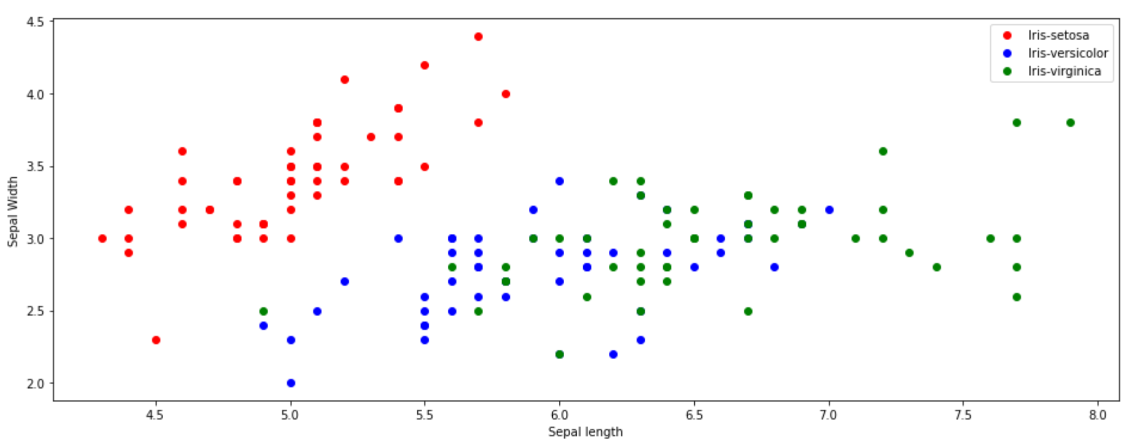
LAB 3

Question 1

In [105]:
```
#Question 1
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import scipy.linalg as la
import math
from scipy.fftpack import fft,fftfreq
from scipy.linalg import toeplitz
from matplotlib import animation


#1
df = pd.read_csv('Iris.csv')
X=df.iloc[:,1:3]
Y=df.iloc[:,5]
X=np.array(X)
Y=np.array(Y)
a=X[np.where(Y=='Iris-setosa')]
b=X[np.where(Y=='Iris-versicolor')]
c=X[np.where(Y=='Iris-virginica')]
plt.figure(figsize=(16,20))
plt.subplot(3, 1, 1)
plt.scatter(a[:,0],a[:,1],c='red',alpha=1,label='Iris-setosa
')
plt.scatter(b[:,0],b[:,1],c='blue',alpha=1,label='Iris-versic
olor')
plt.scatter(c[:,0],c[:,1],c='green',alpha=1,label='Iris-virgi
nica')
plt.legend()
plt.xlabel('Sepal length')
plt.ylabel('Sepal Width')
plt.show()
```

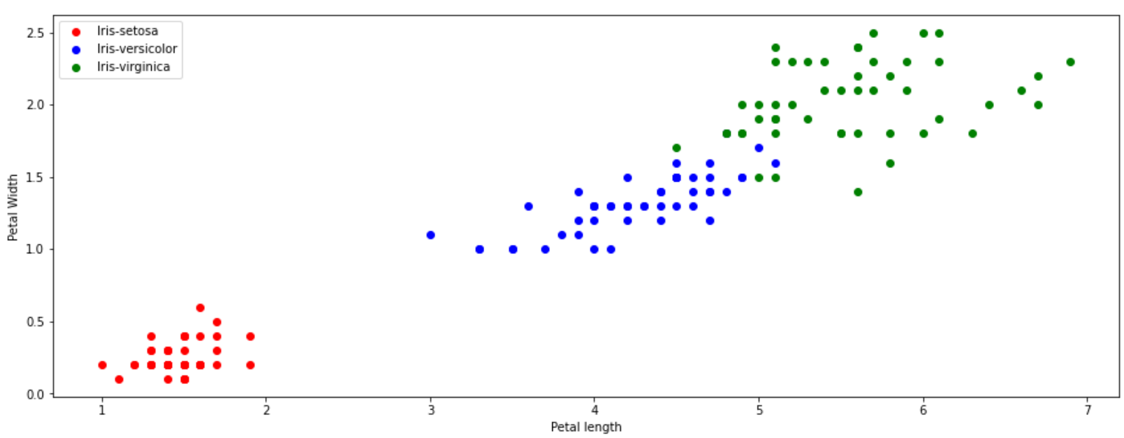


Observation:

