Name: Bryan Cheng Hengze

Task A: Data Exploration and Auditing

import pandas as pd

tao = pd.read_csv('TAO_2006.csv')

Firstly, I import the pandas library with an alias pd which provides a data structure called a DataFrame. Then, I load the Tropical Atmosphere Ocean data (TAO_2006.csv) into a pandas DataFrame called 'tao' using Pandas.

A1. Dataset size

tao.shape

.shape returns a tuple which gives the axis dimensions of the table (Row, Column)

Row: 35136

Column: 8

A2. Min/Max values in each column

tao.min()

.min() shows the minimum values of each column

Minimum values:

- Precipitation (PREC): -9.99
- Air Temperature (AT): -99.9
- Sea surface temperature (SST): -99.9
- Relative humidity (RH): -99.9

tao.max()

.max() shows the maximum values of each column

Maximum values:

- Precipitation (PREC): 75.77
- Air Temperature (AT): 31.57
- Sea surface temperature (SST): 31.346
- Relative humidity (RH): 98.1

A3. Number of records in each month

print(tao['YYYYMMDD'].dtype)

tao['YYYYMMDD'] = pd.to_datetime(tao['YYYYMMDD'], format='%Y%m%d')

print(tao['YYYYMMDD'].dtype)

.dtype gives the data type of the column

The data type for the column 'YYYYMMDD' was an integer.

It did not have the correct data type. Therefore, I converted it to datetime data type using .to_datetime() method with the correct format that it was in which was YYYYMMDD.

The data type for the column 'YYYYMMDD' is now datetime.

tao['Month'] = tao['YYYYMMDD'].dt.month

Then, I extracted the 'Month' from column 'YYYYMMDD' using method .dt.month, which returns the month of the datetime from 'YYYYMMDD'.

func = {'Month':{'Number of records':'count'}}

tao_filt= tao.groupby('Month').agg(func).reset_index()

tao_filt.columns = tao_filt.columns.droplevel(0)

tao_filt= tao_filt.rename(columns = {"":"Month"})

tao_filt

I took the "Month" column because it's the column where I want to apply the aggregation to, followed by the name of the resulting new column which is "Number of records" and the aggregation operation which is "count" and assign it to a variable called func.

After that, I groupby 'Month' which combines the results of the each month and make it group into one row and applied the aggregation operation to it. I also "flatten it" using the 'reset_index()' and 'droplevel()' commands so that the output of the groupby operation can be further manipulated. I assigned these into a variable called tao_filt.

Flattening will lose the column name for the 'Month' attribute. Therefore, I renamed the column by using .rename(columns = { '' : 'Month'})

Therefore, these codes will allow me to print out the number of records in each month.

The two months that have the lowest number of records is September and February. This is because February is not a leap year which makes the days lesser compared to the other months which makes the number of records one of the lowest. September as only 144 records because the dataset only has the 1st day of September.

A4. Missing Values

1)

```
import numpy as np

tao2 = pd.read_csv('TAO_2006.csv')

tao2['YYYYMMDD'] = pd.to_datetime(tao['YYYYMMDD'], format='%Y%m%d')

tao2['Month'] = tao2['YYYYMMDD'].dt.month

cols = ["Timestamp","YYYYMMDD","HHMMSS","PREC","AIRT","SST","RH","Q"]

tao2[cols] = tao2[cols].replace({'-9.99':np.nan, -9.99:np.nan})

tao2[cols] = tao2[cols].replace({'-99.9':np.nan, -99.9:np.nan})
```

First, I import the numpy library with an alias np .

I load the Tropical Atmosphere Ocean data (TAO_2006.csv) and assign it to a variable called tao2 because I do not want to change whatever that is in the main dataset as we might use that later.

Then I took all the columns from tao2 and replaced all the missing values which is -9.99 and -99.9 with NaN using .replace

```
null = tao2[tao2.isnull().any(axis=1)]

null.shape[0]
```

I then check for any null (at least one True per row) and filter with Boolean indexing using .isnull().any(axis=1) and assigned it to a variable called null.

.shape[0] will give me the number of rows containing missing values in the dataset.

The number of rows containing missing values is 401 rows.

2)

```
set(tao2['Month'].unique())-set(null['Month'].unique())
```

.unique displays the unique values into a list

I used set difference to check the months with no missing values in them (all the months minus the months which contain null values)

Therefore, months with no missing values in them are 2,5,9 (February, May, September).

