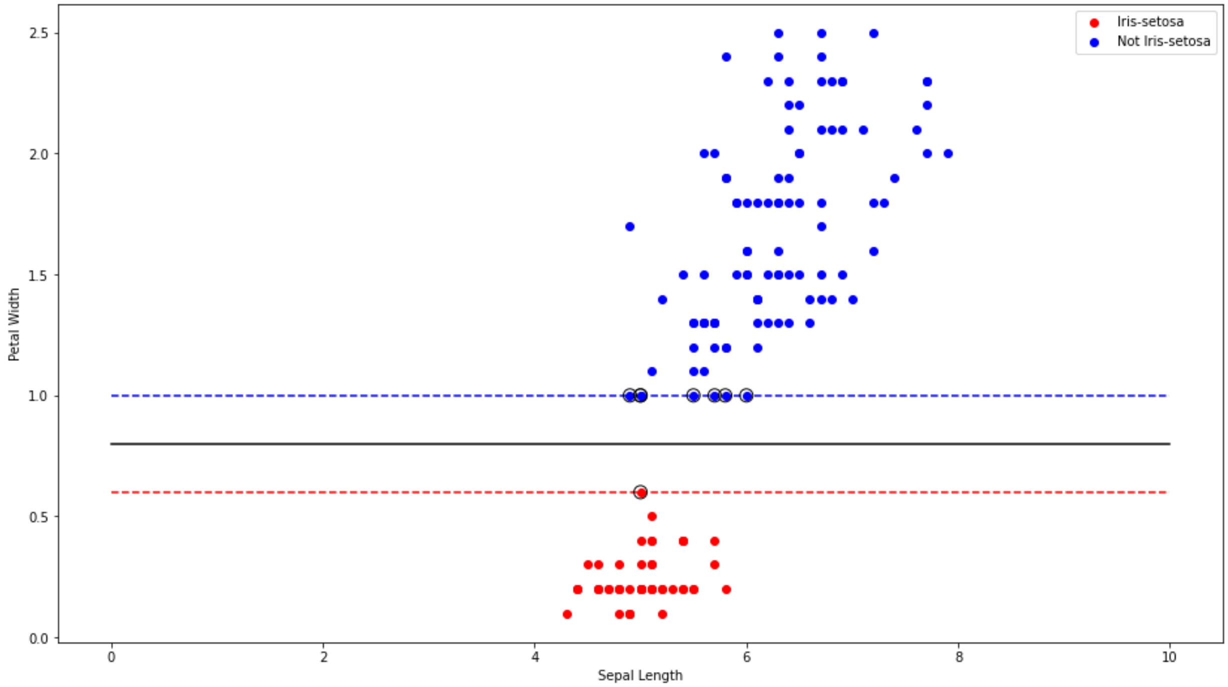
LAB 7

SVM using cvxpy

Here, i have implemented svm using cvxpy , by passing constraints and objective function.After that , iris dataset is fitted in the model.The support vectors ,decision boundary and margins are plotted.

In [144]:
```python
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
import cvxpy as cp



def hardmargin(X,Y):
  beta=cp.Variable(2)
  beta0=cp.Variable()
  constraints=[(Y[i]*(beta.T@X[i] + beta0)) >= 1  for i in range(Y.shape[0])]
  objective=cp.pnorm(beta,p=2)**2
  prob = cp.Problem(cp.Minimize(objective), constraints)
  prob.solve()
  return beta.value,beta0.value
df = pd.read_csv('Iris.csv')
X=df.iloc[:,[1,4]]
Y=df.iloc[:,5]
X=np.array(X)
Y=np.array(Y)
Y[np.where(Y=='Iris-setosa')]=1
Y[np.where(Y!=1)]=-1
w,w0=hardmargin(X,Y)
print("beta = ")
print(w)
print("beta0 = "+ str(w0))

w1=np.array(w.copy())
w01=w0.copy()
a=X[np.where(Y==1)]
b=X[np.where(Y!=1)]
c=X@w1+w01

d=[]

for i in range(X.shape[0]):
  if (c[i]-1<=0.00001 and c[i]>1 ) or (1-c[i]<=0.00001 and c[i]<1) :
    d.append(X[i])
  if (c[i]+1<=0.00001 and c[i]>-1 ) or (-1-c[i]<=0.00001 and c[i]<-1) :
    d.append(X[i])
d=np.array(d)
plt.figure(figsize=(16,9))
x= np.linspace(0, 10, 100)
plt.plot(x, (-w0 - (w[0]*x))/w[1] , 'k')
plt.plot(x, (-w0- (w[0]*x)+1)/w[1] ,'r--')
plt.plot(x, (-w0 - (w[0]*x)-1)/w[1] , 'b--')
plt.scatter(d[:, 0],d[:, 1], s=100,facecolors="none", zorder=10, edgecolors="k")

plt.scatter(a[:,0],a[:,1],c='red',alpha=1,label='Iris-setosa')
plt.scatter(b[:,0],b[:,1],c='blue',alpha=1,label='Not Iris-setosa')
plt.legend()
plt.xlabel('Sepal Length')
plt.ylabel('Petal Width')
plt.show()
```

```
beta =
[-2.19658355e-10 -5.00000000e+00]
beta0 = 4.000000000705559
```



