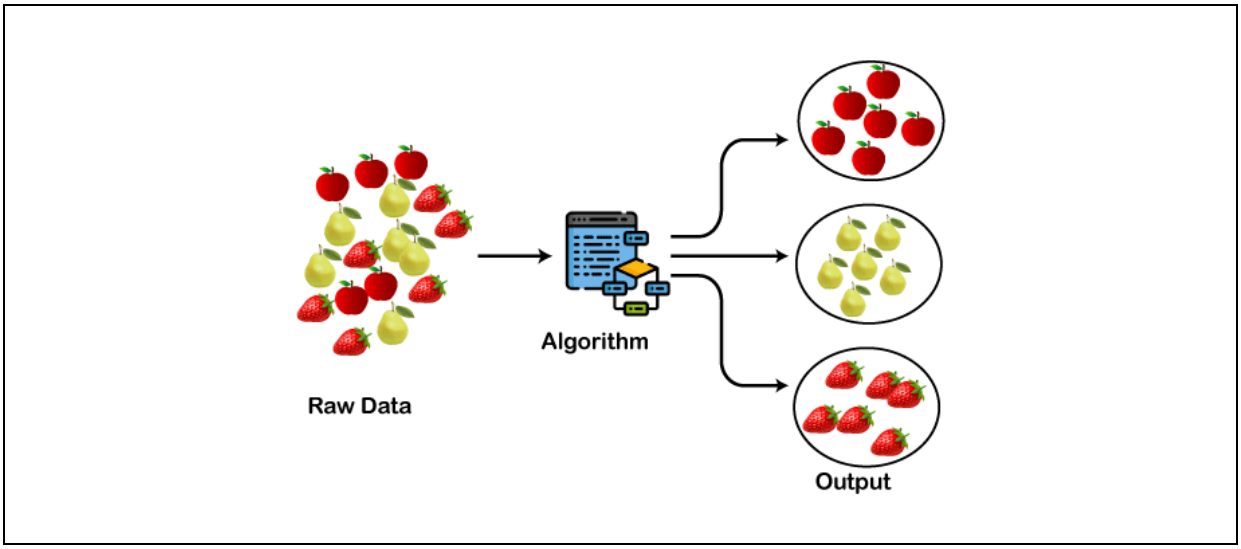
# 28. Data Science – Machine Learning – K – Means Clustering

## Contents

# 28. Data Science – Machine Learning – K – Means Clustering

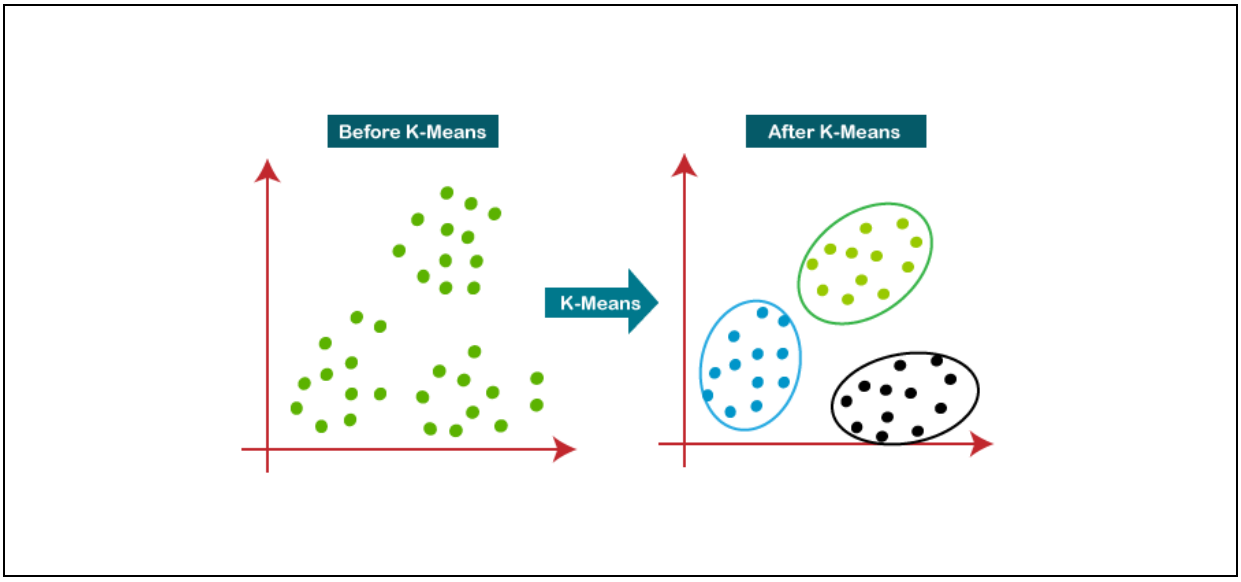
## 1. Clustering

- ✓ Clustering or cluster analysis is a machine learning technique.
- ✓ It groups the unlabelled dataset.
- ✓ It is a way of grouping the data points into different clusters based on similarities.




## 2. K means clustering algorithm

- ✓ K-Means Clustering is an Unsupervised Learning algorithm.
- ✓ This algorithm groups the unlabeled dataset into different clusters based on similar properties.
- ✓ Here K defines the number of pre-defined clusters that need to be created in the process,
  - If K = 2 then there will be two clusters,
  - If K = 3 then there will be three clusters etc.
- ✓ It is a centroid-based algorithm, where each cluster is associated with a centroid.

