Data Science Assignment 2

Name: Germaine Pok Yi Min Student ID: 2979782

Task A: Data Wrangling and Analysis on TAO dataset

In this task, you are required to explore the dataset and do some data analysis on the Tropical Atmosphere Ocean dataset. Have a look at the csv file (TAO_2006.csv) and then answer a series of questions about the data using Python.

A1. Dataset size

How many rows and columns exist in this dataset?

35125	35125	20060901	221000	-1.03	29.28	29.592	74.0	HighQ
35126	35126	20060901	222000	0.72	29.25	29.630	72.4	HighQ
35127	35127	20060901	223000	-0.58	29.32	29.660	72.7	HighQ
35128	35128	20060901	224000	-0.24	29.23	29.636	74.6	HighQ
35129	35129	20060901	225000	0.04	29.24	29.669	73.9	HighQ
35130	35130	20060901	230000	0.23	29.32	29.687	74.5	HighQ
35131	35131	20060901	231000	0.40	29.20	29.680	75.6	HighQ
35132	35132	20060901	232000	-0.61	29.25	29.686	75.4	HighQ
35133	35133	20060901	233000	-0.11	29.15	29.697	75.2	HighQ
35134	35134	20060901	234000	-0.01	29.18	29.704	74.2	HighQ
35135	35135	20060901	235000	0.12	29.20	29.718	74.9	HighQ

35136 rows × 8 columns

The name of my data frame is tao, the number of rows and columns that exist in the dataset can be obtained at the bottom of the dataset or by:

tao.shape

```
Out[905]: (35136, 8)
```

There are 35136 columns and 8 rows in this data set.

A2. Min/Max values in each column

Find maximum and minimum values for Precipitation (PREC), Air temperature (AT), Sea surface temperature (SST) and Relative humidity (RH) in this dataset.

The min values are found using ***tao[['PREC', 'AIRT','SST','RH']].min()***

```
Out[906]: PREC    -9.99
          AIRT   -99.90
          SST    -99.90
          RH     -99.90
          dtype: float64
```

The max values are found using ***tao[['PREC', 'AIRT','SST','RH']].max()***

```
Out[907]: PREC    75.770
          AIRT    31.570
          SST     31.346
          RH      98.100
          dtype: float64
```

A3. Number of records in each month

List the number of records in each month. In which two months are the number of records at their lowest? Why?

The data is first converted to string through:

tao['YYYYMMDD'] = tao['YYYYMMDD'].astype('str')

then it is converted to datetime[64]ns through:

tao['YYYYMMDD'] = tao['YYYYMMDD'].astype('datetime64[ns]')

This is because that originally Timestamp's data type is incorrect hence it needs to be forcefully change to create a new column called 'Month'.

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

When tao is printed:

```
In [986]: tao
```

Out[986]:

	Timestamp	YYYYMMDD	HHMMSS	PREC	AIRT	SST	RH	Q	Month
0	0	2006-01-01	0	-0.17	28.75	29.690	79.8	HighQ	1
1	1	2006-01-01	1000	0.00	28.86	29.708	79.2	HighQ	1
2	2	2006-01-01	2000	-0.02	28.91	29.749	79.4	HighQ	1
3	3	2006-01-01	3000	-0.01	28.85	29.757	78.4	HighQ	1
4	4	2006-01-01	4000	-0.02	28.87	29.787	77.8	HighQ	1
5	5	2006-01-01	5000	-0.04	28.71	29.793	78.2	HighQ	1
6	6	2006-01-01	10000	-0.02	28.66	29.803	77.5	HighQ	1
7	7	2006-01-01	11000	0.01	28.64	29.815	77.5	HighQ	1
8	8	2006-01-01	12000	-0.01	28.64	29.788	78.3	HighQ	1
9	9	2006-01-01	13000	-0.01	28.63	29.760	79.6	HighQ	1
10	10	2006-01-01	14000	0.00	28.74	29.786	79.6	HighQ	1

There is a new column called Month, where it display the month where the data was recorded

To obtain the number of records in a month:

groupbyMonth = tao.groupby('Month')['Timestamp'].count()

the number of occurrences of Timestamp in a month is counted to produce the number of record in each month:

```
Out[989]: Month
          1    4464
          2    4032
          3    4464
          4    4320
          5    4464
          6    4320
          7    4464
          8    4464
          9     144
          Name: Timestamp, dtype: int64
```

A4. Missing values

There are some missing values: -9.990000 and -99.900000 represent missing values.

1. How many rows contain missing values (-9.990000 or -99.900000) in this dataset?

condition1 = tao[(tao['PREC'] == -9.990000) | (tao['PREC'] == -99.900000) |(tao['AIRT'] == -9.99000) | (tao['AIRT'] == -99.900000)| (tao['SST'] == -9.990000) | (tao['SST'] == -99.900000) | (tao['RH'] == -9.990000) | (tao['RH'] == -99.900000)]

This code creates a new data frame called condition1 that has 401 rows. This data frame contains data with missing values.

