Machine Learning Practicals

Scatter Plot On Dataset

In [74]:
```
df3=pd.read_csv('iris.csv')
df3.head()
```

Out[74]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [75]:
```
from sklearn.preprocessing import LabelEncoder
```

In [81]:
```
Encoding=LabelEncoder()
```

In [76]:
```
df3['Species']=Encoding.fit_transform(df3['Species'])
```

In [77]:
```
df3
```

Out[77]:

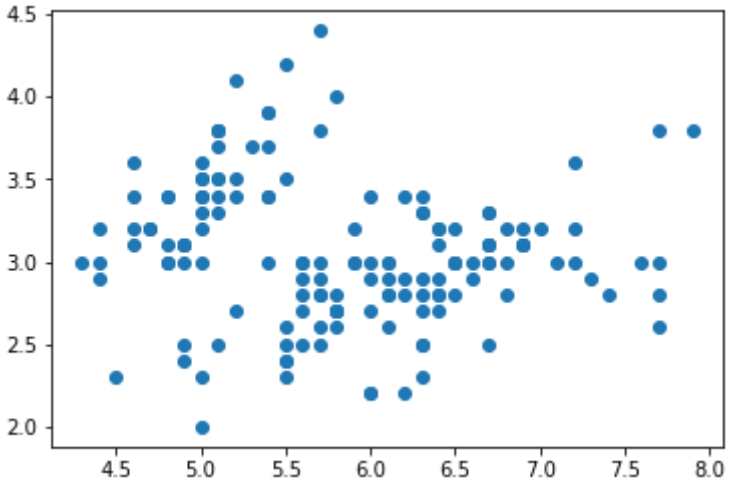
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	0
1	2	4.9	3.0	1.4	0.2	0
2	3	4.7	3.2	1.3	0.2	0
3	4	4.6	3.1	1.5	0.2	0
4	5	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	2
146	147	6.3	2.5	5.0	1.9	2
147	148	6.5	3.0	5.2	2.0	2
148	149	6.2	3.4	5.4	2.3	2
149	150	5.9	3.0	5.1	1.8	2

150 rows × 6 columns

In [78]:
```
import matplotlib.pyplot as plt
```

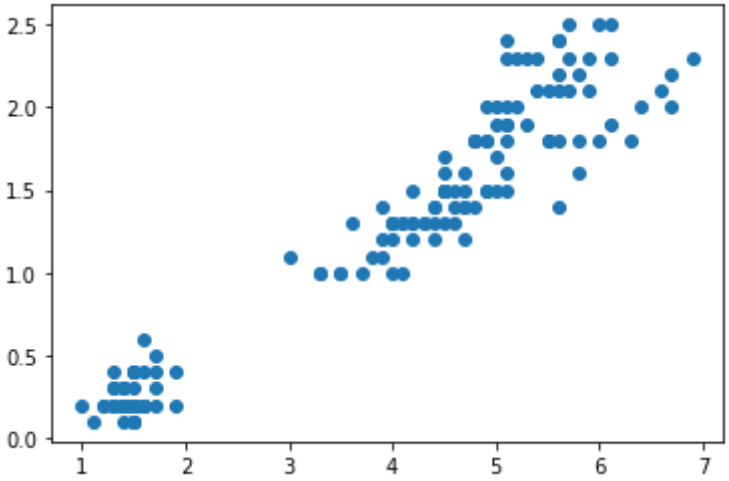
In [79]:
```
plt.scatter(df3['SepalLengthCm'],df3['SepalWidthCm'])
```

Out[79]:
```
<matplotlib.collections.PathCollection at 0x1f80e073820>
```



In [80]:
```
plt.scatter(df3['PetalLengthCm'],df3['PetalWidthCm'])
```

Out[80]:
```
<matplotlib.collections.PathCollection at 0x1f80e0d0a30>
```



Adding Missing Values

In [51]:
```
df6=pd.read_csv("auto.csv")
```

In [52]:
```
df6
```

Out[52]:

	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	mpg
0	8	307.0	130.0	3504.0	12.0	70	1	18.0
1	8	350.0	165.0	3693.0	11.5	70	1	15.0
2	8	318.0	150.0	3436.0	11.0	70	1	18.0
3	8	304.0	150.0	3433.0	12.0	70	1	16.0
4	8	302.0	140.0	3449.0	10.5	70	1	17.0
...	...	...	...	...	...	...	...	...
387	4	140.0	86.0	2790.0	15.6	82	1	27.0
388	4	97.0	52.0	2130.0	24.6	82	2	44.0
389	4	135.0	84.0	2295.0	11.6	82	1	32.0
390	4	120.0	79.0	2625.0	18.6	82	1	28.0

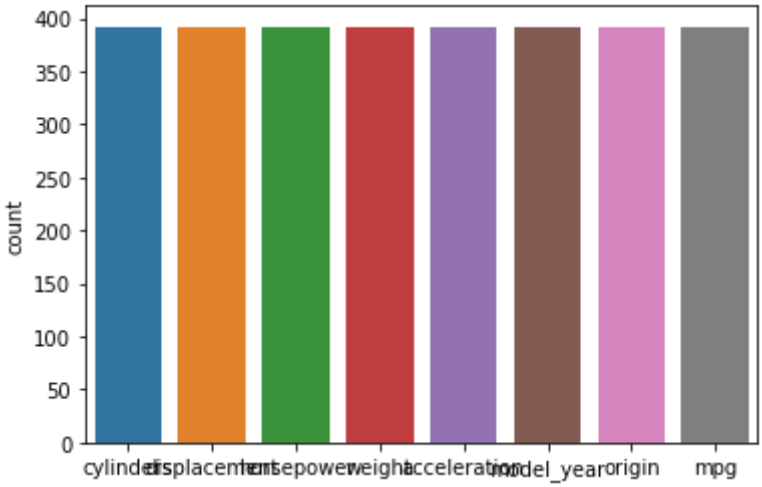
	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	mpg
391	4	119.0	82.0	2720.0	19.4	82	1	31.0

392 rows × 8 columns

In [55]:
```
import seaborn as sns
```

In [57]:
```
sns.countplot(data=df6)
```

Out[57]: <AxesSubplot:ylabel='count'>



Categorical Values into Numerical Values

In [1]:
```
import pandas as pd
```

In [3]:
```
df5=pd.read_csv("PlayTennis.csv")
df5
```

Out[3]:

	outlook	temp	humidity	windy	play
0	sunny	hot	high	False	no
1	sunny	hot	high	True	no
2	overcast	hot	high	False	yes
3	rainy	mild	high	False	yes
4	rainy	cool	normal	False	yes
5	rainy	cool	normal	True	no
6	overcast	cool	normal	True	yes
7	sunny	mild	high	False	no
8	sunny	cool	normal	False	yes
9	rainy	mild	normal	False	yes
10	sunny	mild	normal	True	yes
11	overcast	mild	high	True	yes
12	overcast	hot	normal	False	yes
13	rainy	mild	high	True	no