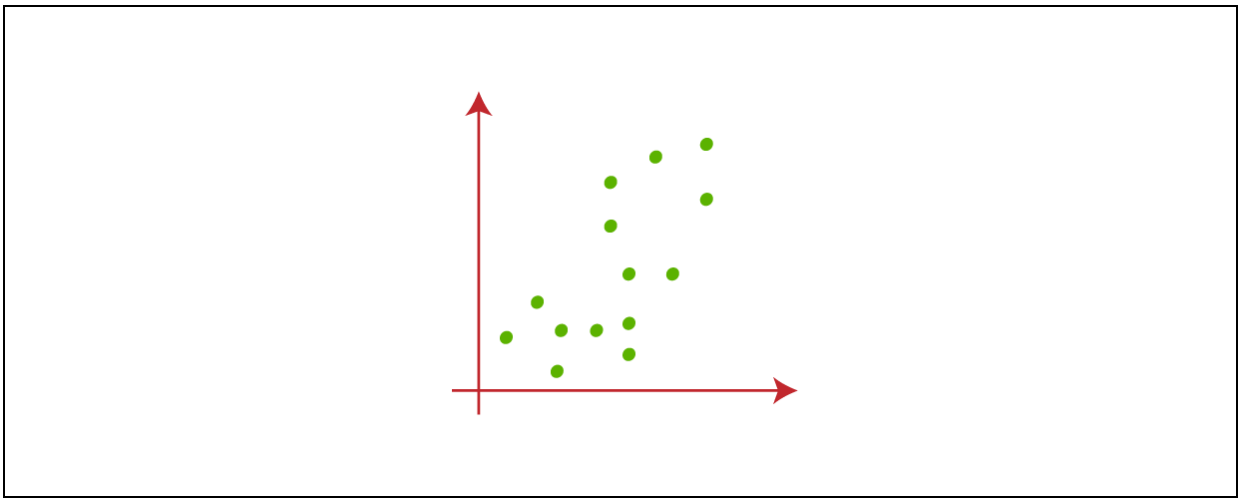



## 3. Steps in K-Means Algorithm

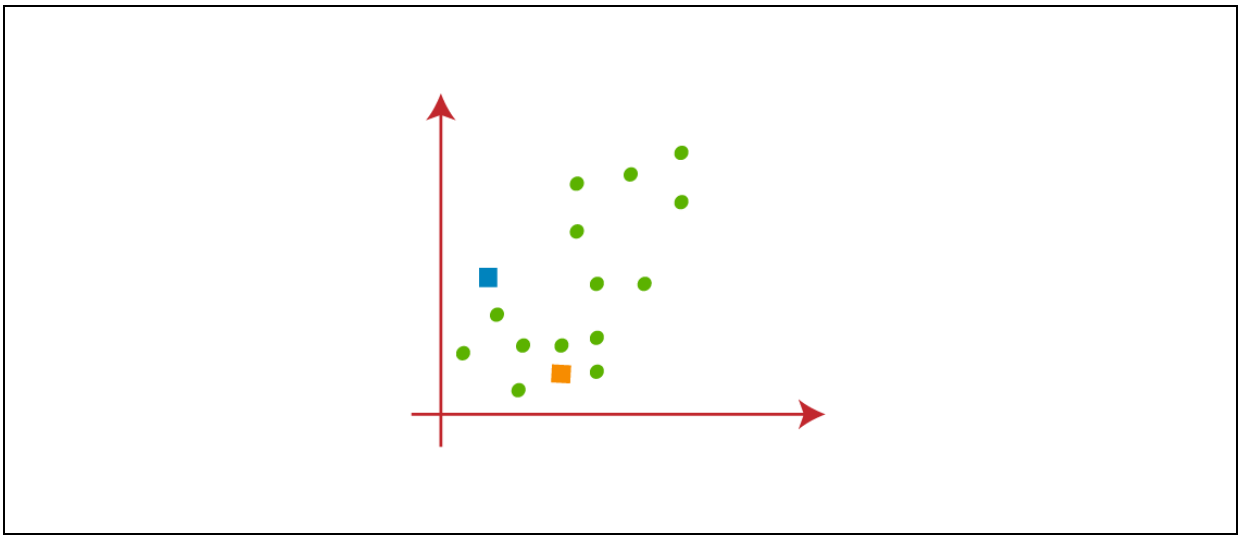
- ✓ Step-1: Select the number K to decide the number of clusters.
- ✓ Step-2: Select random K points or centroids.
- ✓ Step-3: Assign each data point to their closest centroid, which will form K clusters.
- ✓ Step-4: Calculate the variance and place a new centroid of each cluster.
- ✓ Step-5: Repeat the initial 3 steps, which mean reassign each data point to the new closest centroid of each cluster.
- ✓ Step-6: If any reassignment occurs, then go to step-4 else FINISH.
- ✓ Step-7: The model is ready.

## 4. Scenario

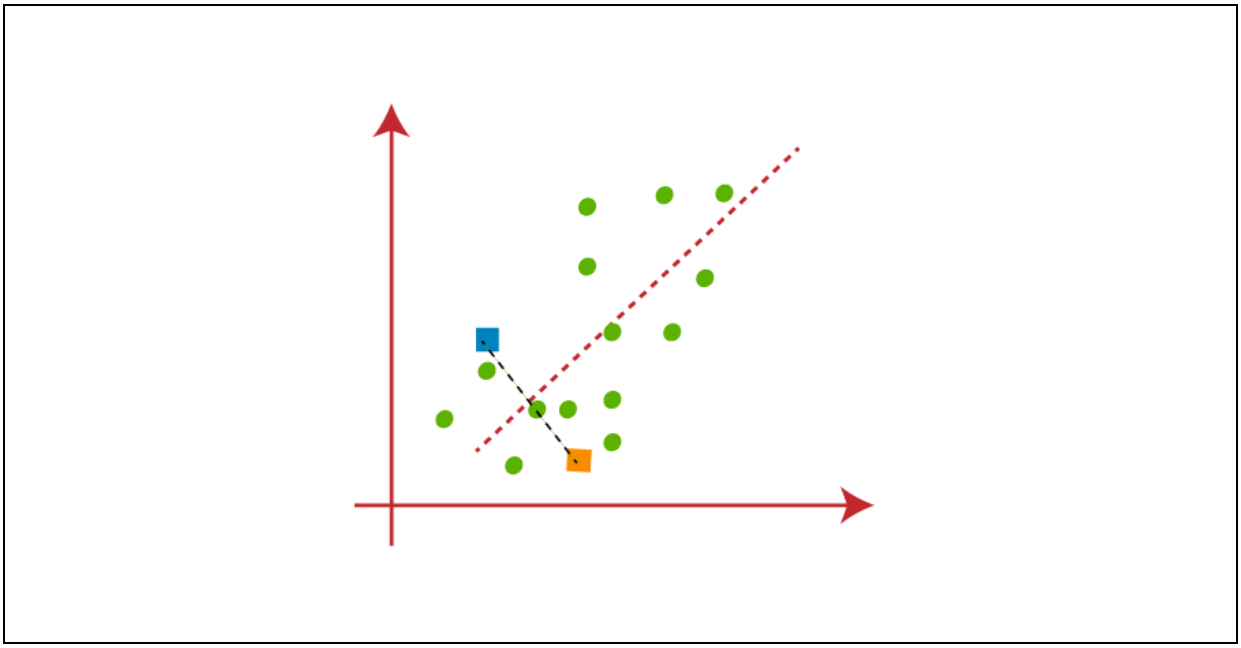
- ✓ Assuming that we have scattered two variables in x and y axis



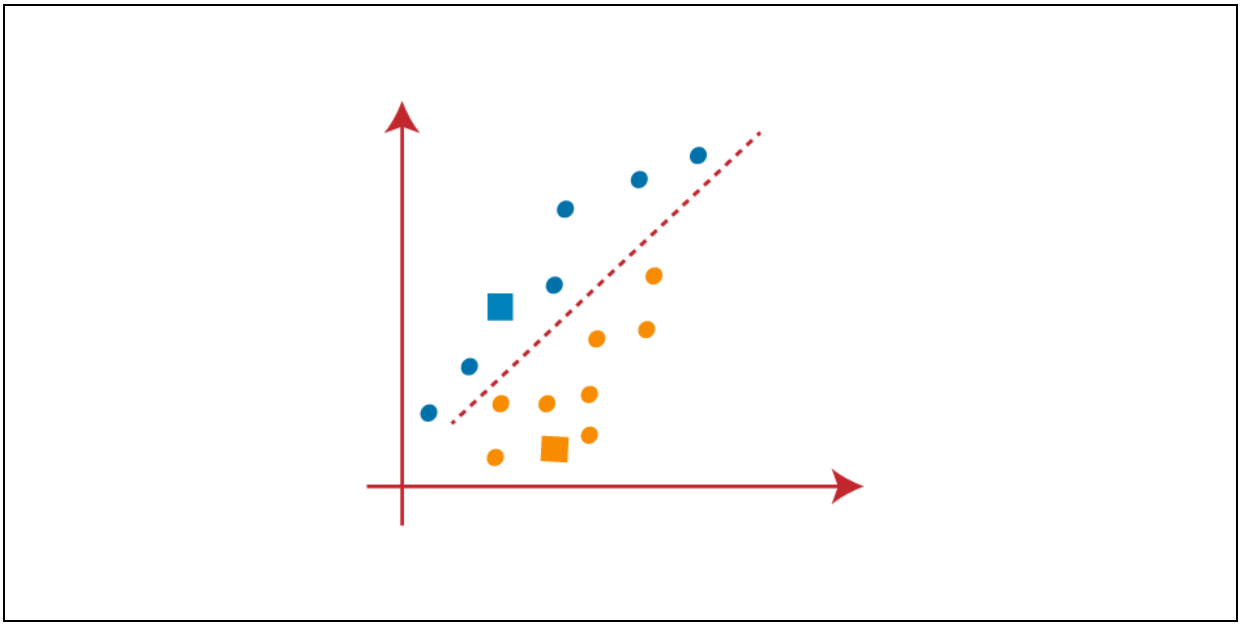
- ✓ Let's take some random k points or centroid to form the cluster.
- ✓ These points can be either the points from the dataset or any other point.
- ✓ So, here we are selecting the below two points as k points, which are not the part of our dataset.