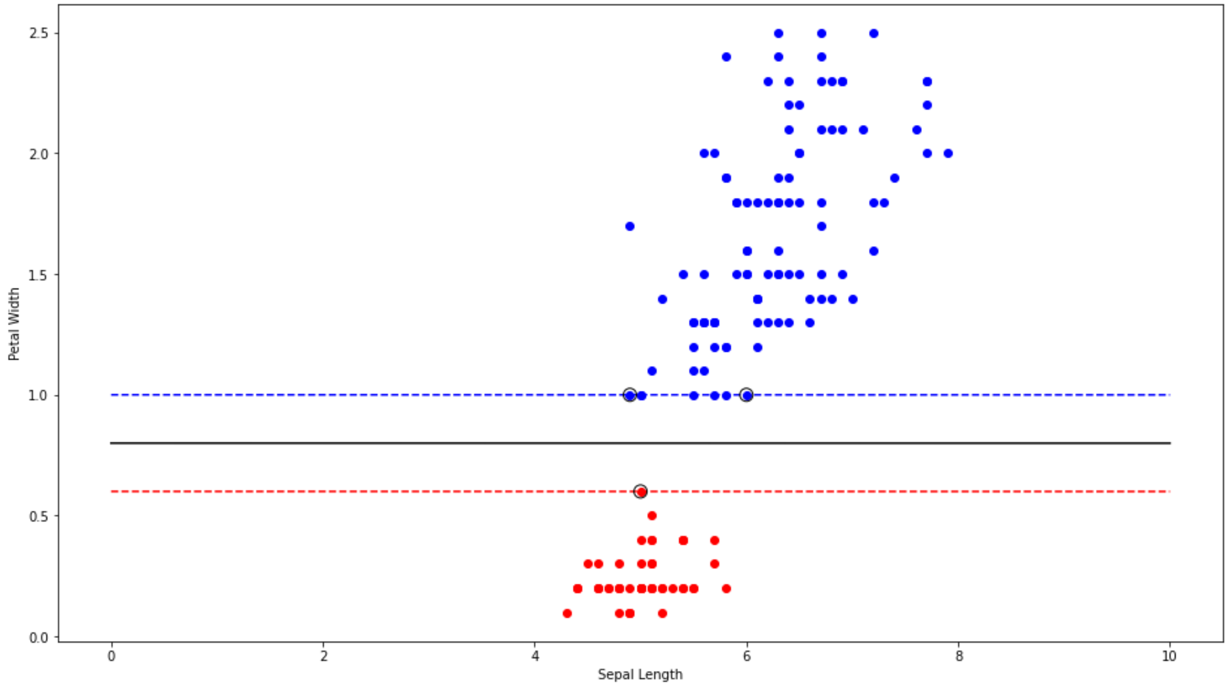
SVM using sklearn library:

Here , I have implemented svm from sklearn library.Here, linear kernel is considered and C is taken very large so that model takes support vectors as close as possible.Overall boundary is similar to the svm implemented using cvxpy.

In [ ]:

```python
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
import cvxpy as cp
from cvxpy import *

from sklearn.svm import SVC
df = pd.read_csv('Iris.csv')
X=df.iloc[:,[1,4]]
Y=df.iloc[:,5]
X=np.array(X)
Y=np.array(Y)
Y[np.where(Y=='Iris-setosa')]=1
Y[np.where(Y!=1)]=-1

Y=Y.astype('int')

svm = SVC(kernel='linear',C=1E10)
svm.fit(X,Y )
w=svm.coef_.T
w0=svm.intercept_
print(w)
print(w0)

a=X[np.where(Y==1)]
b=X[np.where(Y!=1)]
plt.figure(figsize=(16,9))
x= np.linspace(0, 10, 100)
plt.plot(x, (-w0 - (w[0]*x))/w[1] , 'k')
plt.plot(x, (-w0- (w[0]*x)+1)/w[1] ,'r--')
plt.plot(x, (-w0 - (w[0]*x)-1)/w[1] , 'b--')
plt.scatter(a[:,0],a[:,1],c='red',alpha=1,label='Iris-setosa')
plt.scatter(b[:,0],b[:,1],c='blue',alpha=1,label='Not Iris-setosa')
print(svm.support_vectors_)
plt.scatter(svm.support_vectors_[:, 0],svm.support_vectors_[:, 1], s=100,facecolors="none"
, zorder=10, edgecolors="k")
#plt.scatter(svm.support_vectors_[:, 0],svm.support_vectors_[:, 1],s=80, linewidth=0.05, f
acecolors='k');plt.legend()
plt.xlabel('Sepal Length')
plt.ylabel('Petal Width')
plt.show()
```

```
[[-1.46160676e-04]
 [-4.99965020e+00]]
[4.00046664]
[[4.9 1. ]
 [6.  1. ]
 [5.  0.6]]
```



If I remove points other than the support vectors,the decision boundary will remain same.As lagrange multiplier for non support vectors will be 0, so removing them wouldnt affect the beta*.Hence,the decision boundary will remain same.