As shown in the figure, scatter plot for 3 classes are plotted having sepal length and width as x axis and y axis.Here, red dots(class setosa) and blue dots(class versicolor) can be separated by drawing a line and we can achieve 100% train accuracy as both class dots are not overlapping with eachothers majority area.Similarly,red dots and green dots(class virginica) can be separated by drawing a line and can achieve a 100% train accuracy.Here , blue and green dots are inseparable by a line , as they are overlapping with each other.

In [107]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import scipy.linalg as la
import math
from scipy.fftpack import fft,fftfreq
from scipy.linalg import toeplitz

from matplotlib import animation


#1
df = pd.read_csv('Iris.csv')
X=df.iloc[:,3:5]
Y=df.iloc[:,5]
X=np.array(X)
Y=np.array(Y)
a=X[np.where(Y=='Iris-setosa')]
b=X[np.where(Y=='Iris-versicolor')]
c=X[np.where(Y=='Iris-virginica')]
plt.figure(figsize=(16,20))
plt.subplot(3, 1, 1)
plt.scatter(a[:,0],a[:,1],c='red',alpha=1,label='Iris-setosa
')
plt.scatter(b[:,0],b[:,1],c='blue',alpha=1,label='Iris-versic
olor')
plt.scatter(c[:,0],c[:,1],c='green',alpha=1,label='Iris-virgi
nica')
plt.legend()
plt.xlabel('Petal length')
plt.ylabel('Petal Width')
plt.show()
```



Observation:

As shown in the figure, scatter plot for 3 classes are plotted having petal length and width as x axis and y axis.Here, red dots(class setosa) and blue dots(class versicolor) can be separated by drawing a line and we can achieve 100% train accuracy as both class dots are not overlapping with eachothers majority area.Similarly,red dots and green dots(class virginica) can be separated by drawing a line and can achieve a 100% train accuracy.Here , blue and green dots are separable with a line, but cant guarantee 100% train accuracy as there are some dots overlapping.Due, to this dots near decision boundary will not guarantee accurate predictions.

