Faculty of Computing and Information Technology (FCIT)
Department of Computing Indus University, Karachi

STUDENT: AHMED ALI ANSARI ID No: 1402-2020

Lab no 8
Unsupervised Learning in AI

Objectives:

- What is unsupervised learning?
- Implementation of K-means clustering algorithm.
- Implementation of KNN (k-nearest neighbors).

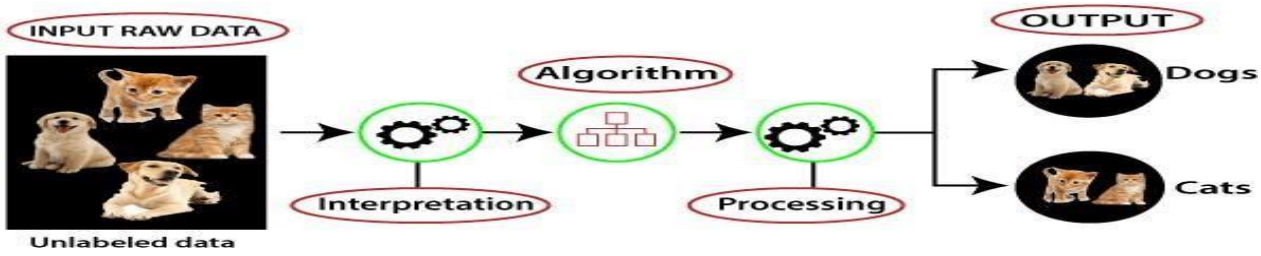
Unsupervised Learning:

We learned supervised machine learning in which models are trained using labeled data under the supervision of training data. But there may be many cases in which we do not have labeled data and need to find the hidden patterns from the given dataset. So, to solve such types of cases in machine learning, we need unsupervised learning techniques.

As the name suggests, unsupervised learning is a machine learning technique in which models are not supervised using training dataset. Instead, models itself find the hidden patterns and insights from the given data. It can be compared to learning which takes place in the human brain while learning new things. It can be defined as:

> *Unsupervised learning is a type of machine learning in which models are trained using unlabeled dataset and are allowed to act on that data without any supervision.*

Working of unsupervised learning can be understood by the below diagram:



Working of Unsupervised Learning

Types of Unsupervised Learning Algorithm:

STUDENT: AHMED ALI ANSARI ID No: 1402-2020

The unsupervised learning algorithm can be further categorized into two types of problems:

- **Clustering:** Clustering is a method of grouping the objects into clusters such that objects with most similarities remains into a group and has less or no similarities with the objects of another group. Cluster analysis finds the commonalities between the data objects and categorizes them as per the presence and absence of those commonalities.

- **Association:** An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database. It determines the set of items that occurs together in the dataset. Association rule makes marketing strategy more effective. Such as people who buy X item (suppose a bread) are also tend to purchase Y (Butter/Jam) item. A typical example of Association rule is Market Basket Analysis.

K-means clustering

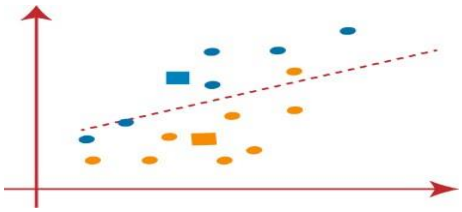
The K-means clustering algorithm computes centroids and repeats until the optimal centroid is found. It is presumptively known how many clusters there are. It is also known as the flat clustering algorithm. The number of clusters found from data by the method is denoted by the letter 'K' in K-means.

In this method, data points are assigned to clusters in such a way that the sum of the squared distances between the data points and the centroid is as small as

possible. It is essential to note that reduced diversity within clusters leads to more identical data points within the same cluster. **Working of K-Means Algorithm**

The following stages will help us understand how the K-Means clustering technique works-

- **Step 1:** First, we need to provide the number of clusters, K, that need to be generated by this algorithm.
- **Step 2:** Next, choose K data points at random and assign each to a cluster. Briefly, categorize the data based on the number of data points.
- **Step 3:** The cluster centroids will now be computed.
- **Step 4:** Iterate the steps below until we find the ideal centroid, which is the assigning of data points to clusters that do not vary.
- **4.1** The sum of squared distances between data points and centroids would be calculated first.
- **4.2** At this point, we need to allocate each data point to the cluster that is closest to theothers (centroid).
- **4.3** Finally, compute the centroids for the clusters by averaging all of the cluster's data points.



Step 1: Create a dataset.