In [4]:
```
from sklearn.preprocessing import LabelEncoder
```

In [5]:
```
encoder=LabelEncoder()
```

In [6]:
```
df5['outlook']=encoder.fit_transform(df5['outlook'])
df5['temp']=encoder.fit_transform(df5['temp'])
df5['humidity']=encoder.fit_transform(df5['humidity'])
df5['windy']=encoder.fit_transform(df5['windy'])
df5['play']=encoder.fit_transform(df5['play'])
```

In [7]:
```
df5
```

Out[7]:

	outlook	temp	humidity	windy	play
0	2	1	0	0	0
1	2	1	0	1	0
2	0	1	0	0	1
3	1	2	0	0	1
4	1	0	1	0	1
5	1	0	1	1	0
6	0	0	1	1	1
7	2	2	0	0	0
8	2	0	1	0	1
9	1	2	1	0	1
10	2	2	1	1	1
11	0	2	0	1	1
12	0	1	1	0	1
13	1	2	0	1	0

REGRESSION

Simple Linear Regression

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

In [6]:
```
dataset = pd.read_csv('Salary_Data.csv')
x= dataset.iloc[:,:-1].values
y= dataset.iloc[:,-1].values
print(y)
```

```
[ 39343  46205  37731  43525  39891  56642  60150  54445  64445  57189
  63218  55794  56957  57081  61111  67938  66029  83088  81363  93940
  91738  98273 101302 113812 109431 105582 116969 112635 122391 121872]
```

In [8]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=1/3,random_state=0)
```

In [13]:
```
from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(x_train,y_train)
```

Out[13]:
```
LinearRegression()
```

In [14]:
```
y_pred=regressor.predict(x_test)
```

In [15]:
```
plt.scatter(x_train,y_train,color = 'red')
plt.plot(x_train,regressor.predict(x_train),color='blue')
plt.title('Salary vs Experience (Training set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```

In [16]:
```
plt.scatter(x_test,y_test,color = 'red')
plt.plot(x_test,regressor.predict(x_test),color='blue')
plt.title('Salary vs Experience (Testing set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```

Multiple Linear Regression

In [4]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

```
import numpy as np
```

In [14]:
```
dataset = pd.read_csv('50_Startups.csv')
dataset.drop(dataset[['State']],axis=1,inplace=True)
x= dataset[['R&D Spend','Administration','Marketing Spend']]
y= dataset[['Profit']]
print(y)
#dataset
#dataset
```

```
      Profit
0   192261.83
1   191792.06
2   191050.39
3   182901.99
4   166187.94
5   156991.12
6   156122.51
7   155752.60
8   152211.77
9   149759.96
10  146121.95
11  144259.40
12  141585.52
13  134307.35
14  132602.65
15  129917.04
16  126992.93
17  125370.37
18  124266.90
19  122776.86
20  118474.03
21  111313.02
22  110352.25
23  108733.99
24  108552.04
25  107404.34
26  105733.54
27  105008.31
28  103282.38
29  101004.64
30   99937.59
31   97483.56
32   97427.84
33   96778.92
34   96712.80
35   96479.51
36   90708.19
37   89949.14
38   81229.06
39   81005.76
40   78239.91
41   77798.83
42   71498.49
43   69758.98
44   65200.33
45   64926.08
46   49490.75
47   42559.73
48   35673.41
49   14681.40
```

In [15]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)
```

In [16]:
```
from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(x_train,y_train)
```

Out[16]:
```
LinearRegression()
```

In [18]:
```
y_pred=regressor.predict(x_test)
```

In [27]:
```
from sklearn.metrics import r2_score
```

In [25]:
```
print(r2_score(y_test,y_pred))
```

```
0.9355188337118217
```

Polynomial Regression

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
dataset=pd.read_csv("Position_Salaries.csv")
```

In [3]:
```
dataset
```

Out[3]:

	Position	Level	Salary
0	Business Analyst	1	45000
1	Junior Consultant	2	50000
2	Senior Consultant	3	60000
3	Manager	4	80000
4	Country Manager	5	110000
5	Region Manager	6	150000
6	Partner	7	200000
7	Senior Partner	8	300000
8	C-level	9	500000
9	CEO	10	1000000

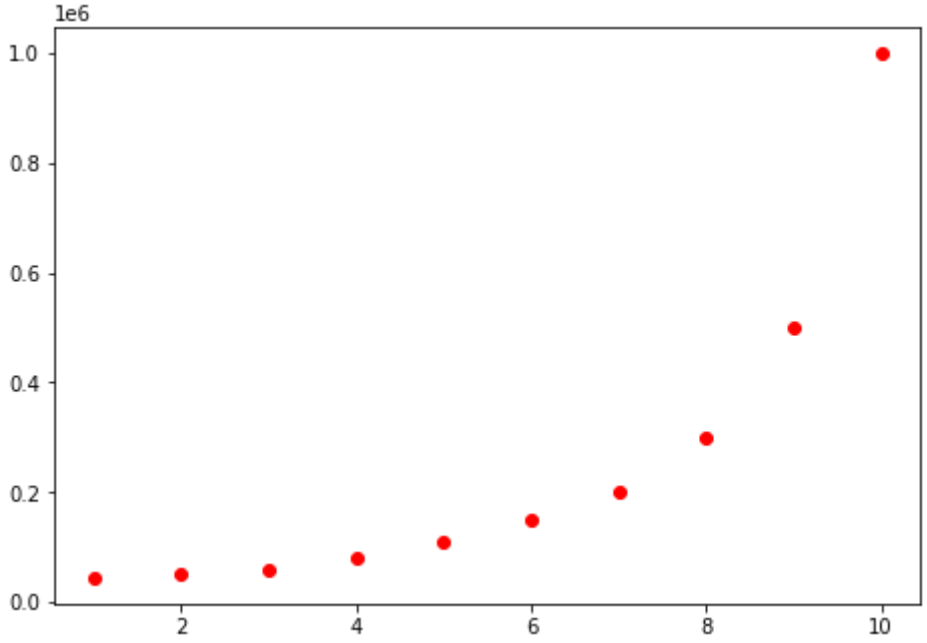
In [4]:
```
x=dataset.iloc[:,1:2].values
y=dataset.iloc[:,2:].values
```

In [5]:
```
x
```

Out[5]:
```
array([[ 1],
       [ 2],
       [ 3],
       [ 4],
       [ 5],
       [ 6],
       [ 7],
       [ 8],
       [ 9],
       [10]], dtype=int64)
```

In [6]:
```
fig=plt.figure()
ax=fig.add_axes([0,0,1,1])
ax.scatter(x,y,color='r')
```

Out[6]: <matplotlib.collections.PathCollection at 0x2291d7ef1c0>



In [7]:
```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
```

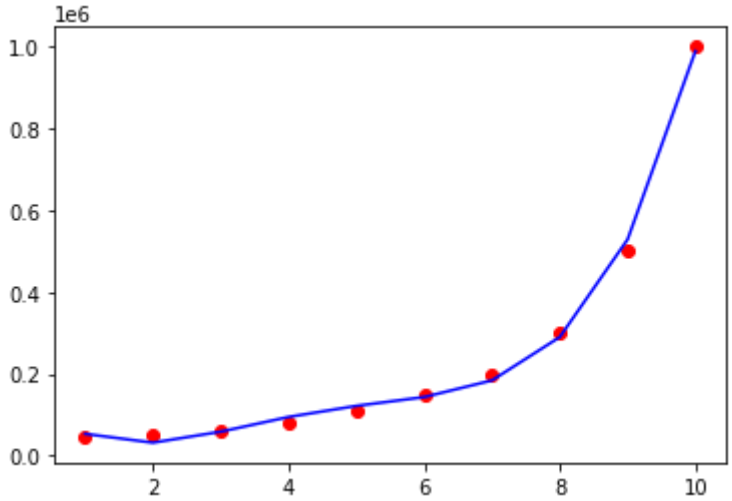
In [8]:
```
poly=PolynomialFeatures(degree=4)
x_poly=poly.fit_transform(x)
```

In [9]:
```
pilreg=LinearRegression()
pilreg.fit(x_poly,y)
```

Out[9]: LinearRegression()

In [10]:
```
plt.scatter(x,y,color="red")
plt.plot(x,pilreg.predict(poly.fit_transform(x)),color="blue")
```

Out[10]: [<matplotlib.lines.Line2D at 0x2291ff77c10>]



In [11]:
```
pilreg.predict(poly.fit_transform([[10]]))
```

Out[11]: array([[988916.08391594]])

CLASSIFIACTION

K Nearest Neighbors Classification Algorithm

In [30]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [33]:
```
df1=pd.read_csv('Social_Network_Ads.csv')
```

In [34]:
```
df1.head()
```

Out[34]:

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0

In [43]:
```
x=df1.iloc[:,:2]
y=df1.iloc[:,-1]
```

In [50]:
```
from sklearn.model_selection import train_test_split
```

In [51]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)
```

