In [388]:
```
#RF
from sklearn import datasets
```

In [389]:
```
iris = datasets.load_iris()
```

In [390]:
```
print(iris.target_names)
```

```
['setosa' 'versicolor' 'virginica']
```

In [391]:
```
print(iris.feature_names)
```

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

In [392]:
```
print(iris.data[0:5])
```

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]]
```

In [393]:
```
print(iris.target)
```

```
[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 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 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

In [394]:
```
import pandas as pd
data=pd.DataFrame({
    'sepal length':iris.data[:,0],
    'sepal width':iris.data[:,1],
    'petal length':iris.data[:,2],
    'petal width':iris.data[:,3],
    'species':iris.target
})
data.head()
```

Out[394]:

	sepal length	sepal width	petal length	petal width	species
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 [395]:
```
from sklearn.model_selection import train_test_split
X=data[['sepal length', 'sepal width', 'petal length', 'petal width']]
y=data['species']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

In [396]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [397]:
```
clf=RandomForestClassifier(n_estimators=100)
```

In [398]:
```
clf.fit(X_train,y_train)

y_pred=clf.predict(X_test)
```

In [399]:
```
from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9777777777777777
```

In [400]:
```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[18  0  0]
 [ 0 14  1]
 [ 0  0 12]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        18
           1       1.00      0.93      0.97        15
           2       0.92      1.00      0.96        12

    accuracy                           0.98        45
   macro avg       0.97      0.98      0.98        45
weighted avg       0.98      0.98      0.98        45
```

In [401]:
```
clf.feature_importances_
```

Out[401]:
```
array([0.11200592, 0.0224508 , 0.41274349, 0.45279979])
```

In [402]:
```
clf.predict([[3, 5, 4, 2]])
```

Out[402]:
```
array([2])
```

In [403]:
```
from sklearn.ensemble import RandomForestClassifier
clf=RandomForestClassifier(n_estimators=100)
clf.fit(X_train,y_train)
```

Out[403]:
```
RandomForestClassifier()
```

In [404]:
```
import pandas as pd
data=pd.DataFrame({
    'sepal length':iris.data[:,0],
    'sepal width':iris.data[:,1],
    'petal length':iris.data[:,2],
    'petal width':iris.data[:,3],
    'species':iris.target
})
data.head()
```

Out[404]:

	sepal length	sepal width	petal length	petal width	species
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 [405]:
```
data.drop('sepal width',axis=1,inplace=True)
```

In [406]:
```
from sklearn.model_selection import train_test_split
X=data[['sepal length', 'petal length', 'petal width']]
y=data['species']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

In [407]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [408]:
```
clf=RandomForestClassifier(n_estimators=100)
```

In [409]:
```
clf.fit(X_train,y_train)

y_pred=clf.predict(X_test)
```

In [410]:
```
from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9555555555555556
```

In [ ]:

In [ ]:

In [411]:
```
#SVM
from sklearn import datasets
```

In [412]:
```
iris = datasets.load_iris()
```

In [413]:
```
print("Features: ",iris.feature_names)
```

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

In [414]:
```
print("labels: ",iris.target_names)
```

```
labels:  ['setosa' 'versicolor' 'virginica']
```

In [415]:
```
iris.data.shape
```

Out[415]:
```
(150, 4)
```

In [416]:
```
print(iris.target)
```

```
[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 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 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

In [417]:
```
from sklearn.model_selection import train_test_split
```

In [418]:
```
x_train,x_test,y_train,y_test=train_test_split(iris.data,iris.target,test_size=0.3,random_state=100)
```

In [419]:
```
from sklearn import svm
```

In [420]:
```
clf=svm.SVC(kernel='linear')
```

In [421]:
```
clf.fit(x_train,y_train)
```

Out[421]:
```
SVC(kernel='linear')
```

In [422]:
```
y_pred = clf.predict(x_test)
```

In [423]:
```
from sklearn import metrics
```

In [424]:
```
print("Accuracy:",metrics.accuracy_score(y_test,y_pred))
```

```
Accuracy: 1.0
```

In [425]:
```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[16  0  0]
 [ 0 11  0]
 [ 0  0 18]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       1.00      1.00      1.00        11
           2       1.00      1.00      1.00        18