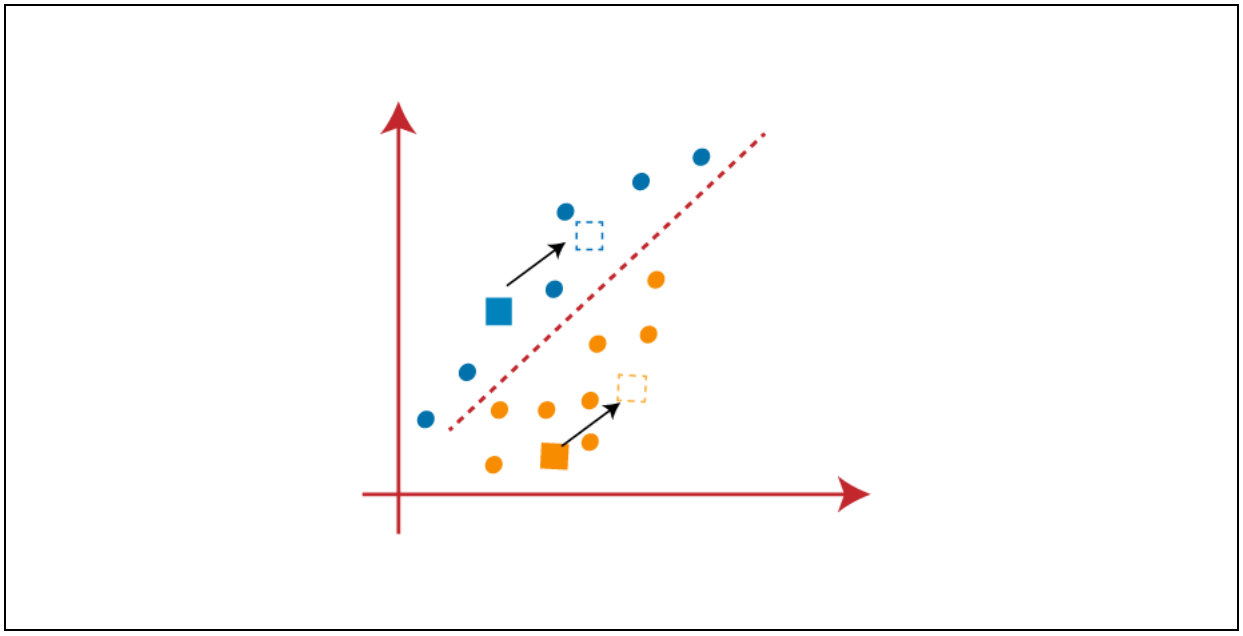
- ✓ Now we will assign each data point of the scatter plot to its closest K-point or centroid.
- ✓ Let's compute the distance between two points.
- ✓ So, we will draw a median between both the centroids.

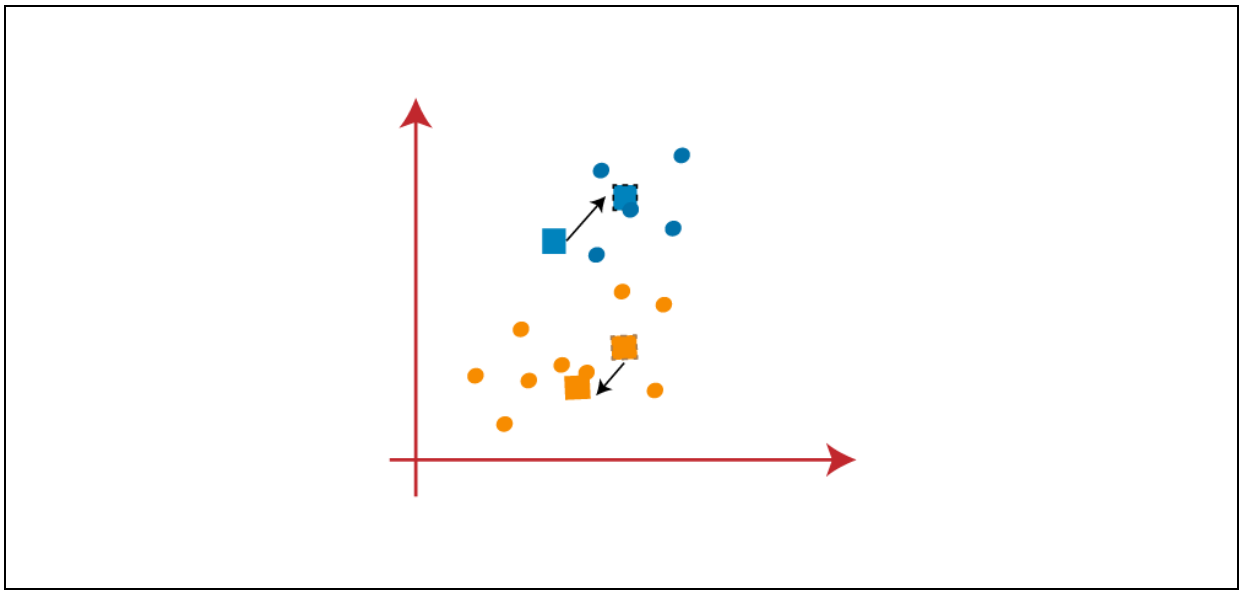


- ✓ From the above image, it is clear that points left side of the line is near to the K1 or blue centroid.
- ✓ Points to the right of the line are close to the yellow centroid.

