بسم الله الرحمن الرحيم

مشروع Data Mining & Warehousing

مادة: data mining and warehousing

الشعبة: IBBB

الاسم	الرقم الجامعي
عبد الرحمن محمود العيد	3901009
معاذ عبد الرحمن العوفي	3901555
علي عدنان محبت	3901498

First of all general information about data: -

1. Title: Haberman's Survival Data

2. Sources:

(a) Donor: Tjen-Sien Lim (limt@stat.wisc.edu)

(b) Date: March 4, 1999

3. Past Usage:

1. Haberman, S. J. (1976). Generalized Residuals for Log-Linear Models, Proceedings of the 9th International Biometrics Conference, Boston, pp. 104-122.
2. Landwehr, J. M., Pregibon, D., and Shoemaker, A. C. (1984), Graphical Models for Assessing Logistic Regression Models (with discussion), Journal of the American Statistical Association 79: 61-83.
3. Lo, W.-D. (1993). Logistic Regression Trees, PhD thesis, Department of Statistics, University of Wisconsin, Madison, WI.

4. Relevant Information:

The dataset contains cases from a study that was conducted between 1958 and 1970 at the University of Chicago's Billings Hospital on the survival of patients who had undergone surgery for breast cancer.

5. Number of Instances: 306

6. Number of Attributes: 4 (including the class attribute)

7. Attribute Information:

1. Age of patient at time of operation (numerical)
2. Patient's year of operation (year - 1900, numerical)
3. Number of positive axillary nodes detected (numerical)
4. Survival status (class attribute)

    1 = the patient survived 5 years or longer

    2 = the patient died within 5 year

8. Missing Attribute Values: None

The following cell shows the importing of the dataset chosen for this project which is haberman and printing of the first 9 rows in the dataset using head function.

```
# importing required libraies
import pandas as pd
import numpy as np
import seaborn as sns #visualisation
import matplotlib.pyplot as plt #drawing library
from sklearn import datasets
%matplotlib inline
sns.set(color_codes=True)
dataset=pd.read_csv(r'C:\Users\2\Desktop\theProject\haberman.data'
                    , delimiter=',',
                        header=0,
                       names=['Age of patient',
                              'Patients year of operation',
                              'Number of positive axillary',
                             'Survival status']
                        )
dataset.head(9)
```

	Age of patient	Patients year of operation	Number of positive axillary	Survival status
0	30	62	3	1
1	30	65	0	1
2	31	59	2	1
3	31	65	4	1
4	33	58	10	1
5	33	60	0	1
6	34	59	0	2
7	34	66	9	2
8	34	58	30	1

To check the dimensions of the dataset, it is done as follows.

```
## 8 columns and 100 attribute
print(dataset.shape)
```

```
(305, 4)
```

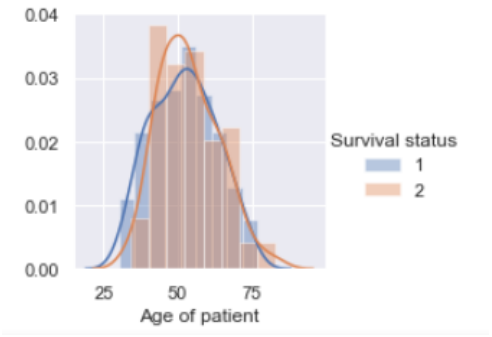
The description of the dataset gives an overview about the mean, std, min, 25%, 50%, 70% and max of each column as follows.

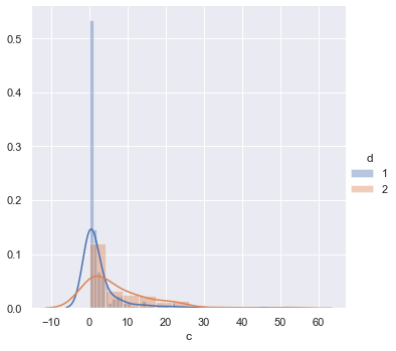
```
## count,mean,standard div,min, quartile for each attribute
print(dataset.describe())
```

