**UNSUPERVISED**

**FOR CLUSTERING**

**What I have seen is that with Bayes you don’t need to transform data to scale or normalize.**

**from** sklearn.cluster **import** KMeans

**import** pandas **as** pd

**from** sklearn.preprocessing **import** MinMaxScaler

**from** matplotlib **import** pyplot **as** plt

**%matplotlib** inline

df **=** pd**.**read\_csv("income.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 |

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

plt**.**xlabel('Age')

plt**.**ylabel('Income($)')

Chart, scatter chart

Description automatically generated

**Preprocessing using min max scaler**

scaler **=** MinMaxScaler()

scaler**.**fit(df[['Income($)']])

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

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 | 0 |
| **3** | Ismail | 0.117647 | 0.128205 | 0 |
| **4** | Kory | 0.941176 | 0.897436 | 1 |

:

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

Chart, scatter chart

Description automatically generated

km **=** KMeans(n\_clusters**=**3)

y\_predicted **=** km**.**fit\_predict(df[['Age','Income($)']])

y\_predicted

Out[206]:

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()

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

km**.**cluster\_centers\_

Out[208]:

array([[ 0.1372549 , 0.11633428],

[ 0.72268908, 0.8974359 ],

[ 0.85294118, 0.2022792 ]])

In [209]:

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()

Chart, scatter chart

Description automatically generated

**Elbow Plot**

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)

Shape

Description automatically generated

**from** sklearn.model\_selection **import** train\_test\_split

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(wine**.**data, wine**.**target, test\_size**=**0.3, random\_state**=**100)

**from** sklearn.naive\_bayes **import** GaussianNB, MultinomialNB

model **=** GaussianNB()

model**.**fit(X\_train,y\_train)

GaussianNB(priors=None, var\_smoothing=1e-09)

model**.**score(X\_test,y\_test)

**1.0**

mn **=** MultinomialNB()

mn**.**fit(X\_train,y\_train)

mn**.**score(X\_test,y\_test)

**0.7777**