FIT 1043 Introduction to Data Science
Assignment 2
Vanessa Joyce Tan, 30556864

Task A: Data Wrangling and Analysis on TAO dataset

A1. Dataset size

This dataset contains 35136 rows and 8 columns.

```
tao = pd.read_csv("TAO_2006.csv")
tao.shape
```

```
Out[2]: (35136, 8)
```

A2. Min/Max values in each column

Precipitation (PREC): Maximum is 75.77 and minimum is -9.99
Air Temperature (AIRT): Maximum is 31.57 and minimum is -99.9
Sea Surface Temperature (SST): Maximum is 31.346 and minimum is -99.9
Relative Humidity (RH): Maximum is 98.1 and minimum is -99.9

```
tao['PREC'].max()
```

```
Out[9]: 75.77
```

```
tao['PREC'].min()
```

```
Out[10]: -9.99
```

```
tao['AIRT'].max()
```

```
Out[11]: 31.57
```

```
tao['AIRT'].min()
```

```
Out[12]: -99.9
```

```
tao['SST'].max()
```

```
Out[13]: 31.346
```

```
tao['SST'].min()
```

```
Out[14]: -99.9
```

```
tao['RH'].max()
```

```
Out[15]: 98.1
```

```
tao['RH'].min()
```

```
Out[16]: -99.9
```

A3. Number of records in each month

Month	Number of records
January	4464
February	4032
March	4464
April	4320
May	4464
June	4320
July	4464
August	4464
September	144

```
Out[25]: Month
         1    4464
         2    4032
         3    4464
         4    4320
         5    4464
         6    4320
         7    4464
         8    4464
         9     144
         dtype: int64
```

February and September have the two lowest number of records. This is because February is a short month with only 28 days while other months have 30 or 31 days, so February will have slightly less records than other months. As for September, the dataset does not have data for the whole of September. In fact, the dataset only contains records from the first day of September.

```
tao['YYYYMMDD'] = pd.to_datetime(tao['YYYYMMDD'], format='%Y%m%d')
tao['Month']=tao['YYYYMMDD'].dt.month
tao.groupby('Month').size()
```

A4. Missing values

1. **How many rows contain missing values?**

   401 rows contain missing values.

```
miss1 = (tao['PREC'] == -9.990000)
miss2 = (tao['AIRT'] == -99.900000)
miss3 = (tao['SST'] == -99.900000)
miss4 = (tao['RH'] == -99.900000)
tao_missing = tao[miss1 | miss2 | miss3 | miss4]
tao_missing.shape
```

```
Out[41]: (401, 9)
```

2. **Months with no missing values in them.**

   February, May, September.

```
no_miss_month = tao[~tao['Month'].isin(tao_missing['Month'])]
no_miss_month.agg({'Month':'unique'})
```

```
Out[65]:
```

	Month
0	2
1	5
2	9

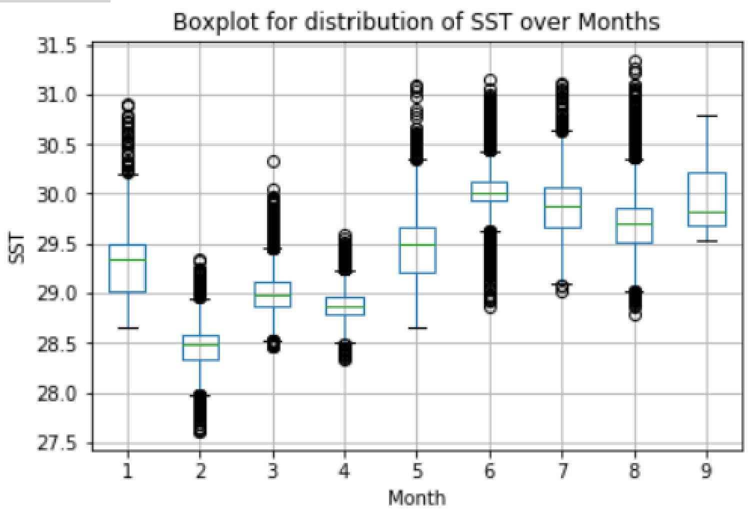
3. **Removal of records.**

```
tao2 = tao[~tao['Timestamp'].isin(tao_missing['Timestamp'])]
tao2.reset_index(drop=True, inplace=True)
```

