In [1]:
```
# Question 1:

# Load libraries
import pandas
from pandas.plotting import scatter_matrix
import matplotlib.pyplot as plt
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
```

In [2]:
```
# Question 1:

# Load dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
dataset = pandas.read_csv(url, names=names)
```

In [3]:
```
# Question 1:

# head
print(dataset.head(20))
```

```
    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
5            5.4          3.9           1.7          0.4  Iris-setosa
6            4.6          3.4           1.4          0.3  Iris-setosa
7            5.0          3.4           1.5          0.2  Iris-setosa
8            4.4          2.9           1.4          0.2  Iris-setosa
9            4.9          3.1           1.5          0.1  Iris-setosa
10           5.4          3.7           1.5          0.2  Iris-setosa
11           4.8          3.4           1.6          0.2  Iris-setosa
12           4.8          3.0           1.4          0.1  Iris-setosa
13           4.3          3.0           1.1          0.1  Iris-setosa
14           5.8          4.0           1.2          0.2  Iris-setosa
15           5.7          4.4           1.5          0.4  Iris-setosa
16           5.4          3.9           1.3          0.4  Iris-setosa
17           5.1          3.5           1.4          0.3  Iris-setosa
18           5.7          3.8           1.7          0.3  Iris-setosa
19           5.1          3.8           1.5          0.3  Iris-setosa
```

In [4]:
```
# Question 1:

# descriptions
print(dataset.describe())
```

```
       sepal-length  sepal-width  petal-length  petal-width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.054000      3.758667     1.198667
std        0.828066     0.433594      1.764420     0.763161
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
```

In [5]:
```
# Question 1:

# class distribution
print(dataset.groupby('class').size())
```

```
class
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

In [6]:
```
# Question 2: Splitting the Iris dataset in 2 parts -- features and species

array = dataset.values
X = array[:,0:4]
Y = array[:, 4]
```

In [7]:
```
# Question 2: Printing the splitted array

print(array)
```

```
[[5.1 3.5 1.4 0.2 'Iris-setosa']
 [4.9 3.0 1.4 0.2 'Iris-setosa']
 [4.7 3.2 1.3 0.2 'Iris-setosa']
 [4.6 3.1 1.5 0.2 'Iris-setosa']
 [5.0 3.6 1.4 0.2 'Iris-setosa']
 [5.4 3.9 1.7 0.4 'Iris-setosa']
 [4.6 3.4 1.4 0.3 'Iris-setosa']
 [5.0 3.4 1.5 0.2 'Iris-setosa']
 [4.4 2.9 1.4 0.2 'Iris-setosa']
 [4.9 3.1 1.5 0.1 'Iris-setosa']
 [5.4 3.7 1.5 0.2 'Iris-setosa']
 [4.8 3.4 1.6 0.2 'Iris-setosa']
 [4.8 3.0 1.4 0.1 'Iris-setosa']
 [4.3 3.0 1.1 0.1 'Iris-setosa']
 [5.8 4.0 1.2 0.2 'Iris-setosa']
 [5.7 4.4 1.5 0.4 'Iris-setosa']
 [5.4 3.9 1.3 0.4 'Iris-setosa']
 [5.1 3.5 1.4 0.3 'Iris-setosa']
 [5.7 3.8 1.7 0.3 'Iris-setosa']
 [5.1 3.8 1.5 0.3 'Iris-setosa']
 [5.4 3.4 1.7 0.2 'Iris-setosa']
 [5.1 3.7 1.5 0.4 'Iris-setosa']
 [4.6 3.6 1.0 0.2 'Iris-setosa']
 [5.1 3.3 1.7 0.5 'Iris-setosa']
 [4.8 3.4 1.9 0.2 'Iris-setosa']
 [5.0 3.0 1.6 0.2 'Iris-setosa']
 [5.0 3.4 1.6 0.4 'Iris-setosa']
 [5.2 3.5 1.5 0.2 'Iris-setosa']
 [5.2 3.4 1.4 0.2 'Iris-setosa']
 [4.7 3.2 1.6 0.2 'Iris-setosa']
 [4.8 3.1 1.6 0.2 'Iris-setosa']
 [5.4 3.4 1.5 0.4 'Iris-setosa']
 [5.2 4.1 1.5 0.1 'Iris-setosa']
 [5.5 4.2 1.4 0.2 'Iris-setosa']
 [4.9 3.1 1.5 0.1 'Iris-setosa']
 [5.0 3.2 1.2 0.2 'Iris-setosa']
 [5.5 3.5 1.3 0.2 'Iris-setosa']
 [4.9 3.1 1.5 0.1 'Iris-setosa']
 [4.4 3.0 1.3 0.2 'Iris-setosa']
 [5.1 3.4 1.5 0.2 'Iris-setosa']
 [5.0 3.5 1.3 0.3 'Iris-setosa']
 [4.5 2.3 1.3 0.3 'Iris-setosa']
 [4.4 3.2 1.3 0.2 'Iris-setosa']
 [5.0 3.5 1.6 0.6 'Iris-setosa']
 [5.1 3.8 1.9 0.4 'Iris-setosa']
 [4.8 3.0 1.4 0.3 'Iris-setosa']
 [5.1 3.8 1.6 0.2 'Iris-setosa']
 [4.6 3.2 1.4 0.2 'Iris-setosa']
 [5.3 3.7 1.5 0.2 'Iris-setosa']
 [5.0 3.3 1.4 0.2 'Iris-setosa']
 [7.0 3.2 4.7 1.4 'Iris-versicolor']
 [6.4 3.2 4.5 1.5 'Iris-versicolor']
 [6.9 3.1 4.9 1.5 'Iris-versicolor']
 [5.5 2.3 4.0 1.3 'Iris-versicolor']
 [6.5 2.8 4.6 1.5 'Iris-versicolor']
 [5.7 2.8 4.5 1.3 'Iris-versicolor']
 [6.3 3.3 4.7 1.6 'Iris-versicolor']
 [4.9 2.4 3.3 1.0 'Iris-versicolor']
 [6.6 2.9 4.6 1.3 'Iris-versicolor']
 [5.2 2.7 3.9 1.4 'Iris-versicolor']
 [5.0 2.0 3.5 1.0 'Iris-versicolor']
 [5.9 3.0 4.2 1.5 'Iris-versicolor']
 [6.0 2.2 4.0 1.0 'Iris-versicolor']
 [6.1 2.9 4.7 1.4 'Iris-versicolor']
 [5.6 2.9 3.6 1.3 'Iris-versicolor']
```

In [8]:
```
# Question 2: Printing all the features

