CPS 844 Lab 7

03/30/2022
Section 1
Ryan Soliven
500840952

 # 1
(0 points) Load the dataset weather.csv

Code
```
moviesDataset = pd.read_csv("DataLab7.csv")
```

 # 2
To perform a k-means analysis on the dataset, extract only the numerical attributes: remove the "user" attribute

Code
```
data = moviesDataset.drop(['user'],axis=1)
```

Suppose you want to determine the number of clusters k in the initial data 'data'

 # 3.
(5 points) Create an empty list to store the SSE of each value of k (so that, eventually, we will be able to compute the optimum number of clusters k)

Code
```
SSE = []
```

 # 4.
(30 points) Apply k-means with a varying number of clusters k and compute the corresponding sum of squared errors (SSE)
Hint1: use a loop to try different values of k. Think about the reasonable range of values k can take (for example, 0 is probably not a good idea).
Hint2: research about cluster.KMeans and more specifically 'inertia_'
Hint3: If you get an AttributeError: 'NoneType' object has no attribute 'split', consider downgrading numpy to 1.21.4 this way: pip install --upgrade numpy==1.21.

Code
```
num = range(1,6)
```

```
for k in num:
    kmeans = KMeans(n_clusters=k, init='k-means++', random_state=42)
    y_kmeans = kmeans.fit_predict(data)
    SSE.append(kmeans.inertia_)
```

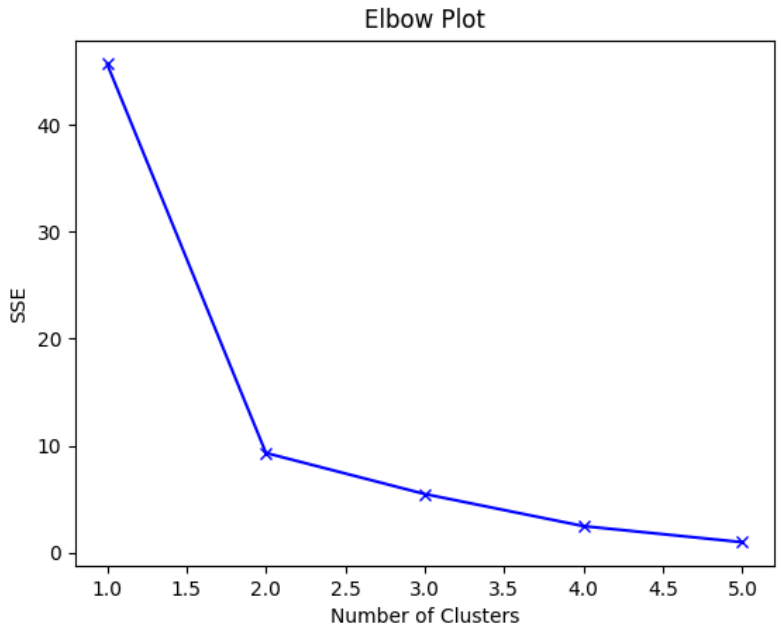
#5

(20 points) Plot to find the SSE vs the Number of Clusters to visually find the "elbow" that estimates the number of clusters. (read online about the "elbow method" for clustering)

Code

```
plt.plot(num, SSE, 'bx-')
plt.title("Elbow Plot")
plt.xlabel("Number of Clusters")
plt.ylabel("SSE")
plt.show()
kl = KneeLocator(num, SSE, curve='convex', direction='decreasing')
```

Results



#6

(10 points) Look at the plot and determine the number of clusters k (read online about the "elbow method" for clustering)

Code

```
k = 2
print("The elbow value for k is:", kl.elbow)
```

#7

(30 points) Using the optimized value for k, apply k-means on the data to partition the data, then store the labels in a variable named 'labels'

Hint1: research about cluster.KMeans and more specifically 'labels_'

Code

```
kmeans = KMeans(n_clusters=2, init='k-means++', random_state=42)
y_kmeans = kmeans.fit(data)
labels = y_kmeans.labels_
```

#8

Display the assignments of each users to a cluster

Code

```
clusters = pd.DataFrame(labels, index=moviesDataset.user, columns=['Cluster ID'])
print(clusters)
```

Results

```
          Cluster ID
user
james              1
margaret           1
brandon            1
linda              0
liam               0
harper             0
```