Hence there are 401 row that contains missing values. Hence there are 401 missing values

condition1.shape

```
Out[991]: (401, 9)
```

2. List the months with no missing values in them.

condition1 is a data frame of months that contains missing values.

Hence, if we find the unique values of the column 'Month' then we would get the months that has no missing values

Eg. ***condition1['Month'].unique()***

```
Out[993]: array([1, 3, 4, 6, 7, 8])
```

As shown as above, **January, March, April, June, July and August** are months that **contains** missing values.

Hence, the missing values which are **February, May and** September are months **without** any missing values.

3. Remove the records with missing values.

condition2 = ((tao["PREC"] != -9.99)&(tao["AIRT"] != -99.9)&(tao["SST"] != -99.9)&(tao["RH"] != -99.9))

A condition was created where it eliminates data with missing values.

tao = tao[condition2]

The tao is then converted to a data frame where the missing values are removed

```
tao
```

35125	35125	2006-09-01	221000	-1.03	29.28	29.592	74.0	HighQ	9
35126	35126	2006-09-01	222000	0.72	29.25	29.630	72.4	HighQ	9
35127	35127	2006-09-01	223000	-0.58	29.32	29.660	72.7	HighQ	9
35128	35128	2006-09-01	224000	-0.24	29.23	29.636	74.6	HighQ	9
35129	35129	2006-09-01	225000	0.04	29.24	29.669	73.9	HighQ	9
35130	35130	2006-09-01	230000	0.23	29.32	29.687	74.5	HighQ	9
35131	35131	2006-09-01	231000	0.40	29.20	29.680	75.6	HighQ	9
35132	35132	2006-09-01	232000	-0.61	29.25	29.686	75.4	HighQ	9
35133	35133	2006-09-01	233000	-0.11	29.15	29.697	75.2	HighQ	9
35134	35134	2006-09-01	234000	-0.01	29.18	29.704	74.2	HighQ	9
35135	35135	2006-09-01	235000	0.12	29.20	29.718	74.9	HighQ	9

34735 rows × 9 columns

Originally, the data frame had 35136 rows of data, the new data frame with the missing values removed have 34735 rows of data.

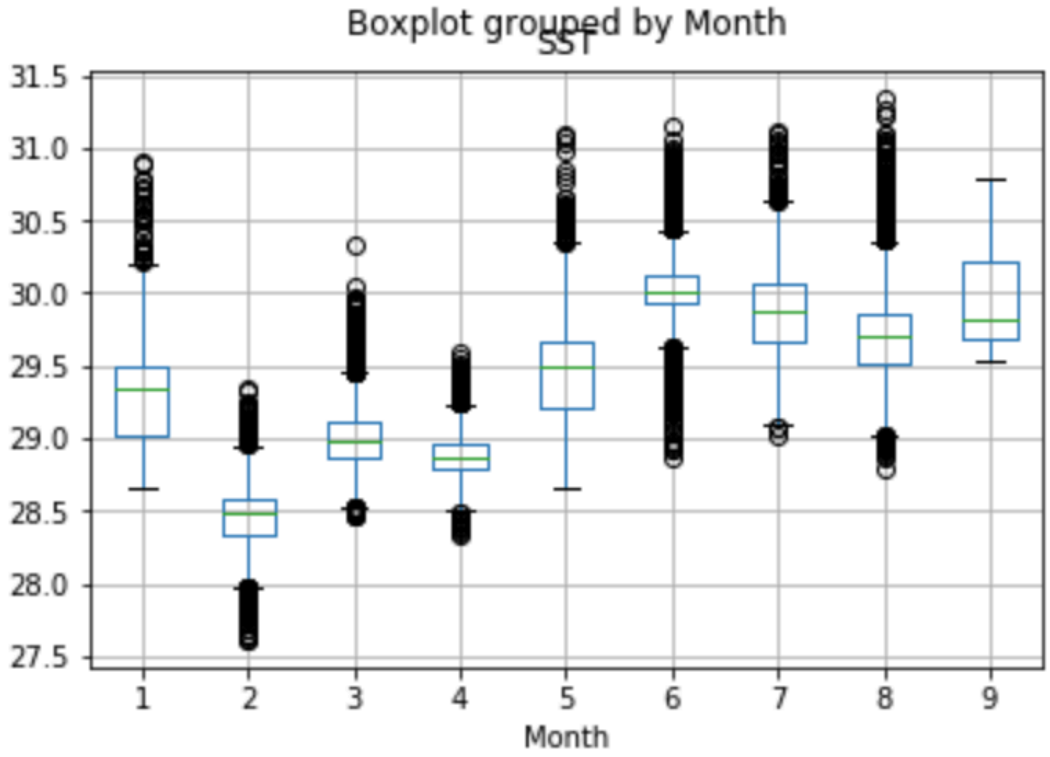
A5. Investigating Sea surface temperature (SST) in different months

Now look at the sea surface temperature (SST) column and answer the following questions

1. Using a boxplot, visualize the distribution of SST over different months.

tao.boxplot(column = 'SST', by = 'Month')

This line of code visualises the boxplot.



