Mohammed Meraj Mohammed Ashfaque

ML 6 : SVM - TY Computer

Support Vector Machine (SVM)

Implemented Test Your Knowledge problem from practical

Importing the libraries

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn import datasets
%pylab inline
pylab.rcParams['figure.figsize'] = (10, 6)
```

```
%pylab is deprecated, use %matplotlib inline and import the required libraries.
Populating the interactive namespace from numpy and matplotlib
```

Importing the dataset

In [2]:
```
iris = datasets.load_iris()
print(iris)
# We'll use the petal length and width only for this analysis
X = iris.data[:, [2, 3]]
y = iris.target
```

```
{'data': array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 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. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 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. , 0.2],
       [5.1, 3.3, 1.7, 0.5],
       [4.8, 3.4, 1.9, 0.2],
       [5. , 3. , 1.6, 0.2],
       [5. , 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, 0.2],
       [5. , 3.2, 1.2, 0.2],
       [5.5, 3.5, 1.3, 0.2],
       [4.9, 3.6, 1.4, 0.1],
       [4.4, 3. , 1.3, 0.2],
       [5.1, 3.4, 1.5, 0.2],
       [5. , 3.5, 1.3, 0.3],
       [4.5, 2.3, 1.3, 0.3],
       [4.4, 3.2, 1.3, 0.2],
       [5. , 3.5, 1.6, 0.6],
       [5.1, 3.8, 1.9, 0.4],
       [4.8, 3. , 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. , 3.3, 1.4, 0.2],
[7. , 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. , 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. ],
[6.6, 2.9, 4.6, 1.3],
[5.2, 2.7, 3.9, 1.4],
[5. , 2. , 3.5, 1. ],
[5.9, 3. , 4.2, 1.5],
[6. , 2.2, 4. , 1. ],
[6.1, 2.9, 4.7, 1.4],
[5.6, 2.9, 3.6, 1.3],
[6.7, 3.1, 4.4, 1.4],
[5.6, 3. , 4.5, 1.5],
[5.8, 2.7, 4.1, 1. ],
[6.2, 2.2, 4.5, 1.5],
[5.6, 2.5, 3.9, 1.1],
[5.9, 3.2, 4.8, 1.8],
[6.1, 2.8, 4. , 1.3],
[6.3, 2.5, 4.9, 1.5],
[6.1, 2.8, 4.7, 1.2],
[6.4, 2.9, 4.3, 1.3],
[6.6, 3. , 4.4, 1.4],
[6.8, 2.8, 4.8, 1.4],
[6.7, 3. , 5. , 1.7],
[6. , 2.9, 4.5, 1.5],
[5.7, 2.6, 3.5, 1. ],
[5.5, 2.4, 3.8, 1.1],
[5.5, 2.4, 3.7, 1. ],
[5.8, 2.7, 3.9, 1.2],
[6. , 2.7, 5.1, 1.6],
[5.4, 3. , 4.5, 1.5],
[6. , 3.4, 4.5, 1.6],
[6.7, 3.1, 4.7, 1.5],
[6.3, 2.3, 4.4, 1.3],
[5.6, 3. , 4.1, 1.3],
[5.5, 2.5, 4. , 1.3],
[5.5, 2.6, 4.4, 1.2],
[6.1, 3. , 4.6, 1.4],
[5.8, 2.6, 4. , 1.2],
[5. , 2.3, 3.3, 1. ],
[5.6, 2.7, 4.2, 1.3],
[5.7, 3. , 4.2, 1.2],
[5.7, 2.9, 4.2, 1.3],
[6.2, 2.9, 4.3, 1.3],
[5.1, 2.5, 3. , 1.1],
[5.7, 2.8, 4.1, 1.3],
[6.3, 3.3, 6. , 2.5],
[5.8, 2.7, 5.1, 1.9],
[7.1, 3. , 5.9, 2.1],
[6.3, 2.9, 5.6, 1.8],
[6.5, 3. , 5.8, 2.2],
[7.6, 3. , 6.6, 2.1],
[4.9, 2.5, 4.5, 1.7],
[7.3, 2.9, 6.3, 1.8],
[6.7, 2.5, 5.8, 1.8],
[7.2, 3.6, 6.1, 2.5],
[6.5, 3.2, 5.1, 2. ],
[6.4, 2.7, 5.3, 1.9],
[6.8, 3. , 5.5, 2.1],
[5.7, 2.5, 5. , 2. ],
[5.8, 2.8, 5.1, 2.4],
[6.4, 3.2, 5.3, 2.3],
[6.5, 3. , 5.5, 1.8],
[7.7, 3.8, 6.7, 2.2],
[7.7, 2.6, 6.9, 2.3],
[6. , 2.2, 5. , 1.5],
[6.9, 3.2, 5.7, 2.3],
[5.6, 2.8, 4.9, 2. ],
[7.7, 2.8, 6.7, 2. ],
[6.3, 2.7, 4.9, 1.8],
[6.7, 3.3, 5.7, 2.1],
[7.2, 3.2, 6. , 1.8],
[6.2, 2.8, 4.8, 1.8],
[6.1, 3. , 4.9, 1.8],
[6.4, 2.8, 5.6, 2.1],
[7.2, 3. , 5.8, 1.6],
[7.4, 2.8, 6.1, 1.9],
```