Question 2

In this question , here basic statistics like mean ,median ,max,mean ,std are plotted for every classes.Also for more insights in data i have done visual representation by using box plot & violin part. In both types of plots I have featured species on X-axis & parameters in Y-axis. From Boxplot, we can get idea of median, interquartile range, some outliers for each of four features for all classes. Well, boxplot does not show that in interquartle range how the data is distributed. It can be seen from violin graph.Here ,we can observe the frequency of the numbers in the any range.

In [ ]:
```
#Question 2
iris=df.iloc[:,1:]
iris.groupby('Species').agg(['mean','median','min','max'])
```

Out[ ]:

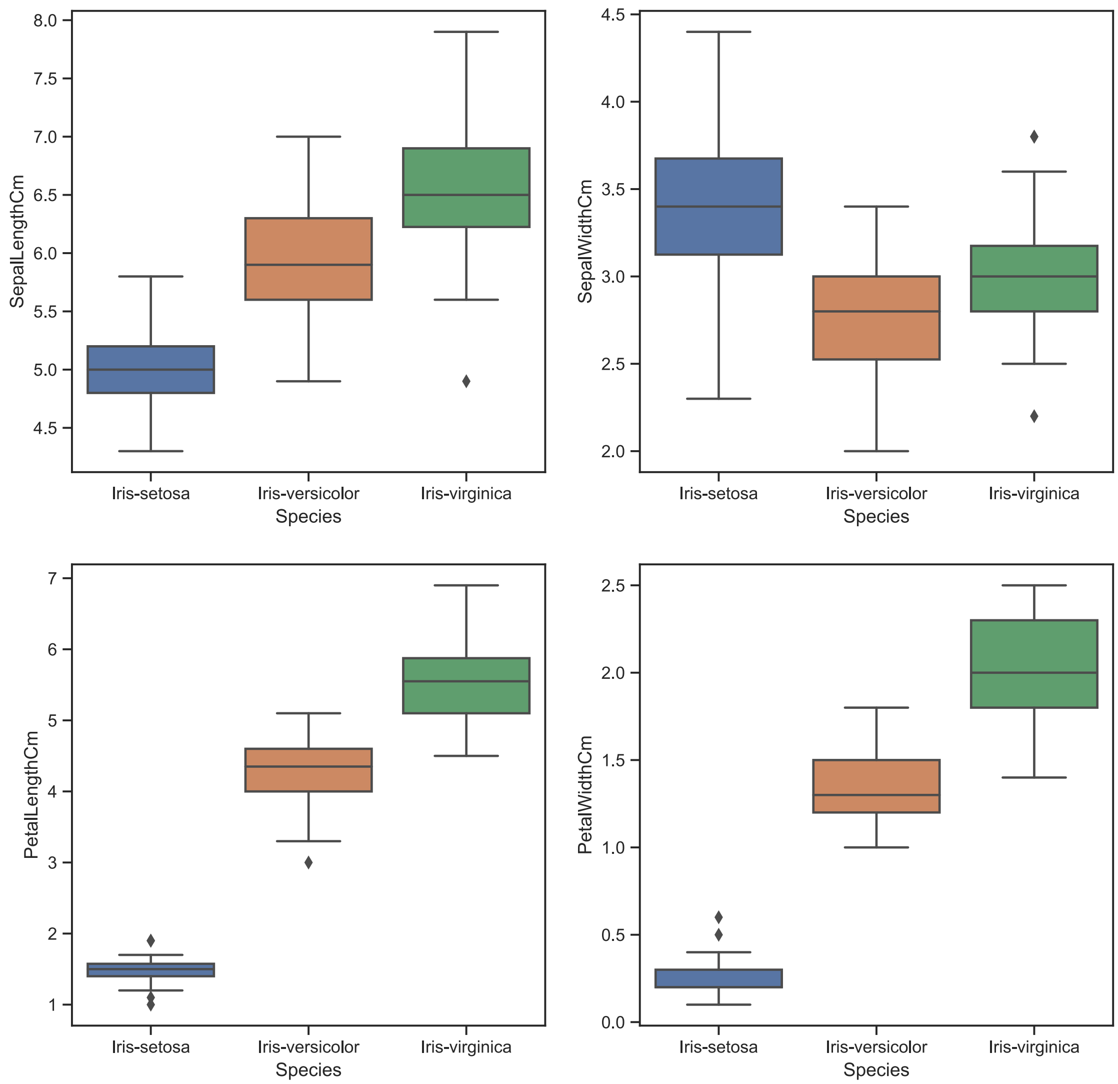
	SepalLengthCm				SepalWidthCm				PetalLengthCm		
	mean	median	min	max	mean	median	min	max	mean	median	ı
Species											
Iris-setosa	5.006	5.0	4.3	5.8	3.418	3.4	2.3	4.4	1.464	1.50	
Iris-versicolor	5.936	5.9	4.9	7.0	2.770	2.8	2.0	3.4	4.260	4.35	
Iris-virginica	6.588	6.5	4.9	7.9	2.974	3.0	2.2	3.8	5.552	5.55	

In [ ]:
```
iris.groupby('Species').std()
```

Out[ ]:

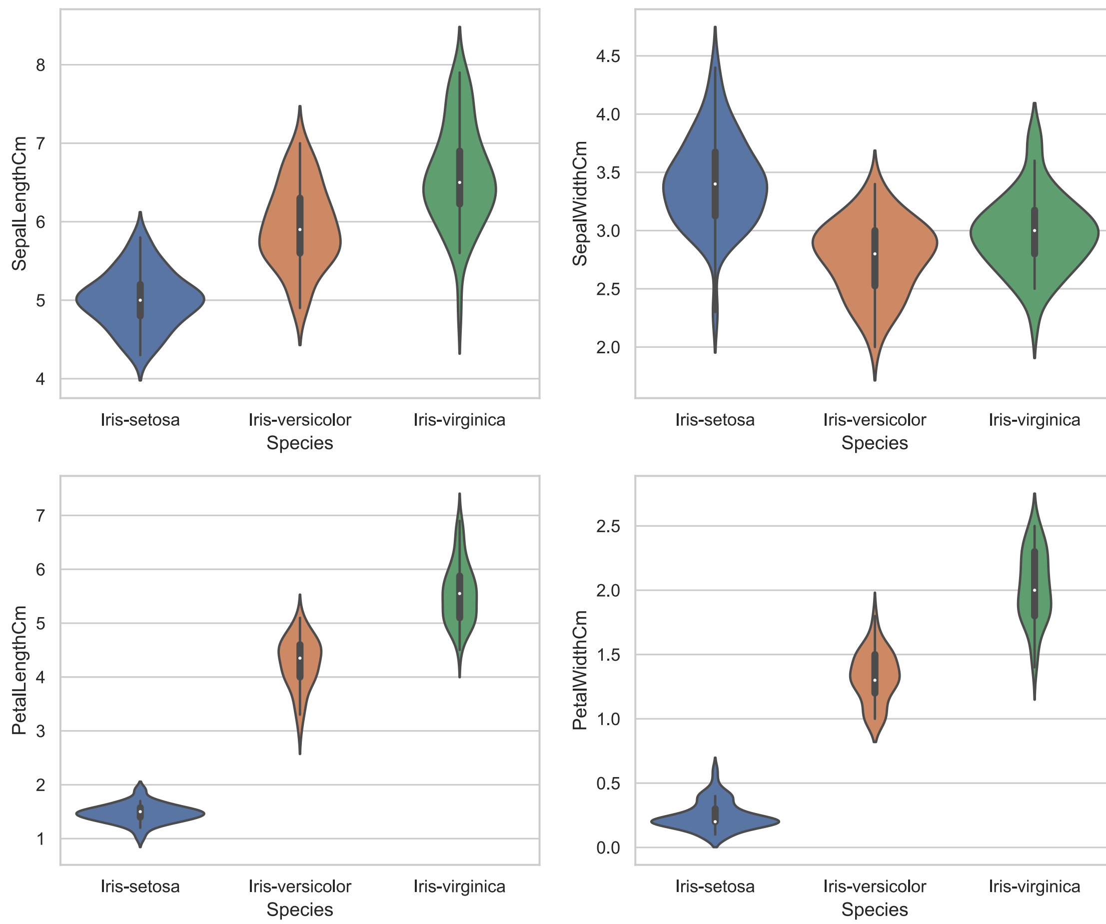
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
Species				
Iris-setosa	0.352490	0.381024	0.173511	0.107210
Iris-versicolor	0.516171	0.313798	0.469911	0.197753
Iris-virginica	0.635880	0.322497	0.551895	0.274650

In [ ]:
```
import seaborn as sns
iris=df.iloc[:,1:]
sns.set(style="ticks")
plt.figure(figsize=(12,12))
plt.subplot(2,2,1)
sns.boxplot(x='Species',y='SepalLengthCm',data=iris)
plt.subplot(2,2,2)
sns.boxplot(x='Species',y='SepalWidthCm',data=iris)
plt.subplot(2,2,3)
sns.boxplot(x='Species',y='PetalLengthCm',data=iris)
plt.subplot(2,2,4)
sns.boxplot(x='Species',y='PetalWidthCm',data=iris)
plt.show()
```



In [ ]:

```
sns.set(style="whitegrid")
plt.figure(figsize=(12,10))
plt.subplot(2,2,1)
sns.violinplot(x='Species',y='SepalLengthCm',data=iris)
plt.subplot(2,2,2)
sns.violinplot(x='Species',y='SepalWidthCm',data=iris)
plt.subplot(2,2,3)
sns.violinplot(x='Species',y='PetalLengthCm',data=iris)
plt.subplot(2,2,4)
sns.violinplot(x='Species',y='PetalWidthCm',data=iris)
plt.show()
```



Question 3

i) Here, ex2data1 is fitted in logistic regression implemented from scratch.Here after analysing the scattered plot, drawing a line to separate the two classes will result lesser accuracy.Here I had taken terms of x1 and x2 till order of 2.so X=[1 x1 x2 x1^2 x2^2 X1*x2],so after fitting X and Y , theta will be obtained of len=6.The accuracy after using 2nd order is obtained to be 100% for both train and test. Due to this, precision,recall and F1 will be 100% or 1.Confusion matrix is been plotted using matshow function and for exact values matrix is printed.Here, runtimewarning can ignored as using exp for very large numbers ,it will decrease its digit precision.

In [2]:
```
#Question 3
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import scipy.linalg as la
import math
from scipy.fftpack import fft,fftfreq
from scipy.linalg import toeplitz
from matplotlib import animation
from sklearn.model_selection import train_test_split

