In [136…

```python
import pandas as pd
import numpy as np
```

In [137…

```python
data=pd.read_csv('team.csv')
```

In [138…

```python
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
dfle=data
dfle.TEAM=le.fit_transform(dfle.TEAM)
dfle
```

Out[138]:

	TEAM	YEAR
0	0	2000
1	1	2002
2	2	2003
3	3	2004
4	0	2005
5	2	2006
6	1	2007
7	0	2008
8	3	2009

In [139…

```python
from sklearn.preprocessing import OneHotEncoder
```

In [140…

```python
enc=OneHotEncoder()
enc_data=pd.DataFrame(enc.fit_transform(dfle[['TEAM']]).toarray())
enc_data
```

Out[140]:

	0	1	2	3
0	1.0	0.0	0.0	0.0
1	0.0	1.0	0.0	0.0
2	0.0	0.0	1.0	0.0
3	0.0	0.0	0.0	1.0
4	1.0	0.0	0.0	0.0
5	0.0	0.0	1.0	0.0
6	0.0	1.0	0.0	0.0
7	1.0	0.0	0.0	0.0
8	0.0	0.0	0.0	1.0

In [142…

```python
abc=dfle.join(enc_data)
```

In [143…

```python
final=abc.drop(['TEAM'],axis='columns')
final
```

Out[143]:

	YEAR	0	1	2	3
0	2000	1.0	0.0	0.0	0.0
1	2002	0.0	1.0	0.0	0.0
2	2003	0.0	0.0	1.0	0.0
3	2004	0.0	0.0	0.0	1.0
4	2005	1.0	0.0	0.0	0.0
5	2006	0.0	0.0	1.0	0.0
6	2007	0.0	1.0	0.0	0.0
7	2008	1.0	0.0	0.0	0.0
8	2009	0.0	0.0	0.0	1.0

In [ ]:

In [1]:

```
import pandas as pd
import numpy as np
```

In [2]:

```
data=pd.read_csv('abcde.csv')
```

In [4]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(data[['age']],data.results,test_size=0.5,random_state=10)
```

In [5]:

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(x_train,y_train)
```

Out[5]:

```
LogisticRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [6]:

```
y_pred=model.predict(x_test)
```

In [7]:

```
model.score(x_test,y_test)
```

Out[7]:

```
0.9285714285714286
```

In [ ]:

In [1]:
```
Write Python program to implement classifier using support vector machines.
1) Installation packages , Loading dataset, Show data frame , Create confusion_matrix
2) Describe accuracy_score, precision_score, recall_score, f1_score using confusion matrix
```

In [2]:
```
from sklearn.datasets import load_iris
iris=load_iris()
```

In [7]:
```
df=pd.DataFrame(iris.data,columns=iris.feature_names)
```

In [8]:
```
df['target']=iris.target
df.head()
```

Out[8]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [11]:
```
x=df.drop(['target'],axis='columns')
y=df.target
```

In [12]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

In [13]:
```
from sklearn.svm import SVC
clr=SVC()
clr.fit(x_train,y_train)
```

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

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

In [14]:
```
clr.score(x_test,y_test)
```

Out[14]:
```
1.0
```

In [30]:
```
y_pred=clr.predict(x_test)
y_pred
```

Out[30]:
```
array([2, 1, 0, 2, 0, 2, 0, 1, 1, 1, 2, 1, 1, 1, 1, 0, 1, 1, 0, 0, 2, 1,
       0, 0, 2, 0, 0, 1, 1, 0])
```

In [31]:
```
from sklearn.metrics import classification_report
classification_report(y_test,y_pred)
```

Out[31]:
```
'              precision    recall  f1-score   support\n\n           0       1.00      1.00      1.00
11\n           1       1.00      1.00      1.00        13\n           2       1.00      1.00      1.00
6\n\n    accuracy                           1.00        30\n   macro avg       1.00      1.00      1.00
30\nweighted avg       1.00      1.00      1.00        30\n'
```

In [ ]:
```
```

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:

```
from sklearn.datasets import load_iris
iris=load_iris()
```

In [4]:

```
df=pd.DataFrame(iris.data,columns=iris.feature_names)
df['target']=iris.target
```

In [5]:

```
x=df.drop(['target'],axis=1)
y=df['target']
```

In [6]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=2)
```