	Name	Age	Income($)
1	Rob	27	70000
2	Michael	29	90000
3	Mohan	29	61000
4	Ismail	28	60000
5	Kory	42	150000
6	Gautam	39	155000
7	David	41	160000
8	Andrea	38	162000
9	Brad	36	156000
10	Angelina	35	130000
11	Donald	37	137000
12	Tom	26	45000
13	Arnold	27	48000
14	Jared	28	51000
15	Stark	29	49500
16	Ranbir	32	53000
17	Dipika	40	65000
18	Priyanka	41	63000
19	Nick	43	64000
20	Alia	39	80000
21	Sid	41	82000
22	Abdul	39	58000

Step 2: Import Libraries and class

STUDENT: AHMED ALI ANSARI ID No: 1402-2020

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

Step 3: Import Dataset

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

NAME OF

Step 4: Apply scatter on dataset

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

Step 5: Create reference variable of KMeans

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

```
array([2, 2, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 0])
```

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

Step 6: find center and apply scatter on dataset

```
km.cluster_centers_
```

```
array([[  3.29090909e+01,   5.61363636e+04],
       [  3.82857143e+01,   1.50000000e+05],
       [  3.40000000e+01,   8.05000000e+04]])
```

```
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()
```

Preprocessing using min max scalar

Step 7: apply scalar on dataset

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

```
df.head()
```

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

STUDENT: AHMED ALI ANSARI ID No: 1402-2020

Step 8: Re-create reference variable of KMean

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

```
array([0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2])
```

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

Step 9: Re-create Scatter

```
km.cluster_centers_
```

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

```
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()
```

NAME OF

TASKS:

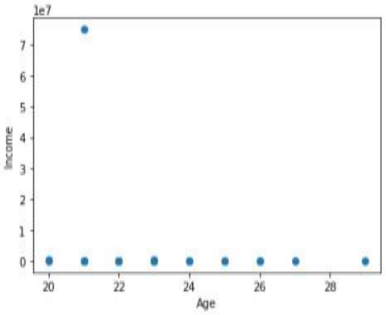
In [1]:
```
from sklearn.cluster import KMeans
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from matplotlib import pyplot as plt
%matplotlib inline                                    #M.SAMI

df = pd.read_csv("C:/Users/Student.DESKTOP-T9AM0KV/Downloads/salaries.csv")
df.head()
```

Out[1]:

	Name	Age	Income
0	Rob	21	70000
1	Michael	22	60000
2	Mohan	23	80000
3	Ali	21	58000
4	Andrea	26	89000

In [2]:
```
plt.scatter(df.Age,df['Income'])
plt.xlabel('Age')                                     #M.SAMI
plt.ylabel('Income')
```

Out[2]: Text(0, 0.5, 'Income')



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

Out[3]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       2])
```

STUDENT: AHMED ALI ANSARI ID No: 1402-2020

```
In [4]: df['cluster'] = y_predicted                    #M.SAMI
        df.head()
```

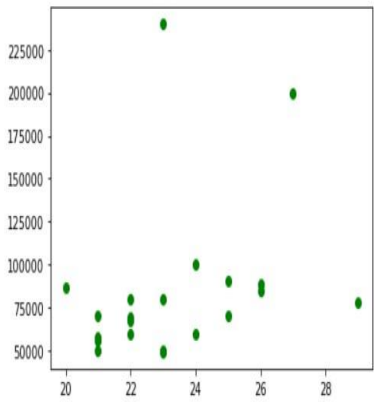
Out[4]:

	Name	Age	Income	cluster
0	Rob	21	70000	0
1	Michael	22	60000	0
2	Mohan	23	80000	0
3	Ali	21	58000	0
4	Andrea	26	89000	0

```
In [5]: km.cluster_centers_
```

```
Out[5]: array([[2.33333333e+01, 8.51428571e+04],
               [2.10000000e+01, 7.50000000e+07],
               [2.00000000e+01, 5.60000000e+05]])
```

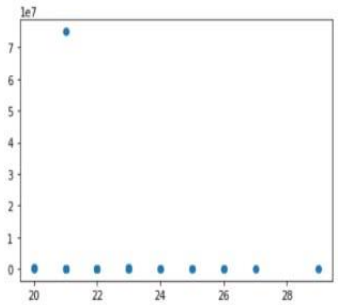
```
In [7]: df1 = df[df.cluster == 0]
        df2 = df[df.cluster == 1]
        df3 = df[df.cluster == 2]
        plt.scatter(df1.Age,df1['Income'], color = 'green')          #M.SAMI
        plt.scatter(df2.Age,df1['Income'], color = 'pink')
        plt.scatter(df3.Age,df1['Income'], color = 'black')
        plt.scatter(km.clutter_centers_[:, 0], km.cluster_centers_[:,1], color = 'pink' , marker = '<3', label
        plt.xlabel('Age')
        plt.ylabel('Income')
        plt.legend()
```




```
In [10]: scaler = MinMaxScaler()
         scaler.fit(df[['Income']])
         df['Income'] = scaler.tansform(df[['Income']])
         scaler.fit(df[['Age']])                                    #M.SAMI
         df['Age'] = scaler.transform(df[['Age']])
```

```
In [12]: df.head()
         plt.scatter(df.Age,df['Income'])
```

```
Out[12]: <matplotlib.collections.PathCollection at 0x18d74dca6d0>
```




```
In [13]: km = KMeans(n_clusters = 3)
         y_predicted = km.fit_predict(df[['Age','Income']])
         y_predicted
```

```
Out[13]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
```

```
In [17]: plt.xlabel('K')
         plt.ylabel('Sum of squared error')                         #M.SAMI
         plt.plot(k_rng, sse)
```

```
Out[17]: [<matplotlib.lines.Line2D at 0x18d74efc400>]
```