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

```
from google.colab import files
uploaded = files.upload()
```

Choose Files income (1).csv

- **income (1).csv**(text/csv) - 363 bytes, last modified: 7/27/2024 - 100% done

```
Saving income (1).csv to income (1).csv
```

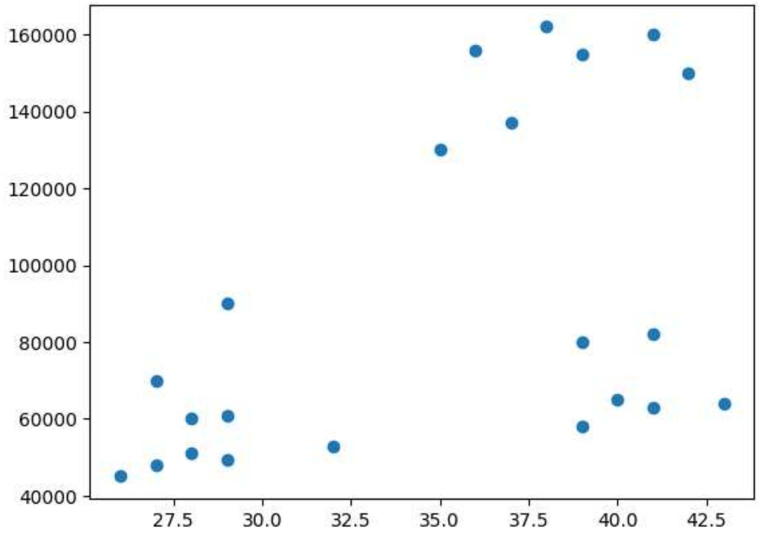
```
df =  pd.read_csv("income (1).csv")
df.head()
```

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

Next steps: Generate code with df View recommended plots New interactive sheet

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

```
<matplotlib.collections.PathCollection at 0x7f6603122770>
```



```
#k = 3
km = KMeans(n_clusters=3)
km
```

```
▾ KMeans
KMeans(n_clusters=3)
```

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

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarni
  super()._check_params_vs_input(X, default_n_init=10)
```

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

Next steps: Generate code with df | View recommended plots | New interactive sheet

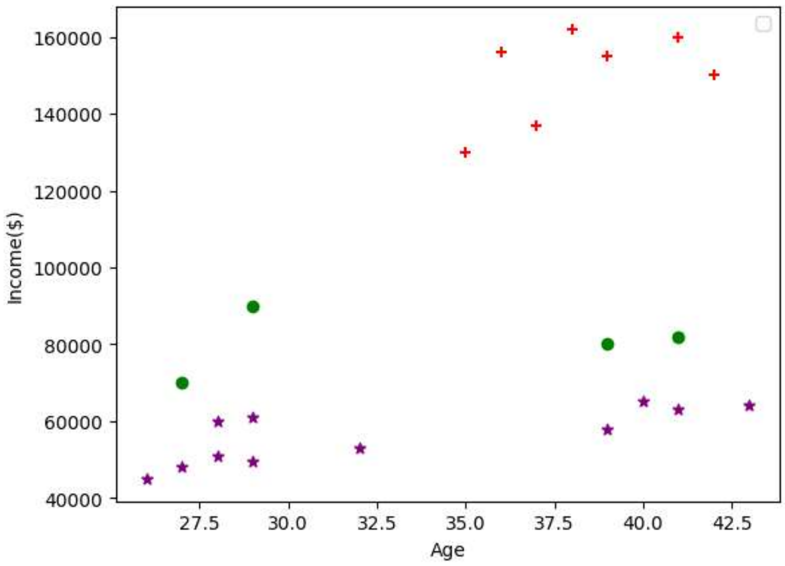
What happened here it, it applied the Kmeans algorithm and then it formed 3 clusters. It assigned them 3 clusters with labels 0,1,2

```
df_1 =df[df.cluster == 0]
df_2 =df[df.cluster == 1]
df_3 =df[df.cluster == 2]

plt.scatter(df_1['Age'], df_1['Income($)'],color = 'red', marker = "+")
plt.scatter(df_2['Age'], df_2['Income($)'],color = 'purple', marker = "*")
plt.scatter(df_3['Age'], df_3['Income($)'],color = 'green')

plt.xlabel("Age")
plt.ylabel("Income($)")
plt.legend()
```

```
WARNING:matplotlib.legend:No artists with labels found to put in legend.  Note that 
<matplotlib.legend.Legend at 0x7f6602d97130>
```



The plot does not look okay as the green ones and the purples ones look merged, it is happening because of scaling, the range of x axis is too narrow compared to the x axis, this is the reason why we use a MinMaxScaler()

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

