**Name:- Medhansh K**

**PRN:-122A8035**

**Batch:-B2**

**Experiment No 4**

**Aim:-** To perform Software Vector Machine

**Theory:-**

Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression tasks. While it can handle regression problems, SVM is particularly well-suited for classification tasks.

SVM aims to find the optimal hyperplane in an N-dimensional space to separate data points into different classes. The algorithm maximizes the margin between the closest points of different classes.

**Support Vector Machine (SVM) Terminology**

* **Hyperplane**: A decision boundary separating different classes in feature space, represented by the equation **wx + b = 0** in linear classification.
* **Support Vectors**: The closest data points to the hyperplane, crucial for determining the hyperplane and margin in SVM.
* **Margin**: The distance between the hyperplane and the support vectors. SVM aims to maximize this margin for better classification performance.
* **Kernel**: A function that maps data to a higher-dimensional space, enabling SVM to handle non-linearly separable data.
* **Hard Margin**: A maximum-margin hyperplane that perfectly separates the data without misclassifications.
* **Soft Margin**: Allows some misclassifications by introducing slack variables, balancing margin maximization and misclassification penalties when data is not perfectly separable.
* **C**: A regularization term balancing margin maximization and misclassification penalties. A higher C value enforces a stricter penalty for misclassifications.
* **Hinge Loss**: A loss function penalizing misclassified points or margin violations, combined with regularization in SVM.
* **Dual Problem**: Involves solving for Lagrange multipliers associated with support vectors, facilitating the kernel trick and efficient computation.

**CODE:**

%matplotlib inline

import numpy as np

import matplotlib.pyplot as plt

plt.style.use('seaborn-whitegrid')

from scipy import stats

from sklearn.datasets 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='autumn');



xfit = np.linspace(-1, 3.5)

plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='autumn')

plt.plot([0.6], [2.1], 'x', color='red', 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, '-k')

plt.xlim(-1, 3.5);



xfit = np.linspace(-1, 3.5)

plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='autumn')

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, '-k')

plt.fill\_between(xfit, yfit - d, yfit + d, edgecolor='none',

color='lightgray', alpha=0.5)

plt.xlim(-1, 3.5);



from sklearn.svm import SVC # "Support vector classifier"

model = SVC(kernel='linear', C=1E10)

model.fit(X, y)

def plot\_svc\_decision\_function(model, ax=None, plot\_support=True): """Plot the decision function for a 2D SVC""" if ax is None: ax = plt.gca() xlim = ax.get\_xlim() ylim = ax.get\_ylim()

# create grid to evaluate model  
x = np.linspace(xlim[0], xlim[1], 30)  
y = np.linspace(ylim[0], ylim[1], 30)  
Y, X = np.meshgrid(y, x)  
xy = np.vstack([X.ravel(), Y.ravel()]).T  
P = model.decision\_function(xy).reshape(X.shape)  
  
# plot decision boundary and margins  
ax.contour(X, Y, P, colors='k',  
 levels=[-1, 0, 1], alpha=0.5,  
 linestyles=['--', '-', '--'])  
  
# plot support vectors  
if plot\_support:  
 ax.scatter(model.support\_vectors\_[:, 0],  
 model.support\_vectors\_[:, 1],  
 s=300, linewidth=1, edgecolors='black',  
 facecolors='none');  
ax.set\_xlim(xlim)  
ax.set\_ylim(ylim)  
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='autumn') plot\_svc\_decision\_function(model);



**Conclusion:-** Support Vector Machine (SVM) is an effective machine learning algorithm for classification, which finds the optimal boundary that separates different classes with the maximum margin. It is powerful in high-dimensional spaces and works well with clear margin separations.