Scholaa project

December 1, 2021

1 Scholaa internship Project

2 Checking the results by various Training and Testing ratios using Linear Regression for given data.

2.0.1 For Training and testing ratio of 50:50 i.e., test_size=1/2

[4]:
```
import numpy as np                                                    #import the␣
 ↪required libraries needed.
import pandas as pd
dataset=pd.read_csv('Iris.csv')                                       #loading the␣
 ↪dataset
dataset.drop('Id',axis=1,inplace=True)                                #Removing␣
 ↪the unwanted column(Id) to make it simple
from sklearn.preprocessing import LabelEncoder                        #importing␣
 ↪LabelEncoder from sklearn library
lb=LabelEncoder()
dataset["Species"]=lb.fit_transform(dataset["Species"])               #Converting␣
 ↪species in terms of 0's, 1's and 2's for better understanding
X=dataset.iloc[:,:-1].values                                          #Assigning␣
 ↪all rows and the columns till PetalWidthCm from dataset to X
y=dataset.iloc[:,4].values                                            #Assigning␣
 ↪all rows and the last column i.e., Species from dataset to y
from sklearn.model_selection import train_test_split                  #importing␣
 ↪train_test_split from sklearn library
X_train,X_test,y_train,y_test = train_test_split(X,y, test_size = 1/
 ↪2,random_state=0)       #Dividing the data for training and testing in the␣
 ↪ratio 50:50
from sklearn.linear_model import LinearRegression                     #importing␣
 ↪LinearRegression from sklearn library
lr=LinearRegression()
lr.fit(X_train,y_train)                                               #fitting the␣
 ↪training data using LinearRegression and traing the model
y_pred=lr.predict(X_test)                                             #The model␣
 ↪is predicting the species for testing data
```

```
print("Accuracy score: ",lr.score(X_test,y_test)*100,"%")        #Finding the␣
 ↪accuracy of the model
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score  ␣
 ↪  #importing mean_absolute_error, mean_squared_error, r2_score from sklearn␣
 ↪library for finding losses and score
print("Mean absolute error: ", mean_absolute_error(y_test,y_pred))
print("Mean squared error: ", mean_squared_error(y_test,y_pred))
print("Root Mean squared error: ", np.sqrt(mean_squared_error(y_test,y_pred)))
print("R2_score: ", r2_score(y_test,y_pred))
```

```
Accuracy score:  89.31295013048629 %
Mean absolute error:  0.20240006197941057
Mean squared error:  0.06395130641917014
Root Mean squared error:  0.2528859553616415
R2_score:  0.8931295013048628
```

2.0.2 For Training and testing ratio of 60:40 i.e., test_size=0.4

[5]:
```
import numpy as np                                   #import the required libraries␣
 ↪needed.
import pandas as pd
dataset=pd.read_csv('Iris.csv')                      #loading the dataset
dataset.drop('Id',axis=1,inplace=True)               #Removing the unwanted␣
 ↪column(Id) to make it simple
from sklearn.preprocessing import LabelEncoder
lb=LabelEncoder()
dataset["Species"]=lb.fit_transform(dataset["Species"])
X=dataset.iloc[:,:-1].values
y=dataset.iloc[:,4].values
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y, test_size = 0.
 ↪4,random_state=0)      #Dividing the data for training and testing in the␣
 ↪ratio 60:40
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(X_train,y_train)
y_pred=lr.predict(X_test)
print("Accuracy score: ",lr.score(X_test,y_test)*100,"%")
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
print("Mean absolute error: ", mean_absolute_error(y_test,y_pred))
print("Mean squared error: ", mean_squared_error(y_test,y_pred))
print("Root Mean squared error: ", np.sqrt(mean_squared_error(y_test,y_pred)))
print("R2_score: ", r2_score(y_test,y_pred))
```

```
Accuracy score:  88.86093167492697 %
Mean absolute error:  0.2084165864058451
Mean squared error:  0.06791737492648697
```

```
Root Mean squared error:  0.26060962170742463
R2_score:  0.8886093167492697
```