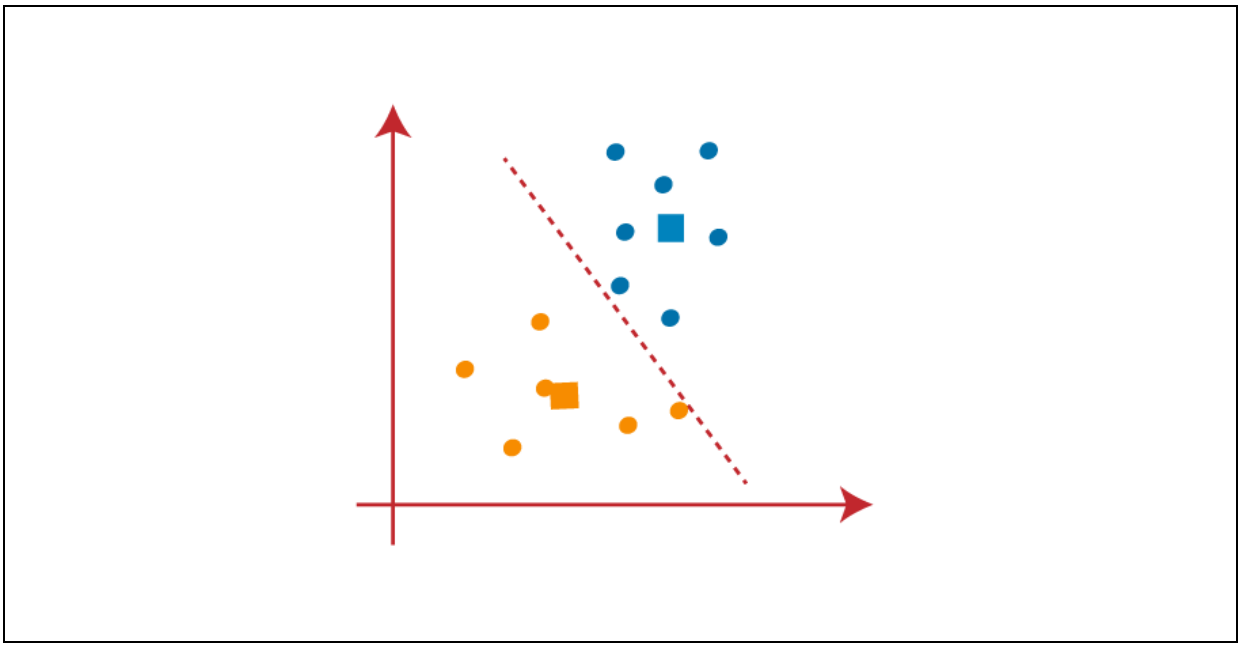


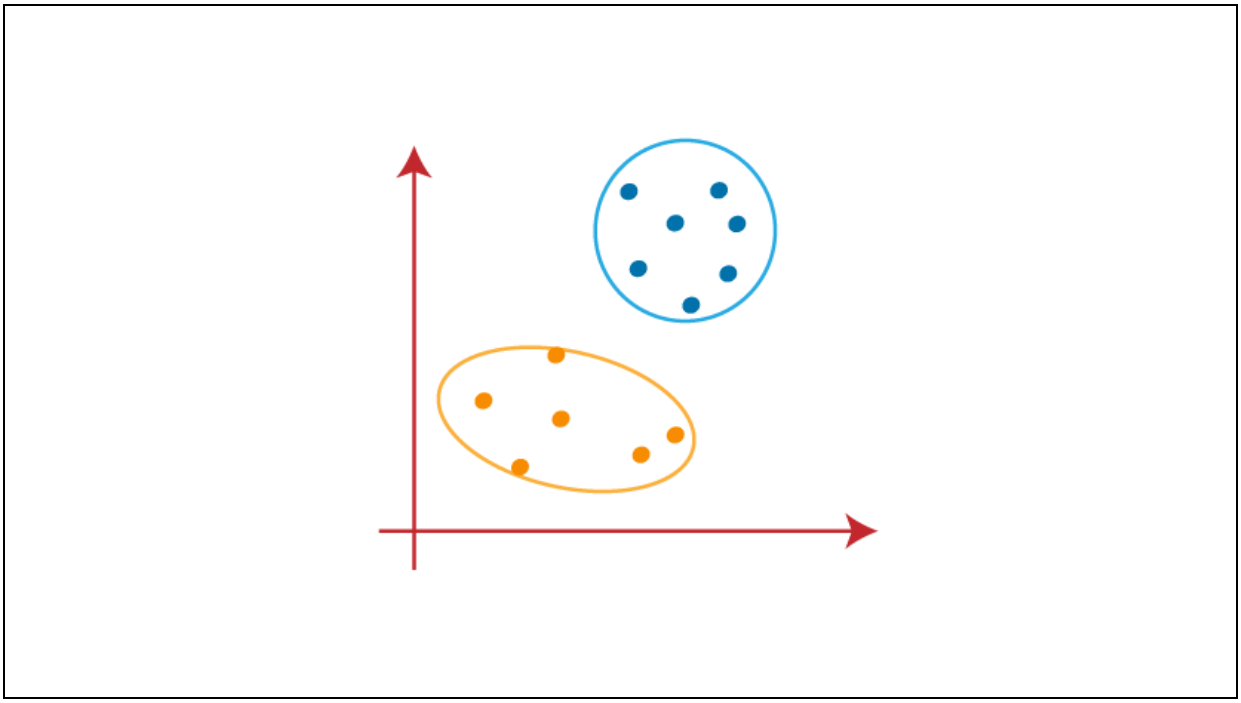
- ✓ As we need to find the closest cluster, so we will repeat the process by choosing a new centroid.
- ✓ To choose the new centroids, we will compute the center of gravity of these centroids, and will find new centroids.





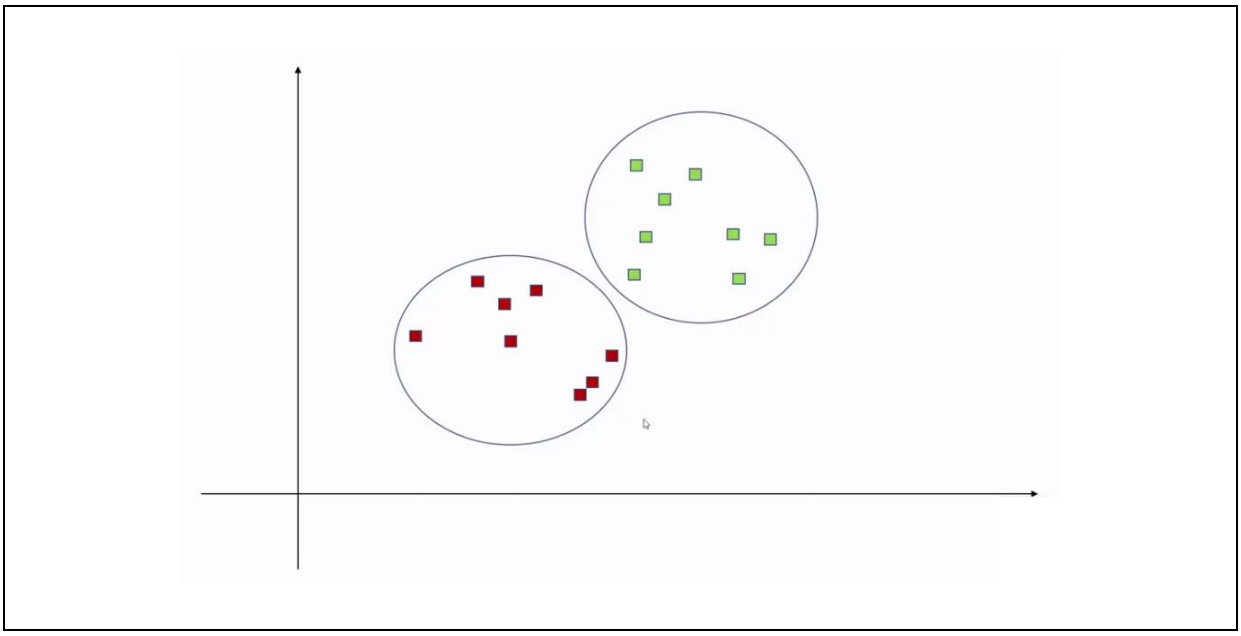
✓ As we got the new centroids so again will draw the median line and reassign the data points.