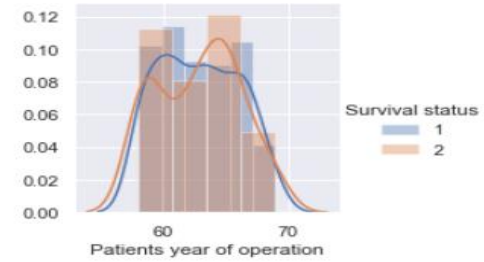
```
       Age of patient  Patients year of operation  \
count      305.000000                  305.000000
mean        52.531148                   62.849180
std         10.744024                    3.254078
min         30.000000                   58.000000
25%         44.000000                   60.000000
50%         52.000000                   63.000000
75%         61.000000                   66.000000
max         83.000000                   69.000000

       Number of positive axillary  Survival status
count                   305.000000       305.000000
mean                      4.036066         1.265574
std                       7.199370         0.442364
min                       0.000000         1.000000
25%                       0.000000         1.000000
50%                       1.000000         1.000000
75%                       4.000000         2.000000
max                      52.000000         2.000000
```

For visualizing the distribution of the data, the following is used. The distribution shows that the data is normally distributed.

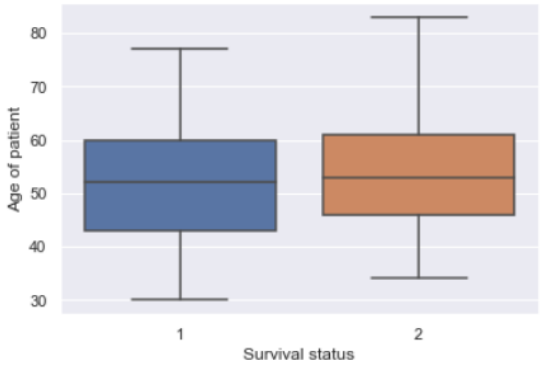


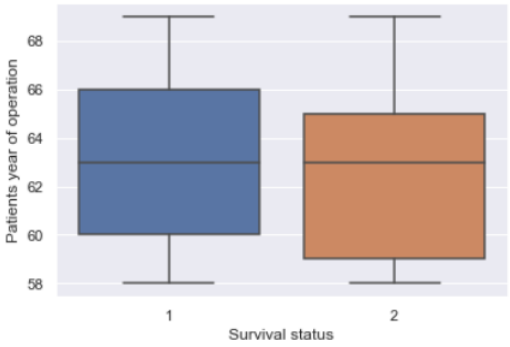


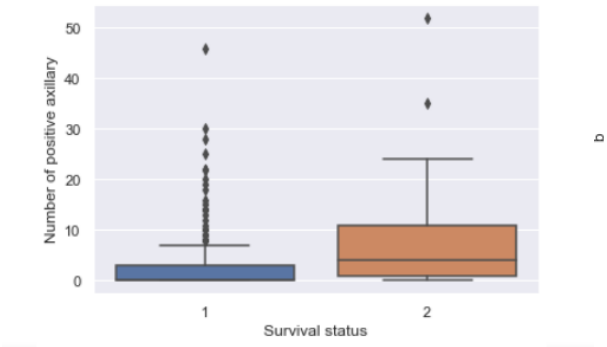


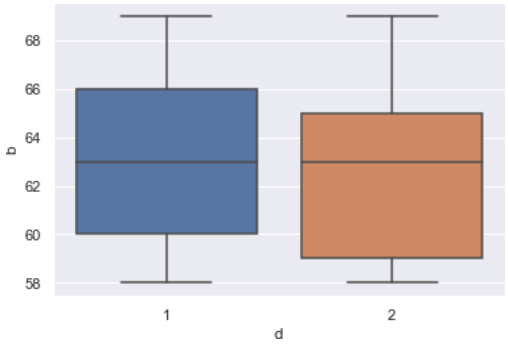
Also, a box plot is used in order to determine the quartiles in the data for the attributes

```
sns.boxplot(x='Survival status', y='Age of patient', data=dataset)
plt.show()
sns.boxplot(x='Survival status', y='Patients year of operation', data=dataset)
plt.show()
sns.boxplot(x='Survival status', y='Number of positive axillary', data=dataset)
plt.show()
```