In [98]:
```
neigh=KNeighborsClassifier(n_neighbors=5)
```

In [99]:
```
neigh.fit(x_train,y_train)
```

Out[99]:
```
KNeighborsClassifier()
```

In [100…
```
y_pred=neigh.predict(x_test)
```

In [101…
```
from sklearn.metrics import accuracy_score,mean_squared_error
```

In [102…
```
print(mean_squared_error(y_test,y_pred))
```

```
0.18333333333333332
```

In [103…
```
print(accuracy_score(y_test,y_pred))
```

```
0.8166666666666667
```

Logistic Regression Classification Model

In [83]:
```
from sklearn.linear_model import LogisticRegression
```

In [84]:
```
df4=pd.read_csv("Social_Network_Ads.csv")
```

In [85]:
```
df4.head()
```

Out[85]:

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0

	Age	EstimatedSalary	Purchased
2	26	43000	0
3	27	57000	0
4	19	76000	0

In [86]:
```
x1=df4[['Age','EstimatedSalary']]
y1=df4[['Purchased']]
```

In [90]:
```
from sklearn.model_selection import train_test_split
x_train1,x_test1,y_train1,y_test1=train_test_split(x1,y1,test_size=0.3,random_state=
```

In [91]:
```
from sklearn.preprocessing import StandardScaler
```

In [93]:
```
scaled=StandardScaler()
x_train_scale=scaled.fit_transform(x_train1)
```

In [94]:
```
x_test_scale=scaled.transform(x_test1)
```

In [95]:
```
logistic=LogisticRegression()
logistic.fit(x_train_scale,y_train1)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConve
rsionWarning: A column-vector y was passed when a 1d array was expected. Please chan
ge the shape of y to (n_samples, ), for example using ravel().
  return f(**kwargs)
```

Out[95]:
```
LogisticRegression()
```

In [96]:
```
y_pred=logistic.predict(x_test_scale)
```

In [97]:
```
from sklearn.metrics import accuracy_score,confusion_matrix
```

In [99]:
```
print(accuracy_score(y_test1,y_pred))
```

```
0.8666666666666667
```

In [100…
```
print(confusion_matrix(y_test1,y_pred))
```

```
[[74  5]
 [11 30]]
```

Decision Tree Classification Model

In [6]:
```
df2=pd.read_csv("PlayTennis.csv")
df2.head()
```

Out[6]:

	outlook	temp	humidity	windy	play
0	sunny	hot	high	False	no
1	sunny	hot	high	True	no
2	overcast	hot	high	False	yes
3	rainy	mild	high	False	yes
4	rainy	cool	normal	False	yes

In [9]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14 entries, 0 to 13
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   outlook   14 non-null     object
 1   temp      14 non-null     object
 2   humidity  14 non-null     object
 3   windy     14 non-null     bool  
 4   play      14 non-null     object
dtypes: bool(1), object(4)
memory usage: 590.0+ bytes
```

In [10]:
```
df2.isnull().sum()
```

Out[10]:
```
outlook     0
temp        0
humidity    0
windy       0
play        0
dtype: int64
```

In [11]:
```
from sklearn.preprocessing import LabelEncoder
```

In [12]:
```
Encoding=LabelEncoder()
```

In [15]:
```
df2['outlook']=Encoding.fit_transform(df2['outlook'])
df2['temp']=Encoding.fit_transform(df2['temp'])
df2['humidity']=Encoding.fit_transform(df2['humidity'])
df2['windy']=Encoding.fit_transform(df2['windy'])
df2['play']=Encoding.fit_transform(df2['play'])
```

In [16]:
```
df2
```

Out[16]:

	outlook	temp	humidity	windy	play
0	2	1	0	0	0
1	2	1	0	1	0
2	0	1	0	0	1
3	1	2	0	0	1
4	1	0	1	0	1
5	1	0	1	1	0
6	0	0	1	1	1
7	2	2	0	0	0
8	2	0	1	0	1
9	1	2	1	0	1
10	2	2	1	1	1
11	0	2	0	1	1
12	0	1	1	0	1
13	1	2	0	1	0

In [20]:
```
x=df2[['outlook','temp','humidity','windy']]
```

```
y=df2[['play']]
```

In [24]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)
```