3)

tao2 = tao2.dropna()

.dropna() drops the rows where at least one element is missing.

Therefore, the records with missing values is removed and I assigned it to tao2 as I will be using the dataset with missing values removed from here onwards.

A5. Investigating Sea surface temperature (SST) in different months

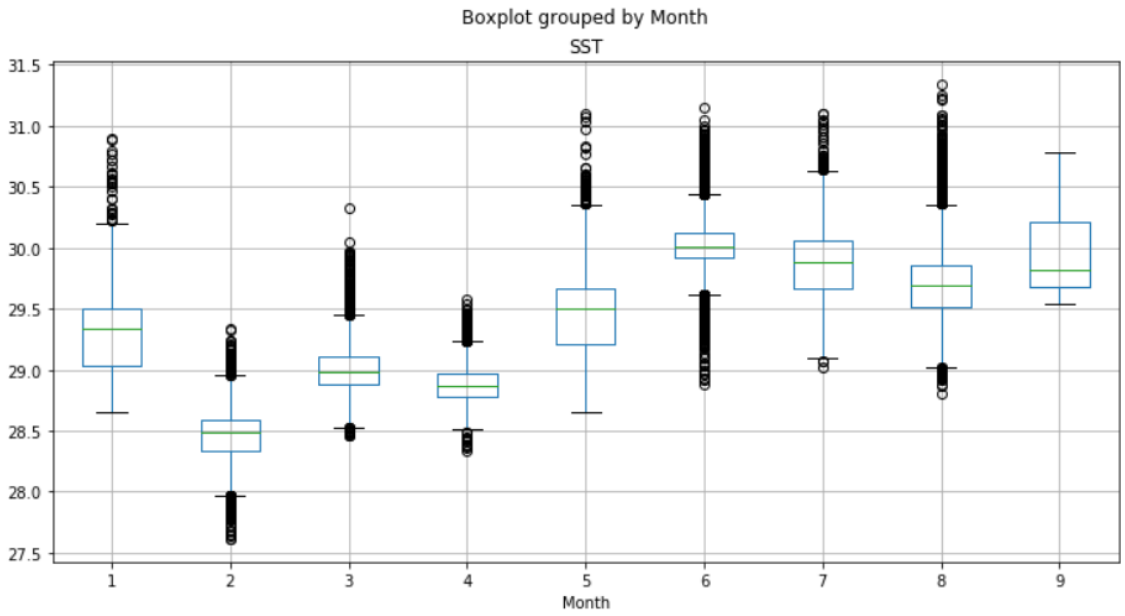
1)

tao2.boxplot(by="Month", column="SST",figsize=(12,6))

.boxplot is used to plot the boxplot

The by option takes an object by which the data can be grouped and the column option lets you separate the data.

Therefore, I created a boxplot grouped by Month with the 'SST' column.



2)

From the graph above, we can see that the median of Sea Surface Temperature (SST) decreases from month 1 (January) to month 2 (February). During month 2, the median of

SST is the lowest between the 9 months. The median of SST then increases from month 2 (February) to month 3 (March) and then slightly drop the month after (April). After month 4 (April) the median of SST rises considerably up to month 6 (June) and is the highest between the 9 months. The median then slowly decreases from month 6 (June) to month 8 (August). Finally, the median of SST increases slight after month 8.

3)

filt = tao2.groupby(['Month'])['SST'].median().reset_index()

To get the median of SST in each month, I groupby 'Month' which combines it with the 'SST' and get the median of 'SST' using the .median() method. I also "flatten it" using the 'reset_index()' so that the output of the groupby operation can be further manipulated. I assigned these into a variable called filt.

filt["Month"][filt["SST"].idxmax()]

filt["Month"][filt["SST"].idxmin()]

.idxmax() gets the index of the maximum value of the median SST

.idxmin() gets the index of the minimum value of the median SST

Therefore, the month with the highest median SST is month 6 (June) and the month with the lowest median SST is month 2 (February).