2.0.3 For Training and testing ratio of 70:30 i.e., test_size=0.3

[3]:
```
import numpy as np  #import the required libraries needed.
import pandas as pd
dataset=pd.read_csv('Iris.csv')     #loading the dataset
dataset.drop('Id',axis=1,inplace=True)  #Removing the unwanted column(Id) to␣
 ↪make it simple
from sklearn.preprocessing import LabelEncoder
lb=LabelEncoder()
dataset["Species"]=lb.fit_transform(dataset["Species"])
X=dataset.iloc[:,:-1].values
y=dataset.iloc[:,4].values
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)     #Dividing the data for training and testing in the␣
 ↪ratio 70:30
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(X_train,y_train)
y_pred=lr.predict(X_test)
print("Accuracy score: ",lr.score(X_test,y_test)*100,"%")
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
print("Mean absolute error: ", mean_absolute_error(y_test,y_pred))
print("Mean squared error: ", mean_squared_error(y_test,y_pred))
print("Root Mean squared error: ", np.sqrt(mean_squared_error(y_test,y_pred)))
print("R2_score: ", r2_score(y_test,y_pred))
```

```
Accuracy score:  89.99447180621179 %
Mean absolute error:  0.19694319022859288
Mean squared error:  0.058797918768434404
Root Mean squared error:  0.24248282159450885
R2_score:  0.899944718062118
```

2.0.4 For Training and testing ratio of 80:20 i.e., test_size=0.2

[4]:
```
import numpy as np  #import the required libraries needed.
import pandas as pd
dataset=pd.read_csv('Iris.csv')     #loading the dataset
dataset.drop('Id',axis=1,inplace=True)  #Removing the unwanted column(Id) to␣
 ↪make it simple
from sklearn.preprocessing import LabelEncoder
lb=LabelEncoder()
dataset["Species"]=lb.fit_transform(dataset["Species"])
X=dataset.iloc[:,:-1].values
```

```
y=dataset.iloc[:,4].values
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)      #Dividing the data for training and testing in the␣
 ↪ratio 80:20
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(X_train,y_train)
y_pred=lr.predict(X_test)
print("Accuracy score: ",lr.score(X_test,y_test)*100,"%")
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
print("Mean absolute error: ", mean_absolute_error(y_test,y_pred))
print("Mean squared error: ", mean_squared_error(y_test,y_pred))
print("Root Mean squared error: ", np.sqrt(mean_squared_error(y_test,y_pred)))
print("R2_score: ", r2_score(y_test,y_pred))
```

```
Accuracy score:  90.59663899067813 %
Mean absolute error:  0.18498283353406483
Mean squared error:  0.05067366766134556
Root Mean squared error:  0.22510812437880948
R2_score:  0.9059663899067814
```

[5]:
```
import numpy as np  #import the required libraries needed.
import pandas as pd
dataset=pd.read_csv('Iris.csv')     #loading the dataset
dataset.drop('Id',axis=1,inplace=True)  #Removing the unwanted column(Id) to␣
 ↪make it simple
from sklearn.preprocessing import LabelEncoder
lb=LabelEncoder()
dataset["Species"]=lb.fit_transform(dataset["Species"])
X=dataset.iloc[:,:-1].values
y=dataset.iloc[:,4].values
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)
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(X_train,y_train)
y_pred=lr.predict(X_test)
print("Accuracy score: ",lr.score(X_test,y_test)*100,"%")
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
print("Mean absolute error: ", mean_absolute_error(y_test,y_pred))
print("Mean squared error: ", mean_squared_error(y_test,y_pred))
print("Root Mean squared error: ", np.sqrt(mean_squared_error(y_test,y_pred)))
print("R2_score: ", r2_score(y_test,y_pred))
```

```
Accuracy score:  90.33582492295453 %
```

```
Mean absolute error:  0.19829850933386622
Mean squared error:  0.05987922877737375
Root Mean squared error:  0.24470232687364
R2_score:  0.9033582492295453
```

From the above results for various ratios of training and testing ratios, we see that the Accuracy of the linear regression model kept decreasing(a small decrease) with increase in the testing data (or) with increase in testing and training ratio. The Loss/Error is minimum for the model, but a slight increment of Loss/Error is found with increase in testing data.

From above examples, let's consider training and testing ratios 80:20 and 50:50 and compare them.

