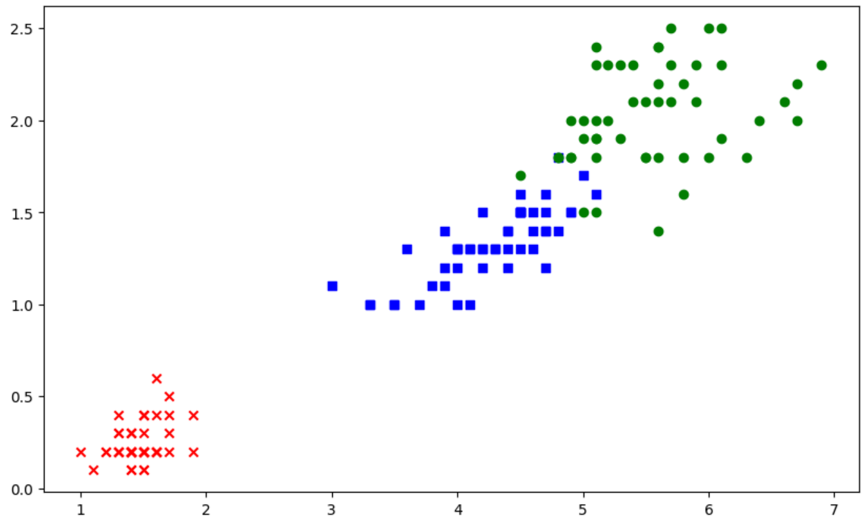
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn import datasets
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
%pylab inline
pylab.rcParams['figure.figsize']=(10,6)
iris_data=datasets.load_iris()
x=iris_data.data[:,[2,3]]
y=iris_data.target
iris_dataframe = pd.DataFrame(iris_data.data[:, [2, 3]],columns=iris_data.feature_names[2:])
print(iris_dataframe.head())
print('\n'+'UniqueLabels contained in this data are'+str(np.unique(y)))
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)
print('The training set contains {} samples and the test set contains {}samples'.format(x_train.shape[0],x_test.shape[0]))
```

```
Populating the interactive namespace from numpy and matplotlib
   petal length (cm)  petal width (cm)
0                1.4               0.2
1                1.4               0.2
2                1.3               0.2
3                1.5               0.2
4                1.4               0.2

UniqueLabels contained in this data are[0 1 2]
The training set contains 105 samples and the test set contains 45samples
```

```
markers = ('x', 's', 'o')
colors = ('red', 'blue', 'green')
cmap = ListedColormap(colors[:len(np.unique(y_test))])
for idx, cl in enumerate(np.unique(y)):
  plt.scatter(x=x[y==cl,0],y=x[y==cl,1],c=cmap(idx),marker=markers[idx],label=cl)
```

```
<ipython-input-9-97065b55afe1>:5: UserWarning: *c* argument looks like a single numer
  plt.scatter(x=x[y==cl,0],y=x[y==cl,1],c=cmap(idx),marker=markers[idx],label=cl)
```

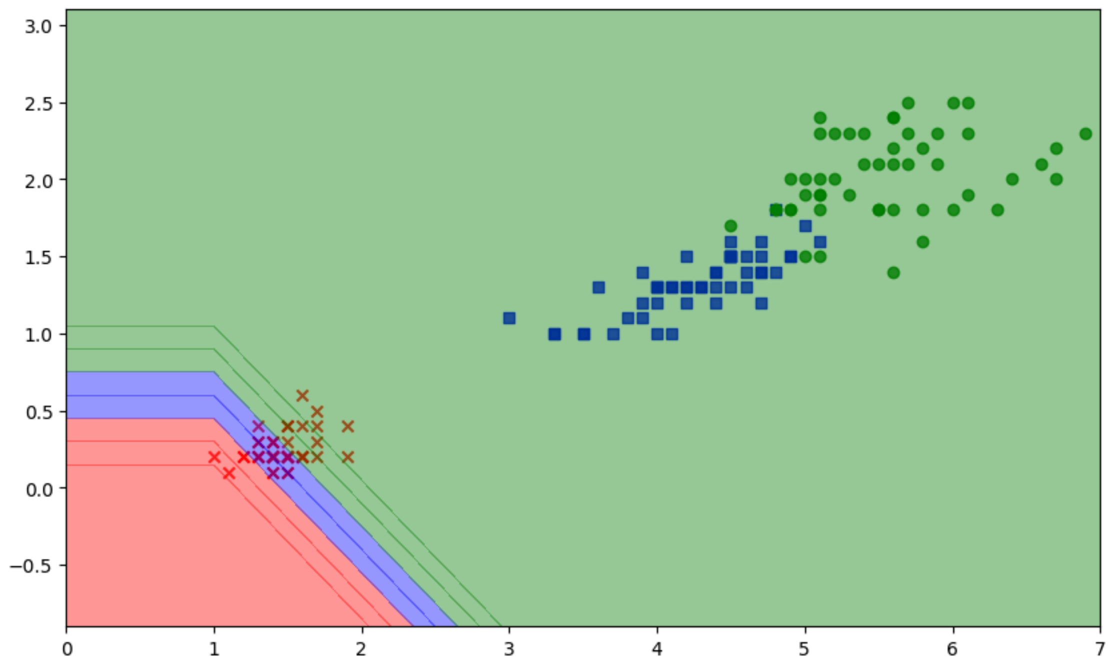


```
standard_scaler=StandardScaler()
standard_scaler.fit(x_train)
x_train_standard = standard_scaler.transform(x_train)
x_test_standard = standard_scaler.transform(x_test)
print('The first five rows after standardisation look like this:\n')
print(pd.DataFrame(x_train_standard, columns=iris_dataframe.columns).head())
SVM = SVC(kernel='rbf',random_state=0,gamma=.10,C=1.0)
SVM.fit(x_train_standard,y_train)
print('Accuracy of our SVM model on the training data is {:.2f} out of 1'.format(SVM.score(x_train_standard,y_train)))
print('Accuracy of our SVM model on the test data is {:.2f} out of 1'.format(SVM.score(x_test_standard,y_test)))
```

```
The first five rows after standardisation look like this:
   petal length (cm)  petal width (cm)
0          -0.182950         -0.293181
1           0.930661          0.737246
2           1.042022          1.638870
3           0.652258          0.350836
4           1.097702          0.737246
Accuracy of our SVM model on the training data is 0.95 out of 1
Accuracy of our SVM model on the test data is 0.98 out of 1
```

```
import warnings
def versiontuple(version):
  return tuple(map(int, (version.split("."))))
def decision_plot(x,y,classifier,test_idx=None,resolution=0.02):
  markers=('s','x','o','^','v')
  colors=('red','blue','green','gray','cyan')
  cmap=ListedColormap(colors[:len(np.unique(y))])
  x1min,x1max=x[:,0].min()-1,x[:,0].max()+1
  x2min,x2max=x[:,1].min()-1,x[:,1].max()+1
  xx1,xx2=np.meshgrid(np.arange(x1min, x1max,resolution),np.arange(x2min,x2max,resolution))
  z=classifier.predict(np.array([xx1.ravel(),xx2.ravel()]).T)
  z=z.reshape(xx1.shape)
  plt.contourf(xx1,xx2,z,alpha=0.4,cmap=cmap)
  plt.xlim(xx1.min(),xx1.max())
  plt.ylim(xx2.min(),xx2.max())
for idx,cl in enumerate(np.unique(y)):
  plt.scatter(x=x[y==cl,0],y=x[y==cl,1],alpha=0.8,c=cmap(idx),marker=markers[idx],label=cl)
decision_plot(x,y,classifier=SVM,test_idx=None,resolution=01.0)
```

```
<ipython-input-44-b211bd3cd3fc>:17: UserWarning: *c* argument looks like a single numeric RGB or RGBA sequence, which should be avoi
  plt.scatter(x=x[y==cl,0],y=x[y==cl,1],alpha=0.8,c=cmap(idx),marker=markers[idx],label=cl)
```


3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
0
1
2
3
4
5
6
7