print(X)
```

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.0 1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.0 3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.0 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 3.0 1.4 0.1]
 [4.3 3.0 1.1 0.1]
 [5.8 4.0 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]
 [5.1 3.8 1.5 0.3]
 [5.4 3.4 1.7 0.2]
 [5.1 3.7 1.5 0.4]
 [4.6 3.6 1.0 0.2]
 [5.1 3.3 1.7 0.5]
 [4.8 3.4 1.9 0.2]
 [5.0 3.0 1.6 0.2]
 [5.0 3.4 1.6 0.4]
 [5.2 3.5 1.5 0.2]
 [5.2 3.4 1.4 0.2]
 [4.7 3.2 1.6 0.2]
 [4.8 3.1 1.6 0.2]
 [5.4 3.4 1.5 0.4]
 [5.2 4.1 1.5 0.1]
 [5.5 4.2 1.4 0.2]
 [4.9 3.1 1.5 0.1]
 [5.0 3.2 1.2 0.2]
 [5.5 3.5 1.3 0.2]
 [4.9 3.1 1.5 0.1]
 [4.4 3.0 1.3 0.2]
 [5.1 3.4 1.5 0.2]
 [5.0 3.5 1.3 0.3]
 [4.5 2.3 1.3 0.3]
 [4.4 3.2 1.3 0.2]
 [5.0 3.5 1.6 0.6]
 [5.1 3.8 1.9 0.4]
 [4.8 3.0 1.4 0.3]
 [5.1 3.8 1.6 0.2]
 [4.6 3.2 1.4 0.2]
 [5.3 3.7 1.5 0.2]
 [5.0 3.3 1.4 0.2]
 [7.0 3.2 4.7 1.4]
 [6.4 3.2 4.5 1.5]
 [6.9 3.1 4.9 1.5]
 [5.5 2.3 4.0 1.3]
 [6.5 2.8 4.6 1.5]
 [5.7 2.8 4.5 1.3]
 [6.3 3.3 4.7 1.6]
 [4.9 2.4 3.3 1.0]
 [6.6 2.9 4.6 1.3]
 [5.2 2.7 3.9 1.4]
 [5.0 2.0 3.5 1.0]
 [5.9 3.0 4.2 1.5]
 [6.0 2.2 4.0 1.0]
 [6.1 2.9 4.7 1.4]
 [5.6 2.9 3.6 1.3]
```