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

1. **Visualize distribution of SST.**

```
tao2.boxplot(column='SST', by='Month')
plt.title('Boxplot for distribution of SST over Months')
plt.suptitle('')
plt.ylabel('SST')
plt.show()
```



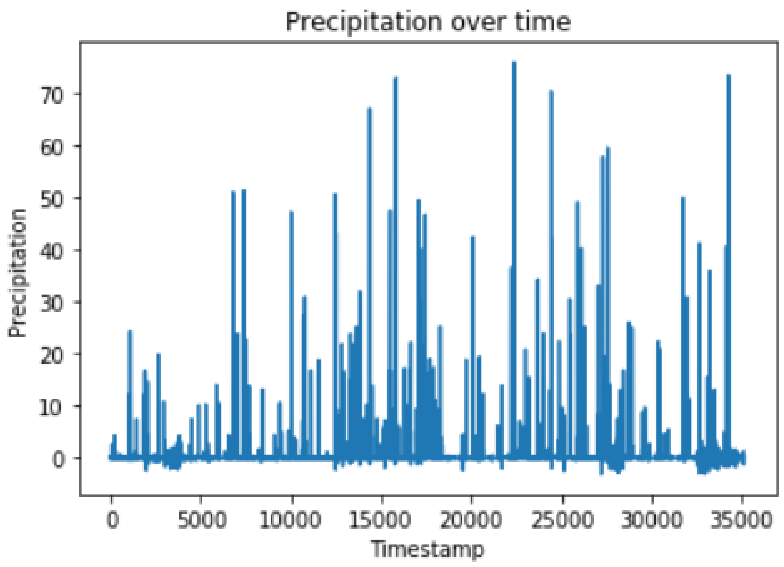
2. **Describe trend of median SST.**

   The median SST dropped a bit from January to February. But from February onwards, there are a few times the median dropped a little bit but generally, there is a positive trend. The sea surface temperature is increasing with each month after February.

3. **Which month has the highest and lowest median SST?**
June has the highest median SST and February has the lowest median SST
```
SST_df = tao2.groupby('Month').agg({'SST':{'Median SST':'median'}})
SST_df = SST_df.reset_index()
SST_df.columns = SST_df.columns.droplevel(0)
SST_df.rename(columns = {'':'Month'}, inplace=True)
print(SST_df)
print(SST_df[SST_df['Median SST'] == SST_df['Median SST'].max()])
print(SST_df[SST_df['Median SST'] == SST_df['Median SST'].min()])
```
```
   Month  Median SST
0      1      29.334
1      2      28.484
2      3      28.979
3      4      28.862
4      5      29.497
5      6      30.013
6      7      29.877
7      8      29.698
8      9      29.822
   Month  Median SST
5      6      30.013
   Month  Median SST
1      2      28.484
```

A6. Exploring precipitation measurements (PREC)

1.
```
plt.plot(tao2['Timestamp'], tao2['PREC'])
plt.title('Precipitation over time')
plt.xlabel('Timestamp')
plt.ylabel('Precipitation')
plt.show()
```



2. **Counter-intuitive values**
Counter-intuitive values are those that are negative because there is no such thing as negative rain rates.
```
tao3 = tao2.copy()
tao3['PREC'][tao3['PREC']<0] = 0
```

A7. Relationship between variables

1. **Pairwise correlation of precipitation, air temperature and sea surface temperature.**
Precipitation (PREC) and Sea Surface Temperature (SST) have the least linear association because the correlation coefficient between both of these two columns is the nearest to 0.