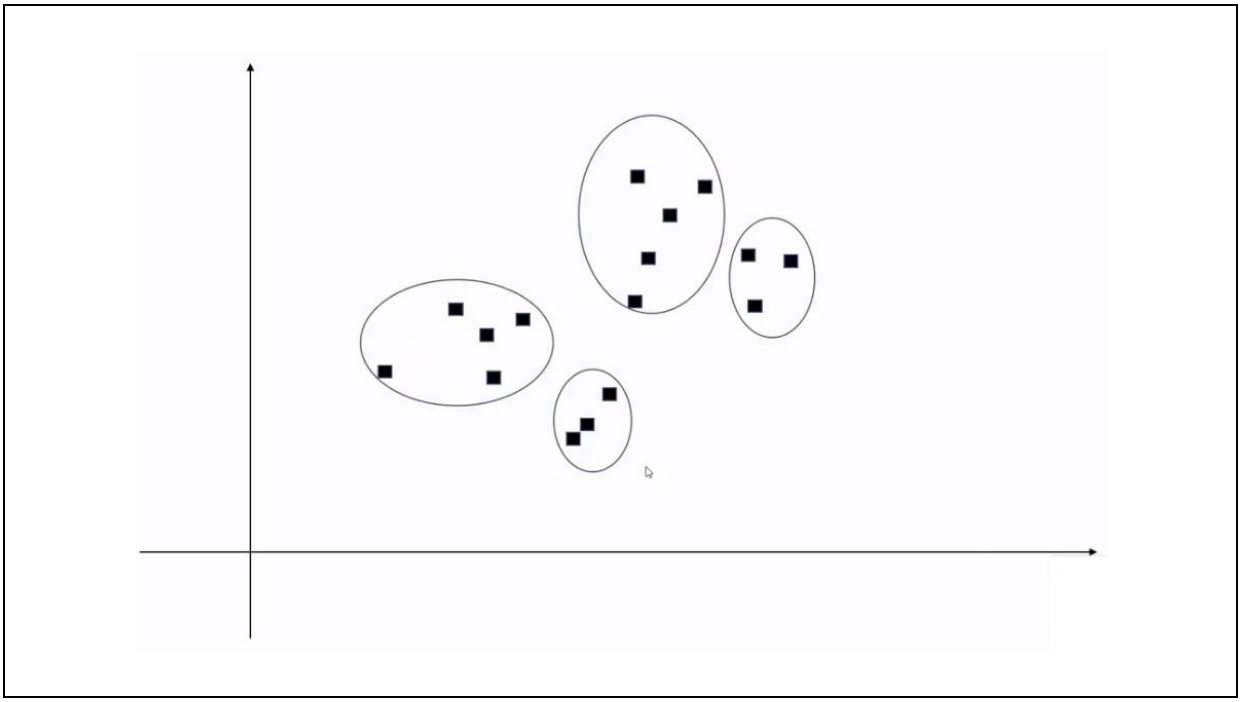


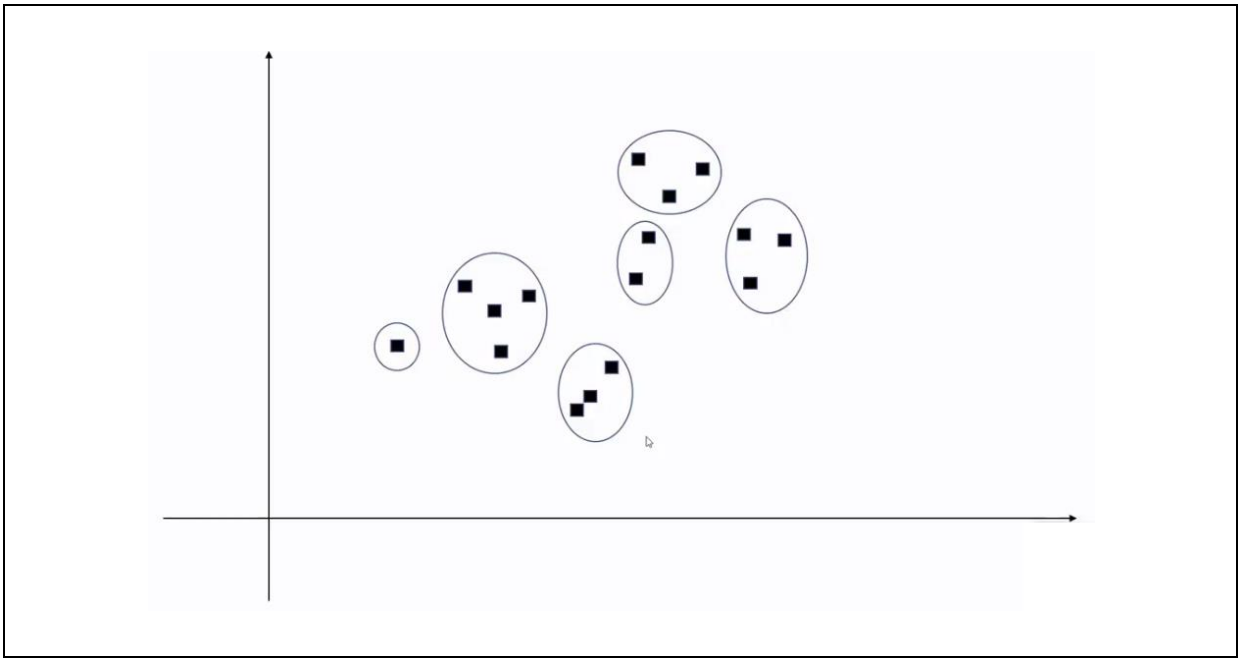


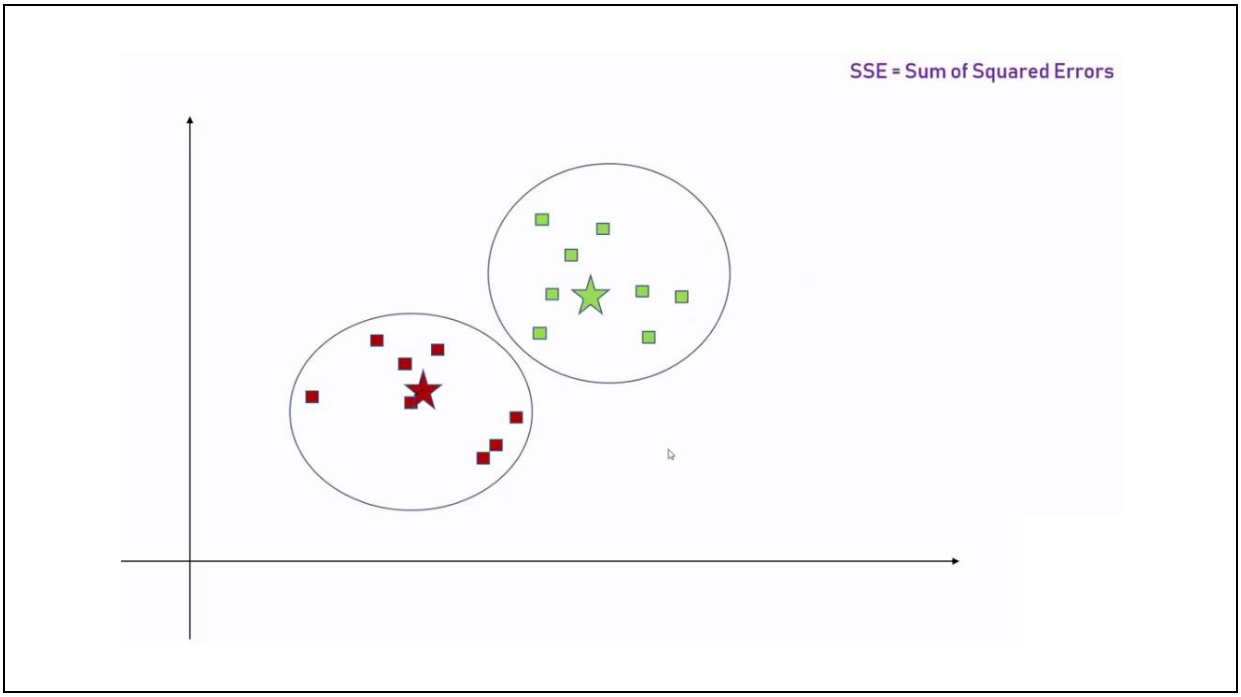
## 5. How to determine the correct number of clusters?