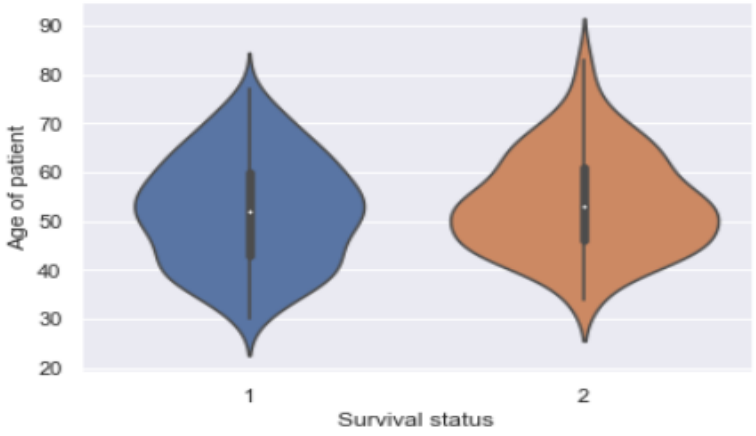








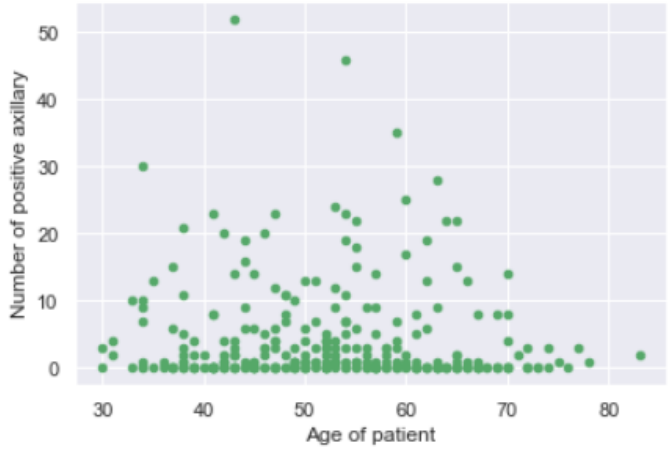
```
sns.violinplot(x='Survival status', y='Age of patient', data=dataset)
plt.show()
```



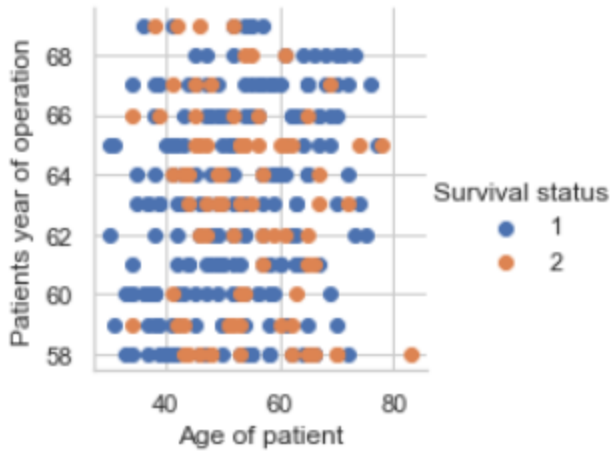
Also, the following scatter plot clearly shows that are outliers in the data as most of the data lies under approximately 25.

```
data=dataset.plot(kind='scatter',x='Age of patient',y='Number of positive axillary',c= 'g')
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



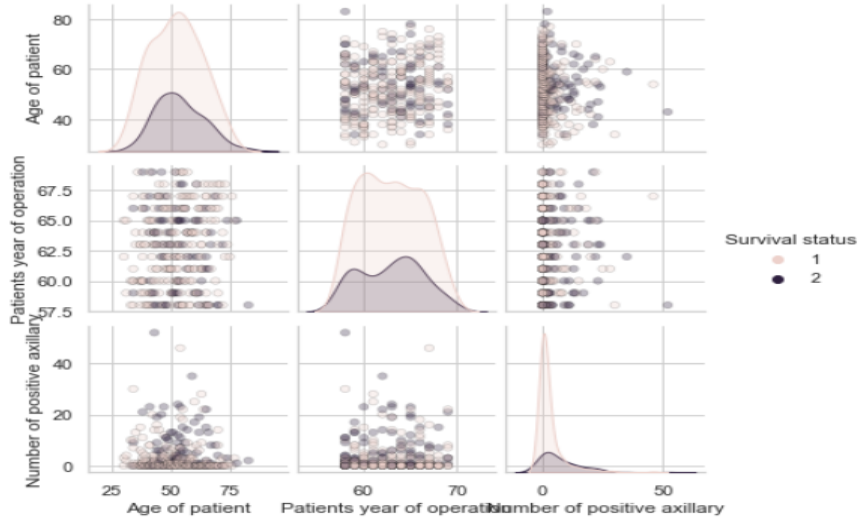
```
sns.set_style("whitegrid")
sns.FacetGrid(dataset, hue="Survival status",height=3) \
    .map(plt.scatter, "Age of patient","Patients year of operation") \
    .add_legend();
plt.show();
```



The falling pair plot creates a grid of Axes such that each attribute in data will by shared in the y-axis across a single row and in the x-axis across a single column.