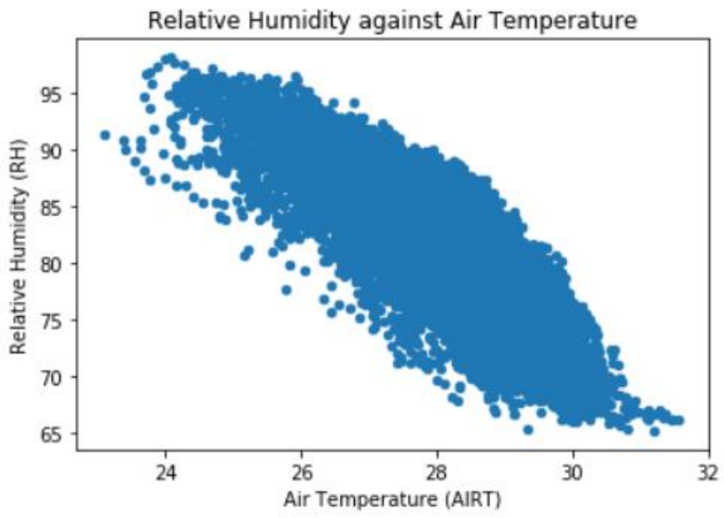
```
columns = ['PREC', 'AIRT', 'SST']
tao3[columns].corr()
```

```
Out[97]:
```

	PREC	AIRT	SST
PREC	1.000000	-0.303513	-0.034852
AIRT	-0.303513	1.000000	0.366119
SST	-0.034852	0.366119	1.000000

2.

```
tao3.plot.scatter(x='AIRT', y='RH')
plt.title('Relative Humidity against Air Temperature')
plt.xlabel('Air Temperature (AIRT)')
plt.ylabel('Relative Humidity (RH)')
plt.show()
```



The relationship between relative humidity and air temperature is somewhat linear and inverse. When one is high, the other is low. When air temperature is high, the relative humidity is low and vice versa.

A8. Predicting quality of measurements (Q)

1. **Divide dataset and train decision tree model.**

```
x = tao3.iloc[:, [3, 4, 5, 6]].values
y = tao3.iloc[:, [7]].values

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, random_state = 0)

from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier.fit(x_train, y_train)
```

2. **Compute confusion matrix and accuracy.**

```
y_pred = classifier.predict(x_test)

from sklearn.metrics import confusion_matrix
matrix = confusion_matrix(y_test, y_pred)
matrix
Out[165]: array([[8670,    8],
                 [   4,    2]], dtype=int64)

from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
Out[178]: 0.9986181483187471
```

3. **Is this a good model? What other metric(s) should be considered and why?**
Yes, this model is quite good in terms of accuracy because its predictions are almost all correct. The model has an accuracy of 0.9986 and the closer the accuracy is to 1, the better the model is. But just because it's accurate, doesn't mean that overall it is a good model. Besides accuracy, we should also consider calculating precision because it shows what proportion of positive predictions is actually positive. A model can be accurate and have low precision at the same time. If the results contain a lot of false positives, the precision is low. Recall/sensitivity should also be calculated to see when the actual result was positive, how often was the prediction correct. In order to minimise false negative results, the recall should be close to 1. Specificity is another metric that can be calculated. It shows what proportion of actual negative results were predicted as negative. When most of these (accuracy, precision, recall and specificity) are high, then the model can be considered good.