- ✓ Lest take few scenarios



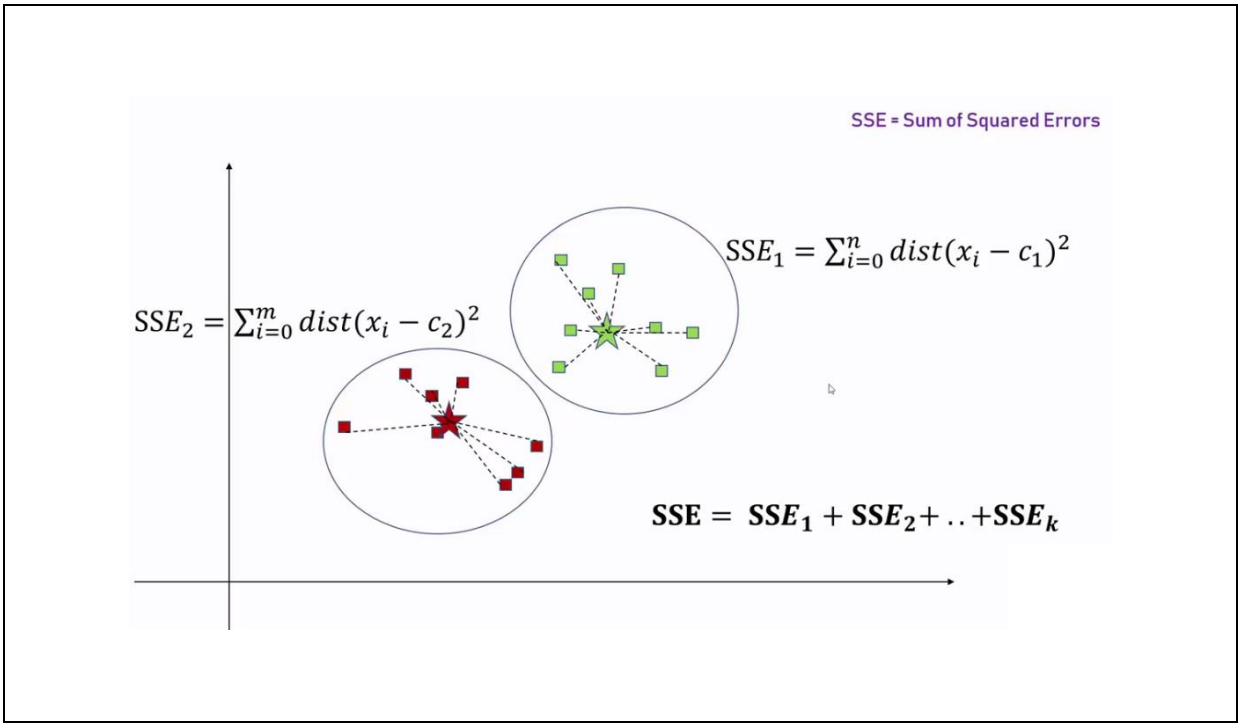


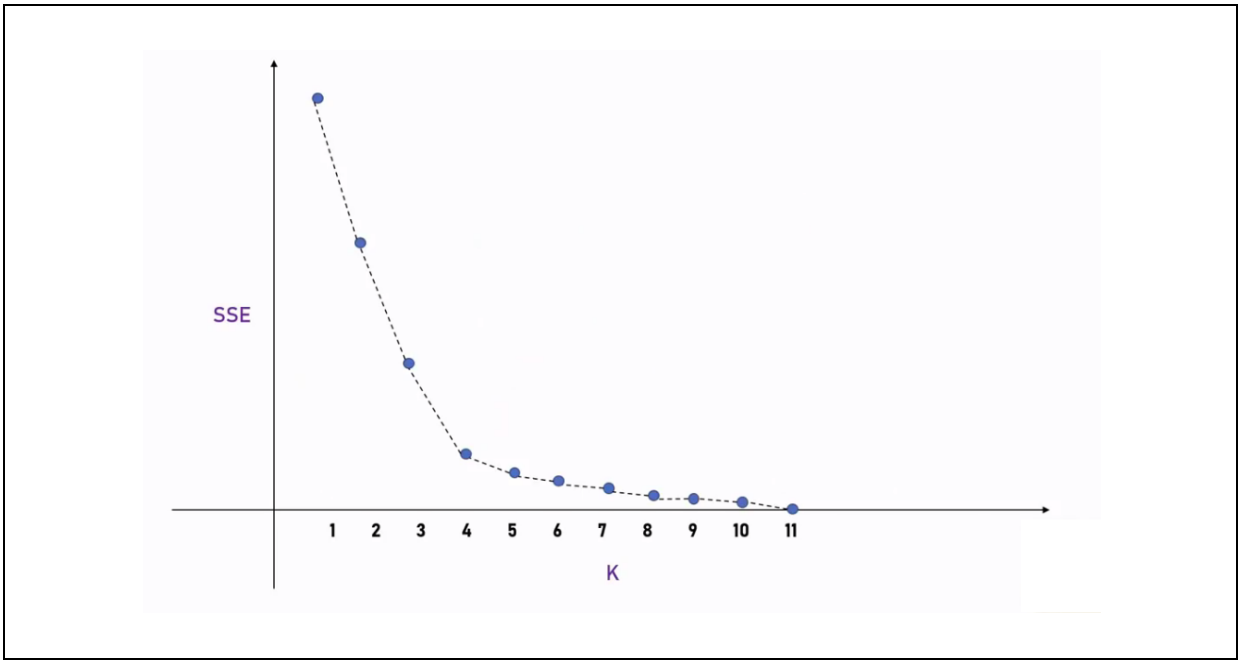


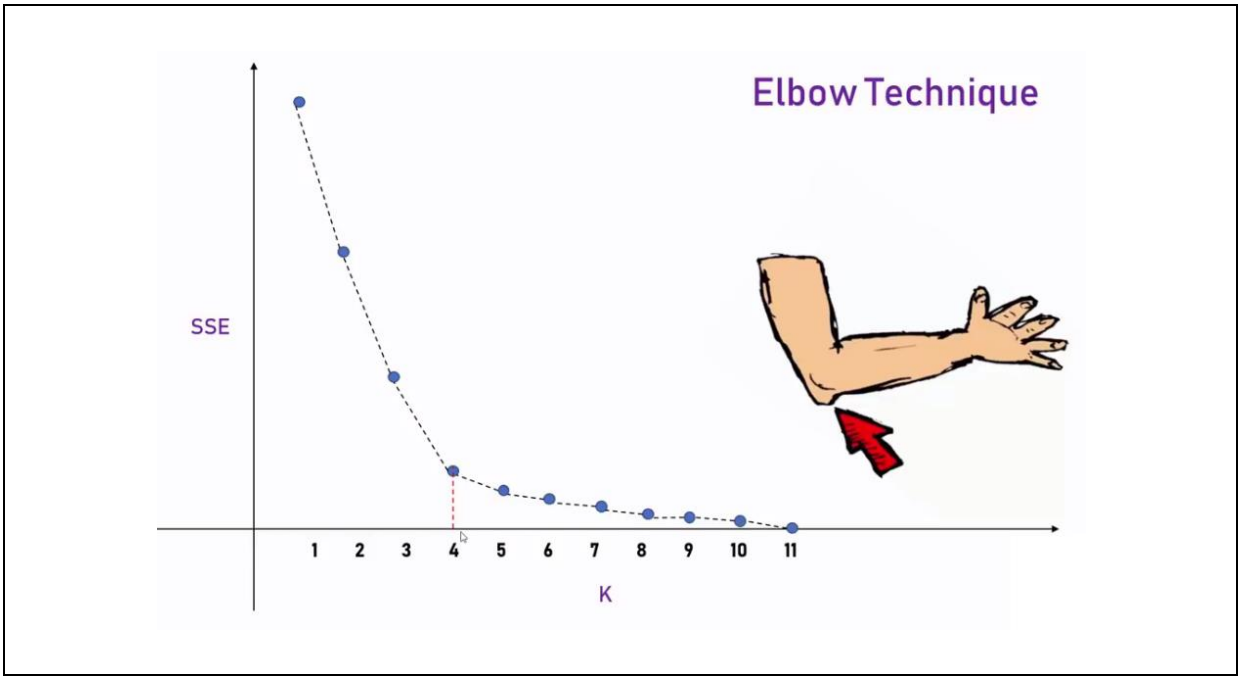

SSE = Sum of Squared Errors

## 6. Elbow Method

- ✓ The Elbow method is one of the most popular ways to find the optimal number of clusters.
- ✓ This method uses the concept of Cluster Sum of Squares.
- ✓ It creates the total variations within a cluster.




Elbow Technique
SSE
1
2
3
4
5
6
7
8
9
10
11
K

Program Name: Loading the dataset
demo1.py

```
import pandas as pd

df = pd.read_csv("income.csv")

print(df.head())
```

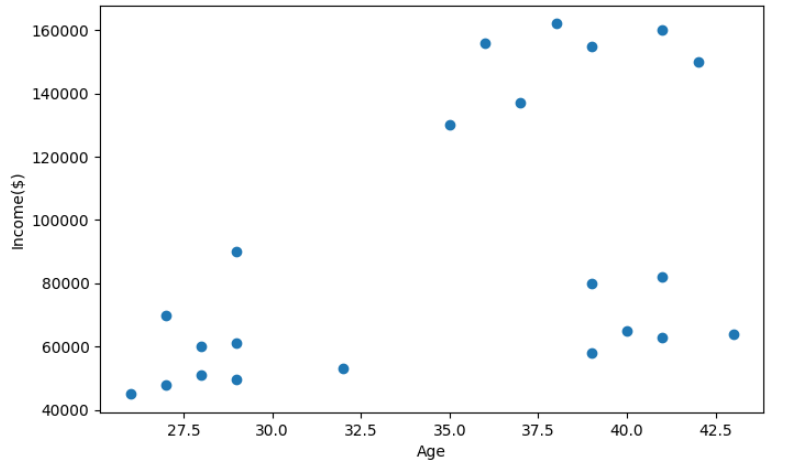
Output

```
      Name  Age  Income($)
0      Rob   27      70000
1  Michael   29      90000
2    Mohan   29      61000
3   Ismail   28      60000
4     Kory   42     150000
```

Program Name: Plotting the data
demo2.py

```
import pandas as pd
from matplotlib import pyplot as plt

df = pd.read_csv("income.csv")

plt.scatter(df.Age, df['Income($)'])
plt.xlabel('Age')
plt.ylabel('Income($)')

plt.show()
```

Output



Program Name: Creating clusters
demo3.py

```
import pandas as pd
from sklearn.cluster import KMeans

df = pd.read_csv("income.csv")

km = KMeans(n_clusters = 3)
y_predicted = km.fit_predict(df[['Age', 'Income($)']])
print(y_predicted)
```

Output

```
[1 1 2 2 0 0 0 0 0 0 0 2 2 2 2 2 2 2 1 1 2]
```

Program Name: Predicting the cluster
demo4.py

```
import pandas as pd
from sklearn.cluster import KMeans

df = pd.read_csv("income.csv")

km = KMeans(n_clusters = 3)

y_predicted = km.fit_predict(df[['Age', 'Income($)']])