A6. Exploring precipitation measurements (PREC)

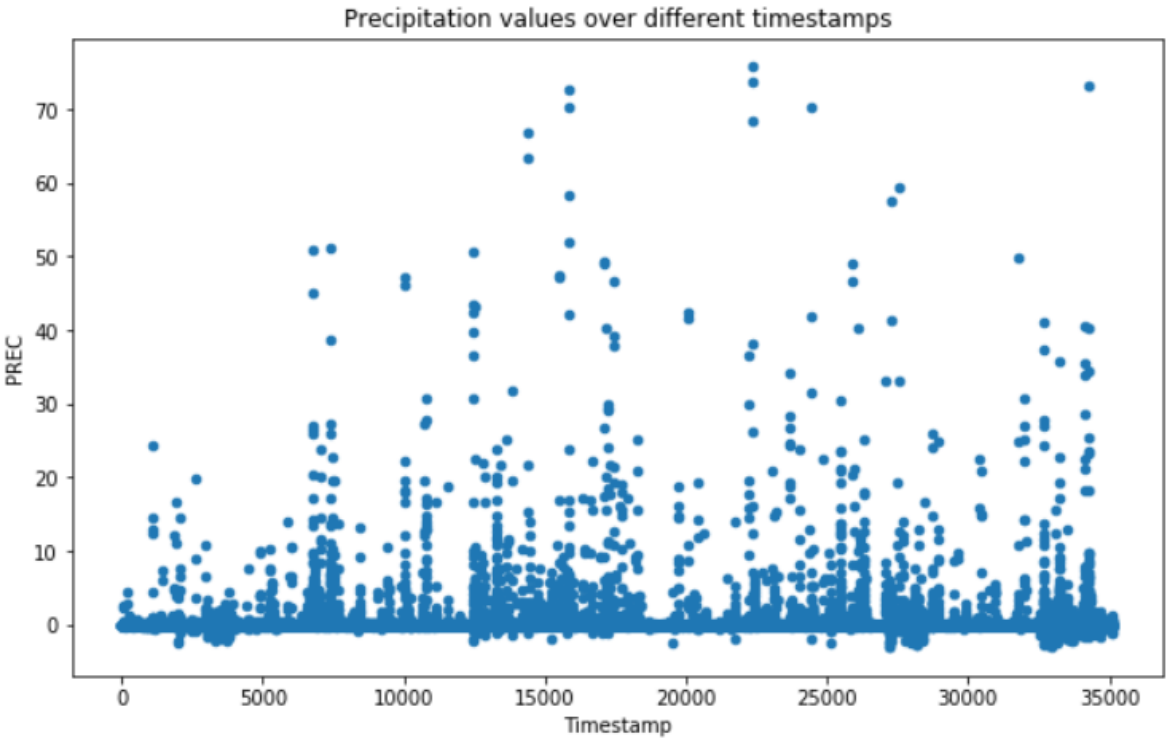
1)

import matplotlib.pyplot as plt

To plot the graph, I first need to import the matplotlib.pyplot library and with an alias plt.

tao2.plot(kind='scatter',x='Timestamp',y='PREC', title ='Precipitation values over different timestamps',figsize=(10,6))

This line of code plots a scatter graph, with the x-axis being Timestamp and y-axis being PREC.



2)

tao2.loc[tao2['PREC'] < 0, 'PREC'] = 0

.loc allows me to access a group of rows and columns by label(s) or a boolean array.

The counter-intuitive values in this Precipitation column is negative precipitation.

Therefore, the line of code above takes all the values which has a precipitation value lesser than 0 and replaces them with zero.

A7. Relationship between variables

1)

tao2['PREC'].corr(tao2['AIRT'])

tao2['PREC'].corr(tao2['SST'])

tao2['AIRT'].corr(tao2['SST'])

Each line of code above calculates the correlation between two columns.