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

In [426]:
```
clf=svm.SVC(kernel='poly')
```

In [427]:
```
clf.fit(x_train,y_train)
```

Out[427]:
```
SVC(kernel='poly')
```

In [428]:
```
y_pred = clf.predict(x_test)
```

In [429]:
```
from sklearn import metrics
```

In [430]:
```
print("Accuracy:",metrics.accuracy_score(y_test,y_pred))
```

```
Accuracy: 1.0
```

In [431]:
```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[16  0  0]
 [ 0 11  0]
 [ 0  0 18]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       1.00      1.00      1.00        11
           2       1.00      1.00      1.00        18

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

In [432]:
```
clf=svm.SVC(kernel='rbf')
```

In [433]:
```
clf.fit(x_train,y_train)
```

Out[433]:
```
SVC()
```

In [434]:
```
y_pred = clf.predict(x_test)
```

In [435]:
```
from sklearn import metrics
```

In [436]:
```
print("Accuracy:",metrics.accuracy_score(y_test,y_pred))
```

```
Accuracy: 0.9777777777777777
```

In [437]:
```
clf=svm.SVC(kernel='sigmoid')
```

In [438]:
```
clf.fit(x_train,y_train)
```

Out[438]:
```
SVC(kernel='sigmoid')
```

In [439]:
```
y_pred = clf.predict(x_test)
```

In [440]:
```
from sklearn import metrics
```

In [441]:
```
print("Accuracy:",metrics.accuracy_score(y_test,y_pred))
```

```
Accuracy: 0.24444444444444444
```

In [ ]:
```

```

In [442]:
```
#KNN
%cd C:\Users\manoj\Downloads
```

```
C:\Users\manoj\Downloads
```

In [443]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [444]:
```
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']
```

In [445]:
```
df=pd.read_csv('iris.data',names=names)
```

In [446]:
```
df.head()
```

Out[446]:

	sepal-length	sepal-width	petal-length	petal-width	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [447]:
```
X = df.iloc[:, :-1].values
y = df.iloc[:, 4].values
```

In [448]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
```

In [449]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

In [450]:
```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(X_train, y_train)
```