df['cluster']=y_predicted

print(df.head())
```

Output

```
      Name  Age  Income($)  cluster
0      Rob   27      70000        2
1  Michael   29      90000        2
2    Mohan   29      61000        0
3   Ismail   28      60000        0
4     Kory   42     150000        1
```

Program Name: Cluster distance
demo5.py

```
import pandas as pd
from sklearn.cluster import KMeans

df = pd.read_csv("income.csv")

km = KMeans(n_clusters = 3)

y_predicted = km.fit_predict(df[['Age', 'Income($)']])
df['cluster']=y_predicted

print(km.cluster_centers_)
```

Output

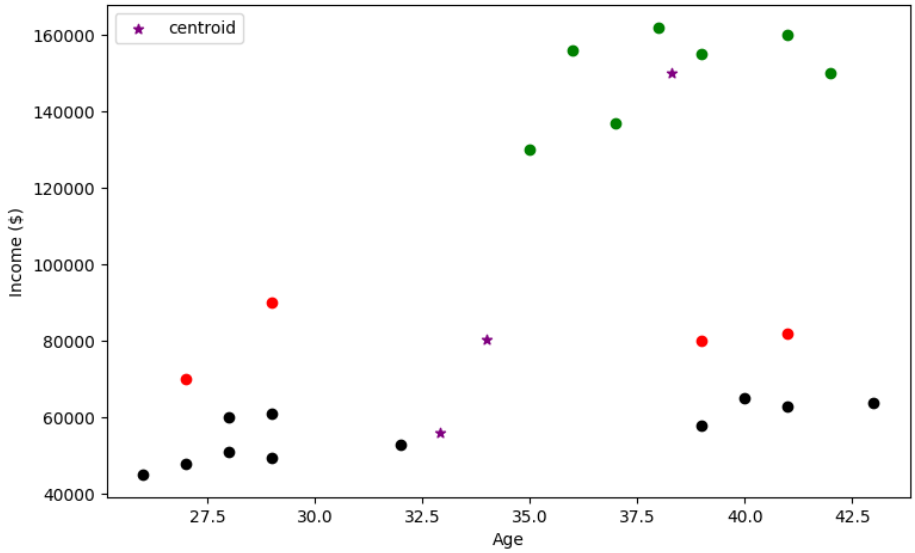
```
[[3.29090909e+01 5.61363636e+04]
 [3.82857143e+01 1.50000000e+05]
 [3.40000000e+01 8.05000000e+04]]
```

Program Name: Plotting the clusters

demo6.py

```
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.cluster import KMeans

df = pd.read_csv("income.csv")

km = KMeans(n_clusters = 3)

y_predicted = km.fit_predict(df[['Age', 'Income($)']])
df['cluster'] = y_predicted

df1 = df[df.cluster==0]
df2 = df[df.cluster==1]
df3 = df[df.cluster==2]

plt.scatter(df1.Age, df1['Income($)'], color='green')
plt.scatter(df2.Age, df2['Income($)'], color='red')
plt.scatter(df3.Age, df3['Income($)'], color='black')

plt.scatter(km.cluster_centers_[:, 0], km.cluster_centers_[:, 1],
color = 'purple', marker='*', label='centroid')

plt.xlabel('Age')
plt.ylabel('Income ($)')
plt.legend()

plt.show()
```

## Output


centroid
160000
140000
120000
100000
80000
60000
40000
Income ($)
27.5
30.0
32.5
35.0
37.5
40.0
42.5
Age