In [7]:

```
from sklearn.svm import SVC
clr=SVC()
clr.fit(x_train,y_train)
```

Out[7]:

```
SVC()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [8]:

```
y_train_pred=clr.predict(x_train)
y_test_pred=clr.predict(x_test)
```

In [15]:

```
from sklearn.metrics import confusion_matrix
train_cm=confusion_matrix(y_train,y_train_pred)
train_cm
```

Out[15]:

```
array([[36,  0,  0],
       [ 0, 41,  1],
       [ 0,  2, 40]], dtype=int64)
```

In [17]:

```
test_cm=confusion_matrix(y_test,y_test_pred)
test_cm
```

Out[17]:

```
array([[14,  0,  0],
       [ 0,  7,  1],
       [ 0,  0,  8]], dtype=int64)
```

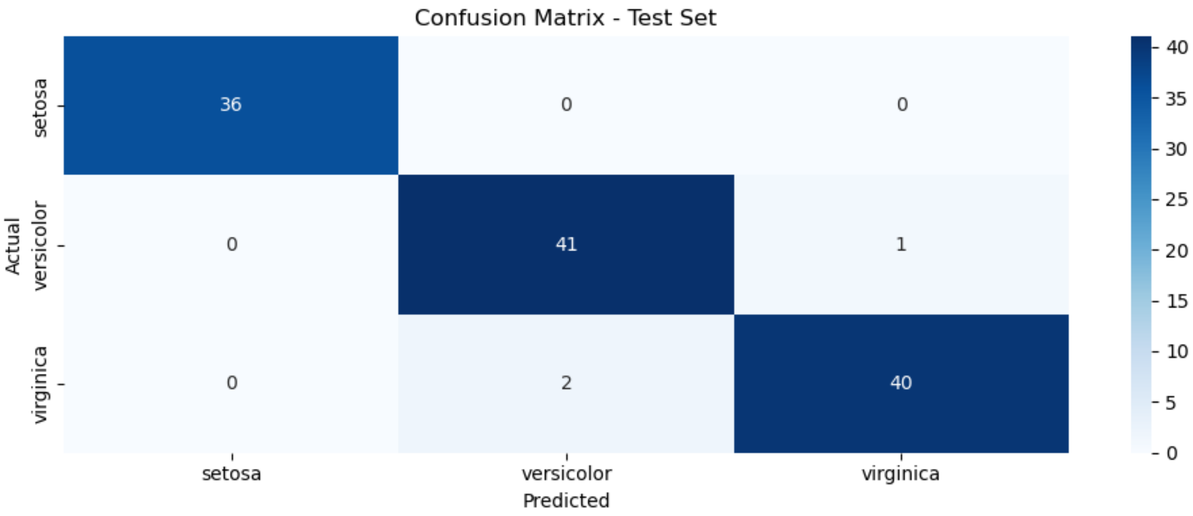
In [31]:

```
#Visualize_Confusion_matrix
import seaborn as sns

plt.figure(figsize=(10,4))
sns.heatmap(train_cm,annot=True,fmt='d',cmap='Blues',
          xticklabels=iris.target_names,
          yticklabels=iris.target_names)

plt.title('Confusion Matrix - Test Set')
plt.xlabel('Predicted')
plt.ylabel('Actual')

plt.tight_layout()
plt.show()
```



In [33]:

```
from sklearn.metrics import classification_report
classification_report(y_test,y_test_pred)
```

Out[33]:

```
'              precision    recall  f1-score   support\n\n           0       1.00      1.00      1.00
14\n           1       1.00      0.88      0.93         8\n           2       0.89      1.00      0.94
8\n\n    accuracy                           0.97        30\n   macro avg       0.96      0.96      0.9
6        30\nweighted avg       0.97      0.97      0.97        30\n'
```

In [ ]:

```
In [2]:  import pandas as pd
         import numpy as np
```

```
In [3]:  from sklearn.datasets import load_iris
         data=load_iris()
```

```
In [4]:  data.feature_names

Out[4]:  ['sepal length (cm)',
          'sepal width (cm)',
          'petal length (cm)',
          'petal width (cm)']
```

