K Means Clustering with Python

Reference: https://www.kaggle.com/code

Method Used

K Means Clustering is an unsupervised learning algorithm that tries to cluster data based on their similarity. Unsupervised learning means that there is no outcome to be predicted, and the algorithm just tries to find patterns in the data. In k means clustering, we have the specify the number of clusters we want the data to be grouped into. The algorithm randomly assigns each observation to a cluster, and finds the centroid of each cluster. Then, the algorithm iterates through two steps: Reassign data points to the cluster whose centroid is closest. Calculate new centroid of each cluster. These two steps are repeated till the within cluster variation cannot be reduced any further. The within cluster variation is calculated as the sum of the euclidean distance between the data points and their respective cluster centroids.

```
!pip install --upgrade scikit-learn==1.0.2
!pip install --upgrade numpy==1.21.5
```

Import Libraries

```
import seaborn as sns
import matplotlib.pyplot as plt
```

Create some Data

```
from sklearn.datasets import make_blobs
```

```
# Create Data
data = make_blobs(n_samples=200, n_features=2,
            centers=4, cluster_std=1.8,random_state=101)
```

```
# data
data[0][0:5]
```

```
# actual cluster group
data[1][0:5]
```

(Optional) Scaler and Inverse Scaler

```
# from sklearn.preprocessing import StandardScaler
# scaler = StandardScaler()
# scaler.fit(data[0])

# trans_data = scaler.transform(data[0])
# print(trans_data[0:5,])

# org_data = scaler.inverse_transform(trans_data)
# print(org_data[0:5,])
```

Visualize Data

```
plt.scatter(data[0][:,0],data[0][:,1],c=data[1],cmap='rainbow')
```

Creating the Clusters

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

```
kmeans = KMeans(n_clusters=4)
```

```
kmeans.fit(data[0])
```

```
kmeans.cluster_centers_
```

```
kmeans.labels_
```

```
kmeans.predict(data[0])
```

```
f, (ax1, ax2) = plt.subplots(1, 2, sharey=True,figsize=(10,6))
ax1.set_title('K Means')
ax1.scatter(data[0][:,0],data[0][:,1],c=kmeans.labels_,cmap='rainbow')
ax2.set_title("Original")
ax2.scatter(data[0][:,0],data[0][:,1],c=data[1],cmap='rainbow')
```

```
# elbow method
import numpy as np
from scipy.spatial.distance import cdist
import matplotlib.pyplot as plt

K = range(1, 10)
meandistortions = []
for k in K:
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(data[0])
    meandistortions.append(sum(np.min(cdist(data[0], kmeans.cluster_centers_, '

plt.plot(K, meandistortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Average distortion')
plt.title('Selecting k with the Elbow Method')
plt.show()
```

You should note, the colors are meaningless in reference between the two plots.