Program Name: Features scaling
demo7.py

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

df = pd.read_csv("income.csv")

scaler = MinMaxScaler()

scaler.fit(df[['Income($)']])
df['Income($)'] = scaler.transform(df[['Income($)']])

scaler.fit(df[['Age']])
df['Age'] = scaler.transform(df[['Age']])

print(df.head())
```

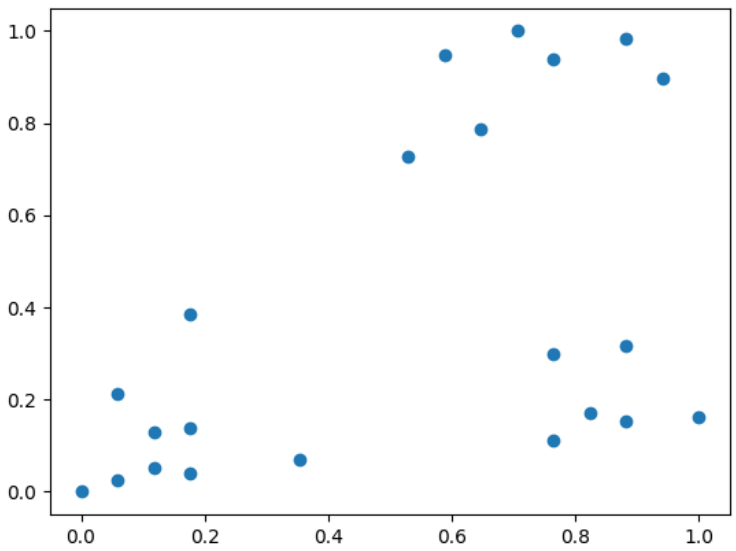
Output

```
      Name       Age  Income($)
0      Rob  0.058824   0.213675
1  Michael  0.176471   0.384615
2    Mohan  0.176471   0.136752
3   Ismail  0.117647   0.128205
4     Kory  0.941176   0.897436
```

Program Name: Plotting after features scaling
demo8.py

```
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.preprocessing import MinMaxScaler

df = pd.read_csv("income.csv")

scaler = MinMaxScaler()

scaler.fit(df[['Income($)']])
df['Income($)'] = scaler.transform(df[['Income($)']])

scaler.fit(df[['Age']])
df['Age'] = scaler.transform(df[['Age']])

plt.scatter(df.Age, df['Income($)'])

plt.show()
```

Output



Program Name: Prediction
demo9.py

```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler

df = pd.read_csv("income.csv")

scaler = MinMaxScaler()

scaler.fit(df[['Income($)']])
df['Income($)'] = scaler.transform(df[['Income($)']])

scaler.fit(df[['Age']])
df['Age'] = scaler.transform(df[['Age']])

km = KMeans(n_clusters = 3)
y_predicted = km.fit_predict(df[['Age', 'Income($)']])
print(y_predicted)
```

Output

```
[1 1 1 1 2 2 2 2 2 2 1 1 1 1 0 0 0 0 0 0]
```

Program Name: Prediction
demo10.py

```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler

df = pd.read_csv("income.csv")

scaler = MinMaxScaler()

scaler.fit(df[['Income($)']])
df['Income($)'] = scaler.transform(df[['Income($)']])

scaler.fit(df[['Age']])
df['Age'] = scaler.transform(df[['Age']])

km = KMeans(n_clusters=3)
y_predicted = km.fit_predict(df[['Age', 'Income($)']])

df['cluster'] = y_predicted
print(df.head())
```

Output

| | Name | Age | Income($) | cluster |
|---|---|---|---|---|
| 0 | Rob | 0.058824 | 0.213675 | 1 |
| 1 | Michael | 0.176471 | 0.384615 | 1 |
| 2 | Mohan | 0.176471 | 0.136752 | 1 |
| 3 | Ismail | 0.117647 | 0.128205 | 1 |
| 4 | Kory | 0.941176 | 0.897436 | 0 |

Program Name: Cluster distance
demo11.py

```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler

df = pd.read_csv("income.csv")

scaler = MinMaxScaler()

scaler.fit(df[['Income($)']])
df['Income($)'] = scaler.transform(df[['Income($)']])

scaler.fit(df[['Age']])
df['Age'] = scaler.transform(df[['Age']])

km = KMeans(n_clusters=3)
y_predicted = km.fit_predict(df[['Age', 'Income($)']])

df['cluster']=y_predicted
print(km.cluster_centers_)
```

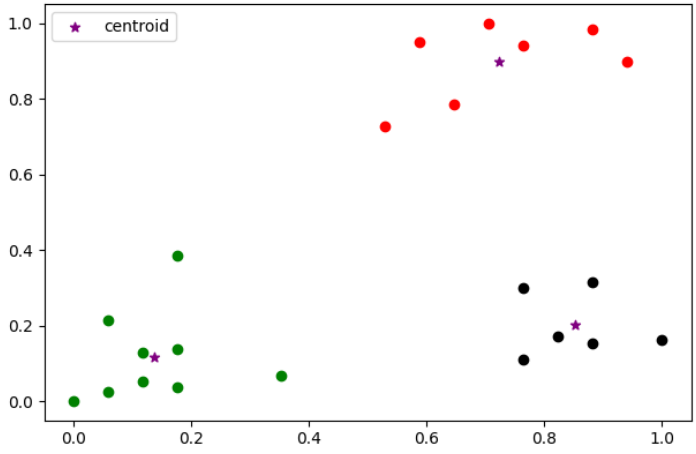
Output

```
[[0.1372549  0.11633428]
 [0.72268908 0.8974359 ]
 [0.85294118 0.2022792 ]]
```

Program Name: Plotting the clusters
demo12.py

```
import pandas as pd
from sklearn.cluster import KMeans
from matplotlib import pyplot as plt
from sklearn.preprocessing import MinMaxScaler

df = pd.read_csv("income.csv")

scaler = MinMaxScaler()

scaler.fit(df[['Income($)']])
df['Income($)'] = scaler.transform(df[['Income($)']])

scaler.fit(df[['Age']])
df['Age'] = scaler.transform(df[['Age']])

km = KMeans(n_clusters=3)
y_predicted = km.fit_predict(df[['Age', 'Income($)']])

df['cluster']=y_predicted

df1 = df[df.cluster==0]
df2 = df[df.cluster==1]
df3 = df[df.cluster==2]

plt.scatter(df1.Age,df1['Income($)'],color='green')
plt.scatter(df2.Age,df2['Income($)'],color='red')
plt.scatter(df3.Age,df3['Income($)'],color='black')

plt.scatter(km.cluster_centers_[:, 0], km.cluster_centers_[:, 1],
color='purple', marker='*',label='centroid')

plt.legend()

plt.show()
```

Output



Program Name: Elbow method
demo13.py

```
import pandas as pd
from sklearn.cluster import KMeans
from matplotlib import pyplot as plt
from sklearn.preprocessing import MinMaxScaler

df = pd.read_csv("income.csv")

scaler = MinMaxScaler()

scaler.fit(df[['Income($)']])
df['Income($)'] = scaler.transform(df[['Income($)']])

scaler.fit(df[['Age']])
df['Age'] = scaler.transform(df[['Age']])

sse = []
k_rng = range(1,10)

for k in k_rng:
        km = KMeans(n_clusters=k)
        km.fit(df[['Age', 'Income($)']])
        sse.append(km.inertia_)

plt.xlabel('K')
plt.ylabel('Sum of squared error')
plt.plot(k_rng, sse)

plt.show()
```

Output