#1
df = pd.read_csv('ex2data1-logisticc.csv')
X=df.iloc[:,0:2]
Y=df.iloc[:,2]
X1=X.copy()
meanx=X1.mean()
stdx=X1.std()
X1=np.array(X1)

X=np.c_[ np.ones(X.shape[0]),X ]
```

In [3]:

```
def g(theta2,X2):
    XT=np.dot(X2,theta2)
    H=1/(1+np.exp(-XT))
    return H
def J(H2,X3,T,Y2):
    error=H2-Y2
    return np.dot(X3.T,error)


theta=np.zeros(6)
alpha=0.01
iterations=1000000
X=np.c_[ X,np.ones(X.shape[0]) ]
for i in range(len( Y)):
    X[i][3]=X[i][1]**2
X=np.c_[ X,np.ones(X.shape[0]) ]
for i in range(len(Y)):
    X[i][4]=X[i][2]**2
X=np.c_[ X,np.ones(X.shape[0]) ]
for i in range(len(Y)):
    X[i][5]=X[i][2]*X[i][1]
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=
0.2)

for i in range(iterations):
    H=g(theta,X_train)
    JJ=J(H,X_train,theta,Y_train)
    if np.sum(JJ**2)==0:
      break

    theta=theta-(alpha/len(Y_train))*JJ
print(theta)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.p
y:4: RuntimeWarning: overflow encountered in exp
  after removing the cwd from sys.path.

[-1.21076630e+02 -3.36988068e+03 -3.80001171e+03  1.6459147
7e+00
  5.35905294e+00  1.17252929e+02]
```

In [40]:

```
z = np.dot(X_train, theta.T)
h = 1 / (1 + np.exp(-z))
t=[]
tp1=0
tn1=0
fp1=0
fn1=0
Y_train=np.array(Y_train)
h=h.reshape(len(Y_train),)
for i in range(len(Y_train)):
    if h[i]>=0.5:
        t.append(1)
    else:
        t.append(0)
t=np.array(t)
for i in range(len(Y_train)):
  if Y_train[i]==t[i]:
    if t[i]==0:
      tn1=tn1+1
    else :
      tp1=tp1+1
  else :
    if t[i]==0:
      fn1=fn1+1
    else :
      fp1=fp1+1
a=(tp1+tn1)/(tp1+tn1+fp1+fn1)
print("accuracy(train cases) : "+ str(a*100)+"%")
```