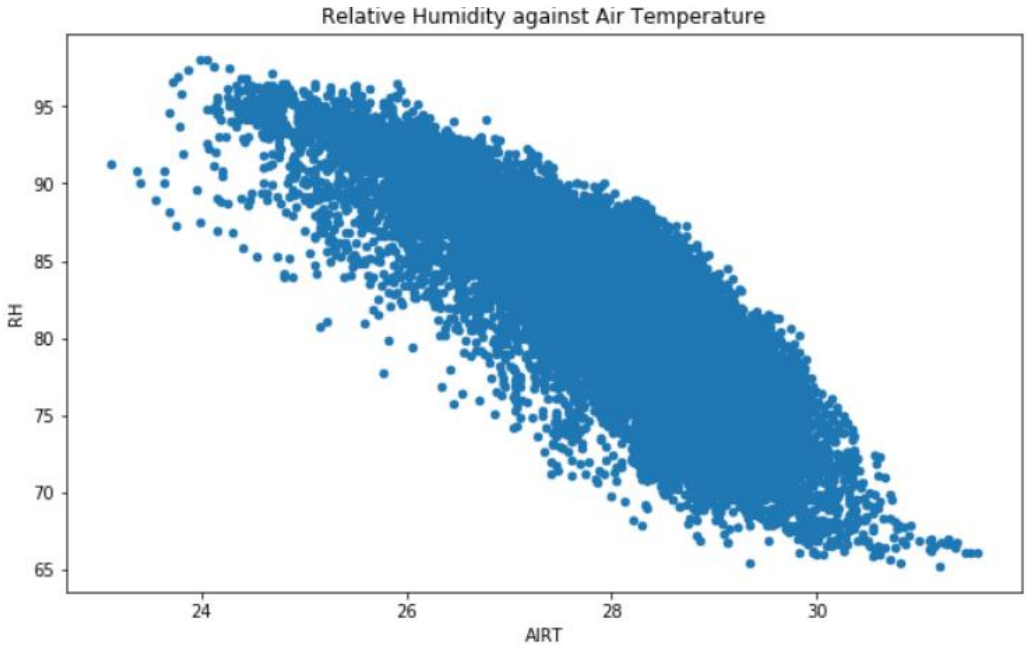
These lines of code computes the pairwise correlation of columns, precipitation, air temperature and surface temperature.

The two features which have the least linear association is the precipitation and surface temperature -0.03485242673000577

2)

tao2.plot(kind='scatter',x='AIRT',y='RH',title = 'Relative Humidity against Air Temperature',figsize=(10,6))

This line of code plots a scatter graph, with the x-axis being AIRT and y-axis being RH.



The graph that I plot above shows the relative humidity against air temperature.

Yes there is a relationship between these two features. From the scatter graph above, we can see that there is a strong negative correlation between the air temperature and relative humidity as they move in opposite directions. That is, if as AIRT increases in value, RH will decrease.

A8. Predicting quality of measurements (Q)

1)

a = tao2.iloc[:, [3, 4, 5, 6]].values

.iloc is an integer-location based indexing for selection by position

I took the values of index 3,4,5,6 which are values of PREC, AIRT, SST, RH and assigned it to a variable called a.

b = tao2.iloc[:, 7].values

I took the value of index 7 which is value Q and assigned it to a variable called b.

from sklearn.model_selection import train_test_split

a_train, a_test, b_train,b_test = train_test_split(a, b, test_size = 0.25, random_state = 0)

Then I import the train_test_split from the sklearn.model_selection module to split the dataset into the Training set and Test set.

2)

from sklearn.tree import DecisionTreeClassifier

classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)

classifier.fit(a_train, b_train)

I import the DecisionTreeClassifier from the sklearn. tree module to fit a decision tree classification to the training set.

b_pred = classifier.predict(a_test)

I then predict the train set results and assigned it to variable b_pred.

from sklearn.metrics import confusion_matrix

confusionm = confusion_matrix(b_test, b_pred)

print('Confusion Matrix : \n')

print(confusionm)

I import the confusion_matrix from sklearn.metrics module to make the confusion matrix and then I assigned it to a variable called confusionm.

Therefore, we can see that the confusion matrix is:

```
[[8670    8]
 [   4    2]]
```

from sklearn.metrics import accuracy_score

print('Accuracy : ', accuracy_score(b_test, b_pred))

I import the accuracy_score from sklearn.metrics module to compute the accuracy.

Therefore, we can see that the accuracy is 0.9986181483187471.

3)

Yes, I think this is a good model because the prediction is correct almost all the time, 99.86181483187471% of the time to be exact.

Sensitivity should also be considered as well. This is because when the actual value is positive we need to see the ones that are correctly identified.

Precision should also be considered as well because we also need to know when a positive value is predicted, how often will the prediction be correct.

A9. Investigating daily relative humidity (RH)

1)

```
tao2['Day'] = tao2['YYYYMMDD'].dt.day
```

```
filt = tao2.groupby('YYYYMMDD')[['RH']].median().reset_index()
```

Then, I extracted the 'day from column 'YYYYMMDD' using method .dt.day, which returns the day of the datetime from 'YYYYMMDD'.