```
       [7.9, 3.8, 6.4, 2. ],
       [6.4, 2.8, 5.6, 2.2],
       [6.3, 2.8, 5.1, 1.5],
       [6.1, 2.6, 5.6, 1.4],
       [7.7, 3. , 6.1, 2.3],
       [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]]), 'target': array([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]), 'frame': None, 'target_names': array(['setosa', '
versicolor', 'virginica'], dtype='<U10'), 'DESCR': '.. _iris_dataset:\n\nIris plants dataset\n-----------------
--\n\n**Data Set Characteristics:**\n\n:Number of Instances: 150 (50 in each of three classes)\n:Number of Attri
butes: 4 numeric, predictive attributes and the class\n:Attribute Information:\n    - sepal length in cm\n    -
sepal width in cm\n    - petal length in cm\n    - petal width in cm\n    - class:\n            - Iris-Setosa\n
- Iris-Versicolour\n            - Iris-Virginica\n\n:Summary Statistics:\n\n============== ==== ==== ======= ===
== ====================\n                Min  Max   Mean    SD   Class Correlation\n============== ==== ==== ===
==== ===== ====================\nsepal length:   4.3  7.9   5.84   0.83    0.7826\nsepal width:    2.0  4.4   3.
05   0.43   -0.4194\npetal length:   1.0  6.9   3.76   1.76    0.9490  (high!)\npetal width:    0.1  2.5   1.20
0.76    0.9565  (high!)\n============== ==== ==== ======= ===== ====================\n\n:Missing Attribute Value
s: None\n:Class Distribution: 33.3% for each of 3 classes.\n:Creator: R.A. Fisher\n:Donor: Michael Marshall (MAR
SHALL%PLU@io.arc.nasa.gov)\n:Date: July, 1988\n\nThe famous Iris database, first used by Sir R.A. Fisher. The da
taset is taken\nfrom Fisher\'s paper. Note that it\'s the same as in R, but not as in the UCI\nMachine Learning
Repository, which has two wrong data points.\n\nThis is perhaps the best known database to be found in the\npatt
ern recognition literature.  Fisher\'s paper is a classic in the field and\nis referenced frequently to this day
.  (See Duda & Hart, for example.)  The\ndata set contains 3 classes of 50 instances each, where each class refe
rs to a\ntype of iris plant.  One class is linearly separable from the other 2; the\nlatter are NOT linearly sep
arable from each other.\n\n.. dropdown:: References\n\n  - Fisher, R.A. "The use of multiple measurements in tax
onomic problems"\n    Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to\n    Mathematical S
tatistics" (John Wiley, NY, 1950).\n  - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysi
s.\n    (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.\n  - Dasarathy, B.V. (1980) "Nosing Ar
ound the Neighborhood: A New System\n    Structure and Classification Rule for Recognition in Partially Exposed\
n    Environments".  IEEE Transactions on Pattern Analysis and Machine\n    Intelligence, Vol. PAMI-2, No. 1, 67
-71.\n  - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions\n    on Information Theory,
May 1972, 431-433.\n  - See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II\n    conceptual c
lustering system finds 3 classes in the data.\n  - Many, many more ...\n', 'feature_names': ['sepal length (cm)'
, 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)'], 'filename': 'iris.csv', 'data_module': 'sklearn.
datasets.data'}
```

Exploratory Data Analysis

Place the iris data into a pandas dataframe

In [3]:

```python
iris_df = pd.DataFrame(iris.data[:, [2, 3]], columns=iris.feature_names[2:])

# View the first 5 rows of the data
print(iris_df.head())

# Print the unique labels of the dataset
print('\n' + 'The unique labels in this data are ' + str(np.unique(y)))
```

```
   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

The unique labels in this data are [0 1 2]
```

Splitting the dataset into the Training set and Test set

In [4]:

```python
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.3, random_state=0)
```

```
print('There are {} samples in the training set and {} samples in the test set'.format(
X_train.shape[0], X_test.shape[0]))
```

```
There are 105 samples in the training set and 45 samples in the test set
```

Feature Scaling

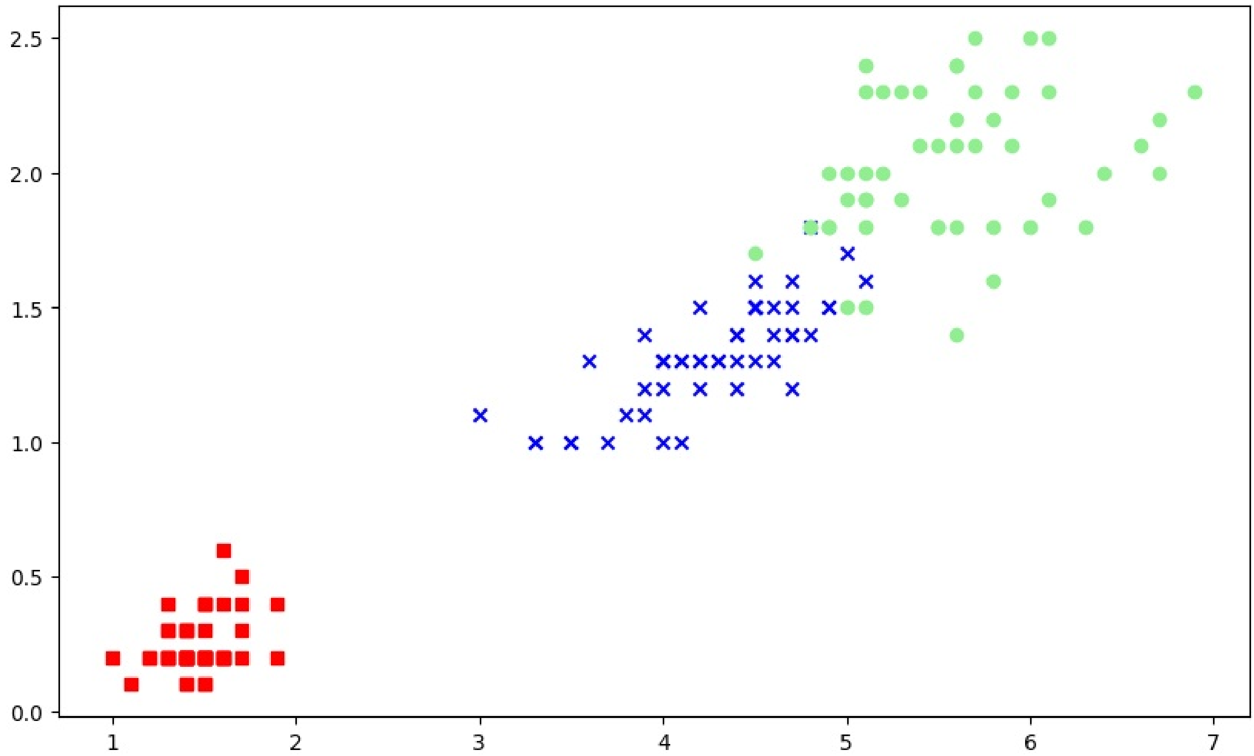
In [5]:
```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

sc.fit(X_train)