```
sns.set_style("whitegrid");
sns.pairplot(dataset, hue = 'Survival status', diag_kind = 'kde',
             plot_kws = {'alpha': 0.3, 's': 30, 'edgecolor': 'k'}, size =2)
plt.show()
```

```
C:\Users\2\anaconda3\lib\site-packages\seaborn\axisgrid.py:1912: UserWarnin
g: The `size` parameter has been renamed to `height`; please update your co
de.
  warnings.warn(msg, UserWarning)
```



Checking if there are any null values.

```
data.isnull().sum()
```

```
Age of patient                  0
Patients year of operation      0
Number of positive axillary     0
target                          0
dtype: int64
```

No null values and data type is int64

PCA is used to reduce the dimensionality of the dataset, increasing interpretability but at the same time minimizing information loss.

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
import pandas as pd
PCA_df = pd.read_csv(r'C:\Users\2\Desktop\theProject\haberman.data',
                     delimiter=',',
                     header=0,
                    names=['Age of patient',
                     'Patients year of operation',
                     'Number of positive axillary',
                     'target'
                     ])
```

```
PCA_df.head()
```

	Age of patient	Patients year of operation	Number of positive axillary	target
0	30	62	3	1
1	30	65	0	1
2	31	59	2	1
3	31	65	4	1
4	33	58	10	1

In order to preprocess the data, the StandardScaler and MinMaxScaler are used to normalize the data as follows.

```
#using normlization as we explained in the slides
from sklearn.preprocessing import StandardScaler, MinMaxScaler

features = ['Age of patient', 'Patients year of operation'
            , 'Number of positive axillary']

# Separating out the features
# :, gives u from the start to all columns
x = PCA_df.loc[:, features].values

# Separating out the target

y = PCA_df.loc[:,['target']].values

# Standardizing the features

#x = StandardScaler().fit_transform(x)
x = StandardScaler().fit_transform(x)
#mini max feature
#x =  MinMaxScaler().fit_transform(x)

print (x)
```

```
[[-2.10053274e+00 -2.61387704e-01 -1.44147084e-01]
 [-2.10053274e+00  6.62047621e-01 -5.61535003e-01]
 [-2.00730479e+00 -1.18482303e+00 -2.83276391e-01]
 [-2.00730479e+00  6.62047621e-01 -5.01777826e-03]
 [-1.82084888e+00 -1.49263480e+00  8.29758059e-01]
 [-1.82084888e+00 -8.77011254e-01 -5.61535003e-01]
 [-1.72762093e+00 -1.18482303e+00 -5.61535003e-01]
 [-1.72762093e+00  9.69859396e-01  6.90628753e-01]
 [-1.72762093e+00 -1.49263480e+00  3.61234418e+00]
 [-1.72762093e+00 -8.77011254e-01 -4.22405697e-01]
 [-1.72762093e+00 -5.69199479e-01  8.29758059e-01]
 [-1.72762093e+00  1.27767117e+00  4.12370140e-01]
 [-1.72762093e+00 -8.77011254e-01 -5.61535003e-01]
 [-1.63439298e+00  3.54235846e-01  1.24714598e+00]
 [-1.63439298e+00  4.64240710e-02 -5.61535003e-01]
 [-1.54116503e+00 -8.77011254e-01 -4.22405697e-01]
 [-1.54116503e+00  1.89329472e+00 -5.61535003e-01]
 [-1.44793708e+00 -8.77011254e-01 -5.61535003e-01]
 [-1.44793708e+00  4.64240710e-02 -5.61535003e-01]
```

