Implement the k_Means Clustering using "Income.csv"

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

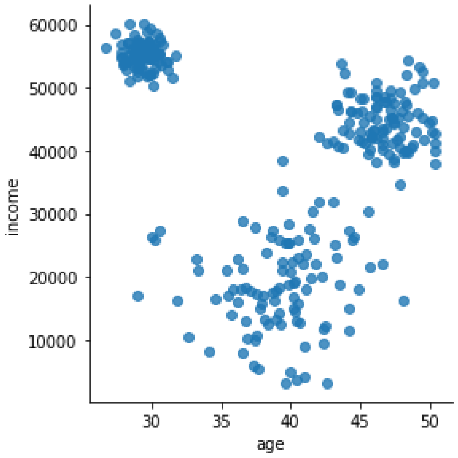
Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Income Data.csv to Income Data (1).csv
```

1. Create a data frame and visualize the natural groupings in the dataset

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sn
df = pd.read_csv("/content/Income Data.csv")
sn.lmplot("age", "income", data = df, fit_reg = False, size = 4)
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
/usr/local/lib/python3.6/dist-packages/seaborn/regression.py:580: UserWarning: The `s
  warnings.warn(msg, UserWarning)
<seaborn.axisgrid.FacetGrid at 0x7f8a0e1b7fd0>
```



2. The above groupings are mostly segmented using income, since it has a huge range. Scale of age is 0 to 60 and income is from 0 to 50000. Hence Euclidean distance will always be dominated by income and not age. Hence all features need to be normalised to a uniform scale before clustering.

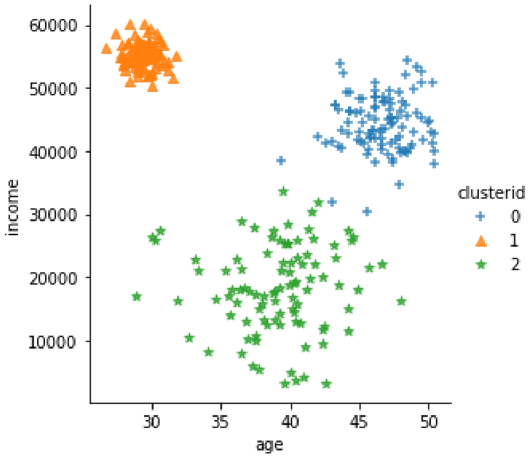
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled_df = scaler.fit_transform(df[["age", "income"]])
scaled_df[0:5]
```

```
array([[ 1.3701637 ,  0.09718548],
       [-1.3791283 ,  0.90602749],
       [ 1.10388844,  0.51405021],
       [ 0.23849387, -1.27162408],
       [-0.35396857, -1.32762083]])
```

3. PLotting customers with their segments

```
from sklearn.cluster import KMeans
clusters = KMeans(3)
clusters.fit(scaled_df)
df["clusterid"] = clusters.labels_
markers = ['+', '^', '*']
sn.lmplot("age", "income", data = df, hue = "clusterid", fit_reg = False, markers  = markers, size = 4)
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
/usr/local/lib/python3.6/dist-packages/seaborn/regression.py:580: UserWarning: The `s
  warnings.warn(msg, UserWarning)
<seaborn.axisgrid.FacetGrid at 0x7f8a0dcf8780>
```



4. Print the cluster centers using the original dataframe. Cluster centres explain the characteristics of the cluster and helps us to interpret the clusters. Print the cluster centres to understand the average age and income of each cluster.

```
clusters = KMeans(3)
clusters.fit(df)
df["new_clusterid"] = clusters.labels_
df.groupby("new_clusterid")['age', 'income'].agg(["mean", 'std']).reset_index()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4: FutureWarning: Indexi
  after removing the cwd from sys.path.
```

	new_clusterid	age		income	
		mean	std	mean	std
0	0	39.174479	3.626068	18144.791667	6745.241906
1	1	31.700435	6.122122	54675.652174	2362.224320
2	2	46.419101	2.289620	43053.932584	3613.769632

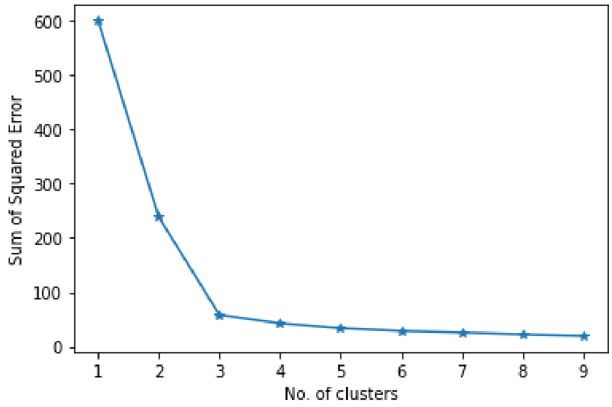
Double-click (or enter) to edit

5. So Cluster 0 has a mean age of 39 and income of 18K. Low age and low income. CLuster 1 has a mean age of 37 and income of 54K. Mid age and high income. CLuster 2 has a mean age of 46 and income of 43K. High age and medium income. The actual age and income of a customer within a cluster will vary from the cluster centers and is called the cluster variance. This is given by WCSS - within cluster sum of squares.

6. Find the optimum number of clusters that may exist using Elbow Method. Try with number of clusters from 1 to 10. In each case print the total variance using "inertia" parameter of the clusters.

```
cluster_range = range(1,10)
cluster_errors = []
for num_clusters in cluster_range:
  clusters = KMeans(num_clusters)
  clusters.fit(scaled_df)
  cluster_errors.append(clusters.inertia_)
plt.figure(figsize = (6,4))
plt.plot(cluster_range, cluster_errors, marker = "*")
plt.xlabel("No. of clusters")
plt.ylabel("Sum of Squared Error")
```

```
Text(0, 0.5, 'Sum of Squared Error')
```



Double-click (or enter) to edit

7. The figure indicates the elbow point is 3, this means there might exist three clusters in the data set.