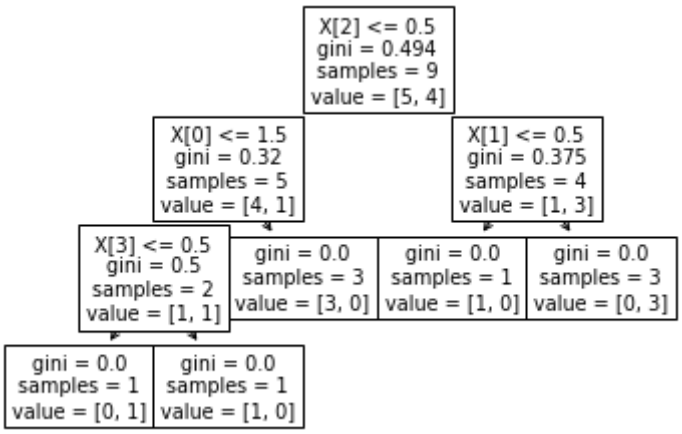
In [59]:
```
from sklearn import tree
dtree=tree.DecisionTreeClassifier(random_state=1234)
```

In [60]:
```
dtree.fit(x_train,y_train)
```

Out[60]:
```
DecisionTreeClassifier(random_state=1234)
```

In [61]:
```
tree.plot_tree(dtree)
```

Out[61]:
```
[Text(186.0, 190.26, 'X[2] <= 0.5\ngini = 0.494\nsamples = 9\nvalue = [5, 4]'),
 Text(111.60000000000001, 135.9, 'X[0] <= 1.5\ngini = 0.32\nsamples = 5\nvalue = [4,
1]'),
 Text(74.4, 81.53999999999999, 'X[3] <= 0.5\ngini = 0.5\nsamples = 2\nvalue = [1,
1]'),
 Text(37.2, 27.180000000000007, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(111.60000000000001, 27.180000000000007, 'gini = 0.0\nsamples = 1\nvalue = [1,
0]'),
 Text(148.8, 81.53999999999999, 'gini = 0.0\nsamples = 3\nvalue = [3, 0]'),
 Text(260.40000000000003, 135.9, 'X[1] <= 0.5\ngini = 0.375\nsamples = 4\nvalue =
[1, 3]'),
 Text(223.20000000000002, 81.53999999999999, 'gini = 0.0\nsamples = 1\nvalue = [1,
0]'),
 Text(297.6, 81.53999999999999, 'gini = 0.0\nsamples = 3\nvalue = [0, 3]')]
```



In [62]:
```
#!pip install graphviz
#import graphviz
#dot_data = tree.export_graphviz(dtree, out_file=None)
#graph = graphviz.Source(dot_data)
#graph
```

In [63]:
```
y_pred=dtree.predict(x_train)
```

In [64]:
```
y_pred
```

Out[64]:
```
array([0, 1, 0, 0, 1, 1, 0, 0, 1])
```

In [65]:
```
y_pred1=dtree.predict(x_test)
```

In [66]:
```
y_pred1
```

Out[66]:
```
array([0, 0, 0, 0, 1])
```

In [67]:
```
y_train
```

Out[67]:

	play
13	0
9	1
1	0
7	0
10	1
3	1
0	0
5	0
12	1

In [68]:
```
y_pred
```

Out[68]:
```
array([0, 1, 0, 0, 1, 1, 0, 0, 1])
```

In [69]:
```
y_test
```

Out[69]:

	play
8	1
6	1
4	1
11	1
2	1

In [70]:
```
y_pred1
```

Out[70]:
```
array([0, 0, 0, 0, 1])
```

In [71]:
```
from sklearn.metrics import accuracy_score,confusion_matrix
```

In [72]:
```
print(accuracy_score(y_test,y_pred1))
```

```
0.2
```

In [82]:
```
print(confusion_matrix(y_test,y_pred1))
```

```
[[0 0]
 [4 1]]
```

Support Vector Machine Classification Algorithm

In [12]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [13]:
```
dataset=pd.read_csv('Position_Salaries.csv')
```

In [14]:
```
dataset
```

Out[14]:

	Position	Level	Salary
0	Business Analyst	1	45000
1	Junior Consultant	2	50000
2	Senior Consultant	3	60000
3	Manager	4	80000
4	Country Manager	5	110000
5	Region Manager	6	150000
6	Partner	7	200000
7	Senior Partner	8	300000
8	C-level	9	500000
9	CEO	10	1000000

In [15]:
```
x=dataset.iloc[:,1:2].values
y=dataset.iloc[:,2:].values
```

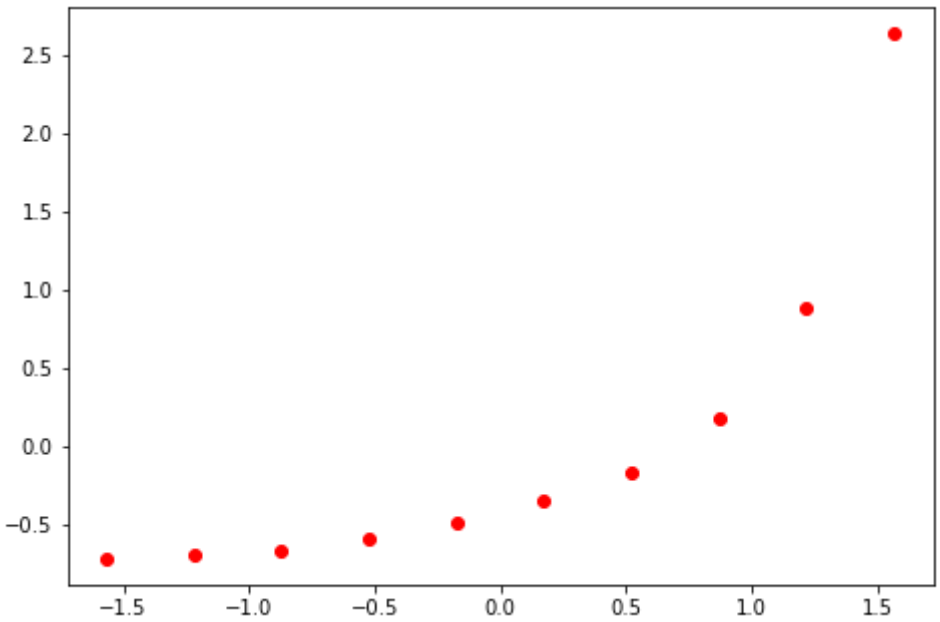
In [16]:
```
from sklearn.preprocessing import StandardScaler
```

In [17]:
```
st_x=StandardScaler()
st_y=StandardScaler()
```

In [18]:
```
X=st_x.fit_transform(x)
Y=st_y.fit_transform(y)
```

In [19]:
```
fig=plt.figure()
ax=fig.add_axes([0,0,1,1])
ax.scatter(X,Y,color='r')
```