The PCA are constructed as follows.

```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
principalComponents = pca.fit_transform(x)
principalDf = pd.DataFrame(data = principalComponents

       , columns = ['principal component 1', 'principal component 2'])
print(principalDf)
```

```
     principal component 1  principal component 2
0                 1.564649              -0.288826
1                 0.858638              -0.089232
2                 1.971183              -0.939584
3                 1.027176               0.363644
4                 2.485094              -0.213370
..                     ...                    ...
300              -1.498467              -0.476315
301              -2.506614               0.309218
302              -2.042908               0.290016
303              -2.225380               0.064862
304              -1.256378              -1.074608

[305 rows x 2 columns]
```

```
finalDf=pd.concat([principalDf,PCA_df[['target']]],axis=1)
print(finalDf)
```

```
     principal component 1  principal component 2  target
0                 1.564649              -0.288826       1
1                 0.858638              -0.089232       1
2                 1.971183              -0.939584       1
3                 1.027176               0.363644       1
4                 2.485094              -0.213370       1
..                     ...                    ...     ...
300              -1.498467              -0.476315       1
301              -2.506614               0.309218       1
302              -2.042908               0.290016       1
303              -2.225380               0.064862       2
304              -1.256378              -1.074608       2

[305 rows x 3 columns]
```

```
pip install plotly
```

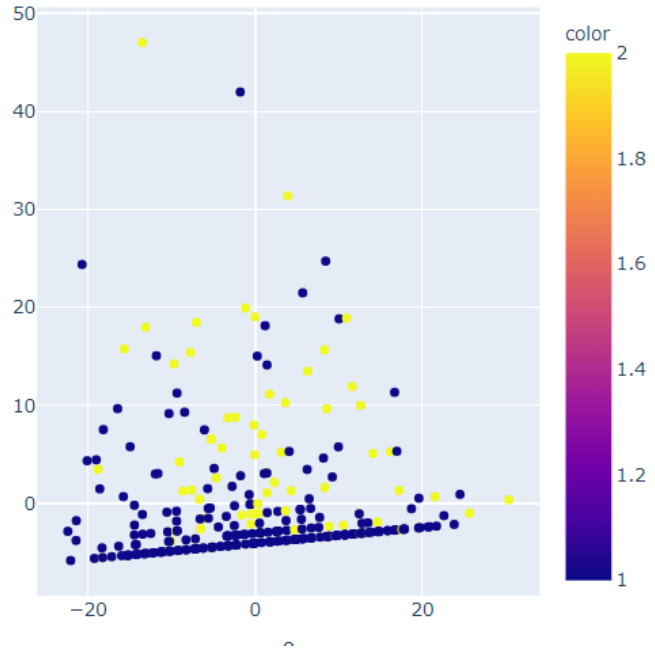
```
Requirement already satisfied: plotly in c:\users\2\anaconda3\lib\site-p
ackages (4.14.3)
Requirement already satisfied: retrying>=1.3.3 in c:\users\2\anaconda3\l
ib\site-packages (from plotly) (1.3.3)
Requirement already satisfied: six in c:\users\2\anaconda3\lib\site-pack
ages (from plotly) (1.15.0)
Note: you may need to restart the kernel to use updated packages.
```

```
import matplotlib.pyplot as plt
import plotly.express as px
%matplotlib inline

#sns.set(color_codes=True)



X = dataset[['Age of patient', 'Patients year of operation'
             , 'Number of positive axillary']]
pca = PCA(n_components=2)
components = pca.fit_transform(X)
fig = px.scatter(components, x=0, y=1, color=finalDf['target'])
fig.show()
```




```
print(pca.explained_variance_ratio_)
```

```
[0.65192362 0.28910141]
```

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns #visualisation
import numpy as np
from sklearn.cluster import KMeans
```

```
# select all columns
x= dataset.iloc[:,[0,1,2,3]].values
```

KMeans is a method of clustering to determine the clusters which the data fall in. Therefore, 5 clusters have been built where points are added to each cluster as follow.