```
In [5]:  data.target_names

Out[5]:  array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

```
In [7]:  x=data.data
         y=data.target
```

```
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=0.2,random_state=10)
```

```
In [9]:  from sklearn.tree import DecisionTreeClassifier
         sc=DecisionTreeClassifier(criterion='entropy')
         sc.fit(x_train,y_train)

Out[9]:  DecisionTreeClassifier(criterion='entropy')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [10]: y_pred=sc.predict(x_test)
```

```
In [16]: from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
         confusion_matrix(y_test,y_pred)

Out[16]: array([[10,  0,  0],
                [ 0, 11,  2],
                [ 0,  0,  7]], dtype=int64)
```

```
In [17]: accuracy_score(y_test,y_pred)

Out[17]: 0.9333333333333333
```

```
In [18]: classification_report(y_test,y_pred)

Out[18]: '              precision    recall  f1-score   support\n\n           0       1.00      1.00      1.00
         10\n           1       1.00      0.85      0.92        13\n           2       0.78      1.00      0.88
         7\n\n    accuracy                           0.93        30\n   macro avg       0.93      0.95      0.93
         30\nweighted avg       0.95      0.93      0.93        30\n'
```

```
In [ ]:
```

In [1]:

```
import numpy as np
import pandas as pd
```

In [2]:

```
from sklearn.datasets import load_breast_cancer
data=load_breast_cancer()
```

In [3]:

```
data.target_names
```

Out[3]:

```
array(['malignant', 'benign'], dtype='<U9')
```

In [6]:

```
df=pd.DataFrame(np.c_[data.data,data.target],columns=[list(data.feature_names)+['target']])
```

In [8]:

```
df.shape
```

Out[8]:

```
(569, 31)
```

In [10]:

```
x=df.iloc[:,0:-1]
y=df.iloc[:,-1]
```

In [13]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=1)
```

In [14]:

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
xtrain=sc.fit_transform(x_train)
xtest=sc.transform(x_test)
```

In [15]:

```
from sklearn.naive_bayes import GaussianNB
clr=GaussianNB()
clr.fit(xtrain,y_train)
```

Out[15]:

```
GaussianNB()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [16]:

```
y_pred=clr.predict(xtest)
```

In [17]:

```
y_pred
```

Out[17]:

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

In [19]:

```
from sklearn.metrics import accuracy_score
print('Accuracy :',accuracy_score(y_test,y_pred))
```

```
Accuracy : 0.9473684210526315
```

In [ ]:

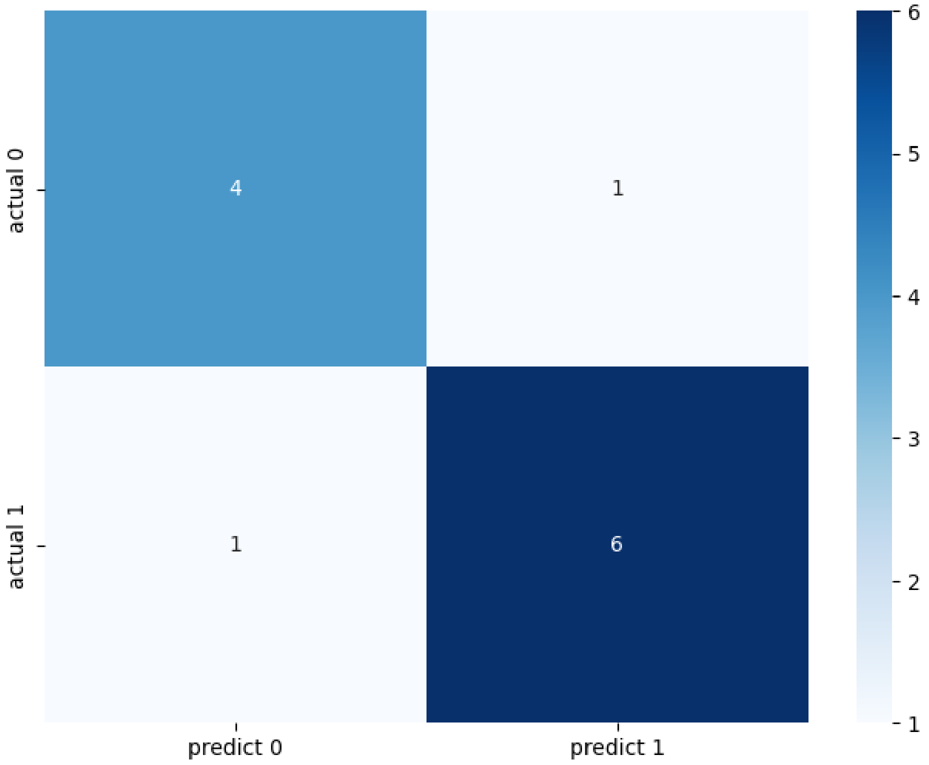
```
In [1]: from sklearn.metrics import confusion_matrix,classification_report
        import matplotlib.pyplot as plt
        import seaborn as sns
```