Out[19]:
```
<matplotlib.collections.PathCollection at 0x2291ffe3790>
```



In [20]:
```
from sklearn.svm import SVR
```

In [21]:
```
regressor=SVR(kernel='rbf')
```

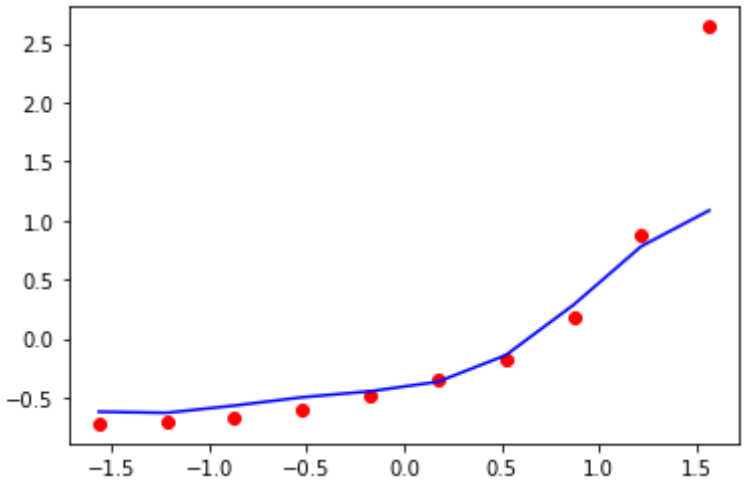
In [22]:
```
regressor.fit(X,Y)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConve
rsionWarning: A column-vector y was passed when a 1d array was expected. Please chan
ge the shape of y to (n_samples, ), for example using ravel().
  return f(**kwargs)
```

Out[22]:
```
SVR()
```

In [23]:
```
plt.scatter(X,Y,color='red')
plt.plot(X,regressor.predict(X),color='blue')
```

Out[23]:
```
[<matplotlib.lines.Line2D at 0x2292005beb0>]
```



Random Forest Classification Model

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
df2=pd.read_csv('PlayTennis.csv')
```

In [3]:
```
df2
```

Out[3]:

	outlook	temp	humidity	windy	play
0	sunny	hot	high	False	no
1	sunny	hot	high	True	no
2	overcast	hot	high	False	yes
3	rainy	mild	high	False	yes
4	rainy	cool	normal	False	yes
5	rainy	cool	normal	True	no
6	overcast	cool	normal	True	yes
7	sunny	mild	high	False	no
8	sunny	cool	normal	False	yes
9	rainy	mild	normal	False	yes

	outlook	temp	humidity	windy	play
10	sunny	mild	normal	True	yes
11	overcast	mild	high	True	yes
12	overcast	hot	normal	False	yes
13	rainy	mild	high	True	no

In [4]:
```
from sklearn.preprocessing import LabelEncoder
```

In [5]:
```
Encoding=LabelEncoder()
```

In [6]:
```
df2['outlook']=Encoding.fit_transform(df2['outlook'])
df2['temp']=Encoding.fit_transform(df2['temp'])
df2['humidity']=Encoding.fit_transform(df2['humidity'])
df2['windy']=Encoding.fit_transform(df2['windy'])
df2['play']=Encoding.fit_transform(df2['play'])
```

In [7]:
```
df2
```

Out[7]:

	outlook	temp	humidity	windy	play
0	2	1	0	0	0
1	2	1	0	1	0
2	0	1	0	0	1
3	1	2	0	0	1
4	1	0	1	0	1
5	1	0	1	1	0
6	0	0	1	1	1
7	2	2	0	0	0
8	2	0	1	0	1
9	1	2	1	0	1
10	2	2	1	1	1
11	0	2	0	1	1
12	0	1	1	0	1
13	1	2	0	1	0

In [11]:
```
x=df2[['outlook','temp','humidity','windy']]
```

In [12]:
```
y=df2[['play']]
```

In [8]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [9]:
```
from sklearn.model_selection import train_test_split
```

In [26]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=150)
```

In [39]:
```
RandomF=RandomForestClassifier(n_estimators=100)
```

In [40]:
```
RandomF.fit(x_train,y_train)
```

```
<ipython-input-40-da23e1510298>:1: DataConversionWarning: A column-vector y was pass
ed when a 1d array was expected. Please change the shape of y to (n_samples,), for e
xample using ravel().
  RandomF.fit(x_train,y_train)
```

Out[40]:
```
RandomForestClassifier()
```

In [41]:
```
y_pred2=RandomF.predict(x_test)
```

In [42]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
```

In [43]:
```
print(accuracy_score(y_test,y_pred2))
```

```
0.8
```

In [44]:
```
print(confusion_matrix(y_test,y_pred2))
```

```
[[1 0]
 [1 3]]
```

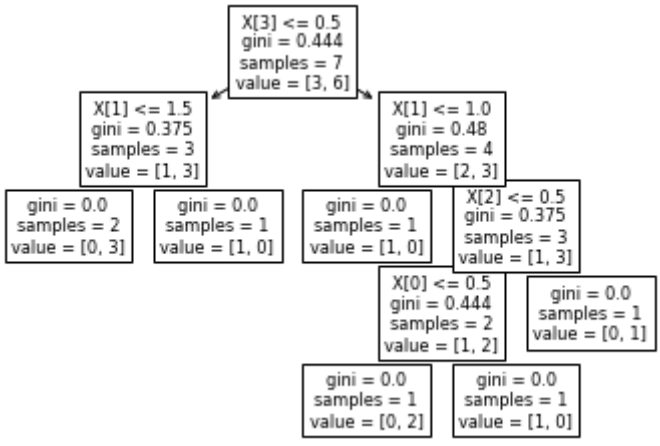
In [45]:
```
from sklearn import tree
```

In [46]:
```
RandomF
```

Out[46]:
```
RandomForestClassifier()
```

In [50]:
```
tree.plot_tree(RandomF.estimators_[2])
```

Out[50]:
```
[Text(148.8, 195.696, 'X[3] <= 0.5\ngini = 0.444\nsamples = 7\nvalue = [3, 6]'),
 Text(74.4, 152.208, 'X[1] <= 1.5\ngini = 0.375\nsamples = 3\nvalue = [1, 3]'),
 Text(37.2, 108.72, 'gini = 0.0\nsamples = 2\nvalue = [0, 3]'),
 Text(111.60000000000001, 108.72, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(223.20000000000002, 152.208, 'X[1] <= 1.0\ngini = 0.48\nsamples = 4\nvalue =
[2, 3]'),
 Text(186.0, 108.72, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(260.40000000000003, 108.72, 'X[2] <= 0.5\ngini = 0.375\nsamples = 3\nvalue =
[1, 3]'),
 Text(223.20000000000002, 65.232, 'X[0] <= 0.5\ngini = 0.444\nsamples = 2\nvalue =
[1, 2]'),
 Text(186.0, 21.744, 'gini = 0.0\nsamples = 1\nvalue = [0, 2]'),
 Text(260.40000000000003, 21.744, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(297.6, 65.232, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]')]
```



Naive Bayes Classifier

In [1]:
```
import pandas as pd
```

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
df6=pd.read_csv('PlayTennis.csv')
```