Out[450]:
```
KNeighborsClassifier()
```

In [451]:
```
y_pred = classifier.predict(X_test)
```

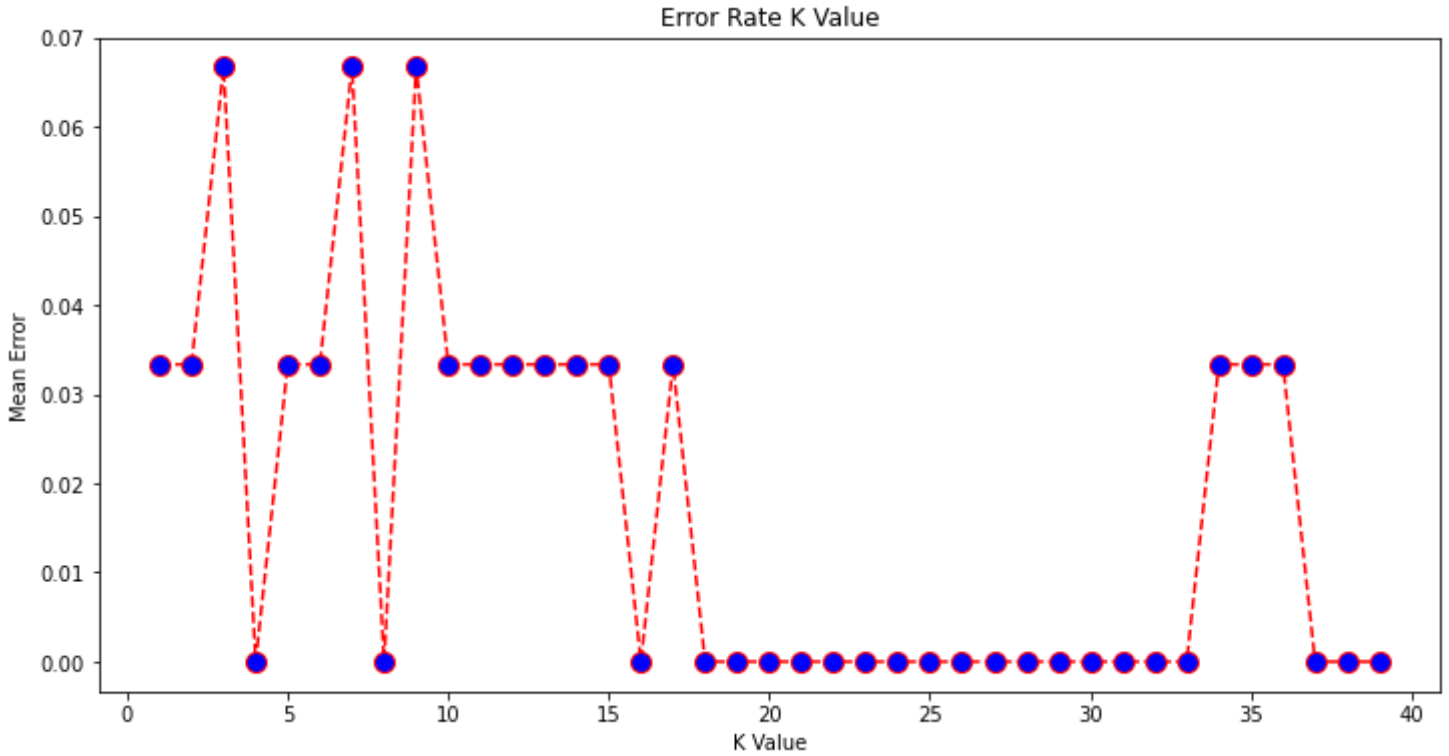
In [452]:
```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[15  0  0]
 [ 0  8  1]
 [ 0  0  6]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        15
Iris-versicolor       1.00      0.89      0.94         9
 Iris-virginica       0.86      1.00      0.92         6

       accuracy                           0.97        30
      macro avg       0.95      0.96      0.95        30
   weighted avg       0.97      0.97      0.97        30
```

In [453]:
```
error = []

for i in range(1, 40):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
    error.append(np.mean(pred_i != y_test))
```

In [454]:
```
plt.figure(figsize=(12, 6))
plt.plot(range(1, 40), error, color='red', linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)
plt.title('Error Rate K Value')
plt.xlabel('K Value')
plt.ylabel('Mean Error')
```

Out[454]:
```
Text(0, 0.5, 'Mean Error')
```



In [ ]:

In [455]:
```
#logistic regression
```

In [456]:
```
%cd C:\Users\manoj\Downloads
```

```
C:\Users\manoj\Downloads
```

In [457]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
import scipy.stats
from sklearn import datasets
```

In [458]:
```
iris = datasets.load_iris()
```

In [459]:
```
print(iris.target_names)
```

```
['setosa' 'versicolor' 'virginica']
```

In [460]:
```
print(iris.target)
```

```
[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 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 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

In [461]:
```
X = df.iloc[:, :-1].values
y = df.iloc[:, 4].values
```

In [462]:
```
print(X)
```

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]
 [5.4 3.7 1.5 0.2]
 [4.8 3.4 1.6 0.2]
 [4.8 3.  1.4 0.1]
 [4.3 3.  1.1 0.1]
 [5.8 4.  1.2 0.2]
 [5.7 4.4 1.5 0.4]
 [5.4 3.9 1.3 0.4]
 [5.1 3.5 1.4 0.3]
 [5.7 3.8 1.7 0.3]
```

In [463]:
```
print(y)
```

```
['Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica']
```

In [464]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=0)
```

In [465]:
```
from sklearn.linear_model import LogisticRegression
```

In [466]:
```
logreg = LogisticRegression(max_iter=300)
```

In [467]:
```
logreg.fit(X_train,y_train)
```

Out[467]:
```
LogisticRegression(max_iter=300)
```

In [468]:
```
y_pred=logreg.predict(X_test)
```

In [469]:
```
print(y_pred)
```

```
['Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica'
 'Iris-setosa' 'Iris-virginica' 'Iris-setosa' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
 'Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
 'Iris-virginica' 'Iris-setosa' 'Iris-setosa' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor'
 'Iris-setosa' 'Iris-virginica' 'Iris-virginica' 'Iris-versicolor'
 'Iris-setosa' 'Iris-virginica']
```

