Lab 6

In this lab, you will learn how to to use k-nearest neighbor algorithm and hierarchical clustering in a Python. We will use those algorithms to analyze historical daily temperatures in Philadelphia.

I. Data Set.

We prepared file called 'historical_temperature_philly.csv' which contains 5 columns called Month, Day, Year, Max, Min. The first 3 columns define the date, the fourth column is the maximal recorded temperature in Philadelphia on a given date and the fifth column is the minimal recorded temperature on the same day. The record of temperatures starts on Jan 1, 1874 and end on Jan 31, 2015 (>141 years of daily temperature data). The data set was preprocessed from files downloaded from the Frankin Institute web page: http://learn.fi.edu/weather/data2/index.html. If you are interested, you can get much more of the historical weather data from sides such as www.openweathermap.org/history, www.wunderground.com/history/, www.weather.gov/help-past-weather, www.ncdc.noaa.gov.

Task 0. Download 'historical_temperature_philly.csv' from the course web site

II. Data Preprocessing and Exploratory Analysis

For this lab, we will study how weather patterns change over years. Therefore, it will be useful to think about the weather data as a matrix of dimension N times 365, where N is the number of years and 365 is the number of days in a year. With this matrix representation, the value in row 2 and column 6 will represent the temperature on Jan 6, 1875. Another useful way to think about the data is to calculate monthly averages and produce a matrix of dimension N times 12 where N is the number of years and 12 is the number of months in a year. With this matrix representation, the value in row 2 and column 6 will represent the average temperature during June of 1875.

The code to convert the original data 'historical_temperature_philly.csv' into the daily and monthly matrices is provided in matrix_generation.py. The following is the summary of both functions.

daily_mtx(years, month, colname)

```
Create a matrix with size N by (number of days in month)
Each row contains daily max or min temperature of the month in the year.

arguments:
years -- a N length list-like years, e.g., [2012, 2011, 2013], or
np.array([2012,2011,2013])
month -- an interger between [1, 12].
colname -- a string either "Max" or "Min".

outputs:
daily_mtx -- a N by 30 or N by 31 or N by 28 table, where N is the length of years.
\
Column number varies with different month selection
usage example:
# To create a daily max temperature matrix in Jan of years from 1874-2015
import matrix_generation
```

```
daily_max_mtx_jan = matrix_generation.daily_mtx(range(1874, 2016), 1, "Max")
```

monthly_ave_mtx(years, colname)

```
Create a matrix with size N by 12.
Each row contains monthly average min or max temperature of a whole year.

arguments:

years -- N length list-like years, e.g., [2012, 2011, 2013], or
np.array([2012,2011,2013])
colname -- a string either "Max" or "Min".

outputs:
month_mtx -- N by 12 matrix

usage example:
# To create a monthly average of max temperature matrix of years 1874 - 2014
import matrix_generation
monthly_maxave_mtx = matrix_generation.monthly_ave_mtx(range(1874,2015), 'Max')
```

Task 1: Study the code and write in one or two paragraphs an explanation how each line of the code works in functions daily_max and monthly_ave_mtx. Could you guess how the leap years are handled by the code?

Task 2: Perform exploratory analysis of the data set. What were the coldest and warmest days during the last 144 years? What are the average monthly temperatures during the 144 years? What are the average daily temperatures during the 144 years? Plot the average monthly and average daily temperatures using the line plot. What is the variation in January temperatures – plot the histogram of 144 January temperatures. What is the mean and standard deviation? How does January 2015 compare to January during the previous 144 years? What is correlation coefficient between January and February temperature? What is the correlation between January and July and what can you conclude from this number (e.g., is July temperature predictable from January temperatures)? What are the average yearly temperatures? Plot the time series of yearly averages. Can you spot any trends? Plot the time series of 144 July temperatures. Do you see the same trends as with the yearly averages?

Task 3 (EXTRA CREDIT). Use Python, Tableau or D3.js to do some additional exploratory analysis of the data. The amount of extra credit will depend on your additional effort and creativity.

III. Nearest Neighbor Algorithm

Let us see how k-nearest neighbor algorithm can help us get some interesting results. In particular, we will explore to what extent temperatures in January are predictable of temperatures later during the year.