In [3]:
```
df6.head()
```

Out[3]:

	outlook	temp	humidity	windy	play
0	sunny	hot	high	False	no
1	sunny	hot	high	True	no
2	overcast	hot	high	False	yes
3	rainy	mild	high	False	yes
4	rainy	cool	normal	False	yes

In [4]:
```
from sklearn.preprocessing import LabelEncoder
```

In [5]:
```
encod=LabelEncoder()
```

In [6]:
```
df6['outlook']=encod.fit_transform(df6['outlook'])
df6['temp']=encod.fit_transform(df6['temp'])
df6['humidity']=encod.fit_transform(df6['humidity'])
df6['windy']=encod.fit_transform(df6['windy'])
df6['play']=encod.fit_transform(df6['play'])
```

In [7]:
```
df6
```

Out[7]:

	outlook	temp	humidity	windy	play
0	2	1	0	0	0
1	2	1	0	1	0
2	0	1	0	0	1
3	1	2	0	0	1
4	1	0	1	0	1
5	1	0	1	1	0
6	0	0	1	1	1
7	2	2	0	0	0
8	2	0	1	0	1
9	1	2	1	0	1
10	2	2	1	1	1
11	0	2	0	1	1
12	0	1	1	0	1
13	1	2	0	1	0

In [15]:
```
x3=df6.iloc[:,:-1]
y3=df6.iloc[:,-1]
```

In [17]:
```
from sklearn.model_selection import train_test_split
```

In [34]:
```
x_train,x_test,y_train,y_test=train_test_split(x3,y3,test_size=0.3,random_state=50)
```

In [35]:
```
from sklearn.naive_bayes import GaussianNB
```

In [36]:
```
gaussian=GaussianNB()
```

In [37]:
```
gaussian.fit(x_train,y_train)
```

Out[37]:
```
GaussianNB()
```

In [38]:
```
y_pred4=gaussian.predict(x_test)
```

In [39]:
```
y_pred4
```

Out[39]:
```
array([1, 0, 1, 0, 1])
```

In [40]:
```
from sklearn.metrics import accuracy_score,confusion_matrix
```

In [41]:
```
print(accuracy_score(y_test,y_pred4)*100)
```

```
60.0
```

In [43]:
```
print(confusion_matrix(y_test,y_pred4))
```

```
[[0 0]
 [2 3]]
```

CLUSTERING

KMeans Clustering Partitional Clustering

In [47]:
```
df7=pd.read_csv("Mall_Customers.csv")
```

In [48]:
```
df7.head()
```

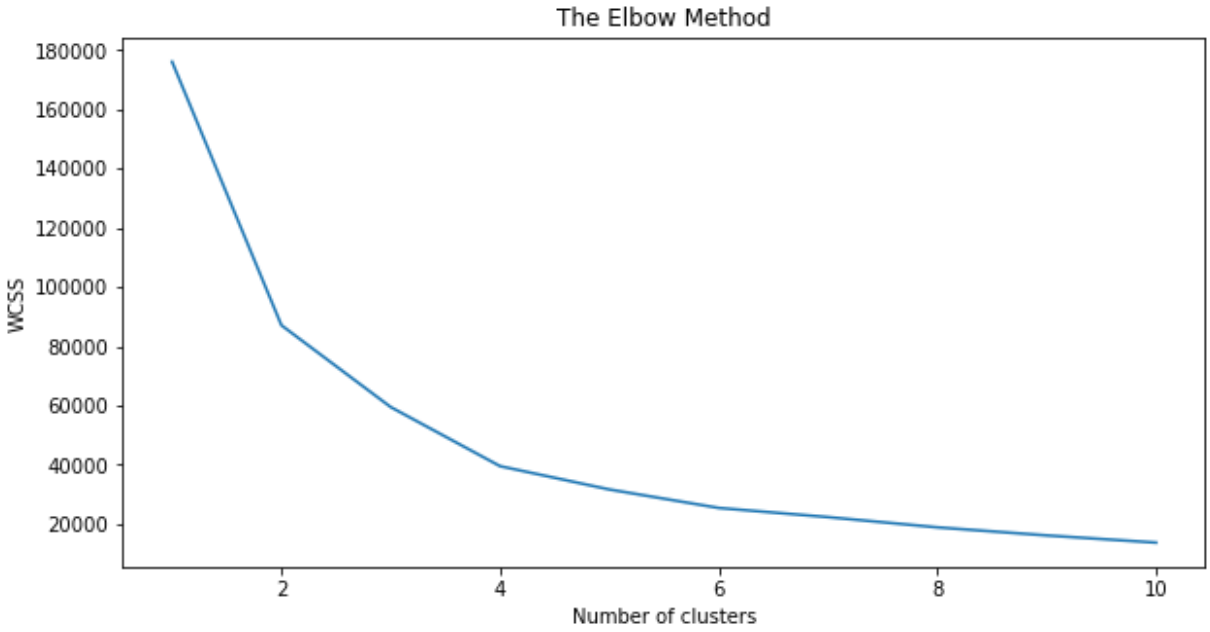
Out[48]:

	CustomerID	Genre	Age	Annual_Income_(k$)	Spending_Score
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

In [49]:
```
X = df7.iloc[:, [2, 3]].values
```

In [51]:
```
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
```

In [52]:
```
plt.figure(figsize=(10,5))
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



In [53]:
```
kmeans=KMeans(n_clusters=5,init='k-means++',random_state=0)
```

In [54]:
```
kmeans.fit(X)
```

Out[54]:
```
KMeans(n_clusters=5, random_state=0)
```

In [65]:
```
y_means=kmeans.predict(X)
```

In [ ]:
```

```

Agglomerative Hierarchical Clustering

In [56]:
```
df8=pd.read_csv("Mall_Customers.csv")
```

In [57]:
```
df8.head()
```

Out[57]:

	CustomerID	Genre	Age	Annual_Income_(k$)	Spending_Score
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

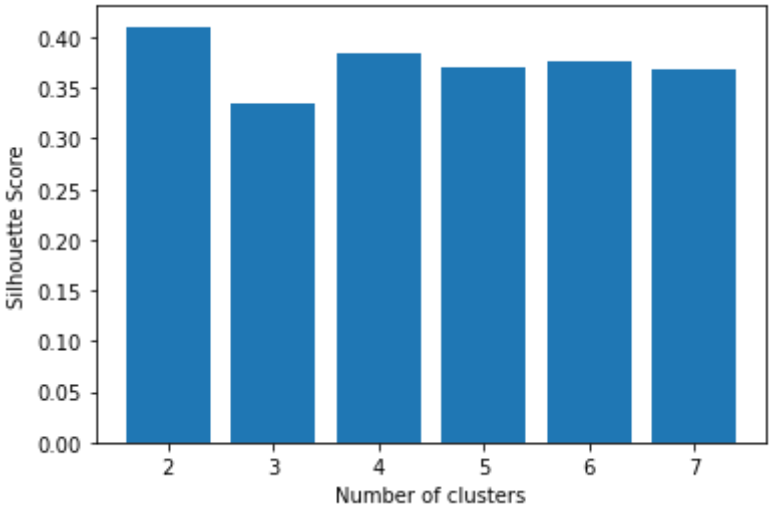
In [60]:
```
X4=df8.iloc[:,[2,3]].values
```

In [63]:
```
from sklearn.metrics import silhouette_score
from sklearn.cluster import AgglomerativeClustering
```

In [64]:
```
silhouette_scores = []