In [470]:
```
from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
```

Out[470]:
```
array([[13,  0,  0],
       [ 0, 15,  1],
       [ 0,  0,  9]], dtype=int64)
```

In [471]:
```
print(metrics.classification_report(y_test,y_pred))
```

```
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        13
Iris-versicolor       1.00      0.94      0.97        16
 Iris-virginica       0.90      1.00      0.95         9

       accuracy                           0.97        38
      macro avg       0.97      0.98      0.97        38
   weighted avg       0.98      0.97      0.97        38
```

In [472]:
```
metrics.accuracy_score(y_test, y_pred)
```

Out[472]:
```
0.9736842105263158
```

In [473]:
```
print(logreg.predict_proba(X_test))
```

```
[[1.16478077e-04 5.59628401e-02 9.43920682e-01]
 [1.26376707e-02 9.60278185e-01 2.70841441e-02]
 [9.84387613e-01 1.56123477e-02 3.88378201e-08]
 [1.25744218e-06 2.34270834e-02 9.76571659e-01]
 [9.70279266e-01 2.97205691e-02 1.64455659e-07]
 [2.00328904e-06 5.98055275e-03 9.94017444e-01]
 [9.81923016e-01 1.80769128e-02 7.13272693e-08]
 [2.83158385e-03 7.47763428e-01 2.49404988e-01]
 [1.50570932e-03 7.39107729e-01 2.59386561e-01]
 [2.04762669e-02 9.35792650e-01 4.37310832e-02]
 [9.19748652e-05 1.59958520e-01 8.39949505e-01]
 [6.95142099e-03 8.10311164e-01 1.82737415e-01]
 [4.06664325e-03 7.93738919e-01 2.02194438e-01]
 [3.04621096e-03 7.60982411e-01 2.35971378e-01]
 [3.85830361e-03 7.10393729e-01 2.85747967e-01]
 [9.63184099e-01 3.68157163e-02 1.84263613e-07]
 [6.69341906e-03 7.56235902e-01 2.37070679e-01]
 [1.13794845e-02 8.44613879e-01 1.44006637e-01]
 [9.67672211e-01 3.23275720e-02 2.17329571e-07]
 [9.82875353e-01 1.71245870e-02 6.02880814e-08]
 [8.25546315e-04 1.92707268e-01 8.06467185e-01]
 [1.02534582e-02 7.10744068e-01 2.79002474e-01]
 [9.44129748e-01 5.58691916e-02 1.06033506e-06]
 [9.75561224e-01 2.44386061e-02 1.70260602e-07]
 [1.36060797e-03 4.26132858e-01 5.72506534e-01]
 [9.94208723e-01 5.79126740e-03 9.79791572e-09]
 [9.50163249e-01 4.98356069e-02 1.14396390e-06]
 [1.06749570e-02 9.00956912e-01 8.83681307e-02]
 [1.40845324e-01 8.52830096e-01 6.32458014e-03]
 [9.61519237e-01 3.84803081e-02 4.55153666e-07]
 [9.87328143e-05 1.16212263e-01 8.83689004e-01]
 [1.18942725e-02 6.83789216e-01 3.04316511e-01]
 [9.68107030e-01 3.18928185e-02 1.51646882e-07]
 [1.27660022e-03 3.57889258e-01 6.40834142e-01]
 [1.47946665e-05 3.39058296e-02 9.66079376e-01]
 [4.79082237e-02 8.80380618e-01 7.17111585e-02]
 [9.44662182e-01 5.53374241e-02 3.93652725e-07]
 [5.99160749e-04 3.11074827e-01 6.88326012e-01]]
```

In [474]:
```
print(y_pred)
```

```
['Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica'
 'Iris-setosa' 'Iris-virginica' 'Iris-setosa' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
 'Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
 'Iris-virginica' 'Iris-setosa' 'Iris-setosa' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor'
 'Iris-setosa' 'Iris-virginica' 'Iris-virginica' 'Iris-versicolor'
 'Iris-setosa' 'Iris-virginica']
```