When training and testing ratio is 80:20(test_size=0.2), we see that Accuracy is 90.59663899067813% whereas when the training and testing ratio is 50:50(test_size=0.5) the Accuracy decreased to 89.31295013048629%.

Hence the result varies with vary in training and testing ratio

3 Preparing and Comparing a model using Logistic Regression, SVM(linear, sigmoid, kernel) and Decision tree for given data. Also plotting the confusion matrix and classification report in all cases.

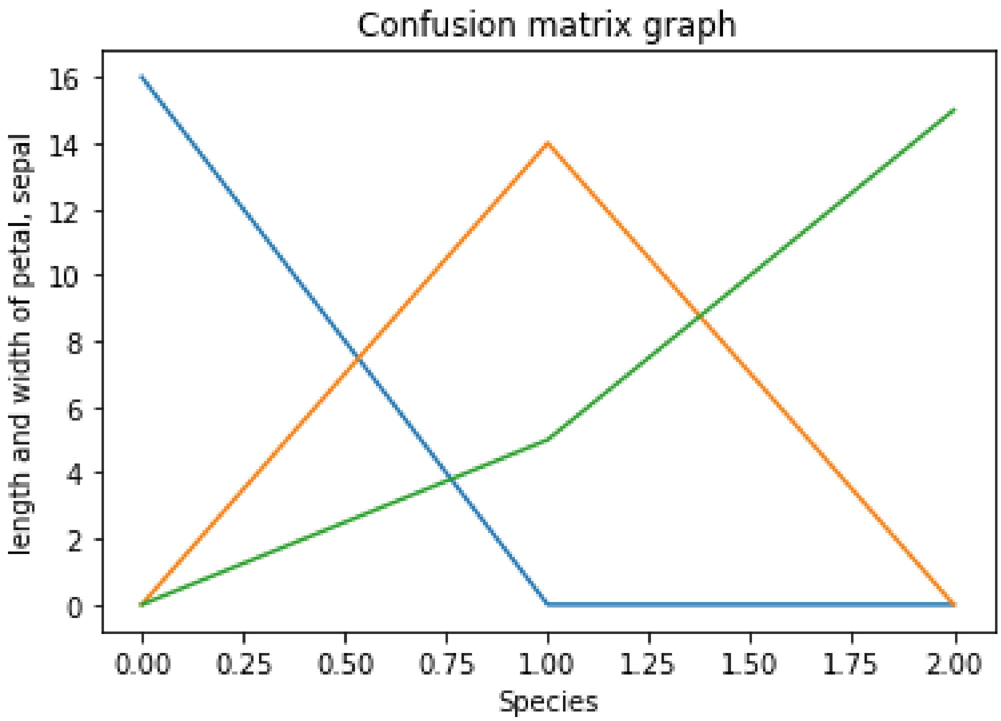
3.0.1 Model using Logistic Regression

[9]:
```
import numpy as np                                    #import the required␣
 ↪libraries needed.
import pandas as pd
import matplotlib.pyplot as plt
dataset=pd.read_csv('Iris.csv')                       #loading the dataset
dataset.drop('Id',axis=1,inplace=True)                #Removing the unwanted␣
 ↪column(Id) to make it simple
from sklearn.preprocessing import LabelEncoder        #importing␣
 ↪LabelEncoder from sklearn library
lb=LabelEncoder()
dataset["Species"]=lb.fit_transform(dataset["Species"])  #Converting species in␣
 ↪terms of 0's, 1's and 2's for better understanding
print(dataset)
X=dataset.iloc[:,:-1].values
y=dataset.iloc[:,4].values
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)
from sklearn.linear_model import LogisticRegression     #importing␣
 ↪LinearRegression from sklearn library
```

```python
logr=LogisticRegression()
logr.fit(X_train,y_train)                                    #fitting the training data using LogisticRegression and traing the model
y_pred=logr.predict(X_test)                                  #The model is predicting the species for testing data
from sklearn.metrics import accuracy_score
print('Accuracy score: ',accuracy_score(y_pred,y_test)*100,"%")       #Finding the accuracy of the model
from sklearn.metrics import confusion_matrix, classification_report  #importing confusion_matrix, classification_report
print(confusion_matrix(y_test,y_pred))                                #printing confusion matrix
print(classification_report(y_test,y_pred))                           #printing classification report
plt.plot(confusion_matrix(y_test,y_pred))                             #plotting confusion matrix
plt.title("Confusion matrix graph")
plt.xlabel("Species")
plt.ylabel("length and width of petal, sepal")
```

```
     SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  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
..             ...           ...            ...           ...      ...
145            6.7           3.0            5.2           2.3        2
146            6.3           2.5            5.0           1.9        2
147            6.5           3.0            5.2           2.0        2
148            6.2           3.4            5.4           2.3        2
149            5.9           3.0            5.1           1.8        2

[150 rows x 5 columns]
Accuracy score:  90.0 %
[[16  0  0]
 [ 0 14  5]
 [ 0  0 15]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       1.00      0.74      0.85        19
           2       0.75      1.00      0.86        15

    accuracy                           0.90        50
   macro avg       0.92      0.91      0.90        50
weighted avg       0.93      0.90      0.90        50
```