```
accuracy(train cases) : 100.0%

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.p
y:2: RuntimeWarning: overflow encountered in exp
```

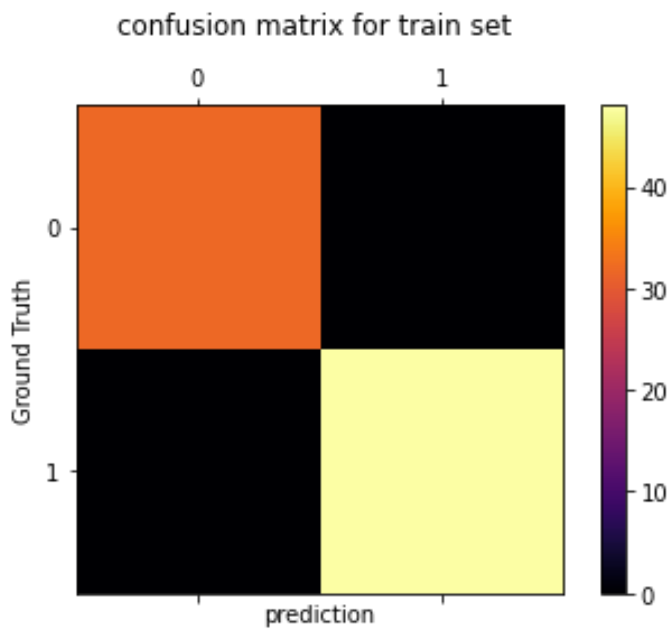
In [41]:

```python
z = np.dot(X_test, theta.T)
h = 1 / (1 + np.exp(-z))
t=[]
tp=0
tn=0
fp=0
fn=0

Y_test=np.array(Y_test)
h=h.reshape(len(Y_test),)
for i in range(len(Y_test)):
    if h[i]>=0.5:
        t.append(1)
    else:
        t.append(0)
t=np.array(t)
for i in range(len(Y_test)):
  if Y_test[i]==t[i]:
    if t[i]==0:
      tn=tn+1
    else :
      tp=tp+1
  else :
    if t[i]==0:
      fn=fn+1
    else :
      fp=fp+1
a=(tp+tn)/(tp+tn+fp+fn)
print("accuracy(test cases): "+ str(a*100)+"%")
```

```
accuracy(test cases): 100.0%

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.p
y:2: RuntimeWarning: overflow encountered in exp
```

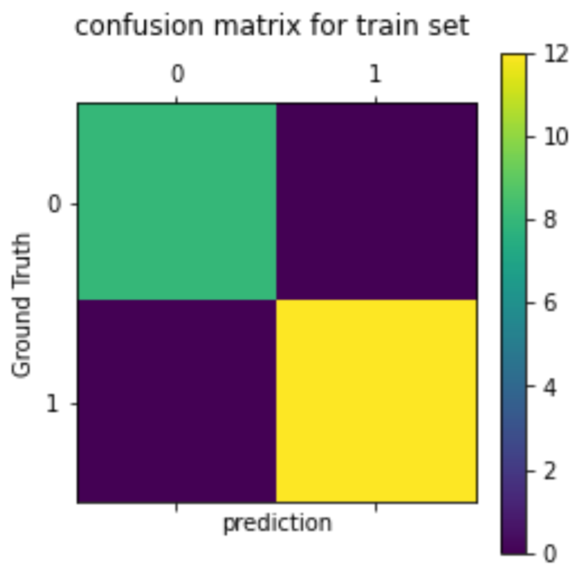
In [38]:

```
confusion=np.array([[tn1,fp1],[fn1,tp1]])
plt.matshow(confusion,0,cmap='inferno')
plt.colorbar()
plt.title("confusion matrix for train set \n")
plt.xlabel("prediction")
plt.ylabel("Ground Truth")
plt.show()
prec=tp1/(tp1+fp1)
rec=tp1/(tp1+fn1)
F=2/((1/rec)+(1/prec))
print(confusion)
print("precision(train) : "+str(prec*100)+"%")
print("recall(train) : "+str(rec*100)+"%")
print("F1-score(train) : "+str(F*100)+"%")
```




```
[[32  0]
 [ 0 48]]
precision(train) : 100.0%
recall(train) : 100.0%
F1-score(train) : 100.0%
```

In [39]:
```
confusion=np.array([[tn,fp],[fn,tp]])
plt.matshow(confusion)
plt.colorbar()
plt.title("confusion matrix for train set \n")
plt.xlabel("prediction")
plt.ylabel("Ground Truth")
plt.show()
prec=tp/(tp+fp)
rec=tp/(tp+fn)
F=2/((1/rec)+(1/prec))
print(confusion)
print("precision(test) : "+str(prec*100)+"%")
print("recall(test) : "+str(rec*100)+"%")
print("F1-score(test) : "+str(F*100)+"%")
```