To get the median of RH in each day, I groupby 'YYYYMMDD' which combines it with the 'RH' and get the median of 'RH' using the .median() method. I also "flatten it" using the 'reset_index()' so that the output of the groupby operation can be further manipulated. I assigned these into a variable called filt.

```
day = []
for i in range(len(filt["YYYYMMDD"])):
    day.append(i+1)
filt["Day"] = day
filt.drop(columns = "YYYYMMDD", inplace = True)
```

After that, I made an empty list and assigned it to a variable called day.

Then I iterate through the length of the column YYYYMMDD and I append I to the day so that it increases by 1 every time even if it past the first month.

So, the days will go up to 244.

I then set the day to the column 'Day' in filt.

I also drop the columns "YYYYMMDD" as we are not going to be using it.

```
from sklearn.linear_model import LinearRegression
a = filt.iloc[:, [1]].values.reshape(-1,1)
b = filt.iloc[:, [0]].values.reshape(-1,1)
```

lin = LinearRegression()

lin.fit(a, b)

I import the LinearRegression from sklearn.linear_model module to compute an ordinary least squares Linear Regression.

Then I divided the dataset into two components which is a and b. a will have the values of column 1 (Day) and b will have the values of column 0 (RH)

After that, I fit the linear regression model on a and b.

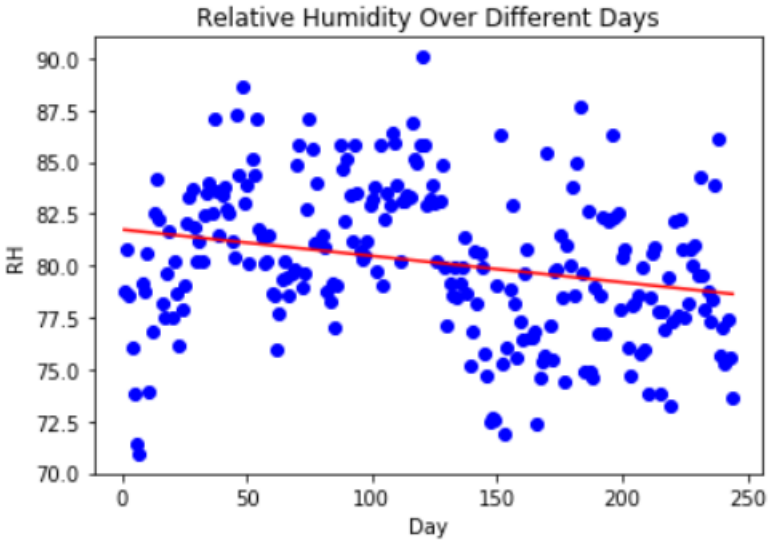
b_pred = lin.predict(a)

plt.scatter(a, b, color = 'blue')

plt.plot(a, b_pred, color = 'red')

plt.show()

The codes above is to plot the linear regression result using scatter plot.



2)

print(lin.predict([[245]]))

To predict the values of the median relative humidity on 2nd September 2006, I used the .predict() function.

The prediction for the median relative humidity on 2nd September 2006 is 78.62522769

3)

from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree = 5)

lin2 = LinearRegression()

```
a = filt.iloc[:, [1]].values

b = filt.iloc[:, [0]].values

a_poly = poly.fit_transform(a)

lin2.fit(a_poly, b)
```

I import the PolynomialFeatures from sklearn.preprocessing module which allows me to generate polynomial and interaction features.

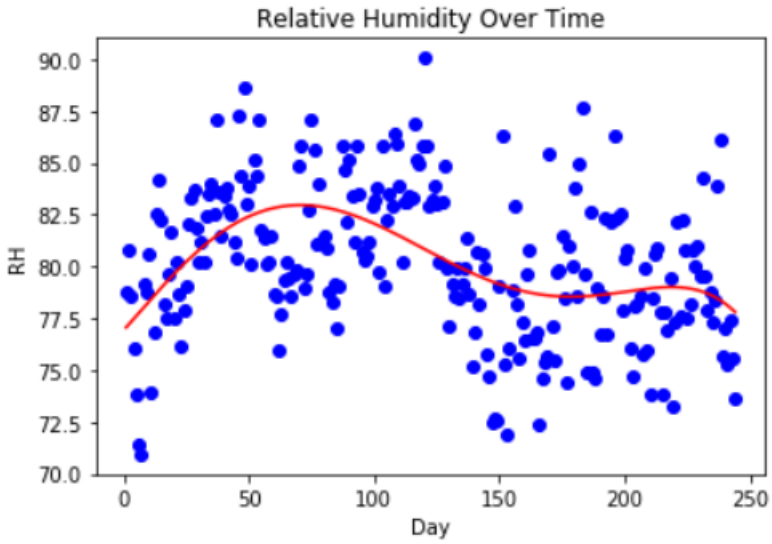
I have set my degree of polynomial features to 5 after trying different of polynomial as I think it is the best fit for the aggregated data.