```
##
kmeans5=KMeans(n_clusters=5)
y_kmeans5=kmeans5.fit_predict(x)
print(y_kmeans5)
kmeans5.cluster_centers_
```

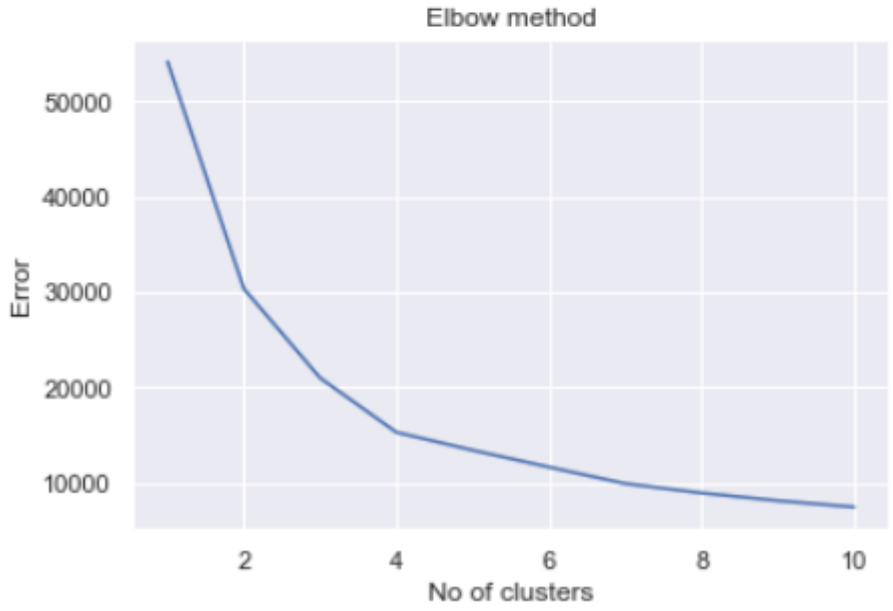
```
[2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2
2 2 2 2 2
 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 3 2 2 2 3 2 2 2
2 2 2 2 2
 3 2 2 2 3 4 4 4 4 4 3 4 4 4 4 4 4 3 4 4 4 3 4 4 4 4 4 4 4 4 4 4
4 4 4 4 4
 4 4 4 4 4 4 4 4 4 4 4 4 3 4 4 4 4 4 4 4 4 4 4 4 4 3 4 4 4 4 4 4 4
4 4 4 4 4
 4 4 4 4 4 4 4 4 4 4 4 3 4 4 4 4 4 4 4 3 4 4 4 4 4 3 4 4 3 4 4 1
3 1 1 1 3
 1 1 3 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1
1 1 1 1 1
 3 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 3 1 3
0 1 0 0 0
 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0
 0 0 0 0 0 0 0 0 0]
```

```
array([[69.16326531, 63.53061224,  1.87755102,  1.28571429],
       [59.03030303, 62.84848485,  1.6969697 ,  1.18181818],
       [38.92957746, 62.28169014,  2.33802817,  1.16901408],
       [51.35714286, 63.10714286, 22.39285714,  1.57142857],
       [49.83516484, 62.84615385,  2.57142857,  1.2967033 ]])
```

The following Elbow algorithm is used in order to choose the optimal number of clusters for the dataset.

```
# elbow algorithm to select optimal
Error=[]
for i in range(1,11):
    kmeans= KMeans(n_clusters=i).fit(x)
    kmeans.fit(x)
    Error.append(kmeans.inertia_)
plt.plot(range(1,11),Error)
plt.title('Elbow method')
plt.xlabel('No of clusters')
plt.ylabel('Error')
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



Thus, 3 clusters are used to model the data as follows using Kmeans.

```
## optimal at 3 as we see before now we cluster at 3
kmeans3=KMeans(n_clusters=3)
y_kmeans3=kmeans3.fit_predict(x)
```

```
# to show
plt.scatter(x[:,0], x[:,1], c=y_kmeans3, cmap='rainbow')
```

```
<matplotlib.collections.PathCollection at 0x21b6a006a00>
```