```
[[ 8  0]
 [ 0 12]]
precision(test) : 100.0%
recall(test) : 100.0%
F1-score(test) : 100.0%
```

ii) Here, ex2data2 is fitted in logistic regression implemented from scratch.Here after analysing the scattered plot, it required a circular decision boundary,drawing a ellipse to separate the two classes resulted in lesser accuracy around 70.As, ellipse will not include x1**x2*,using this term we can increase accuracy till 82.Here I had taken terms of x1 and x2 till order of 2.so X=[1 x1 x2 x1^2 x2^2 X1*x2],so after fitting X and Y , theta will be obtained of len=6.The accuracy after using 2nd order is obtained to be almost 82% for both train and test.Precision,recall and F1 are printed below.Confusion matrix is been plotted using matshow function and for exact values matrix is printed.

In [97]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import scipy.linalg as la
import math
from scipy.fftpack import fft,fftfreq
from scipy.linalg import toeplitz
from matplotlib import animation
from sklearn.model_selection import train_test_split

#1
df = pd.read_csv('ex2data2-logisticc.csv')
X=df.iloc[:,0:2]
Y=df.iloc[:,2]
X1=X.copy()
meanx=X1.mean()
stdx=X1.std()
X1=np.array(X1)

X=np.c_[ np.ones(X.shape[0]),X ]
```

In [98]:
```
def g(theta2,X2):
    XT=np.dot(X2,theta2)
    H=1/(1+np.exp(-XT))
    return H
def J(H2,X3,T,Y2):
    error=H2-Y2
    return np.dot(X3.T,error)


theta=np.zeros(6)
alpha=0.01
iterations=1000000
X=np.c_[ X,np.ones(X.shape[0]) ]
for i in range(len( Y)):
    X[i][3]=X[i][1]**2
X=np.c_[ X,np.ones(X.shape[0]) ]
for i in range(len(Y)):
    X[i][4]=X[i][2]**2
X=np.c_[ X,np.ones(X.shape[0]) ]
for i in range(len(Y)):
    X[i][5]=X[i][2]*X[i][1]
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=
0.2)

for i in range(iterations):
    H=g(theta,X_train)
    JJ=J(H,X_train,theta,Y_train)

    theta=theta-(alpha/len(Y_train))*JJ
print(theta)
```

```
[  5.40995188   3.02816459   3.66022205 -12.07374502 -11.41
989567
  -7.16959293]
```

In [99]:

```
z = np.dot(X_train, theta.T)
h = 1 / (1 + np.exp(-z))
t=[]
tp1=0
tn1=0
fp1=0
fn1=0
Y_train=np.array(Y_train)
h=h.reshape(len(Y_train),)
for i in range(len(Y_train)):
    if h[i]>=0.5:
        t.append(1)
    else:
        t.append(0)
t=np.array(t)
for i in range(len(Y_train)):
  if Y_train[i]==t[i]:
    if t[i]==0:
      tn1=tn1+1
    else :
      tp1=tp1+1
  else :
    if t[i]==0:
      fn1=fn1+1
    else :
      fp1=fp1+1
a=(tp1+tn1)/(tp1+tn1+fp1+fn1)
print("accuracy(train cases) : "+ str(a*100)+"%")
```

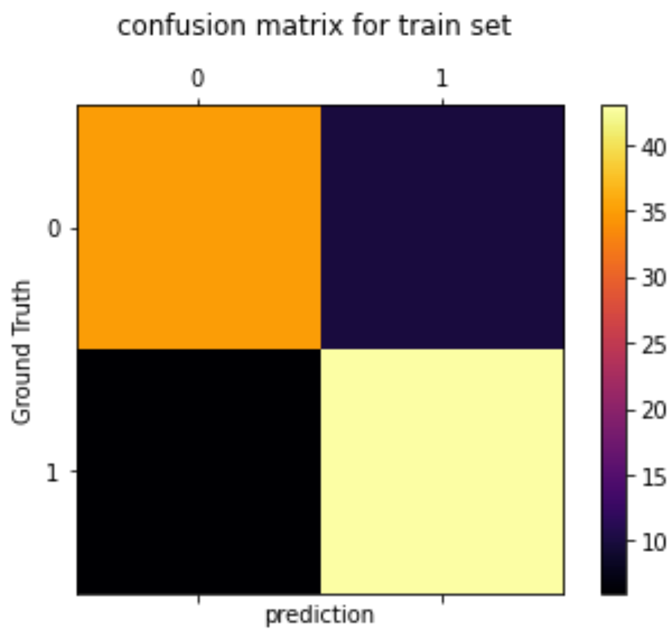
```
accuracy(train cases) : 82.97872340425532%
```