for n_cluster in range(2, 8):
    silhouette_scores.append(silhouette_score(X4, AgglomerativeClustering(n_clusters

k = [2, 3, 4, 5, 6,7]
plt.bar(k, silhouette_scores)
plt.xlabel('Number of clusters', fontsize = 10)
plt.ylabel('Silhouette Score', fontsize = 10)
plt.show()
```



In [69]:
```
agglomerative=AgglomerativeClustering(n_clusters=2)
```

In [70]:
```
agglomerative.fit(X4)
```

Out[70]:
```
AgglomerativeClustering()
```

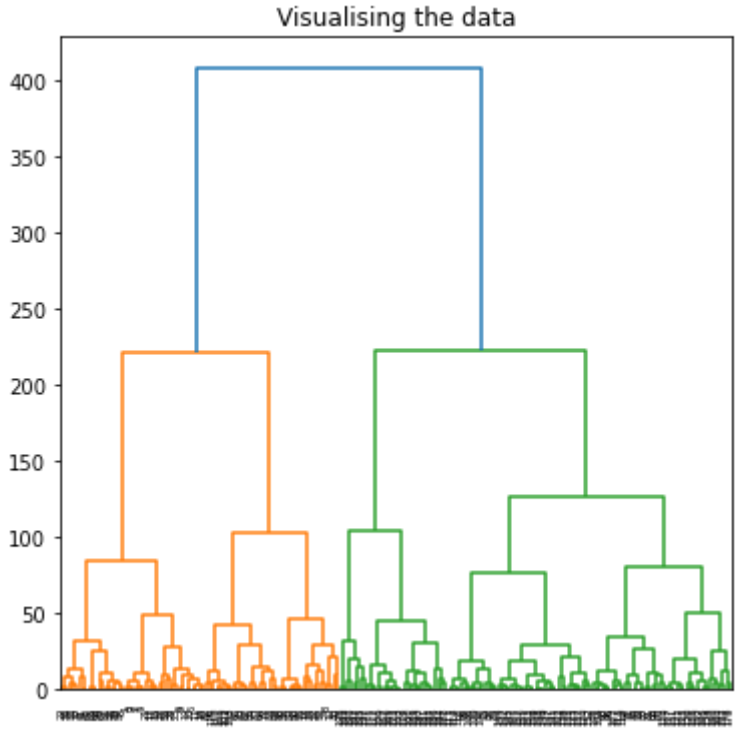
In [73]:
```
y_agglo=agglomerative.fit_predict(X4)
```

In [74]:
```
y_agglo
```

Out[74]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1,
       1, 0, 0, 0, 0, 0, 1, 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, 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], dtype=int64)
```

In [75]:
```
import scipy.cluster.hierarchy as shc
```

In [77]:
```
plt.figure(figsize =(6, 6))
plt.title('Visualising the data')
Dendrogram = shc.dendrogram((shc.linkage(X4, method ='ward')))
```



Density Based Spatial Clustering Of Application with Noise

In [78]:
```
df9=pd.read_csv("Mall_Customers.csv")
```

In [79]:
```
df9.head()
```

Out[79]:

	CustomerID	Genre	Age	Annual_Income_(k$)	Spending_Score
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

In [80]:
```
X5=df9.iloc[:,[2,3]].values
```

In [81]:
```
from sklearn.cluster import DBSCAN
```

In [85]:
```
dbscan=DBSCAN(eps=3,min_samples=4,metric="euclidean")
```

In [88]:
```
model=dbscan.fit(X5)
```

In [89]:
```
dbscan.fit_predict(X5)
```

Out[89]:
```
array([ 0,  0,  0,  0, -1,  0,  1,  0, -1, -1, -1,  1, -1,  0,  1,  0,  1,
        0, -1,  1,  1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
       -1, -1, -1, -1, -1, -1, -1, -1,  2,  3,  2, -1,  2,  3,  3,  3,  2,
        3,  3, -1,  2,  2,  2,  4, -1,  2,  4, -1,  4, -1, -1, -1, -1,  4,
       -1, -1,  4, -1, -1, -1, -1, -1,  5, -1, -1,  5, -1, -1, -1,  5, -1,
        5, -1, -1, -1,  6, -1,  7,  6, -1, -1,  7,  6,  7,  6,  7,  7,  6,
        8,  7,  6,  7,  8, -1,  8,  8,  8,  7,  9,  7,  7,  7,  8,  6,  6,
        6, -1,  9,  9,  9, -1, 10,  9,  9, -1,  9, -1,  9, -1, 10, -1, 10,
```

```
       -1, 10, -1, -1, -1, 10, 10, 10, 10, 10, -1, 10, 10, 10, -1, 10, -1,
       10, -1, 10, 10, 10, 10, 10, -1, 10, -1, 10, -1, 11, -1, -1, 11, 12,
       -1, 12, 11, 11, -1, 12, -1, 12, -1, -1, -1, 13, -1, 13, -1, 13, -1,
       13, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1], dtype=int64)
```

Principle Component Analysis

In [24]:
```
from sklearn.datasets import load_breast_cancer
```

In [29]:
```
cancer=load_breast_cancer()
```

In [31]:
```
cancer.keys()
```

Out[31]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'fil
ename'])
```

In [32]:
```
print(cancer['DESCR'])
```

```
.. _breast_cancer_dataset:

Breast cancer wisconsin (diagnostic) dataset
--------------------------------------------

**Data Set Characteristics:**

    :Number of Instances: 569

    :Number of Attributes: 30 numeric, predictive attributes and the class

    :Attribute Information:
        - radius (mean of distances from center to points on the perimeter)
        - texture (standard deviation of gray-scale values)
        - perimeter
        - area
        - smoothness (local variation in radius lengths)
        - compactness (perimeter^2 / area - 1.0)
        - concavity (severity of concave portions of the contour)
        - concave points (number of concave portions of the contour)
        - symmetry
        - fractal dimension ("coastline approximation" - 1)

        The mean, standard error, and "worst" or largest (mean of the three
        worst/largest values) of these features were computed for each image,
        resulting in 30 features.  For instance, field 0 is Mean Radius, field
        10 is Radius SE, field 20 is Worst Radius.

