Name: Vatsal Arya
Roll No.: 12
Lab7 [DSE]

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn
from sklearn.cluster import KMeans
df = pd.read_csv('data_workout.csv')
df.shape
```

```
(6, 5)
```

```
df.head()
```

	date	distance_km	duration_min	delta_last_workout	day_category
0	17-10-2017	4.3	21.58	1	0
1	04-11-2017	1.9	9.25	18	1
2	18-11-2017	1.9	9.00	14	1
3	23-11-2017	1.9	8.93	5	0
4	28-11-2017	2.3	11.94	5	0

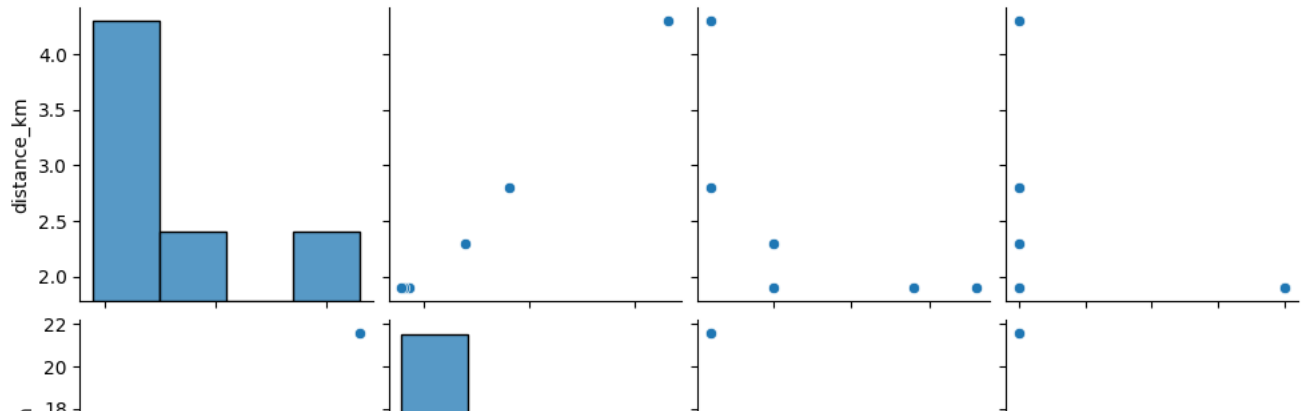
```
df.dtypes
```

```
date                 object
distance_km         float64
duration_min        float64
delta_last_workout    int64
day_category          int64
dtype: object
```

: Pair Plot and Distance versus workout duration, distance versus duration with the number of days, correlation (Scatter plot) to get idea about correlation between different features.

```
seaborn.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7ad690df7370>
```




```
df1=df[['distance_km', 'duration_min', 'delta_last_workout']]
df1
```

	distance_km	duration_min	delta_last_workout
0	4.3	21.58	1
1	1.9	9.25	18
2	1.9	9.00	14
3	1.9	8.93	5
4	2.3	11.94	5
5	2.8	14.05	1

Select K-means clustering for model and get the clusters.

```
kmeans = KMeans(n_clusters=2)
kmeans.fit(df1)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning
  warnings.warn(
```

```
KMeans
KMeans(n_clusters=2)
```

Get Cluster Centers:

```
centroids = kmeans.cluster_centers_
print(centroids)
```

```
[[ 2.825 14.125  3.   ]
 [ 1.9    9.125 16.   ]]
```

Get Cluster Labels:

```
labels = kmeans.labels_
print(labels)
```

```
[0 1 1 0 0 0]
```

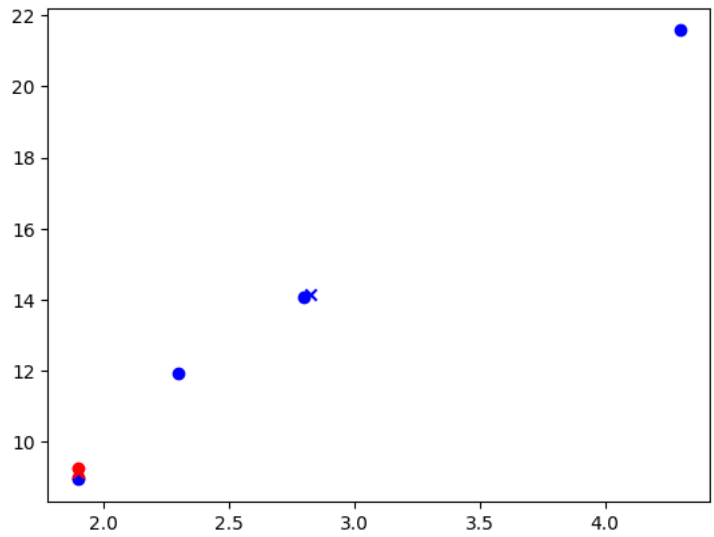
Create array of cluster points:

```
c1 = []
c2 = []
for i,val in enumerate(labels):
    if val==1:
        c1.append(df1.iloc[i])
    else:
        c2.append(df1.iloc[i])
c1 = np.array(c1)
c2 = np.array(c2)
c1[:,0]
```

```
array([1.9, 1.9])
```

plot Cluster using scatter plot:

```
plt.scatter(c1[:,0], c1[:,1], c='red')
plt.scatter(c2[:,0], c2[:,1], c='blue')
plt.scatter(centroids[0,0], centroids[0,1], marker='x', c='blue')
plt.scatter(centroids[1,0], centroids[1,1], marker='x', c='red')
plt.show()
```



Perform prediction of cluster for data points:

```
x=[1.9,8.93,5]
kmeans.predict([x])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but KMeans was fitted
  warnings.warn(
array([0], dtype=int32)
```

```
x=[1.9,9.25,18]
kmeans.predict([x])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but KMeans was fitted
  warnings.warn(
array([1], dtype=int32)
```

```
df2=kmeans.predict(df1)
df2
```

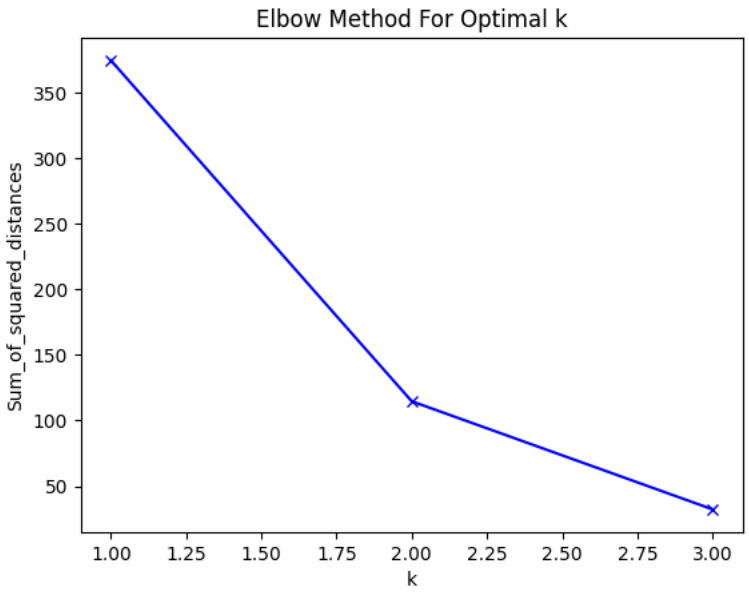
```
array([0, 1, 1, 0, 0, 0], dtype=int32)
```

Evaluate the performance of the model. Find optimized number of clusters For each k value, we will initialise k-means and use the inertia attribute to identify the sum of squared distances of samples to the nearest cluster centre As k increases, the sum of squared distance tends to zero. Imagine we set k to its maximum value n (where n is number of samples) each sample will form its own cluster meaning sum of squared distances equals zero. Below is a plot of sum of squared distances for k in the range specified above. If the plot looks like an arm, then the elbow on the arm is optimal k.

```
Sum_of_squared_distances = []
K = range(1,4)
for k in K:
    km = KMeans(n_clusters=k)
    km = km.fit(df1)
    Sum_of_squared_distances.append(km.inertia_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fro
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fro
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fro
  warnings.warn(
```

```
plt.plot(K, Sum_of_squared_distances, 'bx-')
plt.xlabel('k')
plt.ylabel('Sum_of_squared_distances')
plt.title('Elbow Method For Optimal k')
plt.show()
```



In the plot above the elbow is at k=5 indicating the optimal k for this dataset is 5

```
df1=df[['day_category']]
d=df1.to_numpy()
d
```

```
array([[0],
       [1],
       [1],
       [0],
       [0],
       [0]])
```

Accuracy Accuracy measures how often the model is correct.

```
from sklearn.metrics import accuracy_score
score = accuracy_score(d,df2)
print('Accuracy:{0:f}'.format(score))
```

```
Accuracy:1.000000
```