A9. Investigating daily relative humidity (RH)

1. **Linear Regression**

```
tao_day = tao3.groupby('YYYYMMDD').agg({'RH':{'Median RH':'median'}})
tao_day = tao_day.reset_index()
tao_day.columns = tao_day.columns.droplevel(0)
tao_day.rename(columns = {'':'Day'}, inplace=True)
tao_day['Day_plot'] = tao_day['Day'].apply(lambda date: date.toordinal())
tao_day.head()
```

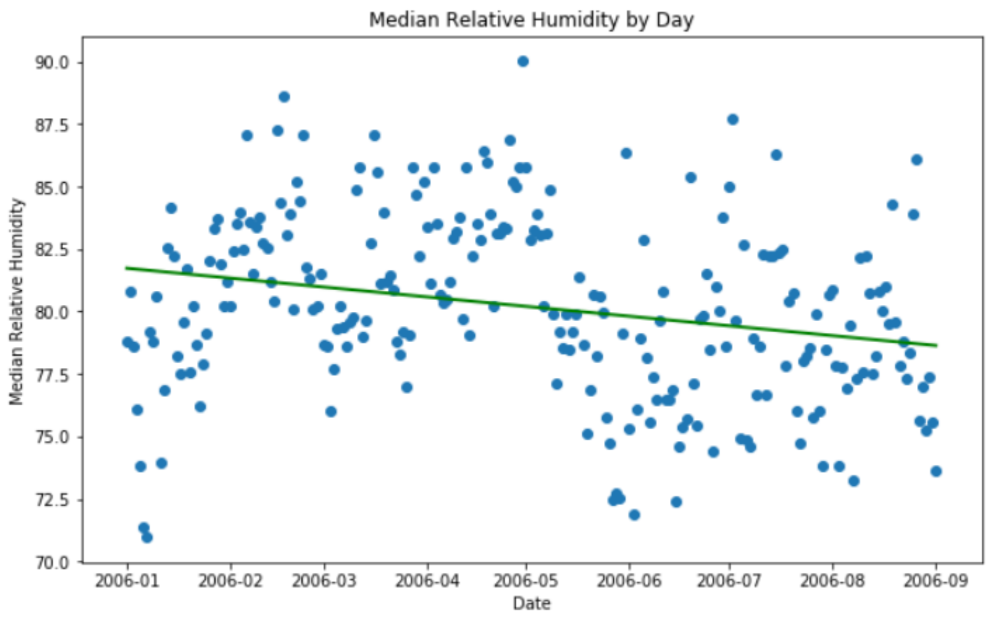
Out[241]:

	Day	Median RH	Day_plot
0	2006-01-01	78.80	732312
1	2006-01-02	80.80	732313
2	2006-01-03	78.60	732314
3	2006-01-04	76.10	732315
4	2006-01-05	73.85	732316

```
from scipy.stats import linregress
slope, intercept, r_value, p_value, std_err = linregress(tao_day['Day_plot'],tao_day['Median RH'])

line = [slope*xi + intercept for xi in tao_day['Day_plot']]

plt.figure(figsize=[10,6])
plt.plot(tao_day['Day_plot'],line,'g-', linewidth=2)
plt.scatter(tao_day['Day'],tao_day['Median RH'])
plt.title('Median Relative Humidity by Day')
plt.xlabel('Date')
plt.ylabel('Median Relative Humidity')
plt.show()
```



2. **Prediction for median relative humidity on 2nd September 2006**
   Basic straight line equation is Y = mX + c. We have found the slope (m) and the intercept (c) in the function linregress earlier. X is taken from the Day_plot column. In the last value, of the dataset tao_day, the last entry is on 1st September 2006 and Day_plot value is 732555 so the Day_plot value for 2nd September 2006 should be 732556. Therefore, the relative humidity for 2nd September 2006 should be given by the formula

   ```
   slope*732556 + intercept
   Out[260]:  78.62522768670169
   ```

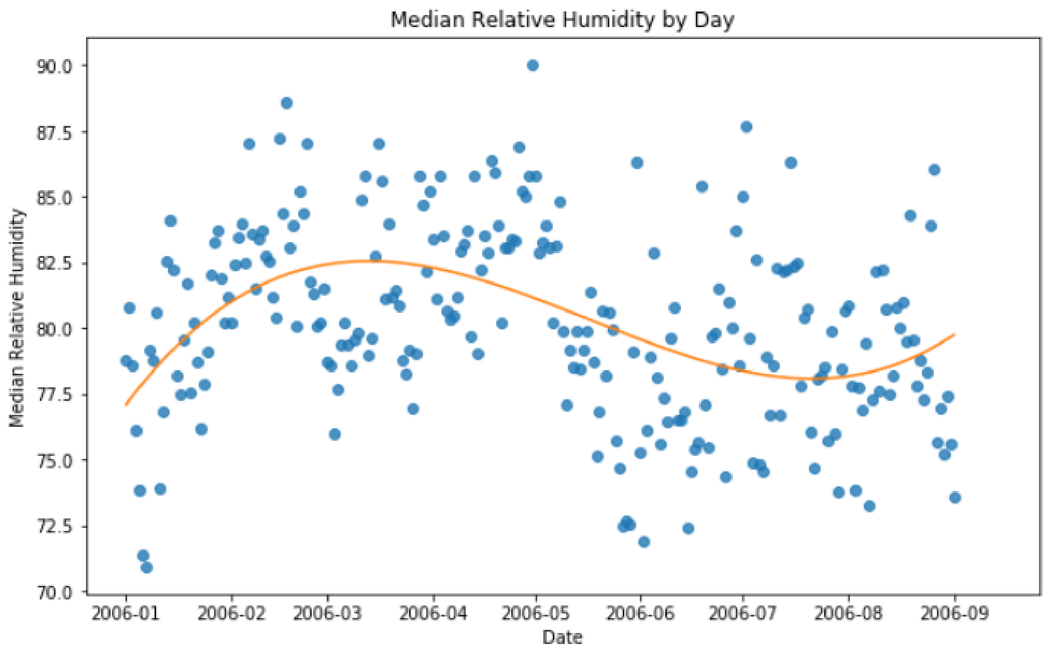
   Hence, relative humidity on 2nd September 2006 is approximately 78.63 based on the linear regression model.

3. **Better model**
   A model that would fit this data better would be a line regression model with a polynomial fit that has a higher degree. This would have a lower bias and lower variance than a linear model. We can see clearly from the plotted graph, that there is a wavy pattern to it and the straight line doesn't reflect the data very well so a linear regression would not be a very good fit.

4. **New model**

   ```
   import numpy as np
   import seaborn as sn

   plt.figure(figsize=[10,6])
   p = np.polyfit(tao_day['Day_plot'], tao_day['Median RH'], deg=3)
   tao_day['fit'] = np.polyval(p, tao_day['Day_plot'])
   sn.regplot(tao_day['Day_plot'], tao_day['Median RH'], fit_reg = False)
   plt.plot(tao_day['Day'], tao_day['fit'], label='fit')
   plt.xlim((732300, 732580))
   plt.title("Median Relative Humidity by Day")
   plt.xlabel("Date")
   plt.ylabel("Median Relative Humidity")
   plt.show()
   ```



We know last entry, 1st September 2006 has Day_plot value 732555 so 2nd September 2006 should have Day_plot value 732556. Apply that to the formula that we used to calculate the polynomial fit:

```
np.polyval(p, 732556)
Out[344]: 79.83973693847656
```

With this model, 2nd September 2006 is predicted to have a relative humidity of 79.84 which is quite similar to the previous prediction with only a slight difference. I assume that coincidentally the two models predict similar results. But there are some days where the model with the polynomial fit is better and much more accurate at predicting than the previous one.