[9]: Text(0, 0.5, 'length and width of petal, sepal')



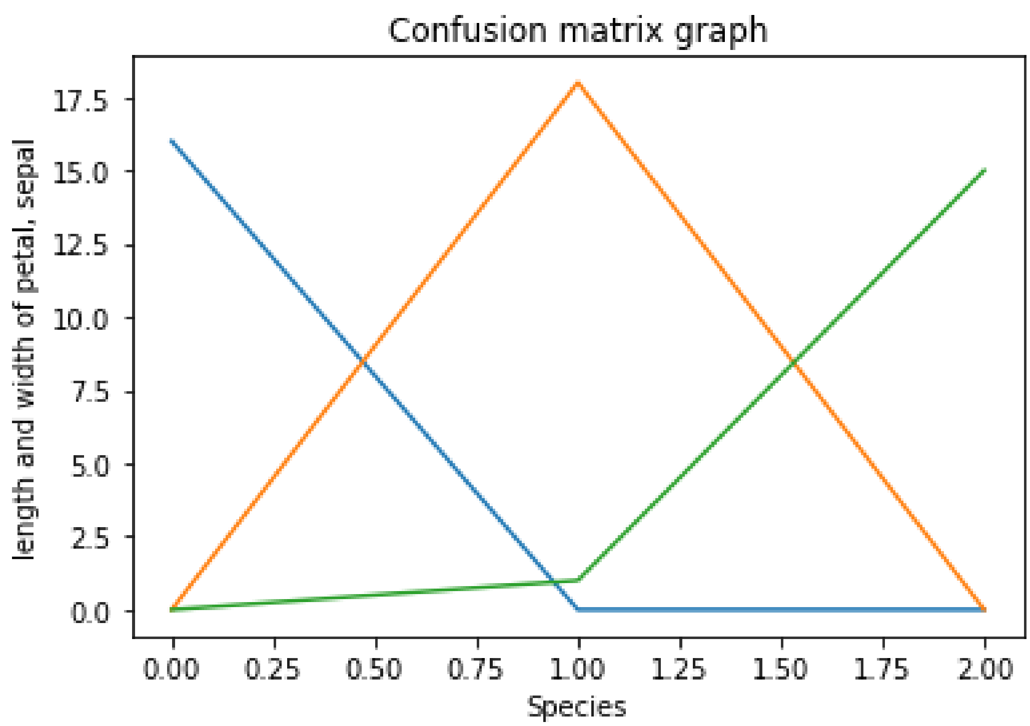
3.0.2 Model using SVM- Support Vector Machines (with kernel='linear')