2. Describe the trend of median SST over different months.

The median of boxplot alternatively increased and decrease from January to May. It then increases in June to the decrease again in July and August to then increase slightly in September.

3. Which month has the highest median SST? Which month has the lowest?

I first grouped SST by Month to only obtain data related to Month and SST.

median = tao.groupby('Month')['SST'].median()

median = median.reset_index()

Then using the code below, I obtained the month with the lowest SST

median[median['SST'] == median['SST'].min()]

Out[1066]:

	Month	SST
1	2	28.484

Then the month with the highest SST.

median[median['SST'] == median['SST'].max()]

Out[1067]:

	Month	SST
5	6	30.013

A6. Exploring precipitation measurements (PREC)

Now look at the Precipitation column and answer the following questions

1. Precipitation values in this dataset show rain rates. Plot Precipitation values over different timestamps.

groupbyTimestamp = tao.groupby('Timestamp')['PREC'].sum()

groupbyTimestamp = groupbyTimestamp.reset_index()

```
In [1113]: groupbyTimestamp
```

Out[1113]:

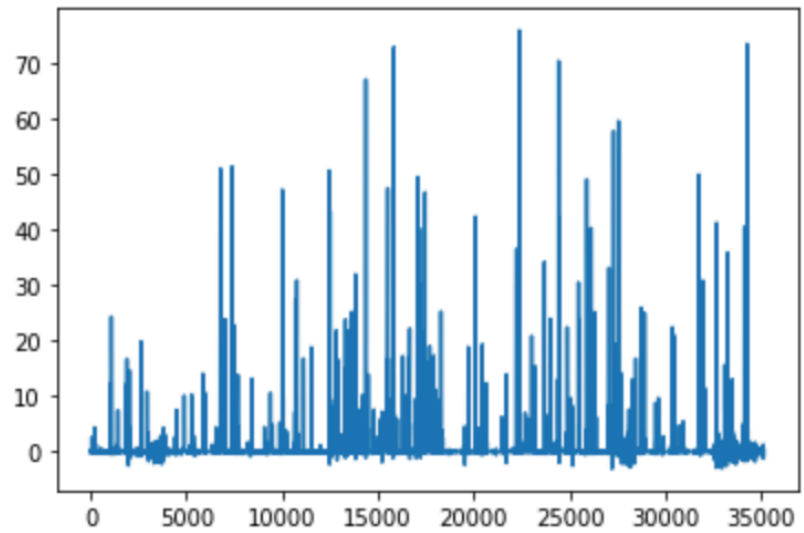
	Timestamp	PREC
0	0	-0.17
1	1	0.00
2	2	-0.02
3	3	-0.01
4	4	-0.02
5	5	-0.04
6	6	-0.02
7	7	0.01
8	8	-0.01
9	9	-0.01
10	10	0.00

Timestamp and PREC were grouped together to form the data frame above.

The data in the data frame was then used to plot a graph of precipitation values over different timestamps using the code below.

plt.plot(groupbyTimestamp['Timestamp'], groupbyTimestamp['PREC'])

plt.show()



2. Due to **measurement error**, there are some counter-intuitive values in Precipitation column. Identify those values and replace them with **zero**.

The interquartile range was obtained through the code below to find the outliers

Q1 = groupbyTimestamp['PREC'].quantile(0.25)

Q3 = groupbyTimestamp['PREC'].quantile(0.75)

IQR = Q3 - Q1

Then a condition was created to obtain the data that are outliers or is negative.

condition2 = ((tao['PREC'] < (Q1 - 1.5 * IQR)) |(tao['PREC'] > (Q3 + 1.5 * IQR)) | (tao['PREC'] < 0))

```
In [1074]: tao[condition2]
```

35124	35124	2006-09-01	220000	0.75	29.35	29.586	72.8	HighQ	9
35125	35125	2006-09-01	221000	-1.03	29.28	29.592	74.0	HighQ	9
35126	35126	2006-09-01	222000	0.72	29.25	29.630	72.4	HighQ	9
35127	35127	2006-09-01	223000	-0.58	29.32	29.660	72.7	HighQ	9
35128	35128	2006-09-01	224000	-0.24	29.23	29.636	74.6	HighQ	9
35130	35130	2006-09-01	230000	0.23	29.32	29.687	74.5	HighQ	9
35131	35131	2006-09-01	231000	0.40	29.20	29.680	75.6	HighQ	9
35132	35132	2006-09-01	232000	-0.61	29.25	29.686	75.4	HighQ	9
35133	35133	2006-09-01	233000	-0.11	29.15	29.697	75.2	HighQ	9
35134	35134	2006-09-01	234000	-0.01	29.18	29.704	74.2	HighQ	9
35135	35135	2006-09-01	235000	0.12	29.20	29.718	74.9	HighQ	9