        - class:
                - WDBC-Malignant
                - WDBC-Benign

    :Summary Statistics:

    ===================================== ====== ======
                                           Min    Max
    ===================================== ====== ======
    radius (mean):                        6.981  28.11
    texture (mean):                       9.71   39.28
    perimeter (mean):                     43.79  188.5
    area (mean):                          143.5  2501.0
    smoothness (mean):                    0.053  0.163
    compactness (mean):                   0.019  0.345
    concavity (mean):                     0.0    0.427
    concave points (mean):                0.0    0.201
    symmetry (mean):                      0.106  0.304
    fractal dimension (mean):             0.05   0.097
    radius (standard error):              0.112  2.873
    texture (standard error):             0.36   4.885
```

```
    perimeter (standard error):           0.757  21.98
    area (standard error):                6.802  542.2
    smoothness (standard error):          0.002  0.031
    compactness (standard error):         0.002  0.135
    concavity (standard error):           0.0    0.396
    concave points (standard error):      0.0    0.053
    symmetry (standard error):            0.008  0.079
    fractal dimension (standard error):   0.001  0.03
    radius (worst):                       7.93   36.04
    texture (worst):                      12.02  49.54
    perimeter (worst):                    50.41  251.2
    area (worst):                         185.2  4254.0
    smoothness (worst):                   0.071  0.223
    compactness (worst):                  0.027  1.058
    concavity (worst):                    0.0    1.252
    concave points (worst):               0.0    0.291
    symmetry (worst):                     0.156  0.664
    fractal dimension (worst):            0.055  0.208
    ==================================== ====== ======

    :Missing Attribute Values: None

    :Class Distribution: 212 - Malignant, 357 - Benign

    :Creator:  Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian

    :Donor: Nick Street

    :Date: November, 1995

This is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.
https://goo.gl/U2Uwz2

Features are computed from a digitized image of a fine needle
aspirate (FNA) of a breast mass.  They describe
characteristics of the cell nuclei present in the image.

Separating plane described above was obtained using
Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree
Construction Via Linear Programming." Proceedings of the 4th
Midwest Artificial Intelligence and Cognitive Science Society,
pp. 97-101, 1992], a classification method which uses linear
programming to construct a decision tree.  Relevant features
were selected using an exhaustive search in the space of 1-4
features and 1-3 separating planes.

The actual linear program used to obtain the separating plane
in the 3-dimensional space is that described in:
[K. P. Bennett and O. L. Mangasarian: "Robust Linear
Programming Discrimination of Two Linearly Inseparable Sets",
Optimization Methods and Software 1, 1992, 23-34].

This database is also available through the UW CS ftp server:

ftp ftp.cs.wisc.edu
cd math-prog/cpo-dataset/machine-learn/WDBC/

.. topic:: References

   - W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction
     for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on
     Electronic Imaging: Science and Technology, volume 1905, pages 861-870,
     San Jose, CA, 1993.
   - O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and
     prognosis via linear programming. Operations Research, 43(4), pages 570-577,
     July-August 1995.
   - W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning techniques
     to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77 (1994)
     163-171.
```

In [33]:
```
df = pd.DataFrame(cancer['data'],columns=cancer['feature_names'])
```

In [34]:
```
df.head()
```

Out[34]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	di
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	

5 rows × 30 columns

In [35]:
```
from sklearn.preprocessing import StandardScaler
```

In [36]:
```
scaler = StandardScaler()
scaler.fit(df)
```

Out[36]:
```
StandardScaler()
```

In [37]:
```
scaled_data = scaler.transform(df)
```

In [38]:
```
from sklearn.decomposition import PCA
```

In [39]:
```
pca = PCA(n_components=2)
```

In [40]:
```
pca.fit(scaled_data)
```

Out[40]:
```
PCA(n_components=2)
```

In [41]:
```
x_pca = pca.transform(scaled_data)
```

In [42]:
```
scaled_data.shape
```

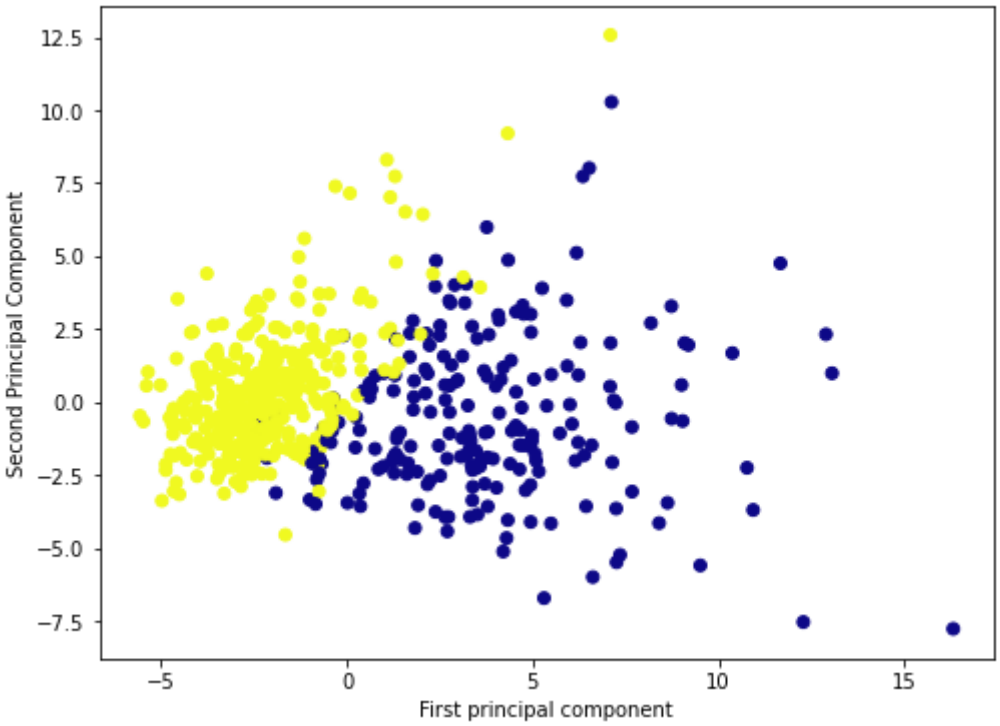
Out[42]:
```
(569, 30)
```

In [43]:
```
x_pca.shape
```

Out[43]:
```
(569, 2)
```

In [44]:
```
plt.figure(figsize=(8,6))
plt.scatter(x_pca[:,0],x_pca[:,1],c=cancer['target'],cmap='plasma')
plt.xlabel('First principal component')
plt.ylabel('Second Principal Component')
```

Out[44]:
```
Text(0, 0.5, 'Second Principal Component')
```


12.5
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
Second Principal Component
−5
0
5
10
15
First principal component

In [ ]: