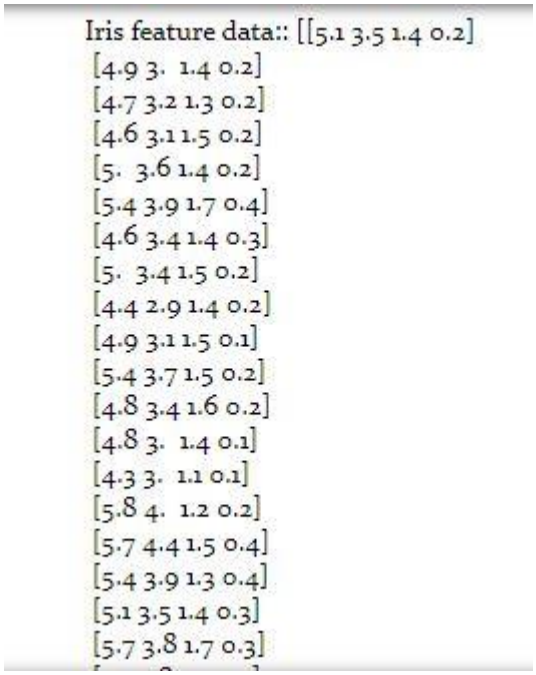
Name : Devashish Prasad
Roll no : 43320
Batch : Q11
Title : : Implement SVM for performing classification and find its accuracy on the given data. (Using Python)
==========

#import libraries
```
from sklearn import datasets
from sklearn import svm
import numpy as np
import matplotlib.pyplot as plt
```

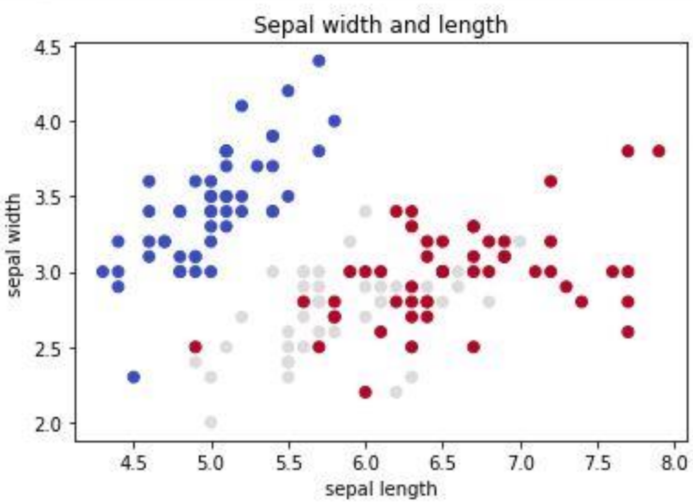
#Load dataset
```
iris_dataset=datasets.load_iris()
print("Iris feature data::",iris_dataset['data'])
print("Iris target::",iris_dataset['target'])
```



```
[6.3 3.4 5.6 2.4]
[6.4 3.1 5.5 1.8]
[6.  3.  4.8 1.8]
[6.9 3.1 5.4 2.1]
[6.7 3.1 5.6 2.4]
[6.9 3.1 5.1 2.3]
[5.8 2.7 5.1 1.9]
[6.8 3.2 5.9 2.3]
[6.7 3.3 5.7 2.5]
[6.7 3.  5.2 2.3]
[6.3 2.5 5.  1.9]
[6.5 3.  5.2 2. ]
[6.2 3.4 5.4 2.3]
[5.9 3.  5.1 1.8]]
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]
```