```
▾ MinMaxScaler
MinMaxScaler()
```

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

	Name	Age	Income($)	cluster
0	Rob	27	0.213675	2
1	Michael	29	0.384615	2
2	Mohan	29	0.136752	1
3	Ismail	28	0.128205	1
4	Kory	42	0.897436	0

Next steps: Generate code with df View recommended plots New interactive sheet

```
scaler.fit(df[["Age"]])
df["Age"]=scaler.transform(df[["Age"]])
df.head()
```

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

Next steps: Generate code with df View recommended plots New interactive sheet

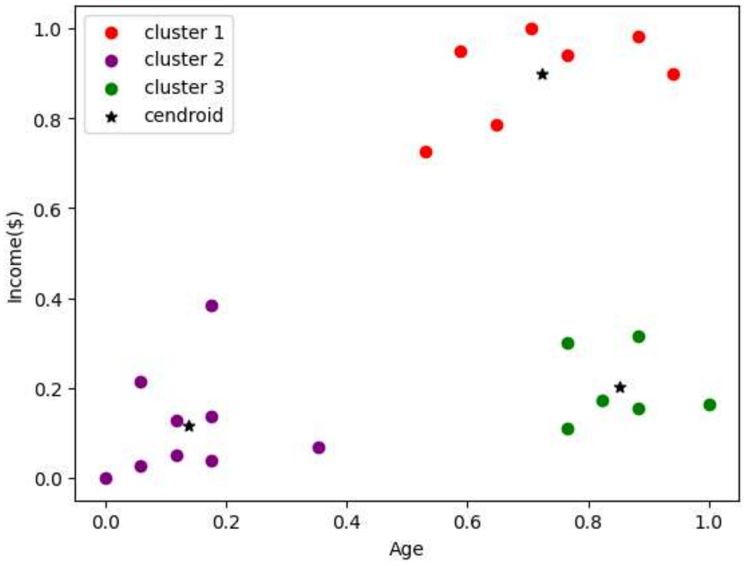
```
km = KMeans(n_clusters=3)
y_predicted = km.fit_predict(df[["Age",'Income($)']])
df['cluster']=y_predicted
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change i
  super()._check_params_vs_input(X, default_n_init=10)
```

```
df_1 =df[df.cluster == 0]
df_2 =df[df.cluster == 1]
df_3 =df[df.cluster == 2]

plt.scatter(df_1['Age'], df_1['Income($)'],color = 'red',label = "cluster 1")
plt.scatter(df_2['Age'], df_2['Income($)'],color = 'purple',label = "cluster 2")
plt.scatter(df_3['Age'], df_3['Income($)'],color = 'green', label = "cluster 3")
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color ="black",marker = "*",label = "cendroid")
plt.xlabel("Age")
plt.ylabel("Income($)")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f65fed5f4f0>
```



It is much better now and it has been solved.

```
km.cluster_centers_ #these are the centroid values for each clusters
```

```
array([[0.72268908, 0.8974359 ],
       [0.1372549 , 0.11633428],
       [0.85294118, 0.2022792 ]])
```

In a real life problem it would be more complicated and we have to use the elbow method. We choose a number of k and we find the SSE

```
k_range = range(1,10)
SSE =[]
for k in k_range:
  km = KMeans(n_clusters=k)
  km.fit(df[["Age","Income($)"]])
  SSE.append(km.inertia_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change t
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change t
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change t
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change t
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change t
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change t
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change t
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change t
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change t
  super()._check_params_vs_input(X, default_n_init=10)
```

```
SSE
```

```
[5.434011511984241,
 2.091136388689264,
 0.4750783498520276,
 0.3491047094404182,
 0.26640301246863574,
 0.21066678487917875,
 0.17796706251972708,
 0.1326541982744777,
 0.101887877250499]
```

```
plt.plot(k_range,SSE)
plt.ylabel("SSE")
plt.xlabel("K")
```

```
Text(0.5, 0, 'K')
```