In [9]:
```
# Question 2: Printing the Species

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 [10]:
```
# Question 3:

# Create SVM model, which takes as the output variable the species, and input, all
of the features.

from sklearn import svm
import numpy

svc = svm.SVC(kernel='linear', C=1, gamma='auto').fit(X, Y)

svc.score(X, Y)

#Predict Output
predicted = svc.predict(X)
```

In [11]:
```
# Question 3:

print(svc.score(X,Y))
```

```
0.9933333333333333
```

In [12]:

```
# Question 4:

# Number of support vectors = 27
# SVM Kernal used = Linear

print(svc.support_vectors_)
```

```
[[5.1 3.3 1.7 0.5]
 [4.8 3.4 1.9 0.2]
 [4.5 2.3 1.3 0.3]
 [6.9 3.1 4.9 1.5]
 [6.3 3.3 4.7 1.6]
 [6.1 2.9 4.7 1.4]
 [5.6 3.  4.5 1.5]
 [6.2 2.2 4.5 1.5]
 [5.9 3.2 4.8 1.8]
 [6.3 2.5 4.9 1.5]
 [6.8 2.8 4.8 1.4]
 [6.7 3.  5.  1.7]
 [6.  2.7 5.1 1.6]
 [5.4 3.  4.5 1.5]
 [5.1 2.5 3.  1.1]
 [4.9 2.5 4.5 1.7]
 [6.5 3.2 5.1 2. ]
 [6.  2.2 5.  1.5]
 [6.3 2.7 4.9 1.8]
 [6.2 2.8 4.8 1.8]
 [6.1 3.  4.9 1.8]
 [7.2 3.  5.8 1.6]
 [6.3 2.8 5.1 1.5]
 [6.  3.  4.8 1.8]
 [6.3 2.5 5.  1.9]
 [6.5 3.  5.2 2. ]
 [5.9 3.  5.1 1.8]]
```

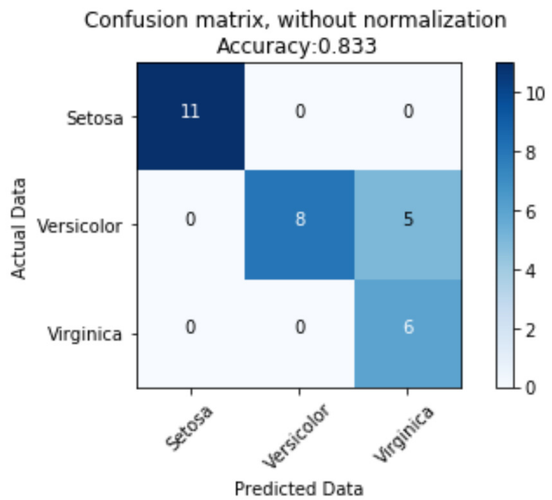
In [13]:

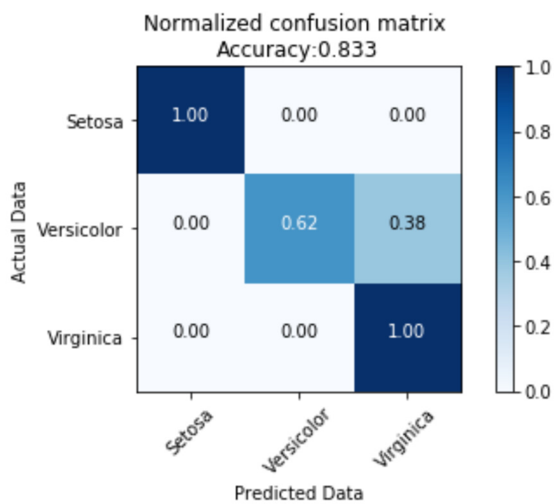
```
# Question 5:

from sklearn.model_selection import train_test_split

# Split the data into a training set and a test set

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = .20, random_s
tate=0)
print("train sample size",X_train.shape,type(X_train))
print("test sample size",X_test.shape,type(X_test))
```