#Visualizing the sepal and target classes

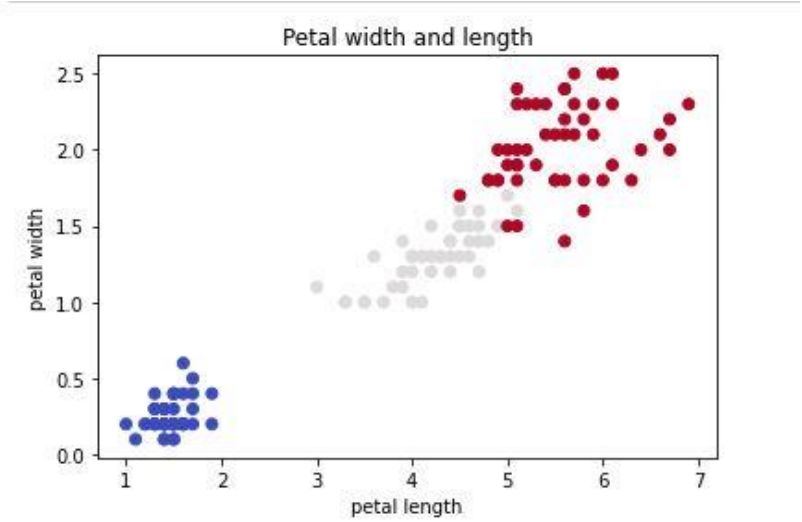
```
def visualize_sepal_data():
    iris=datasets.load_iris()
    X=iris.data[:,:2] #Take only the first two features
    y=iris.target
    plt.scatter(X[:,0],X[:,1],c=y,cmap=plt.cm.coolwarm)
    plt.xlabel('sepal length')
    plt.ylabel('sepal width')
    plt.title('Sepal width and length')
    plt.show()
visualize_sepal_data()
```



#Visualizing the petal and target classes

```
def visualize_petal_data():
    iris=datasets.load_iris()
    X=iris.data[:,2:] #Take only the last two features
```

```
    y=iris.target
    plt.scatter(X[:,0],X[:,1],c=y,cmap=plt.cm.coolwarm)
    plt.xlabel('petal length')
    plt.ylabel('petal width')
    plt.title('Petal width and length')
    plt.show()
visualize_petal_data()
```



#Modelling different kernal svm classifiers

```
iris=datasets.load_iris()
X=iris.data[:,:2]  #We only take sepal two features
y=iris.target
C=1.0 #svm regularization parameter

#SVM with linear kernel
svc=svm.SVC(kernel='linear',C=C).fit(X,y)
svc.score(X, y)

#Linear SVC(linear kernel)
lin_svc=svm.LinearSVC(C=C).fit(X,y)
lin_svc.score(X,y)

#SVC with RBF kernel
rbf_svc=svm.SVC(kernel='rbf',gamma=0.7,C=C).fit(X,y)
rbf_svc.score(X,y)

#Visualize the model svm classifiers
h=0.2
#Creating mesh to plot in
x_min,x_max=X[:,0].min()-1,X[:,0].max()+1
y_min,y_max=X[:,1].min()-1,X[:,1].max()+1
xx,yy=np.meshgrid(np.arange(x_min,x_max,h),np.arange(y_min,y_max,h))
np.arange(y_min,y_max,h)
```

```
titles=['SVC with linear kernel','LinearSVC(linear kernel)','SVC with RBF kernel']

for i,elf in enumerate((svc,lin_svc,rbf_svc)):
    plt.subplot(2,2,i+1)
    plt.subplots_adjust(wspace=0.4,hspace=0.4)
    Z=elf.predict(np.c_[xx.ravel(),yy.ravel()])

    #Put the result into a color plot
    Z=Z.reshape(xx.shape)
    plt.contourf(xx,yy,Z,cmap=plt.cm.coolwarm,alpha=0.8)

    #plot also the training points
    plt.scatter(X[:,0],X[:,1],c=y,cmap=plt.cm.coolwarm)
    plt.xlabel('sepal length')
    plt.ylabel('sepal width')
    plt.xlim(xx.min(),xx.max())
    plt.ylim(yy.min(),yy.max())
    plt.xticks(())
    plt.yticks(())
    plt.title(titles[i])
```