[22]:
```
from sklearn.svm import SVC                         #importing SVC from sklearn␣
 ↪library
svc_classifier=SVC(kernel='linear')                 #assigning kernel='linear'
svc_classifier.fit(X_train,y_train)                 #fitting the training data using␣
 ↪SVC and traing the model
y_pred=svc_classifier.predict(X_test)               #The model is predicting the␣
 ↪species for testing data
from sklearn.metrics import confusion_matrix, classification_report   ␣
 ↪#importing confusion_matrix, classification_report
print(confusion_matrix(y_test,y_pred))                                ␣
 ↪#printing confusion matrix
print(classification_report(y_test,y_pred))                           ␣
 ↪#printing classification report
from sklearn.metrics import accuracy_score
print('Accuracy score: ',accuracy_score(y_pred,y_test)*100,"%")        #Finding␣
 ↪the accuracy of the model
```

```
plt.plot(confusion_matrix(y_test,y_pred))
 ↪#plotting confusion matrix
plt.title("Confusion matrix graph")
plt.xlabel("Species")
plt.ylabel("length and width of petal, sepal")
```

```
[[16  0  0]
 [ 0 18  1]
 [ 0  0 15]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       1.00      0.95      0.97        19
           2       0.94      1.00      0.97        15

    accuracy                           0.98        50
   macro avg       0.98      0.98      0.98        50
weighted avg       0.98      0.98      0.98        50

Accuracy score:  98.0 %
```

[22]:
```
Text(0, 0.5, 'length and width of petal, sepal')
```

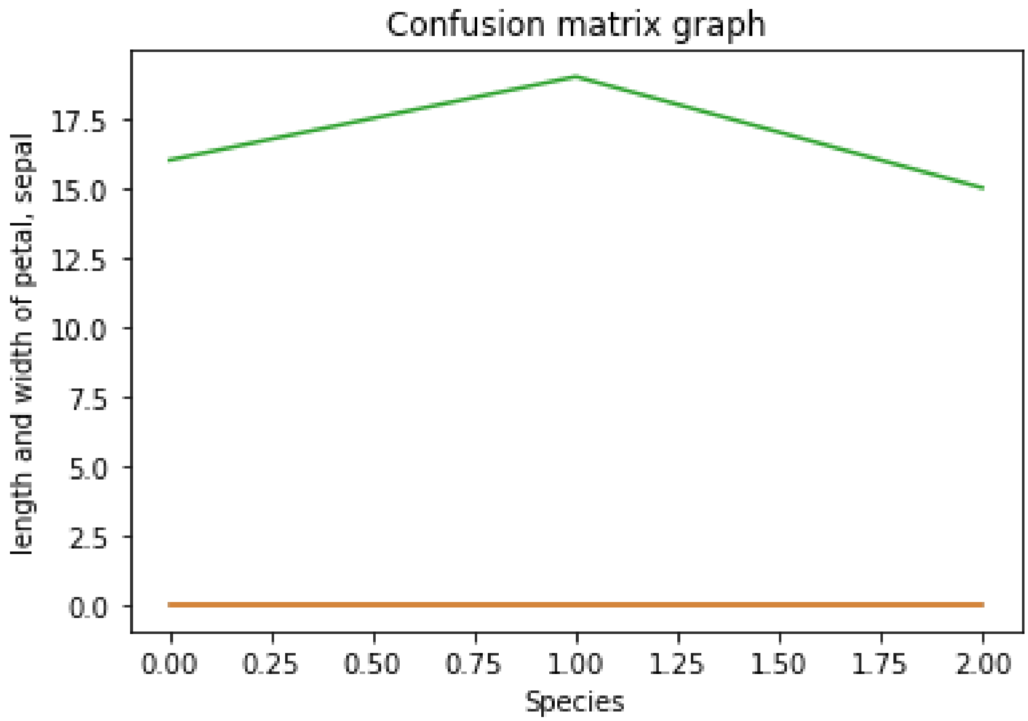


3.0.3 Model using SVM- Support Vector Machines (with kernel='sigmoid')

[25]:
```
from sklearn.svm import SVC
svc_classifier=SVC(kernel='sigmoid')          #assigning kernel='sigmoid'
svc_classifier.fit(X_train,y_train)
y_pred=svc_classifier.predict(X_test)
from sklearn.metrics import confusion_matrix, classification_report
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
from sklearn.metrics import accuracy_score
print('Accuracy score: ',accuracy_score(y_pred,y_test)*100,"%")
plt.plot(confusion_matrix(y_test,y_pred))
plt.title("Confusion matrix graph")
plt.xlabel("Species")
plt.ylabel("length and width of petal, sepal")
```

```
[[ 0  0 16]
 [ 0  0 19]
 [ 0  0 15]]
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        16
           1       0.00      0.00      0.00        19
           2       0.30      1.00      0.46        15

    accuracy                           0.30        50
   macro avg       0.10      0.33      0.15        50
weighted avg       0.09      0.30      0.14        50

Accuracy score:  30.0 %
```