```
train sample size (120, 4) <class 'numpy.ndarray'>
test sample size (30, 4) <class 'numpy.ndarray'>
```

In [14]:
```
# Question 5:

import itertools

import numpy as np

import matplotlib.pyplot as plt

from sklearn import svm, datasets

from sklearn.metrics import accuracy_score, confusion_matrix, precision_recall_fsco
re_support


class_names = ['Setosa','Versicolor', 'Virginica']


# Run classifier on training set and test set

classifier = svm.SVC(kernel='linear', C = .01)

Y_pred = classifier.fit(X_train, Y_train).predict(X_test)


def plot_confusion_matrix(cm, classes,

                          normalize=False,

                          title='Confusion matrix',

                          cmap=plt.cm.Blues):

    """

    This function prints and plots the confusion matrix.

    Normalization can be applied by setting `normalize=True`.

    """

    if normalize:

        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

        print("Normalized confusion matrix")

    else:

        print('Confusion matrix, without normalization')


    print(cm)


    plt.imshow(cm, interpolation='nearest', cmap=cmap)

    plt.title(title)

    plt.colorbar()

    tick_marks = np.arange(len(classes))
```

```
Confusion matrix, without normalization
[[11  0  0]
 [ 0  8  5]
 [ 0  0  6]]
Normalized confusion matrix
[[1.   0.   0.  ]
 [0.   0.62 0.38]
 [0.   0.   1.  ]]
```

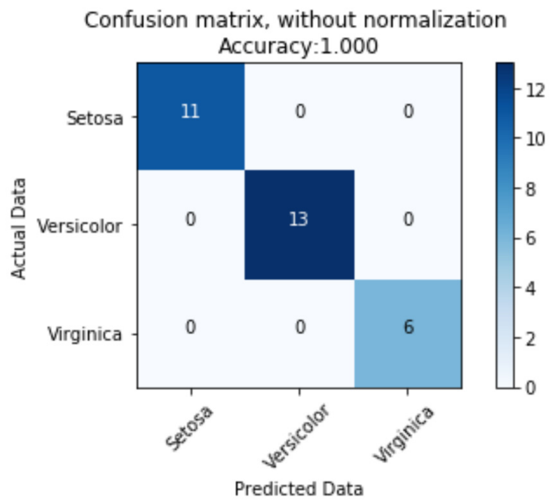


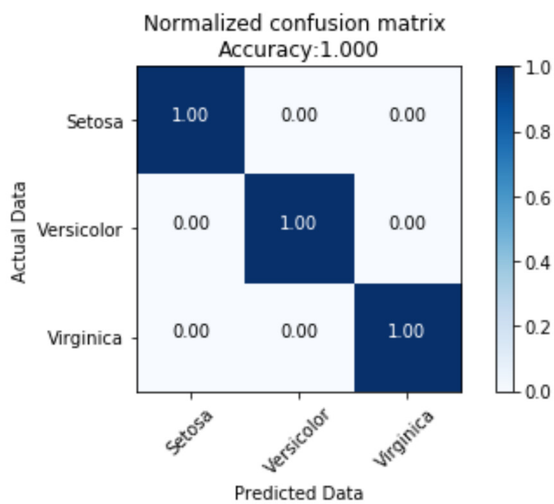


In [15]:

```
# Question 6:

import itertools

import numpy as np

import matplotlib.pyplot as plt

from sklearn import svm, datasets

from sklearn.metrics import accuracy_score, confusion_matrix, precision_recall_fscore_support


class_names = ['Setosa','Versicolor', 'Virginica']


# Run classifier on training set and test set

classifier = svm.SVC(kernel='linear', C= 1, gamma='auto')

Y_pred = classifier.fit(X_train, Y_train).predict(X_test)


def plot_confusion_matrix(cm, classes,

                          normalize=False,

                          title='Confusion matrix',

                          cmap=plt.cm.Blues):

    """

    This function prints and plots the confusion matrix.

    Normalization can be applied by setting `normalize=True`.

    """

    if normalize:

        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

        print("Normalized confusion matrix")

    else:

        print('Confusion matrix, without normalization')


    print(cm)


    plt.imshow(cm, interpolation='nearest', cmap=cmap)

    plt.title(title)

    plt.colorbar()

    tick_marks = np.arange(len(classes))
```

```
Confusion matrix, without normalization
[[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]
Normalized confusion matrix
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```





In [ ]:
```
# Question 6:

# The accuracy improved after changing the C value from ".01" to "1". The Gamma was set to "auto".
```