When making the distance function, there are the attributes DATE, cloudcover, rainfall, min\_temp, windspeed, and humidity to consider. First with the attribute DATE, I chose to ignore it as I viewed the attribute to be a unique identifier to keep track of which day in the year a certain row was and had no significance when measuring the actual similarity of the weather between two different days. When considering the attribute cloudcover, it is a categorical attribute, so it is an ordinal attribute. To assess the similarity between the cloud cover of two days, I ranked each type of cloud cover like such: 1=Fair, 2=Fair/Windy, 3=Partly Cloudy, 4=Partly Cloudy/Windy, 5=Cloudy, 6=Cloudy/Windy, 7=Mostly Cloudy, 8=Mostly Cloudy/Windy, 9=Fog, 10=Haze, 11=Light Rain, 12=Thunder in the vicinity 13=Thunder, 14=Light Rain with Thunder, 15=Rain, 16=Thunder/Windy, 17=T-Storm, 18=Heavy T-Storm. I ranked them from the most calm cloud cover to the most intense or chaotic cloud cover. I then mapped the range of the cloud cover variable onto [0,1] by using the formula zif=(rif - 1)/(Mf – 1) where rif is the rank of the object and Mf is the total number of ordinal categories. After mapping all of the categories you are left with the following values:1= Φ (Heavy T-Storm), 16/17= Φ (T-Storm), 15/17= Φ (Thunder/Windy), 14/17= Φ (Rain), 13/17= Φ (Light Rain with Thunder), 12/17= Φ (Thunder),11/17= Φ (Thunder in the Vicinity), 10/17= Φ (LightRain), 9/17= Φ (Haze), 8/17= Φ (Fog), 7/17= Φ (Mostly Cloudy/Windy), 6/17= Φ (Mostly Cloudy), 5/17= Φ (Cloudy/Windy), 4/17= Φ (cloudy), 3/17= Φ (Partly Cloudy/Windy), 2/17= Φ (Partly Cloudy), 1/17= Φ (Fair/Windy), 0= Φ (fair). Then, utilizing the Manhattan distance function, we get d\_cloudcover(c1,c2)=| Φ(c1) – Φ(c2) |. When calculating the distance function for rainfall, min\_temp, windspeed, and humidity, I standardized all their data using z-scores with the equation of a z-score being z=(xif – mf)/(sf). Where xif is the observed value, mf is the mean of the population, and sf is the standard deviation. After finding the z-scores, I used the Manhattan function to calculate the similarity of that attribute being looked at. The equation looks like this: d\_attribute(x1,x2) = | Zx1 – Zx2 |. When calculating the distance of rainfall, the mean is 0.261, the standard deviation is 0.747, and the distance function is d\_rainfall(r1,r2) = |((r1-0.261)/0.747) - ((r2-0.261)/0.747) |. When calculating the distance of min\_temp, the mean is 64.140, the standard deviation is 15.324, and the distance function is d\_min\_temp(t1,t2) = | ((t1-64.140)/15.324) - ((t2-64.140)/15.324) |. When calculating the distance of windspeed, the mean is 11.630, the standard deviation is 5.408, and the distance function is d\_windspeed(w1,w2) = | ((w1-11.630)/5.408) - ((w2-11.630)/5.408) |. When calculating the distance of humidity, the mean is 53.542, the standard deviation is 17.02, and the distance function is d\_humidity(h1,h2) = | ((h1-53.542)/17.02) - ((w2-53.542)/17.02) |. I assigned the weights as cloudcover=0.5, rainfall=0.8, min\_temp=0.7, wind\_speed=0.8, humidity=0.7. The distance function between the weather of two days is the summation of the products of the distance functions and their respective weights which ends up as: assuming hw1=(c1,r1,t1,w1,h1) and hw2=(c2,r2,t2,w2,h2), then ( (0.5 \* d\_cloudcover(c1,c2)) + (0.8 \* d\_rainfall(r1,r2)) + (0.7 \* d\_min\_temp(t1,t2)) + (0.8 \* d\_windspeed(w1,w2)) + (0.7 \* d\_humidity(h1,h2)) ) / 3.5. For the outlier score function, I had two variables which was df and k where df is the data frame containing all the information in the csv and where k is the hyperparameter which is the number of nearest neighbors we will look at. To calculate the outlier score for all the values, I would iterate through all the rows in the data and create a numpy array containing all the distances of the current day being looked at to the rest of the days in the year. Then I would sort that array in from the closest distance to the farthest. I would add up the k-nearest distances and divide it by the value k to get the outlier score for that certain hyper parameter. I would do this with k values of 5, 50, and 100 which means that I would compute the outlier score of every row by using the 5 nearest neighbors, 50 nearest neighbors, and 100 nearest neighbors to get 3 different outlier scores for each row in the csv. The top 4 outliers in order of highest outlier score for when k is equal to 5 were the days 8/15/2021, 10/27/2021, 4/14/2021, and 6/29/2021. For when k is equal to 50 it was 8/15/2021, 6/28/2021, 10/27/2021, and 4/14/2021. When k is equal to 100, it was 8/15/2021, 6/28/2021, 4/14/2021, and 10/27/2021. When looking at the top 4 outliers for the three different k values, it is pretty consistent with 8/15/2021 being the top outlier for all three values. This is likely due to the minimum temperature being 0 which is an extreme difference compared to the rest of the dataset. Other dates that was in the top 4 outliers for all values of k were 10/27/2021 and 4/14/2021. 6/28/2021 was a top 4 outlier in when k was equal to 50 and 100. The reason for all of these dates to have the highest outlier scores was due to extremes in at least one attribute value. The 2 most normal days in order of lowest outlier score for when k is equal to 5 is 6/25/2021 and 8/8/2021. For when k is equal to 50 it is 5/30/2021 and 3/27/2021. When k is equal to 100 it is 8/7/2021 and 5/26/2021. Now when looking at the two most normal dates for each value of k, there was none that was consistent among the three but when looking at the values of the attributes for each day, they are all very similar. When looking at all of the most normal days, they are most likely the most normal because they all had 0 rainfall and low ranking cloud covers. The outlier technique that I used worked effectively as it was able to identify both global and local outliers using the differing k values. Using the k value of 5 allowed for me to be able to find outliers in a local context and using larger values like 100 allowed for me to be able to see outliers that are compared to a much broader range of neighbors. My outlier detection technique does a good job at finding extremes such as days with extreme temperatures like the minimum being 0 degrees on 8/15/2021 or unusual cloud cover where 4/14/21 had heavy thunderstorms.