```
In [2]: actual=[1,0,1,0,1,0,1,0,1,1,1,0]
        predict=[1,0,1,0,1,0,1,1,1,1,0,0]
```

```
In [4]: cm=confusion_matrix(actual,predict)
```

```
In [10]: plt.figure(figsize=(8,6))
         sns.heatmap(cm,annot=True,fmt='d',cmap='Blues',
                    xticklabels=['predict 0','predict 1'],
                    yticklabels=['actual 0','actual 1'])
```

```
Out[10]: <Axes: >
```



```
In [12]: classification_report(actual,predict)
```

```
Out[12]: '              precision    recall  f1-score   support\n\n           0       0.80      0.80      0.80
         5\n           1       0.86      0.86      0.86         7\n\n    accuracy                           0.83
         12\n   macro avg       0.83      0.83      0.83        12\nweighted avg       0.83      0.83      0.83
         12\n'
```

```
In [ ]:
```

In [1]:
```
#Random_Forest
import pandas as pd
import numpy as np
from sklearn import metrics
```

In [2]:
```
data=pd.read_csv('user_data.csv')
```

In [3]:
```
x=data.iloc[:,[2,3]].values
y=data.iloc[:,4].values
```

In [4]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=2)
```

In [5]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.transform(x_test)
```

In [6]:
```
data
```

Out[6]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
...	...	...	...	...	...
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

400 rows × 5 columns

In [7]:
```
from sklearn.ensemble import RandomForestClassifier
clr=RandomForestClassifier(n_estimators=10,criterion='entropy')
clr.fit(x_train,y_train)
```

Out[7]:
```
RandomForestClassifier(criterion='entropy', n_estimators=10)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [9]:
```
y_pred=clr.predict(x_test)
y_pred
```

Out[9]:
```
array([0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1,
       1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0], dtype=int64)
```

In [12]:
```
from sklearn.metrics import confusion_matrix,classification_report
cm=confusion_matrix(y_test,y_pred)
cm
```

Out[12]:
```
array([[43,  5],
       [ 6, 26]], dtype=int64)
```

```
In [13]: from sklearn.metrics import classification_report
         classification_report(y_test,y_pred)
```

```
Out[13]: '              precision    recall  f1-score   support\n\n           0       0.88      0.90      0.89
         48\n           1       0.84      0.81      0.83        32\n\n    accuracy                           0.
         86        80\n   macro avg       0.86      0.85      0.86        80\nweighted avg       0.86      0.86
         0.86        80\n'
```

```
In [ ]: #Visualization Not Possible The Program Going To Confusion
```

In [1]:

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from matplotlib import pyplot as plt
```

In [2]:

```
df=pd.read_csv('Book1.csv')
df.head()
```

Out[2]:

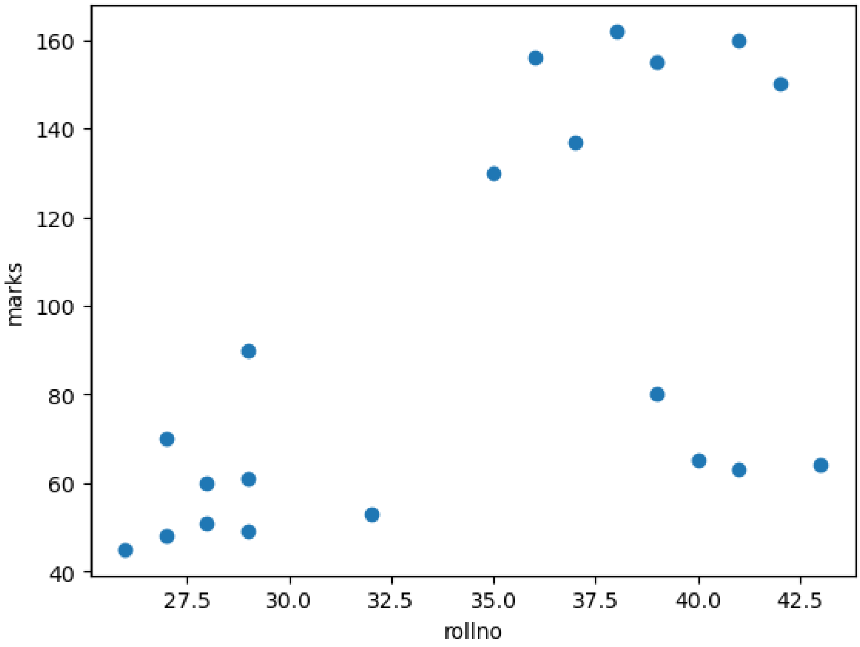
	name	rollno	marks
0	A	40	65
1	B	41	63
2	C	43	64
3	D	39	80
4	E	36	156

In [3]:

```
plt.scatter(df.rollno,df['marks'])
plt.xlabel('rollno')
plt.ylabel('marks')
```

Out[3]:

```
Text(0, 0.5, 'marks')
```



In [5]:

```
from sklearn.cluster import KMeans
km=KMeans(n_clusters=3)
pred=km.fit_predict(df[['rollno','marks']])
pred
```

```
C:\Users\Admin\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default
value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppre
ss the warning
  warnings.warn(
C:\Users\Admin\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is kno
wn to have a memory leak on Windows with MKL, when there are less chunks than available threads. You c
an avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[5]:

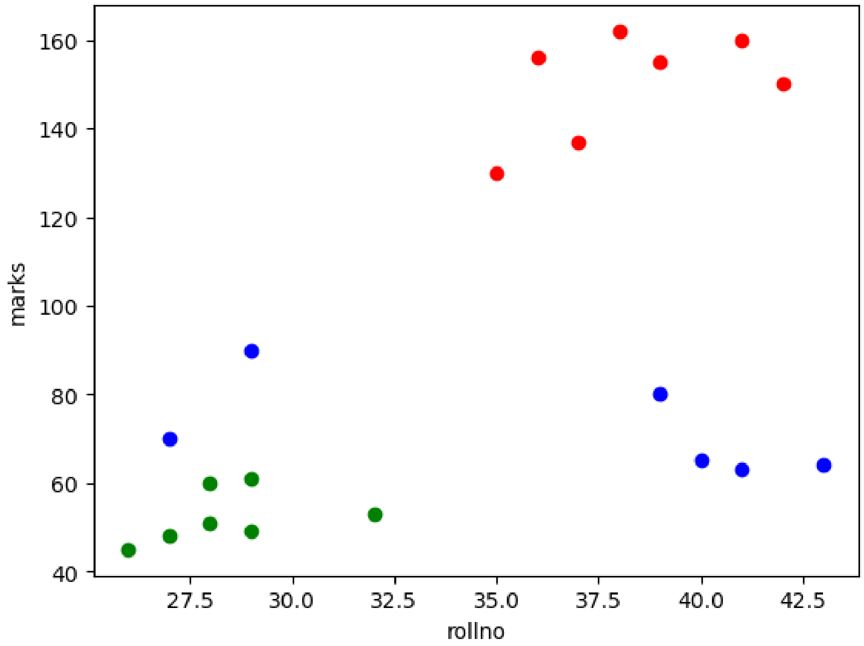
```
array([2, 2, 2, 2, 1, 1, 1, 0, 0, 0, 0, 0, 2, 2, 0, 0, 1, 1, 1, 1])
```

In [7]:

```
df['cluster']=pred
df.head()
df1=df[df.cluster==0]
df2=df[df.cluster==1]
df3=df[df.cluster==2]
plt.scatter(df1.rollno,df1['marks'],color='green')
plt.scatter(df2.rollno,df2['marks'],color='red')
plt.scatter(df3.rollno,df3['marks'],color='blue')
plt.xlabel('rollno')
plt.ylabel('marks')
```

Out[7]:

```
Text(0, 0.5, 'marks')
```



In [9]:

```
scale=MinMaxScaler()
scale.fit(df[['marks']])
df['marks']=scale.transform(df[['marks']])

scale.fit(df[['rollno']])
df['rollno']=scale.transform(df[['rollno']])
```

