TechPoint Xtern Data Science Assessment

October 19, 2020

[1]:
```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import pandas as pd
import numpy as np

df = pd.read_csv(r'C:\Users\zhaoe\Downloads\2020-XTern-DS.csv')

df[:10]
```

[1]:
```
  Restaurant   Latitude  Longitude  \
0    ID_6321  39.262605 -85.837372
1    ID_2882  39.775933 -85.740581
2    ID_1595  39.253436 -85.123779
3    ID_5929  39.029841 -85.332050
4    ID_6123  39.882284 -85.517407
5    ID_5221  39.370441 -85.739516
6    ID_3777  39.821806 -85.005577
7     ID_745  39.280324 -85.144363
8    ID_2970  39.268816 -85.602168
9    ID_3474  39.874521 -85.439963

                                            Cuisines Average_Cost  \
0            Fast Food, Rolls, Burger, Salad, Wraps       $20.00
1                                Ice Cream, Desserts       $10.00
2                       Italian, Street Food, Fast Food       $15.00
3                       Mughlai, North Indian, Chinese       $25.00
4                                    Cafe, Beverages       $20.00
5                  South Indian, North Indian, Chinese       $15.00
6                               Beverages, Fast Food       $15.00
7                               Chinese, Thai, Asian       $65.00
8                                Mithai, Street Food       $10.00
9  Fast Food, North Indian, Rolls, Chinese, Momos…      $20.00

  Minimum_Order Rating Votes Reviews   Cook_Time
0        $50.00    3.5    12       4  30 minutes
1        $50.00    3.5    11       4  30 minutes
```

```
2         $50.00    3.6    99      30   65 minutes
3         $99.00    3.7   176      95   30 minutes
4         $99.00    3.2   521     235   65 minutes
5         $50.00    3.8    46      18   30 minutes
6         $50.00    3.7   108      31   30 minutes
7         $50.00    4.0  1731    1235   45 minutes
8         $50.00    3.9   110      26   30 minutes
9         $50.00    3.9   562     294   65 minutes
```

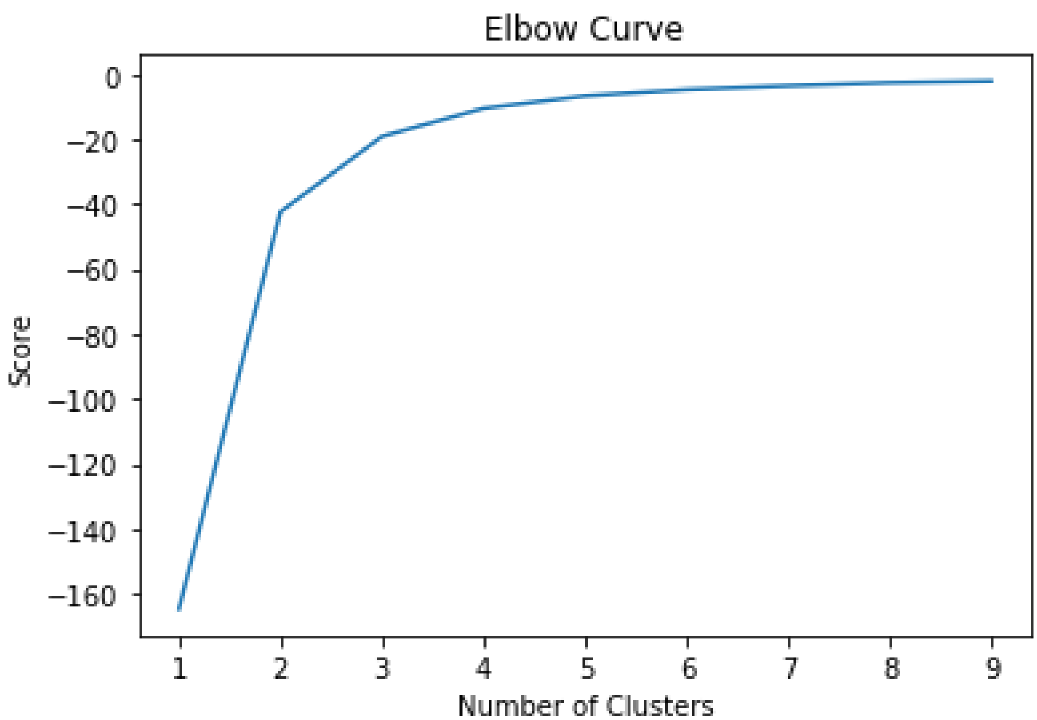
[2]:

```
# Conclusion 1

X = df.loc[:,['Restaurant','Latitude','Longitude']].dropna()

K_clusters = range(1, 10)
kmeans = [KMeans(n_clusters = i) for i in K_clusters]
Y_axis = df[['Latitude']]
X_axis = df[['Longitude']]
score = [kmeans[i].fit(Y_axis).score(Y_axis) for i in range(len(kmeans))]

plt.plot(K_clusters, score)
plt.xlabel('Number of Clusters')
plt.ylabel('Score')
plt.title('Elbow Curve')
plt.show()
```



[ ]:
```
# The graph levls off after 3 clusters so 3 is the best choice here.
```

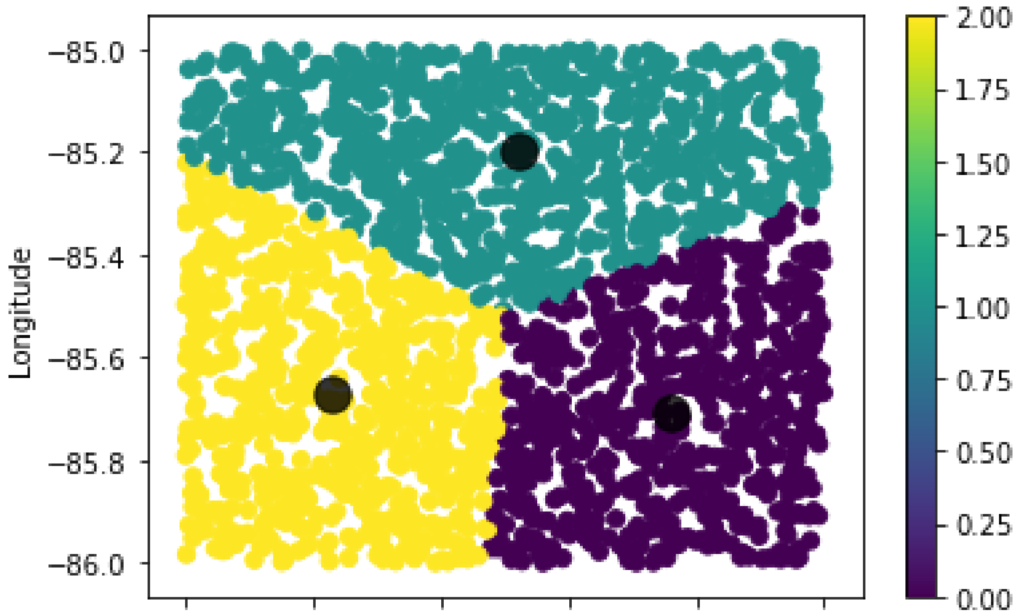
[7]:
```
kmeans = KMeans(n_clusters = 3, init ='k-means++')
kmeans.fit(X[X.columns[1:3]])
X['cluster_label'] = kmeans.fit_predict(X[X.columns[1:3]])
centers = kmeans.cluster_centers_
labels = kmeans.predict(X[X.columns[1:3]])
X[:10]
```

[7]:

	Restaurant	Latitude	Longitude	cluster_label
0	ID_6321	39.262605	-85.837372	2
1	ID_2882	39.775933	-85.740581	0
2	ID_1595	39.253436	-85.123779	1
3	ID_5929	39.029841	-85.332050	2
4	ID_6123	39.882284	-85.517407	0
5	ID_5221	39.370441	-85.739516	2
6	ID_3777	39.821806	-85.005577	1
7	ID_745	39.280324	-85.144363	1
8	ID_2970	39.268816	-85.602168	2
9	ID_3474	39.874521	-85.439963	0

[ ]:
```
# The cluster_label shows which of the 3 clusters the restaurant has been
# grouped into using Kmeans and sci-kit learn. This information could
# help FoodieX optimize pick up zones. Having a FoodieX driver pick up
# food from restaurants that are all in the same cluster would be more
# efficient than having that driver go all over the city to restaurants
# in different clusters. Of course, this does not account for the drop
# off locations but at least in terms of pick up locations, this would
# be a reasonable conclusion to draw from these clusters.
```

[9]:
```
X.plot.scatter(x = 'Latitude', y = 'Longitude', c = labels, s = 40, cmap =␣
↪'viridis')
plt.scatter(centers[:, 0], centers[:, 1], c = 'black', s = 180, alpha = 0.8)
plt.show()
```



[ ]:
```
# This is the visualization of the clusters.
```

[10]:
```
# Conclusion 2

df['cook_time_numerical'] = df.Cook_Time.astype(str).str[:2]
df['cook_time_numerical'] = df['cook_time_numerical'].astype(int)

df['ratings_wholenum'] = df.Rating.astype(str).str[:1]
df['ratings_wholenum'] = pd.to_numeric(df['ratings_wholenum'], errors =␣
 ↪'coerce')

df['cook_time_numerical'].corr(df['ratings_wholenum'])
```

[10]: 0.12418929158599083

[ ]:
```
# The correlation between rating and cooking time is very low at 0.124. There
# is basically no correlation between rating and cooking time. This makes
# sense since there are more important factors when rating a restaurant such
# as the quality of the food or the price. Drawing the conclusion that
# cooking time has very little correlation with ratings helps identify the
# most popular restaurants.
```

[11]:
```
# Conclusion 3

df['avg_cost'] = df['Average_Cost'].str.replace('$', '')
df['avg_cost'] = pd.to_numeric(df['avg_cost'], errors = 'coerce')
```

```python
df['min_order'] = df['Minimum_Order'].str.replace('$', '')

print(df['avg_cost'].describe())
print()
print(df['min_order'].describe())
```

```
count    2017.000000
mean       20.034705
std        12.676288
min         5.000000
25%        10.000000
50%        20.000000
75%        20.000000
max       150.000000
Name: avg_cost, dtype: float64

count      2019
unique        7
top       50.00
freq       1856
Name: min_order, dtype: object
```

[ ]:
```python
# It seems strange that the total average of the average cost of each
# restaurants is $20.03, yet the minimum delivery for the vast
# majority of the restaurants is $50. This means that someone
# ordering from FoodieX would likey have to order for a group of at
# least 3 people. This sort of disincentivizes a single person from
# using FoodieX unless they order from a more expensive restaurant,
# or order more portions than they can eat in one meal. This could
# perhaps point to FoodieX lowering the minimum delivery amount or
# the restaurants lowering it (the prompt doesn't specify which one
# handles this).
```

[13]:
```python
# Conclusion 4

df['Rating'] = pd.to_numeric(df['Rating'], errors = 'coerce')
print(df['Rating'].describe())
filtered = df[(df['Rating'] >= 3.9) & (df['Cuisines'].str.contains('Salad'))]
filtered[:10]
```

```
count    1666.000000
mean        3.609304
std         0.422452
min         2.400000
25%         3.300000
50%         3.600000
75%         3.900000
```

```
max         4.800000
Name: Rating, dtype: float64
```

[13]:

```
    Restaurant   Latitude  Longitude  \
35     ID_1160  39.246289 -85.152915
62     ID_6967  39.971490 -85.104787
197    ID_2041  39.169006 -85.230237
267    ID_6013  39.291499 -85.576338
385    ID_4973  39.734465 -85.641486
504    ID_7302  39.610318 -85.830259
733    ID_4360  39.032331 -85.744339
759    ID_6915  39.303801 -85.960137
781    ID_6952  39.022785 -85.893902
822    ID_8117  39.391951 -85.076733

                                              Cuisines Average_Cost  \
35   Asian, Burmese, Bubble Tea, Desserts, Salad, T…       $60.00
62   Cafe, European, Continental, Sandwich, Salad, …       $60.00
197  Italian, Pizza, Salad, Healthy Food, Mexican, …       $65.00
267     Cafe, Spanish, Italian, Mexican, Salad, Juices       $25.00
385  Salad, European, Steak, Healthy Food, Beverage…       $55.00
504  Finger Food, Salad, Continental, Italian, Sand…       $60.00
733         Italian, Mexican, Pizza, Salad, Beverages       $60.00
759                European, Italian, American, Salad       $65.00
781  Pizza, Salad, Burger, Sandwich, Lebanese, Italian       $15.00
822                                     Italian, Salad         1,00

    Minimum_Order  Rating Votes Reviews   Cook_Time  cook_time_numerical  \
35         $50.00     4.7   914     499  45 minutes                   45
62         $50.00     4.6   391     174  30 minutes                   30
197        $50.00     4.4  3248    1603  45 minutes                   45
267        $50.00     4.1  1307     794  45 minutes                   45
385        $50.00     4.2  1319     659  45 minutes                   45
504        $50.00     4.2  1392     739  45 minutes                   45
733        $50.00     4.2  1114     453  45 minutes                   45
759        $50.00     4.6  2858    1673  30 minutes                   30
781        $50.00     4.4   315     248  30 minutes                   30
822        $50.00     4.3  1276     671  45 minutes                   45

     ratings_wholenum  avg_cost min_order
35                4.0      60.0     50.00
62                4.0      60.0     50.00
197               4.0      65.0     50.00
267               4.0      25.0     50.00
385               4.0      55.0     50.00
504               4.0      60.0     50.00
733               4.0      60.0     50.00
```

759	4.0	65.0	50.00
781	4.0	15.0	50.00
822	4.0	NaN	50.00

[ ]:
```
# If someone wanted to find a restaurant that for example, serves salads,
# FoodieX could find restaurants with Salad in the Cuisines description
# as well as filter by high ratings. Here specifically, I found the top
# quartile for rating and filtered only that top quartile, so only
# restaurants with 3.9 rating or higher. This, along with the cuisine
# type function could also easily be applied to a search function on a
# FoodieX website/app. This would filter and return the best options
# for a customer.
```