24393 rows × 9 columns

There are 24394 data of PREC that are outliers or is negative

Referring to the link below,

https://stackoverflow.com/questions/34653215/pandas-replacing-values-on-specific-columns

the function ‘loc’ was used to replace all the outliers and negative number with zero

tao.loc[condition2, 'PREC'] = 0

Now, when tao is called, the outliers and negative data in the column ‘PREC’ are replaced with 0

Before converting:

Out[1064]:

	Timestamp	YYYYMMDD	HHMMSS	PREC	AIRT	SST	RH	Q	Month
0	0	2006-01-01	0	-0.17	28.75	29.690	79.8	HighQ	1
1	1	2006-01-01	1000	0.00	28.86	29.708	79.2	HighQ	1
2	2	2006-01-01	2000	-0.02	28.91	29.749	79.4	HighQ	1
3	3	2006-01-01	3000	-0.01	28.85	29.757	78.4	HighQ	1
4	4	2006-01-01	4000	-0.02	28.87	29.787	77.8	HighQ	1
5	5	2006-01-01	5000	-0.04	28.71	29.793	78.2	HighQ	1
6	6	2006-01-01	10000	-0.02	28.66	29.803	77.5	HighQ	1
7	7	2006-01-01	11000	0.01	28.64	29.815	77.5	HighQ	1
8	8	2006-01-01	12000	-0.01	28.64	29.788	78.3	HighQ	1
9	9	2006-01-01	13000	-0.01	28.63	29.760	79.6	HighQ	1
10	10	2006-01-01	14000	0.00	28.74	29.786	79.6	HighQ	1

After converting:

Out[1076]:

	Timestamp	YYYYMMDD	HHMMSS	PREC	AIRT	SST	RH	Q	Month
0	0	2006-01-01	0	0.00	28.75	29.690	79.8	HighQ	1
1	1	2006-01-01	1000	0.00	28.86	29.708	79.2	HighQ	1
2	2	2006-01-01	2000	0.00	28.91	29.749	79.4	HighQ	1
3	3	2006-01-01	3000	0.00	28.85	29.757	78.4	HighQ	1
4	4	2006-01-01	4000	0.00	28.87	29.787	77.8	HighQ	1
5	5	2006-01-01	5000	0.00	28.71	29.793	78.2	HighQ	1
6	6	2006-01-01	10000	0.00	28.66	29.803	77.5	HighQ	1
7	7	2006-01-01	11000	0.01	28.64	29.815	77.5	HighQ	1
8	8	2006-01-01	12000	0.00	28.64	29.788	78.3	HighQ	1
9	9	2006-01-01	13000	0.00	28.63	29.760	79.6	HighQ	1
10	10	2006-01-01	14000	0.00	28.74	29.786	79.6	HighQ	1

A7. Relationship between variables

1. Compute pairwise correlation of columns, precipitation, air temperature and surface temperature. Which two features have the least linear association?

Referring to the link below:

https://medium.com/brdata/correlation-straight-to-the-point-e692ab601f4c

I learned that the correlation of columns, precipitation, air temperature and surface temperature can be obtained through this code:

tao[["PREC", "AIRT", "SST"]].corr()

Out[1077]:

	PREC	AIRT	SST
PREC	1.000000	-0.039989	0.021276
AIRT	-0.039989	1.000000	0.366119
SST	0.021276	0.366119	1.000000

According to the data, PREC and AIRT has the least linear association

2. Now let's look at the relationship between air temperature and relative humidity. Plot the values of these features against each other. Is there any relationship between these two features? Describe it.

To find out the relationship of air temperature and relative humidity, a linear regression is plotted using the following codes:

from scipy.stats import linregress

slope, intercept, r_value, p_value, std_err = linregress(tao['AIRT'],tao['RH'])

line = [slope*xi + intercept for xi in tao['AIRT']]

fig= plt.figure(figsize=(8,5))

plt.plot(tao['AIRT'],line,'g-', linewidth=2)

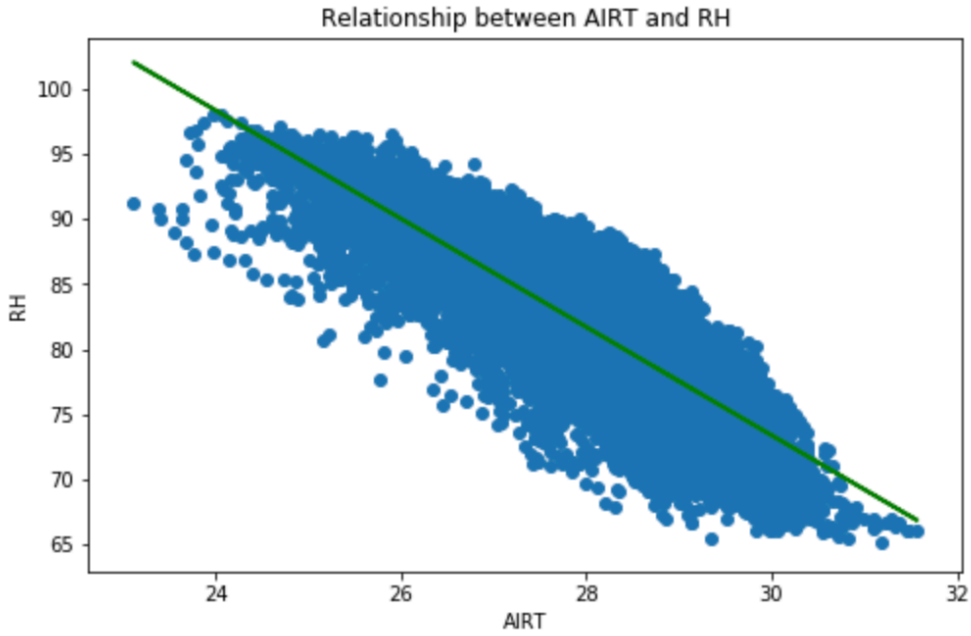
plt.scatter(tao['AIRT'], tao['RH'])

plt.xlabel('AIRT')

plt.ylabel('RH')

plt.title('Relationship between AIRT and RH')

plt.show()



When air temperature (AIRT) increases, the Relative Humidity (RH) decreases hence, they are both inversely proportional

A8. Predicting quality of measurements (Q)

We now want to build a predictive model to predict the quality of measurements (Q) in the dataset based on four features: Precipitation (PREC), Air temperature (AIRT), Sea surface temperature (SST) and Relative humidity (RH).

1. Divide the dataset into a 75% training set and a 25% testing set and train a decision tree model.

X = tao.iloc[:, [3,4,5,6]].values

y = tao.iloc[:, 7].values

The rows were selected using 'iloc'

Then, it is split into Training and Testing set:

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

Next, feature scaling is done so that range of all features is normalized so that each feature contributes approximately proportionately to the final results.

X_train = sc.fit_transform(X_train)

X_test = sc.transform(X_test)

Next, a decision tree classification is fitted to the training set.

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

classifier.fit(X_train, y_train)

```
Out[1085]: DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
                                  max_features=None, max_leaf_nodes=None,
                                  min_impurity_decrease=0.0, min_impurity_split=None,
                                  min_samples_leaf=1, min_samples_split=2,
                                  min_weight_fraction_leaf=0.0, presort=False,
                                  random_state=0, splitter='best')
```

The results of the test set is then predicted using the code below.

y_pred = classifier.predict(X_test)

2. Using test set, compute the confusion matrix and accuracy.

To make the confusion matrix, the code below was used:

from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)

```
Out[1087]: array([[8664,   14],
                  [   6,    0]])
```

Then the accuracy model was computed using:

from sklearn.metrics import accuracy_score

print(accuracy_score(y_test, y_pred))

```
0.9976969138645785
```

The accuracy is 0.9976969138645785.

3. Considering accuracy only, do you think that this is a good model? What other metric(s) should we consider as well? Why? Elaborate your answer.

Accuracy is one of the few metrics that can be used to predict a model. By using accuracy only its not a very good model, but it can be improved to obtain better model prediction. Instead of using accuracy only, other metrics like sensitivity, specificity, false positive rate and precision can be used together. Sensitivity and precision can be used to check how often our prediction will be correct when is positive. Specificity and false positive rate is used to check when negative values prediction would be correct.

A9. Investigating daily relative humidity (RH)

We will now investigate the trend in the daily relative humidity over time. For this, you will need to aggregate the median relative humidity by day.

1. Fit a linear regression using Python to this data (i.e., relative humidity over different days) and plot the linear fit.

day = tao[['RH', 'YYYYMMDD']]

import datetime as dt

groupbyDay = day.groupby('YYYYMMDD')['RH'].median()

groupbyDay = groupbyDay.reset_index()

groupbyDay['YYYYMMDD'] = pd.to_datetime(groupbyDay['YYYYMMDD'])

groupbyDay['YYYYMMDD']=groupbyDay['YYYYMMDD'].map(dt.datetime.toordinal)

I first created a data frame that contains ‘RH’ and ‘YYYYMMDD’. Then I grouped by ‘YYYYMMDD’ and created a data frame that contains the median of ‘RH’ .

Then using to.oordinal, the ‘YYYYMMDD’ is converted into a readable data as because the regression doesn’t read datetime format values.

Out[1133]:

	YYYYMMDD	RH
0	732312	78.80
1	732313	80.80
2	732314	78.60
3	732315	76.10
4	732316	73.85
5	732317	71.40
6	732318	70.95
7	732319	79.20
8	732320	78.80
9	732321	80.60
10	732322	73.95

The data frame above shows that the data type of ‘YYYYMMDD’ has been converted to a readable data to visualise the linear regression.