In [13]:

```
df=df.drop(['cluster'],axis='columns')
df['cluster']=pred
df.head()
```

Out[13]:

	name	rollno	marks	cluster
0	A	0.823529	0.170940	2
1	B	0.882353	0.153846	2
2	C	1.000000	0.162393	2
3	D	0.764706	0.299145	2
4	E	0.588235	0.948718	1

In [14]:

```
km.cluster_centers_
```

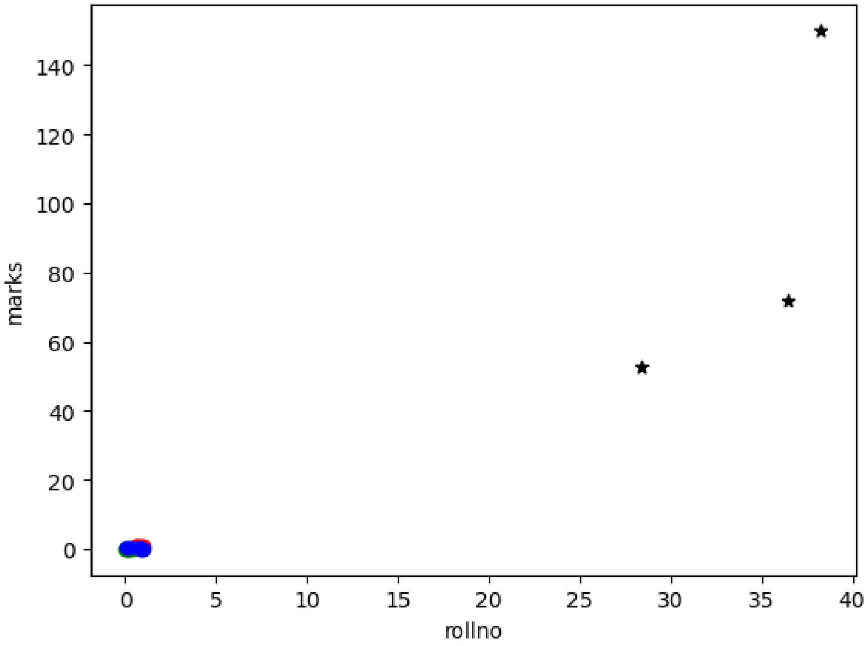
Out[14]:

```
array([[ 28.42857143,  52.42857143],
       [ 38.28571429, 150.        ],
       [ 36.5       ,  72.        ]])
```

In [15]:
```
plt.scatter(df1.rollno,df1['marks'],color='green')
plt.scatter(df2.rollno,df2['marks'],color='red')
plt.scatter(df3.rollno,df3['marks'],color='blue')

plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color='black',marker='*')
plt.xlabel('rollno')
plt.ylabel('marks')
```

Out[15]:
```
Text(0, 0.5, 'marks')
```



In [ ]:

In [1]:
```
#KNN
import pandas as pd
import numpy as np
```

In [2]:
```
data=pd.read_csv('User_Data.csv')
```

In [4]:
```
x=data.iloc[:,[2,3]].values
y=data.iloc[:,4].values
```

In [5]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=2)
```

In [6]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
xtrain=sc.fit_transform(x_train)
xtest=sc.transform(x_test)
```

In [8]:
```
from sklearn.neighbors import KNeighborsClassifier
clr=KNeighborsClassifier(n_neighbors=5)
clr.fit(xtrain,y_train)
```

Out[8]:
```
KNeighborsClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [9]:
```
y_pred=clr.predict(xtest)
y_pred
```

Out[9]:
```
array([0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1,
       1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0], dtype=int64)
```

In [10]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
cm=confusion_matrix(y_test,y_pred)
cm
```

Out[10]:
```
array([[46,  2],
       [ 7, 25]], dtype=int64)
```

In [11]:
```
classification_report(y_test,y_pred)
```

Out[11]:
```
'              precision    recall  f1-score   support\n\n           0       0.87      0.96      0.91
48\n           1       0.93      0.78      0.85        32\n\n    accuracy                           0.89
80\n   macro avg       0.90      0.87      0.88        80\nweighted avg       0.89      0.89      0.89
80\n'
```

In [12]:
```
accuracy_score(y_test,y_pred)
```

Out[12]:
```
0.8875
```

In [ ]:
```
#Visualization_Not_Possible
```