[25]:
```
Text(0, 0.5, 'length and width of petal, sepal')
```



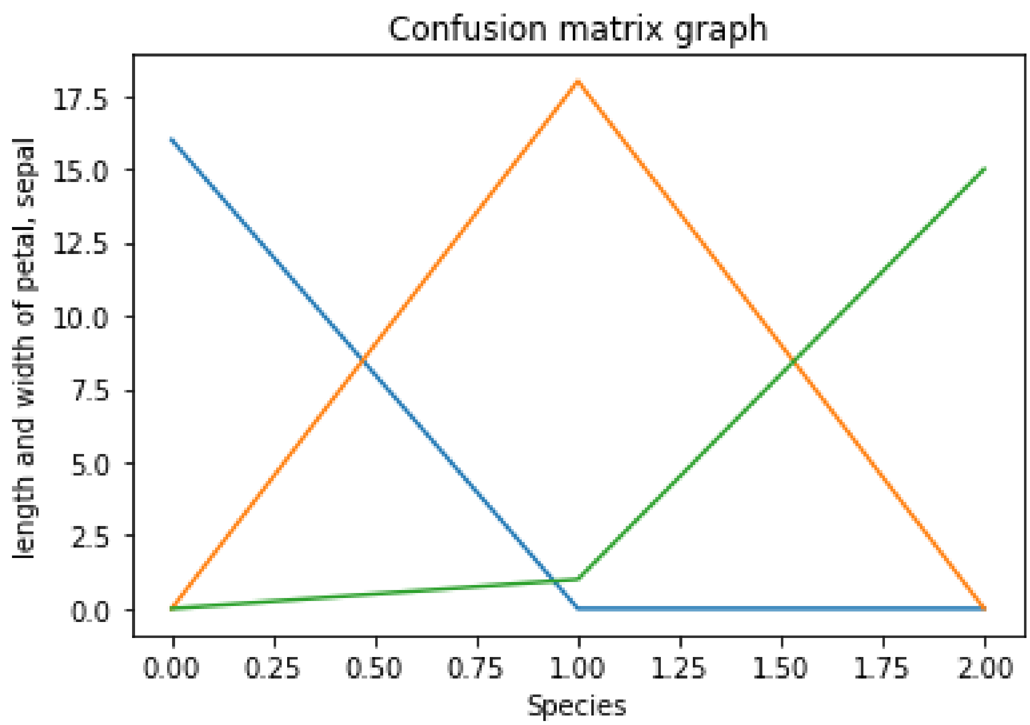
3.0.4 Model using SVM- Support Vector Machines (with kernel='rbf')

[24]:
```
from sklearn.svm import SVC
svc_classifier=SVC(kernel='rbf')                        #assigning kernel='rbf'
svc_classifier.fit(X_train,y_train)
y_pred=svc_classifier.predict(X_test)
from sklearn.metrics import confusion_matrix, classification_report
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
from sklearn.metrics import accuracy_score
print('Accuracy score: ',accuracy_score(y_pred,y_test)*100,"%")
plt.plot(confusion_matrix(y_test,y_pred))
plt.title("Confusion matrix graph")
plt.xlabel("Species")
plt.ylabel("length and width of petal, sepal")
```

```
[[16  0  0]
 [ 0 18  1]
 [ 0  0 15]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       1.00      0.95      0.97        19
```

```
           2       0.94      1.00      0.97        15

    accuracy                           0.98        50
   macro avg       0.98      0.98      0.98        50
weighted avg       0.98      0.98      0.98        50

Accuracy score:  98.0 %
```

[24]: Text(0, 0.5, 'length and width of petal, sepal')