1). Find the nearest neighbors of 2014 with January daily max or min temperature.

The following code shows how you can find 5 nearest neighbors of January 2014 by comparing daily temperatures.

```
# Create the required matrix
import matrix_generation
import numpy as np
import matplotlib.pyplot as plt
years = np.arange(1874, 2015)
daily_max_mtx_jan = matrix_generation.daily_mtx(years, 1, "Max")


# Find the nearest neighbors of 2014 based on the daily January max temperature matrix.
from scipy.spatial.distance import pdist, squareform
data_dist_flat = pdist(daily_max_mtx_jan, 'euclidean') # computing the distance
data_dist_square = squareform(data_dist_flat) # make the flat distance as a matrix
nearest_neighbors_indices = data_dist_square.argsort(axis = 1) # sort each row in ascending
order
k = 5
knn_years = years[nearest_neighbors_indices[-1, 2:(2+k)]]
print('The 5 nearest years of 2014 based on January daily max temperature is: ' +
str(knn_years))
```

This piece of code printed the following line:

The 5 nearest years of 2014 based on January daily max temperature is: [1987 1882 1961 1905 1895]

Task 4. Study the code and summarize what it does in one or two paragraphs.

Task 5. Using the 10 nearest neighbors of January 2014, let us check if we can predict daily temperatures during July 2014. To do this, calculate average of daily temperatures of the 10 nearest neighbors during July. Compare the prediction with the actual temperatures observed in July 2014. Start by plotting the line plots of the predicted and actual daily temperatures. Then, calculate the averaged squared error of the predictions during the 31 days. Are you happy with the accuracy? Let us compare this result to the prediction based on averaging of all 143 years before 2014. What result is better?

Task 5. Repeat Task 5, but this time find nearest neighbors based on daily temperatures during the first 3 months of the year. Compare the results and conclude if it resulted in better accuracy. Did you expect it to happen?

2). Find the nearest neighbors of 2014 with whole year monthly average of max or min temperature.

Instead of comparing daily temperatures, we can compare monthly temperatures. The following is the code you can use to perform the nearest neighbor algorithm on monthly data.

```
# Create the required matrix
years = np.arange(1874, 2015)
monthly_maxave_mtx = matrix_generation.monthly_ave_mtx(years, 'Max')
# Find the nearest neighbors of 2014 based on the monthly average of max temperature
matrix.
from scipy.spatial.distance import pdist, squareform
data_dist_flat = pdist(monthly_maxave_mtx, 'euclidean') # computing the distance
data_dist_square = squareform(data_dist_flat) # make the flat distance as a matrix
```

```
nearest_neighbors_indices = data_dist_square.argsort(axis = 1) # sort each row in ascending
order
k = 5
knn_years = years[nearest_neighbors_indices[-1, 2:(2+k)]]
print('The 5 nearest years of 2014 based on monthly average of max temperature matrix is: '
+ str(knn_years))
```

This piece of code printed the following line:

The 5 nearest years of 2014 based on monthly average of max temperature matrix is: [1959 1957 1971 1996 1951]

Task 7. Study the code and summarize what it does in one or two paragraphs.

Task 8. Repeat Tasks 5 and 6, this time using the monthly data instead of daily data. What do you expect should give better neighbors, daily or monthly data? Why? Discuss in one paragraph. Look at the results and check if your reasoning was correct.

Task 9. In Tasks 5, 6, and 8 you tried to predict July temperatures in 2014. Whatever prediction accuracy you observed might have occurred due to pure luck. Let us now perform a more extensive experiment. Pick July 1874 as the prediction task and find 10 nearest neighbor years among the remaining years. Record the mean squared prediction error for that year. Then, repeat this in July 1875. Keep repeating for all other years. In this way, you will get a total of 144 mean squared prediction errors. What is the average of these numbers? What is the histogram of these numbers? How does finding nearest years based on January temperatures compare to finding nearest neighbors based on January + February + March temperatures? Do you think results make sense and why?

Task 10 (EXTRA CREDIT). Feel free to explore nearest neighbor algorithm more to try to improve the results and provide additional insights about Philadelphia weather. You will ge graded on your effort and creativity.