled\_df, contains the pitch identifying key, encoded categorical features, scaled numerical features, and the pitch type as encoded values as well.

DBSCAN has two primary parameters for clustering, eps (epsilon value) and min\_points (number of neighboring points).

To find eps, I used NearestNeighbors to calculate the distance between each point and its closest neighboring point within the same pitch type, based on the numerical and categorical columns. Then I pass the calculated distances into KneeLocator to calculate the point of maximum curvature. Admittedly, just eyeballing this plot it looks like the epsilon value should be lower, but I am just a human whose brain has its own biases but I do not think that is enough to rationalize that interpretation of the graph, so I used the value provided for each pitch type. The two plots that were not fed into KneeLocator were the two pitch types that had fewer than 5 points each, so I plotted those separately to achieve the same end result.

To find min\_points, I used GridSearch to iterate through several values but quickly realized that there wasn’t a one-size-fits-all value, especially because of the differing ranges for each pitch type, so each pitch required a different value. I then scrapped the GridSearch and adjusted the values manually for each pitch type. I plotted each pitch type by X/Z values again with colors based on clusters (added to the dataframe as ‘clusters’). I attempted to minimize the number of points considered noise (sorted into cluster ‘-1’ by DBSCAN) while also avoiding excessive amounts of clusters and overfitting. Since the ‘-1’ cluster is considered noise by DBSCAN, I worked with the assumption that the ‘-1’ cluster (or ‘0’ for one pitch type) is returned when the pitch has a 100% probability that it was affected by the dew point greater than 65F.

With that assumption, each following cluster would be considered a proportional step-down in probability that the pitch was affected. For example, in a pitch type where the only clusters are [-1,0,1] where cluster ‘-1’ is 1.0 probability, cluster ‘0’ would have a 0.5 probability of being affected, and cluster ‘1’ has a 0.0 probability. Once the clusters were determined, I created a column in the dataframe (‘DEWPOINT\_AFFECTED’) with the same values as the clusters. I looped through the pitch types and the cluster values for each pitch type to calculate the probabilities according to the number of clusters, and paired these values in a dictionary.

Once the values were added to the dictionary, I mapped them to the ‘DEWPOINT\_AFFECTED’ column to get the corresponding probabilities with the correct cluster for each pitch type.

The last step was producing and exporting the ‘PID’ and ‘DEWPOINT\_AFFECTED’ values as a csv as requested.