A10. Filling in missing values

Using the same polynomial regression model in A9 part 4, we can predict the average median value of Relative Humidity on that day and replace missing values as follows:

```
tao['Day'] = tao['YYYYMMDD'].dt.day
polynomial = np.polyval(p, tao['Day'])
RH_index = tao.index[tao['RH']==-99.900000].tolist()
for index in RH_index:
   tao['RH'].replace(-99.900000, polynomial[index], inplace=True)
```

Task B: K-means Clustering on Other Data

Dataset chosen: World Happiness Report 2019
https://www.kaggle.com/PromptCloudHQ/world-happiness-report-2019

Reading csv file and checking how the data looks like.

```
from sklearn.cluster import KMeans
happiness = pd.read_csv("world-happiness-report-2019.csv")
happiness.head()
```

Out[158]:

	Country (region)	Ladder	SD of Ladder	Positive affect	Negative affect	Social support	Freedom	Corruption	Generosity	Log of GDP per capita	Healthy life expectancy
0	Finland	1	4	41.0	10.0	2.0	5.0	4.0	47.0	22.0	27.0
1	Denmark	2	13	24.0	26.0	4.0	6.0	3.0	22.0	14.0	23.0
2	Norway	3	8	16.0	29.0	3.0	3.0	8.0	11.0	7.0	12.0
3	Iceland	4	9	3.0	3.0	1.0	7.0	45.0	3.0	15.0	13.0
4	Netherlands	5	1	12.0	25.0	15.0	19.0	12.0	7.0	12.0	18.0

Checking for null values and cleaning data.

```
happiness.isnull().any()
```

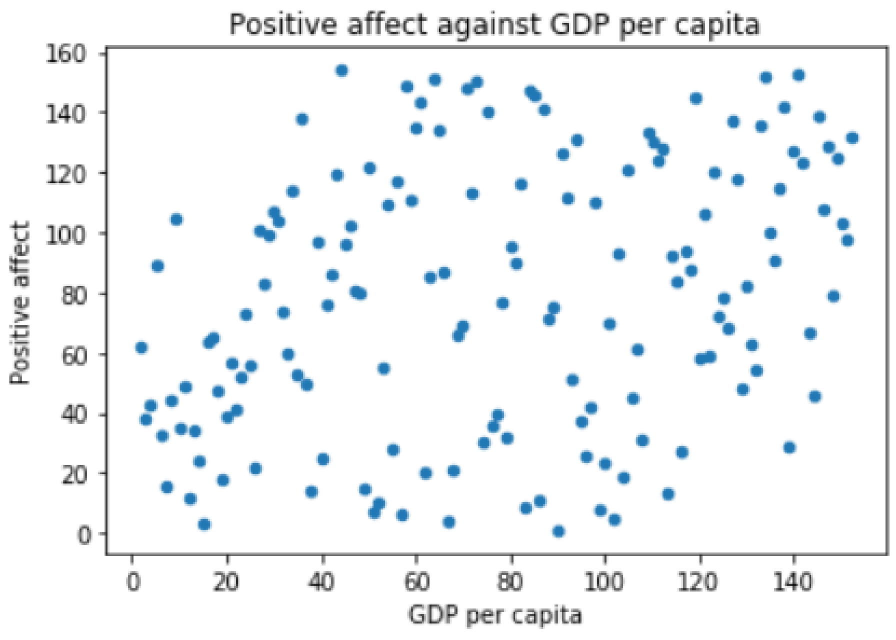
```
Out[159]: Country (region)             False
          Ladder                       False
          SD of Ladder                 False
          Positive affect               True
          Negative affect               True
          Social support                True
          Freedom                       True
          Corruption                    True
          Generosity                    True
          Log of GDP\nper capita        True
          Healthy life\nexpectancy      True
          dtype: bool
```

```
happiness2 = happiness.dropna(how = 'any')
```

Columns chosen: Positive affect (Measure of positive emotion) and Log of GDP per capita (The extent to which GDP contributes to the calculation of the Happiness score)

Seeing how the graph looks like before applying k-means.