X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

print('After standardizing our features, the first 5 rows of our data now look like this:\n')
print(pd.DataFrame(X_train_std, columns=iris_df.columns).head())
```

```
After standardizing our features, the first 5 rows of our data now 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
```

Plot the original Data

In [6]:
```
from matplotlib.colors import ListedColormap
import matplotlib.pyplot as plt

markers = ('s', 'x', 'o')
colors = ('red', 'blue', 'lightgreen')
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)
```

```
C:\Users\Mohammed Meraj\AppData\Local\Temp\ipykernel_12804\4156015251.py:8: UserWarning: *c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2D array with a single row if you intend to specify the same RGB or RGBA value for all points.
  plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],
```



If we plot the original data, we can see that one of the classes is linearly separable, but the other two are not.

Training the SVM model on the Training set

In [9]:
```
from sklearn.svm import SVC

svm = SVC(kernel='poly', random_state=4, degree=2, gamma=.10, C=1.0)
svm.fit(X_train_std, y_train)
```

Out[9]:
```
SVC
SVC(degree=2, gamma=0.1, kernel='poly', random_state=4)
```

Dispplay the support Vectors of model

In [10]:
```
print("Support Vector for model are :",svm.support_vectors_)
```

```
Support Vector for model are : [[-1.24088089 -1.32360858]
 [-1.29656144 -1.32360858]
 [-1.51928365 -1.45241201]
 [-1.18520034 -1.32360858]
 [-1.29656144 -1.06600172]
 [-1.29656144 -1.32360858]
 [-1.35224199 -1.19480515]
 [-1.35224199 -1.45241201]
 [-1.24088089 -1.32360858]
 [-1.24088089 -1.32360858]
 [-1.40792255 -1.19480515]
 [-1.35224199 -1.32360858]
 [-1.40792255 -1.32360858]
 [-1.35224199 -1.32360858]
 [-1.35224199 -1.19480515]
 [-1.40792255 -1.32360858]
 [-1.18520034 -1.06600172]
 [-1.35224199 -1.32360858]
 [-1.29656144 -1.32360858]
 [-1.35224199 -1.32360858]
 [-1.4636031  -1.32360858]
 [-1.18520034 -0.93719829]
 [-1.29656144 -1.06600172]
 [-1.29656144 -1.45241201]
 [-1.4636031  -1.32360858]
 [-1.29656144 -1.19480515]
 [-1.24088089 -1.32360858]
 [-1.35224199 -1.32360858]
 [-1.24088089 -1.32360858]
 [-1.29656144 -1.32360858]
 [-1.40792255 -1.32360858]
 [-1.29656144 -1.06600172]
 [-1.29656144 -1.45241201]
 [-1.35224199 -1.32360858]
 [-0.18295039 -0.29318114]
 [ 0.03977182 -0.16437771]
 [ 0.48521625  0.47963944]
 [-0.01590873 -0.16437771]
 [ 0.48521625  0.22203258]
 [ 0.37385514  0.47963944]
 [ 0.20681348  0.09322915]
 [-0.12726983  0.09322915]
 [ 0.09545238  0.09322915]
 [ 0.42953569  0.22203258]
 [ 0.20681348 -0.03557428]
 [ 0.59657735  0.35083601]
 [-0.46135315 -0.16437771]
 [ 0.37385514  0.35083601]
 [ 0.20681348  0.09322915]
 [ 0.09545238  0.09322915]
 [ 0.37385514  0.09322915]
 [ 0.31817459  0.22203258]
 [ 0.26249403  0.09322915]
 [-0.29431149 -0.29318114]
 [ 0.31817459  0.22203258]
 [ 0.15113293  0.09322915]
 [ 0.03977182 -0.03557428]
 [-0.18295039 -0.29318114]
 [ 0.59657735  0.35083601]
 [ 0.6522579   0.60844287]
 [-0.07158928 -0.29318114]
 [ 0.42953569  0.09322915]
 [ 0.15113293  0.09322915]
 [ 0.5408968   0.7372463 ]
 [ 0.31817459  0.09322915]
 [ 0.15113293 -0.29318114]
```