Then I divided the dataset into two components which is a and b. a will have the values of column 1 (Day) and b will have the values of column 0 (RH)

After that, I fit the polynomial regression model on components a and b

```
plt.scatter(a, b, color = 'blue')

plt.plot(a, lin2.predict(poly.fit_transform(a)), color = 'red')

plt.title('Relative Humidity Over Time')

plt.xlabel('Day')

plt.ylabel('RH')

plt.show()
```

The codes above is to plot the polynomial regression result using scatter plot.



The model that I suggested above shows the near best fit to the aggregated data. The model is not under fitting nor overfitting. For higher orders, these fits are not even close to the best possible fit.

This model is better suited for this task because it has a wide spread. Therefore, polynomial can fit a wide range of curvature and can provide the best estimation of the relationship between the day and RH.

4)

lin2.predict(poly.transform([[245]]))

To predict the values of the median relative humidity on 2nd September 2006 with polynomial regression, I used the .predict(poly.transform()) function.

The prediction for the median relative humidity of my new model on 2nd September 2006 is 77.68150497

The difference between the prediction of my new model with the previous linear fit is close to 1.

A10. Filling in missing values

poly = PolynomialFeatures(degree = 5)

lin2 = LinearRegression()

a = tao.iloc[:, [0]].values

b = tao.iloc[:, [6]].values

a_poly = poly.fit_transform(a)

lin2.fit(a_poly, b)

PolynomialFeatures from sklearn.preprocessing module which allows me to generate polynomial and interaction features.

I have set my degree of polynomial features to 5 after trying different of polynomial as I think it is the appropriate regression model and best fit for the data.

Then I divided the dataset into two components which is a and b. a will have the values of column 0 (Timestamp) and b will have the values of column 6 (RH)

*I am using tao now as I need to fill in the missing values

After that, I fit the polynomial regression model on components a and b

plt.scatter(a, b, color = 'blue')

```
plt.plot(a, lin2.predict(poly.fit_transform(a)), color = 'red')

plt.title('Relative Humidity Over Time')

plt.xlabel('Timestamp')

plt.ylabel('RH')

plt.show()
```

The codes above is to plot the polynomial regression result using scatter plot to make me visualize the model better.

```
for i in range(len(tao['RH'])):

    if (tao['RH'][i] == -9.99 or tao['RH'][i] == -99.9):

        tao.loc[tao['Timestamp'] == i,'RH'] =
lin2.predict(poly.transform([[tao['Timestamp'][i]]]))[0]
```

Then in order to fill in the missing values, I iterated through the length of RH and if any row of RH is equal to -9.99 or -99.9, I will take the predicted value of RH from the polynomial fit and assign it to the RH.

Therefore, this will fill in the missing values for column RH using an appropriate regression model.

Task B: K-means Clustering on Other Data

Question 1 and 2

```
from sklearn.cluster import KMeans
```

I import the KMeans from sklearn.cluster module to apply k-means clustering on my data into k clusters.

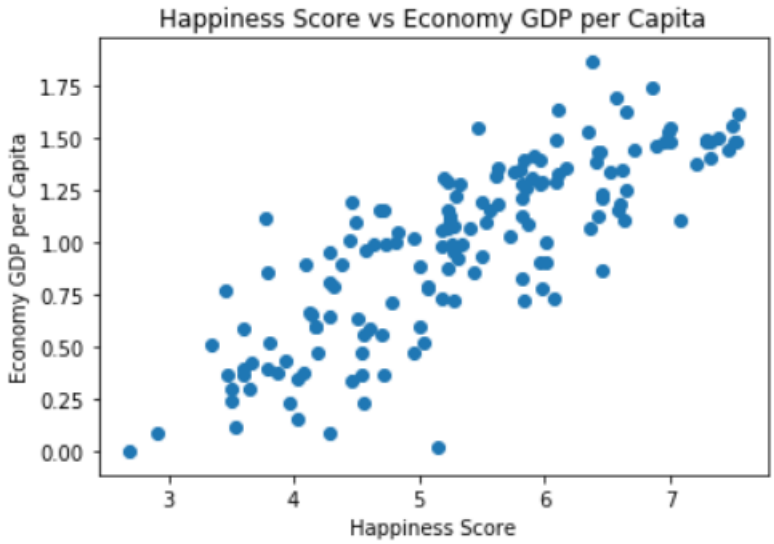
```
df = pd.read_csv('happiness.csv')
```

The dataset is taken from: https://www.kaggle.com/unsdsn/world-happiness

Then, I load the happiness data (happiness.csv) into a pandas DataFrame called 'happy' using Pandas.

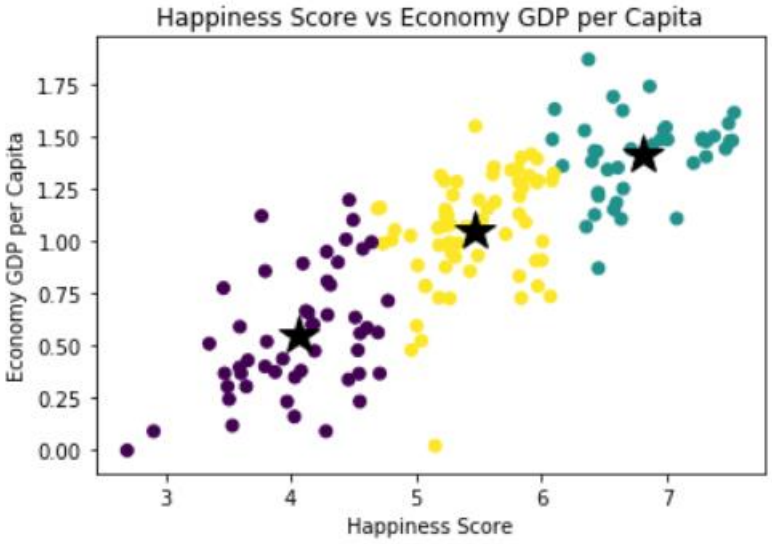
```
plt.scatter(x=happy['Happiness.Score'],y=happy['Economy..GDP.per.Capita.'])

plt.title('Happiness Score vs Economy GDP per Capita')

plt.xlabel('Happiness.Score')

plt.ylabel('Economy..GDP.per.Capita.')
```

These line of code plots a scatter graph, with the x-axis being the happiness score and y-axis being economy GDP per capita.



The chart above shows the dataset for 155 countries, with the Happiness Score on the x-axis and the economy GDP per Capita on the y-axis.

```
cluster = KMeans(n_clusters=3).fit(happy[['Happiness.Score','Economy..GDP.per.Capita.']])

plt.scatter(x=happy['Happiness.Score'],y=happy['Economy..GDP.per.Capita.'],
c=cluster.labels_)

plt.plot(cluster.cluster_centers_[:,0],cluster.cluster_centers_[:,1],'k*',markersize=20)

plt.xlabel('Happiness')

plt.ylabel('GDP per Capita')

plt.show()
```



3)

After testing over several K values, I chose 3 clusters as I could see three distinct groups that have been identified by the algorithm. We can see now that the happiness score has been divided into 3 groups. The purple dots are the countries that are a less happy and have a relatively low economy GDP per capita. The yellow dots are the countries with an average happiness and a rather stable economy GDP per Capita. Lastly, the green dots are the countries with high happiness and a high economy GDP per capita. Therefore, through this clustering, we can clearly deduce that the happiness of a country is highly dependent on the economy GDP per Capita. The higher the economy GDP per capita the higher the happiness score of the country.

4)

The quality of my clusters is not exactly perfect. This is because a survey conducted by Gallup has shown that the although the US has the world's highest GDP, they sit at number 19 in the world happiest country. Jeffrey Sach (2019) states that the reason behind that is because of the worsening health conditions and declines in social trust and trust in government were making Americans less happy. This clearly shows that money can't buy you happiness and the higher a countries GDP does not mean that they have highest happiness score. Therefore, this means that we should also consider other factors affecting happiness of a country such as freedom, generosity, corruption levels, etc. and cannot compare with two numerical features and make a overall conclusion.

Taken from: https://www.weforum.org/agenda/2019/03/finland-is-the-world-s-happiest-country-again/