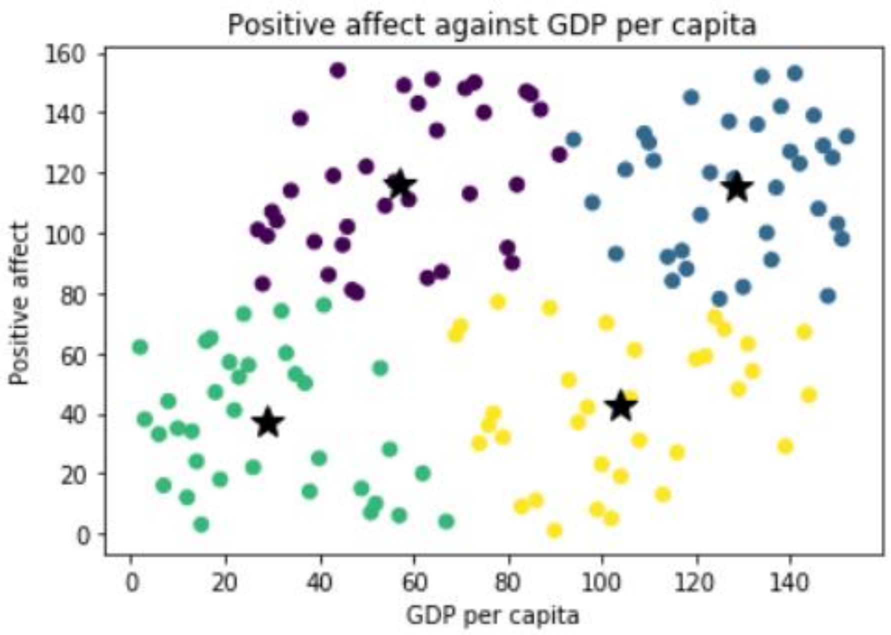
```
plt.scatter(x = happiness2['Log of GDP\nper capita'], y = happiness2['Positive affect'])
plt.title('Positive affect against GDP per capita')
plt.xlabel('GDP per capita')
plt.ylabel('Positive affect')
plt.show()
```



Applying k-means.

```
kmeans = KMeans(n_clusters=4, init='random').fit(happiness2[['Log of GDP\nper capita','Positive affect']])
plt.scatter(x=happiness2['Log of GDP\nper capita'], y=happiness2['Positive affect'], c=kmeans.labels_)
plt.plot(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], 'k*', markersize=15)

plt.title('Positive affect against GDP per capita')
plt.xlabel('GDP per capita')
plt.ylabel('Positive affect')
plt.show()
```



3. The data is grouped into four clusters. The green cluster represents countries whose people are generally not very happy but GDP per capita is not the main reason for their unhappiness. The purple cluster represents countries whose people are generally very happy and GDP per capita somewhat contributes to their happiness. The yellow cluster represents countries whose people are relatively not happy and GDP per capita is a large contribution to their low measure of positive emotion. The blue cluster represents countries whose people are quite happy and the GDP per capita largely contributes to their high measure of positive emotion.

4. One way to evaluate clusters is through their purity. It is used to check how pure the cluster is, or in other words, how much of a single class they contain. The closer the purity of a cluster is to 1, the better the cluster is. In my case, I don’t think my clusters are very pure because the data is scattered everywhere and there are no clear, defined clusters. We could also measure cluster validity by correlation. High correlation means that points that are close to each other belong to the same cluster. Other than that, some internal measures that can be done include cluster cohesion and cluster separation. Cluster cohesion checks the objects in a cluster to see how closely related they are to each other. Meanwhile, cluster separation is how well each cluster is separated from each other, no two objects in different clusters should be similar. The cluster separation and cluster cohesion in my clusters are not very good, because they are quite spread out and some points from the clusters are very close to the other clusters.

REFERENCES

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval.* Cambridge University Press. Retrieved from: https://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-clustering-1.html

Cluster Validation, (n.d.). Retrieved from: http://www.cs.kent.edu/~jin/DM08/ClusterValidation.pdf