```
 [ 0.93066067  0.7372463 ]
 [ 1.04202177  1.63887031]
 [ 0.6522579   0.35083601]
 [ 1.09770233  0.7372463 ]
 [ 1.26474398  1.38126345]
 [ 0.59657735  0.7372463 ]
 [ 0.70793846  0.35083601]
 [ 0.37385514  0.60844287]
 [ 0.6522579   0.86604973]
 [ 0.76361901  0.99485316]
 [ 0.70793846  0.86604973]
 [ 1.20906343  0.7372463 ]
 [ 0.81929956  0.86604973]
 [ 0.70793846  0.7372463 ]
 [ 1.26474398  1.63887031]
 [ 1.15338288  1.12365659]
 [ 1.04202177  1.38126345]
 [ 1.09770233  1.25246002]
 [ 0.98634122  1.12365659]
 [ 0.70793846  0.99485316]
 [ 1.04202177  1.12365659]
 [ 0.87498011  1.38126345]
 [ 0.6522579   0.99485316]
 [ 0.87498011  1.12365659]
 [ 1.09770233  0.47963944]
 [ 1.15338288  1.38126345]
 [ 1.54314675  1.12365659]
 [ 1.71018841  1.38126345]
 [ 1.43178564  0.99485316]
 [ 0.5408968   0.7372463 ]
 [ 1.5988273   0.99485316]
 [ 1.26474398  0.86604973]
 [ 0.98634122  1.51006688]
 [ 0.81929956  1.38126345]
 [ 0.76361901  1.38126345]
 [ 0.98634122  1.51006688]
 [ 0.70793846  0.86604973]
 [ 0.98634122  0.7372463 ]
 [ 1.5988273   1.25246002]]
```

In [20]:

```
print("Number of suppoort Vectors of each class 0 : - ",svm.n_support_[0])
print("Number of suppoort Vectors of each class 1 : - ",svm.n_support_[1])
print("Number of suppoort Vectors of each class 2 : - ",svm.n_support_[2])
```

```
Number of suppoort Vectors of each class 0 : -  34
Number of suppoort Vectors of each class 1 : -  32
Number of suppoort Vectors of each class 2 : -  39
```

In [21]:

```
print("Indices for support vectors are : ",svm.support_)
```

```
Indices for support vectors are :  [ 16  23  24  27  28  30  33  37  42  43  46  48  51  52  54  55  56  60
  61  62  65  66  68  73  75  77  78  80  89  93  98  99 100 104   0   5
   7   8  13  15  18  19  20  21  26  29  32  34  36  57  63  64  67  70
  72  82  83  84  87  88  90  94  95  96  97 102   1   2   3   4   6   9
  10  11  12  14  17  22  25  31  35  38  39  40  41  44  45  47  49  50
  53  58  59  69  71  74  76  79  81  85  86  91  92 101 103]
```

Finding Accuracy of model on Test and Train Set

In [22]:

```
print('The accuracy of the svm classifier on training data is {:.2f} out of 1'.format(svm.score(X_train_std, y_

print('The accuracy of the svm classifier on test data is {:.2f} out of 1'.format(svm.score(X_test_std, y_test)
```

```
The accuracy of the svm classifier on training data is 0.58 out of 1
The accuracy of the svm classifier on test data is 0.58 out of 1
```

Finding Accuracy of model on using confiusion matrix

In [23]:

```
from sklearn import metrics
from sklearn.metrics import confusion_matrix

confusion_matrix = metrics.confusion_matrix(y_test, svm.predict(X_test_std))
print(confusion_matrix)
```

```
[[ 1  0 15]
 [ 0 18  0]
 [ 0  4  7]]
```

In [24]:

```
Accuracy = metrics.accuracy_score(y_test, svm.predict(X_test_std))
Precision = metrics.precision_score(y_test, svm.predict(X_test_std),average='macro')
Sensitivity_recall = metrics.recall_score(y_test, svm.predict(X_test_std),average='macro')
Specificity = metrics.recall_score(y_test, svm.predict(X_test_std), pos_label=0,average='macro')
```

```
F1_score = metrics.f1_score(y_test, svm.predict(X_test_std),average='macro')
```

```
C:\Users\Mohammed Meraj\AppData\Roaming\Python\Python312\site-packages\sklearn\metrics\_classification.py:1618:
UserWarning: Note that pos_label (set to 0) is ignored when average != 'binary' (got 'macro'). You may use label
s=[pos_label] to specify a single positive class.
  warnings.warn(
```

In [25]:

```
#metrics
print({"Accuracy":Accuracy,
       "Precision":Precision,
       "Sensitivity_recall":Sensitivity_recall,
       "Specificity":Specificity,
       "F1_score":F1_score},end = "")
```

```
{'Accuracy': 0.5777777777777777, 'Precision': 0.7121212121212123, 'Sensitivity_recall': 0.5662878787878788, 'Spe
cificity': 0.5662878787878788, 'F1_score': 0.4806298276886512}
```

Create the function for Visualizing Testing and Training model

In [26]:

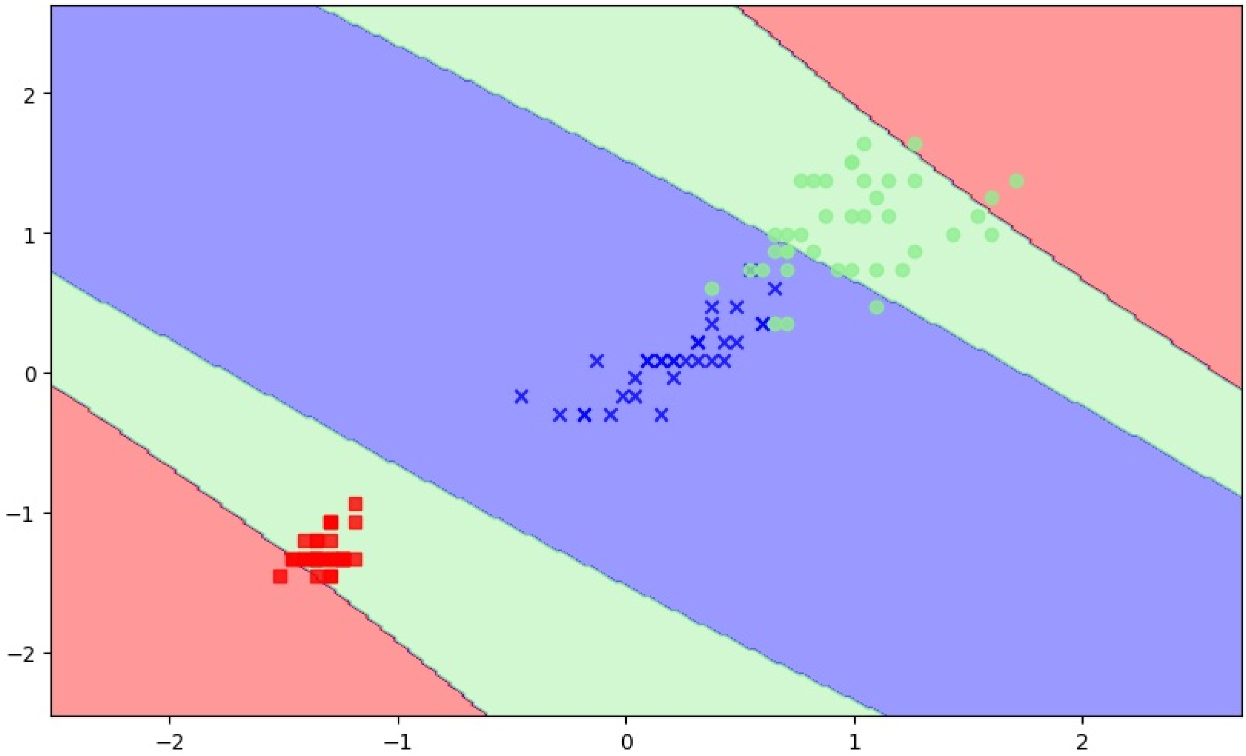
```
def versiontuple(v):
    return tuple(map(int, (v.split("."))))


def plot_decision_regions(X, y, classifier, test_idx=None, resolution=0.02):

    # setup marker generator and color map
    markers = ('s', 'x', 'o', '^', 'v')
    colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])

    # plot the decision surface
    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
                           np.arange(x2_min, x2_max, 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)
plt.show()
```

Visualising the Train set results

In [27]:

```
plot_decision_regions(X_train_std, y_train, svm)
```

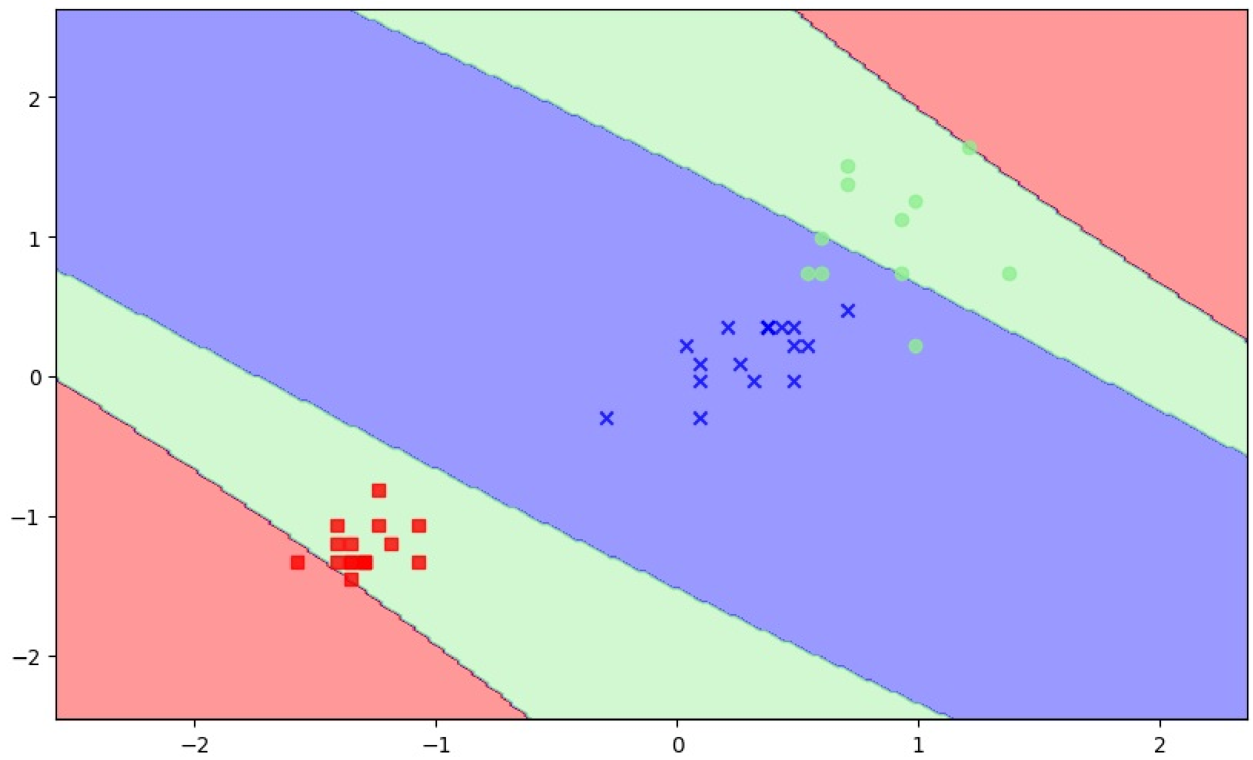
```
C:\Users\Mohammed Meraj\AppData\Local\Temp\ipykernel_12804\1574677956.py:24: UserWarning: *c* argument looks lik
e a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case i
ts length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2D array with a single r
ow if you intend to specify the same RGB or RGBA value for all points.
  plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],
```



Visualising the Test set results

In [28]:
```
plot_decision_regions(X_test_std, y_test, svm)
```

```
C:\Users\Mohammed Meraj\AppData\Local\Temp\ipykernel_12804\1574677956.py:24: UserWarning: *c* argument looks lik
e a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case i
ts length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2D array with a single r
ow if you intend to specify the same RGB or RGBA value for all points.
  plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],
```



In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js