In [100]:

```
z = np.dot(X_test, theta.T)
h = 1 / (1 + np.exp(-z))
t=[]
tp=0
tn=0
fp=0
fn=0

Y_test=np.array(Y_test)
h=h.reshape(len(Y_test),)
for i in range(len(Y_test)):
    if h[i]>=0.5:
        t.append(1)
    else:
        t.append(0)
t=np.array(t)
for i in range(len(Y_test)):
  if Y_test[i]==t[i]:
    if t[i]==0:
      tn=tn+1
    else :
      tp=tp+1
  else :
    if t[i]==0:
      fn=fn+1
    else :
      fp=fp+1
a=(tp+tn)/(tp+tn+fp+fn)
print("accuracy(test cases): "+ str(a*100)+"%")
```

```
accuracy(test cases): 83.33333333333334%
```

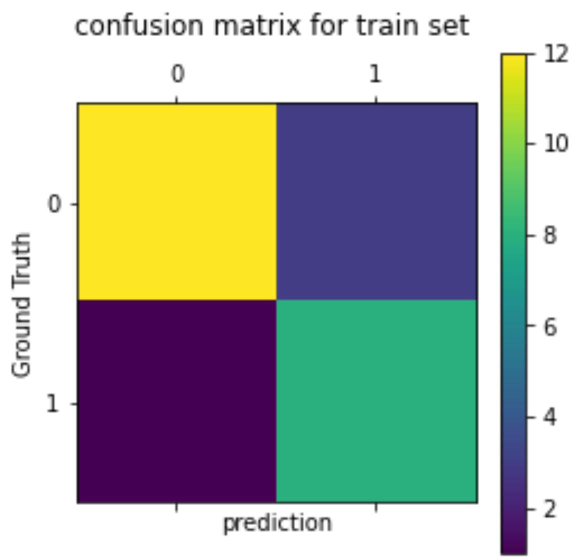
In [101]:
```
confusion=np.array([[tn1,fp1],[fn1,tp1]])
plt.matshow(confusion,0,cmap='inferno')
plt.colorbar()
plt.title("confusion matrix for train set \n")
plt.xlabel("prediction")
plt.ylabel("Ground Truth")
plt.show()
prec=tp1/(tp1+fp1)
rec=tp1/(tp1+fn1)
F=2/((1/rec)+(1/prec))
print(confusion)
print("precision(train) : "+str(prec*100)+"%")
print("recall(train) : "+str(rec*100)+"%")
print("F1-score(train) : "+str(F*100)+"%")
```




```
[[35 10]
 [ 6 43]]
precision(train) : 81.13207547169812%
recall(train) : 87.75510204081633%
F1-score(train) : 84.31372549019609%
```

In [102]:

```
confusion=np.array([[tn,fp],[fn,tp]])
plt.matshow(confusion)
plt.colorbar()
plt.title("confusion matrix for train set \n")
plt.xlabel("prediction")
plt.ylabel("Ground Truth")
plt.show()
prec=tp/(tp+fp)
rec=tp/(tp+fn)
F=2/((1/rec)+(1/prec))
print(confusion)
print("precision(test) : "+str(prec*100)+"%")
print("recall(test) : "+str(rec*100)+"%")
print("F1-score(test) : "+str(F*100)+"%")
```




```
[[12  3]
 [ 1  8]]
precision(test) : 72.72727272727273%
recall(test) : 88.88888888888889%
F1-score(test) : 80.0%
```