In [ ]:
```

```

In [ ]:
```

```

In [475]:
```
#Linear regression
```

In [476]:
```
%cd C:\Users\manoj\Downloads
```

```
C:\Users\manoj\Downloads
```

In [477]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [478]:
```
iris = datasets.load_iris()
```

In [479]:
```
print(iris.target_names)
```

```
['setosa' 'versicolor' 'virginica']
```

In [480]:
```
print(iris.feature_names)
```

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

In [481]:
```
import pandas as pd
data=pd.DataFrame({
    'sepal length':iris.data[:,0],
    'sepal width':iris.data[:,1],
    'petal length':iris.data[:,2],
    'petal width':iris.data[:,3],
    'species':iris.target
})
data.head()
```

Out[481]:

	sepal length	sepal width	petal length	petal width	species
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 [482]:
```
print(iris.target)
```

```
[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 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 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

In [483]:
```
from sklearn.model_selection import train_test_split
X=data[['sepal length', 'petal length', 'petal width']]
y=data['species']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

In [484]:
```
print(X)
```

```
     sepal length  petal length  petal width
0             5.1           1.4          0.2
1             4.9           1.4          0.2
2             4.7           1.3          0.2
3             4.6           1.5          0.2
4             5.0           1.4          0.2
..            ...           ...          ...
145           6.7           5.2          2.3
146           6.3           5.0          1.9
147           6.5           5.2          2.0
148           6.2           5.4          2.3
149           5.9           5.1          1.8

[150 rows x 3 columns]
```

In [485]:
```
print(y)
```

```
0      0
1      0
2      0
3      0
4      0
      ..
145    2
146    2
147    2
148    2
149    2
Name: species, Length: 150, dtype: int32
```

In [486]:
```
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 [487]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[487]:
```
LinearRegression()
```

In [488]:
```
print(regressor.intercept_)
```

```
0.1257143152925959
```

In [489]:
```
print(regressor.coef_)
```

```
[-0.13142356  0.25264791  0.58844661]
```

In [490]:
```
y_pred=regressor.predict(X_test)
```

In [491]:
```
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df
```

Out[491]:

	Actual	Predicted
114	2	2.064234
62	1	0.936211
33	0	-0.125719
107	2	1.817208
7	0	-0.034742
100	2	2.284750
40	0	-0.026427
86	1	1.315292
76	1	1.268569
71	1	1.099603
134	2	1.562684
51	1	1.304189
73	1	1.217612
54	1	1.316311
63	1	1.335301
37	0	-0.105709
78	1	1.356758
90	1	1.220671
45	0	0.025122
16	0	-0.020152
121	2	1.804610
66	1	1.409328

	Actual	Predicted
24	0	0.092602
8	0	0.018847
126	2	1.582802
22	0	-0.108497
44	0	0.170864
97	1	1.162255
93	1	0.890781
26	0	0.108212

In [492]:
```
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Mean Absolute Error: 0.186582317748203
Mean Squared Error: 0.05110989851762263
Root Mean Squared Error: 0.22607498428092976
```

In [493]:
```
df.describe()
```

Out[493]:

	Actual	Predicted
count	30.000000	30.000000
mean	0.833333	0.898119
std	0.746640	0.753513
min	0.000000	-0.125719
25%	0.000000	0.041992
50%	1.000000	1.189933
75%	1.000000	1.351394
max	2.000000	2.284750

In [494]:
```
print(y_pred)
```

```
[ 2.06423389  0.93621122 -0.12571886  1.81720808 -0.03474229  2.28474989
 -0.02642721  1.31529157  1.26856934  1.09960285  1.56268416  1.30418906
  1.21761172  1.31631149  1.33530104 -0.10570939  1.35675848  1.22067148
  0.02512229 -0.02015197  1.80461038  1.4093279   0.09260158  0.01884705
  1.58280213 -0.10849682  0.17086384  1.16225486  0.89078124  0.10821182]
```

In [ ]:
```

```