The code below is used to visualise the linear regression:

slope, intercept, r_value, p_value, std_err = linregress(groupbyDay['YYYYMMDD'],groupbyDay['RH'])

line = [slope*xi + intercept for xi in groupbyDay['YYYYMMDD']]

fig= plt.figure(figsize=(8,5))

plt.plot(groupbyDay['YYYYMMDD'],line,'g-', linewidth=2)

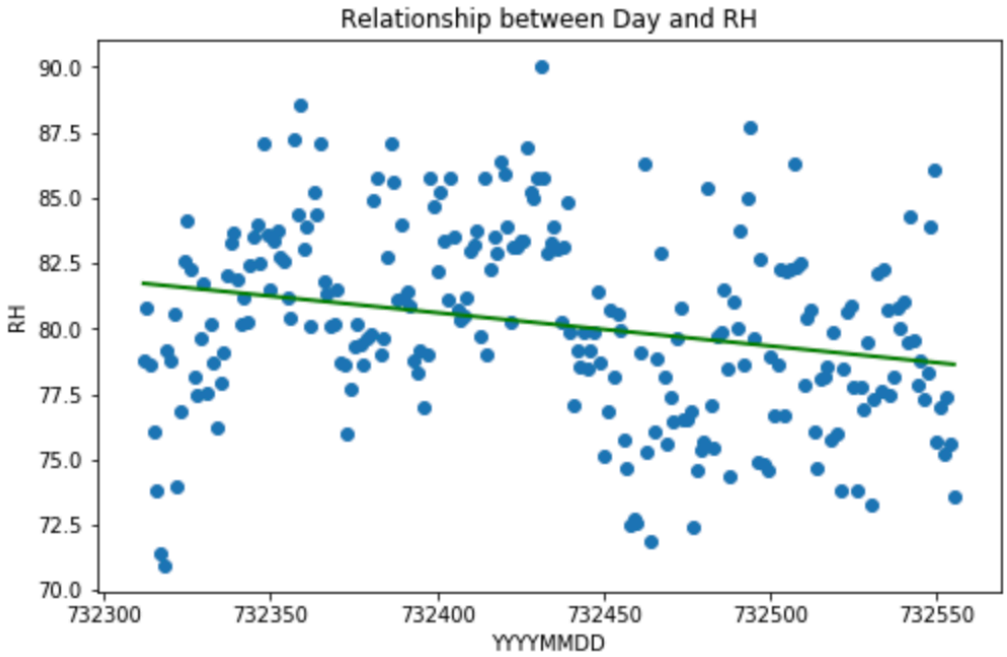
plt.scatter(groupbyDay['YYYYMMDD'], groupbyDay['RH'])

plt.xlabel('YYYYMMDD')

plt.ylabel('RH')

plt.title('Relationship between Day and RH')

plt.show()



2. Use the linear fit to predict median relative humidity on 2nd September 2006.

   from datetime import datetime as dt
   predict = dt.strptime('2006-09-02', '%Y-%m-%d').date()

predict
predict.toordinal()

The predicted median of RH on 2 September 2006 is obtained by converting the date toordinal, then using the concept of y=mx+c , the prediction value is found.

line = slope*732556 + intercept

```
Out[1093]: 78.62522768670169
```

The predicted median of 2 September 2006 is 78.62522768670169

3. Can you think of a better model that fits all of the aggregated data to capture the trend in relative humidity over time? Describe the model you suggested and explain why it is better suited for this task.

A polynomial regression is a better model that fits all aggregated data because it will curve and try to fit the graph compared to linear regression where it is just a straight line. A polynomial regression provides a better approximation because it fits a wide range of curvature

The code below visualises the polynomial regression

import operator

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error, r2_score

from sklearn.preprocessing import PolynomialFeatures

x = groupbyDay['YYYYMMDD']

y = groupbyDay['RH']

 # transforming the data to include another axis

x = x[:, np.newaxis]

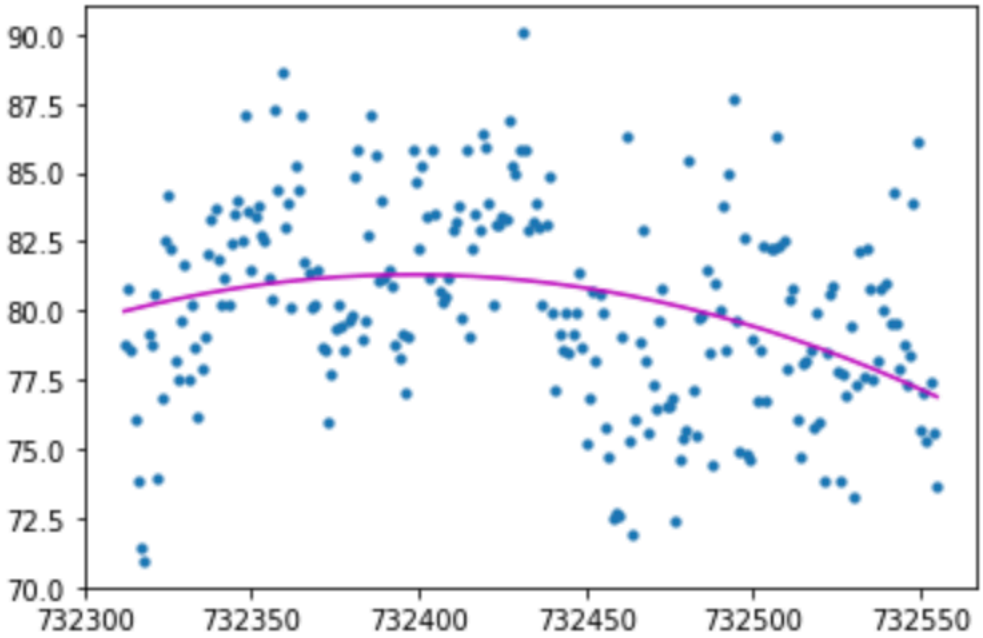
y = y[:, np.newaxis]

```
polynomial_features= PolynomialFeatures(degree=2)
x_poly = polynomial_features.fit_transform(x)

model = LinearRegression()
model.fit(x_poly, y)
y_poly_pred = model.predict(x_poly)

rmse = np.sqrt(mean_squared_error(y,y_poly_pred))
r2 = r2_score(y,y_poly_pred)
print(rmse)
print(r2)

plt.scatter(x, y, s=10)
sort_axis = operator.itemgetter(0)
sorted_zip = sorted(zip(x,y_poly_pred), key=sort_axis)
x, y_poly_pred = zip(*sorted_zip)
plt.plot(x, y_poly_pred, color='m')
plt.show()
```



4. Use your new model to predict median relative humidity on 2nd September 2006 and

   compare with the prediction of your previous linear fit.

The same code used to predict median on 2 September 2006 for linear regression is used here.

from datetime import datetime as dt

predict = dt.strptime('2006-09-02', '%Y-%m-%d').date()

predict

predict.toordinal()

The line of code below predicts the median for polynomial regression

array = model.predict(polynomial_features.fit_transform([[732556]]))

print(array[0][0])

```
76.82593207061291
```

The predicted median of 2 September 2006 is 78.82593207061291

A10. Filling in missing values

Rather than removing the missing values in task A4, fill in the missing values (for column, RH only) using an appropriate regression model.

The csv file is read again and is named tao2

condition1 = ((tao2['RH'] == -9.990000) | (tao2['RH'] == -99.900000))

The code above contains the condition that outputs 'RH' that only contains (-9.990000) and (-99.900000)

According to the data frame, only 10 July 2006 contains -9.990000 and -99.900000

Hence, we use the toordinal and prediction function again to convert 10 July 2006 to obtain the median of 10 July 2006.

```
from datetime import datetime as dt
predict = dt.strptime('2006-07-10', '%Y-%m-%d').date()
predict
predict.toordinal()
array = model.predict(polynomial_features.fit_transform([[732502]]))
```

```
Out[1162]: 732502

Out[1164]: array([[79.3609094]])
```

Then using the 'loc' function, we convert all the data that contains -9.990000 and (-99.900000) to the median, which is 79.3609094

tao2.loc[condition1, 'RH'] = array[0][0]

tao2 is a now a data frame where all the missing values (-9.990000 and -99.900000) are replaced with the median value.

Task B: K-means Clustering on Other Data

We have demonstrated k-means clustering algorithm in week 7. Your task in this part is to find an interesting dataset and apply k-means clustering on a dataset using Python. Kaggle, a private company which runs data science competitions, provides a list of their publicly available datasets:

Dataset I choose:

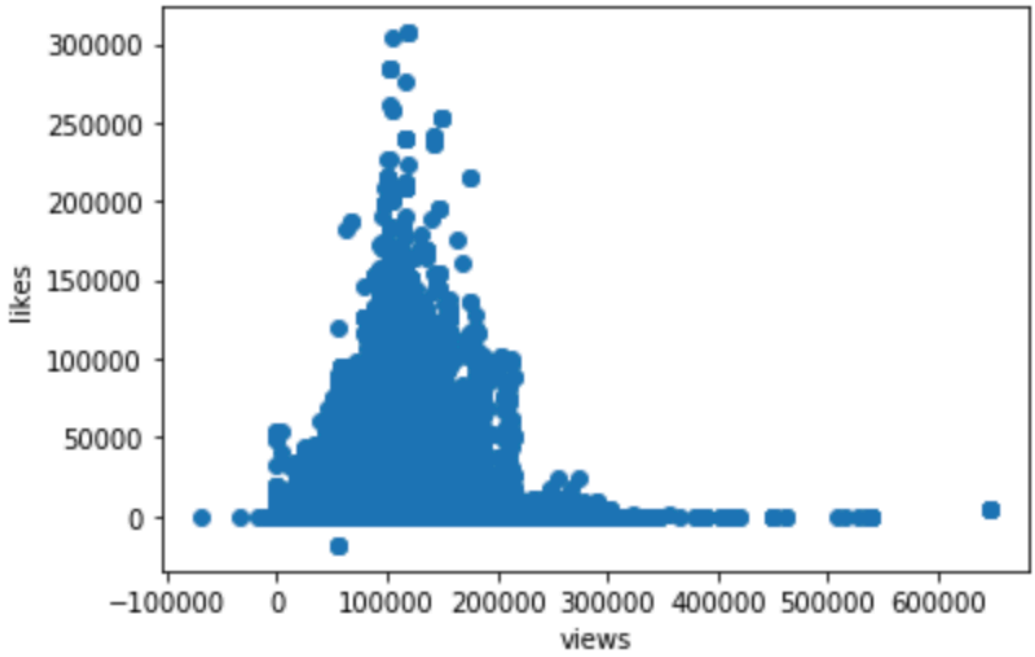
https://www.kaggle.com/datasnaek/youtube-new

