**Practical 7**

**Aim:**

**Write a python program to classify various types of from with iris dataset using support vector machine.**

**Code:**

from sklearn import datasets

iris = datasets.load\_iris()

print(&quot;Features: &quot;, iris.feature\_names)

print(&quot;Labels: &quot;, iris.target\_names)

iris.data.shape

print(iris.target)

from sklearn import svm

import numpy as np

from scipy import stats

import seaborn as sns; sns.set()

from sklearn.model\_selection import train\_test\_split

%matplotlib inline

X\_train, X\_test, y\_train, y\_test = train\_test\_split(iris.data, iris.target,

test\_size=0.3,random\_state=109)

clf = svm.SVC(kernel=&#39;linear&#39;)

clf.fit(X\_train, y\_train)

y\_pred = clf.predict(X\_test)

from sklearn import metrics

from sklearn.datasets.samples\_generator import make\_blobs

import matplotlib.pyplot as plt

print(&quot;Accuracy:&quot;,metrics.accuracy\_score(y\_test, y\_pred))

from sklearn.datasets.samples\_generator import make\_blobs

X, y = make\_blobs(n\_samples=50, centers=2,

random\_state=0, cluster\_std=0.60)

plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap=&#39;autumn&#39;);

xfit = np.linspace(-1, 3.5)

plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap=&#39;autumn&#39;)

plt.plot([0.6], [2.1], &#39;x&#39;, color=&#39;red&#39;, markeredgewidth=2, markersize=10)

for m, b in [(1, 0.65), (0.5, 1.6), (-0.2, 2.9)]:

plt.plot(xfit, m \* xfit + b, &#39;-k&#39;)

plt.xlim(-1, 3.5);

xfit = np.linspace(-1, 3.5)

plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap=&#39;autumn&#39;)

for m, b, d in [(1, 0.65, 0.33), (0.5, 1.6, 0.55), (-0.2, 2.9, 0.2)]:

yfit = m \* xfit + b

plt.plot(xfit, yfit, &#39;-k&#39;)

plt.fill\_between(xfit, yfit - d, yfit + d, edgecolor=&#39;none&#39;,

color=&#39;#AAAAAA&#39;, alpha=0.4)

plt.xlim(-1, 3.5);

**Output:**

Features: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']

Labels: ['setosa' 'versicolor' 'virginica']

[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]

Accuracy: 0.9555555555555556