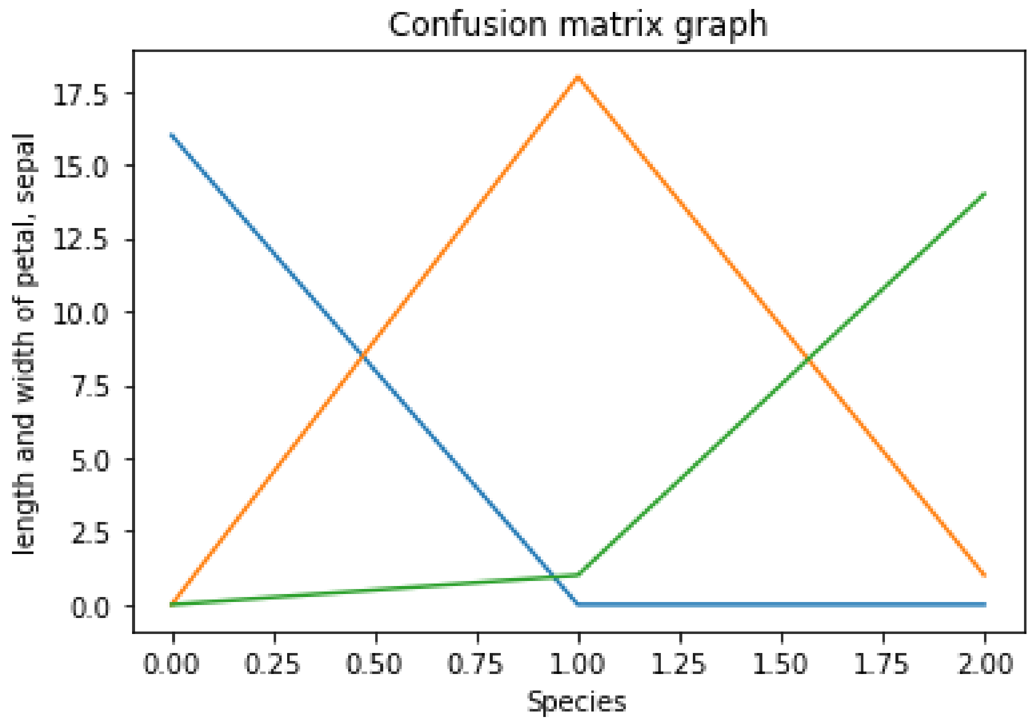
3.0.5 Model using Decision Tree

[26]:
```
from sklearn.tree import DecisionTreeClassifier                              ␣
 ↪#importing SVC from sklearn library
dtc_classifier=DecisionTreeClassifier()
dtc_classifier.fit(X_train,y_train)                                           #fitting␣
 ↪the training data using SVC and traing the model
y_pred=dtc_classifier.predict(X_test)                                         #The␣
 ↪model is predicting the species for testing data
from sklearn.metrics import confusion_matrix, classification_report          ␣
 ↪#importing confusion_matrix, classification_report
print(confusion_matrix(y_test,y_pred))                                       ␣
 ↪#printing confusion matrix
```

```
print(classification_report(y_test,y_pred))                                    ␣
 ↪#printing classification report
from sklearn.metrics import accuracy_score
print('Accuracy score: ',accuracy_score(y_pred,y_test)*100,"%")        #finding␣
 ↪accuracy of the model
plt.plot(confusion_matrix(y_test,y_pred))                                      ␣
 ↪#plotting confusion matrix
plt.title("Confusion matrix graph")
plt.xlabel("Species")
plt.ylabel("length and width of petal, sepal")
```

```
[[16  0  0]
 [ 0 18  1]
 [ 0  1 14]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       0.95      0.95      0.95        19
           2       0.93      0.93      0.93        15

    accuracy                           0.96        50
   macro avg       0.96      0.96      0.96        50
weighted avg       0.96      0.96      0.96        50

Accuracy score:  96.0 %
```

[26]:
```
Text(0, 0.5, 'length and width of petal, sepal')
```



Hence models are created for the given data using logistic regression, SVM and Decision tree.

But the Accuracy varies in each case.

For the model using Logistic Regression, the Accuracy is 90%

For the model using SVM(kernel='linear'), the Accuracy is 98%

For the model using SVM(kernel='sigmoid'), the Accuracy is 30%

For the model using SVM(kernel='rbf'), the Accuracy is 98%

For the model using Decision Tree, the Accuracy is 96%

Therefore, the accuracy varies for each model. The model made using SVM(kernel='linear') and SVM(kernel='rbf') has the highest accuracy of all i.e., 98%.

Classification report and confusion matrix also differs from one another. ### –By Prathi Harshith

[ ]:

```

```