1. choose two numerical features in your dataset and apply k-means clustering on your data into k clusters in Python, where k>=2.

The two data I choose are views and likes from the dataset.

A scatterplot was first plotted

```
%matplotlib inline
plt.scatter(x=video['views'],y=video['likes'])
plt.xlabel('views')
plt.ylabel('likes')
```



Then the kmeans was calculated, including with the cluster center and labels.

kmeans = KMeans(n_clusters=2).fit(video[['views','likes']])

kmeans.cluster_centers_

```
Out[1253]: array([[ 1799645.25485343,    60207.31113575],
                  [58676122.01485164,  1485251.9009901 ]])
```

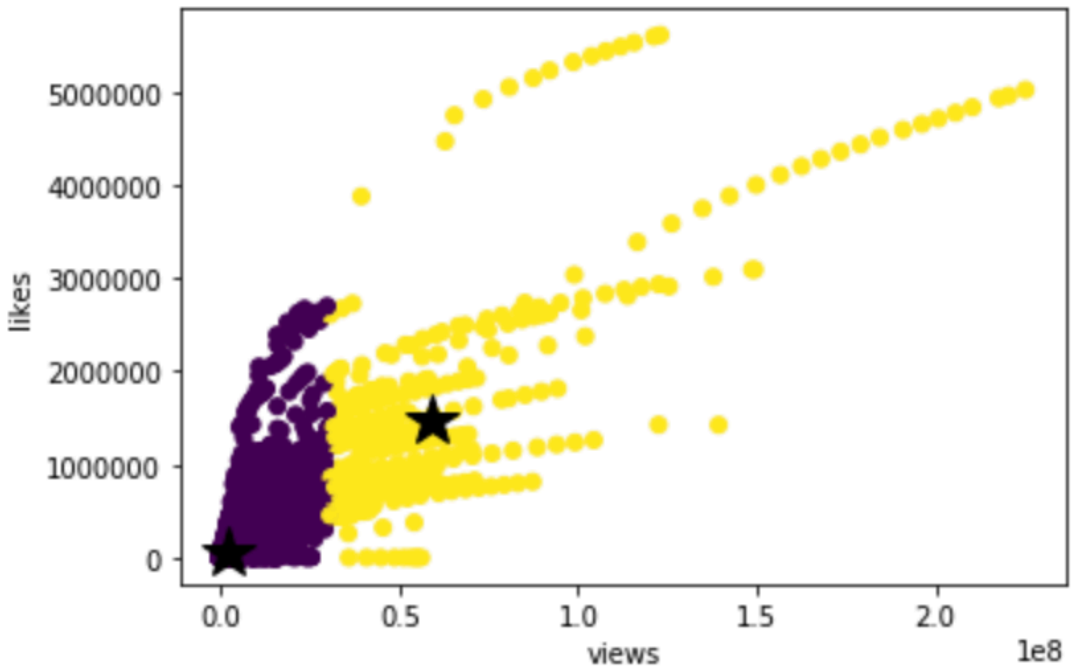
kmeans.labels_

```
Out[1241]: array([0, 1, 0, ..., 0, 0, 0], dtype=int32)
```

2. visualise the data as well as the results of the k-means clustering. Ideally each cluster is shown in a different colour.

```
# Visualise the output labels
plt.scatter(x=video['views'],y=video['likes'], c=kmeans.labels_)

# Visualise the cluster centers (black stars)
plt.plot(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],'k*',markersize=20)
plt.xlabel('views')
plt.ylabel('likes')
plt.show()
```



3. describe your findings about the identified clusters.

The data chosen are represented by clusters using K-means clustering. The purple clusters shows us a low likes and low views where the yellow clusters shows us high likes and high views. Through my the visualised data, it is shown that the more number of views that a video have, the more likes it has.

4. investigate/suggest some appropriate measures to evaluate the quality of your clusters. You can search online for this task.

Silhouette Analysis is another measures to evaluate the quality of clusters. It is used to determine the degree of separation between clusters

Obtained from: https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a