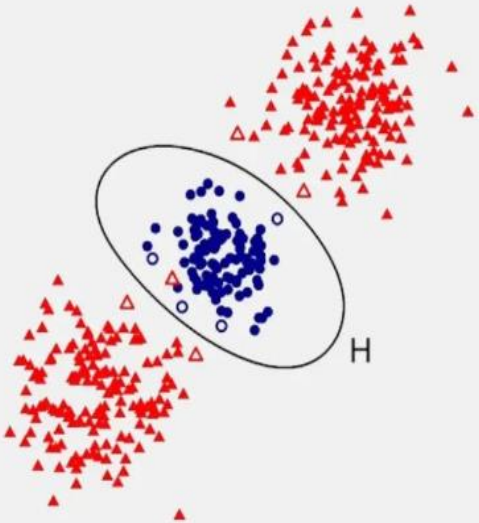
Support Vector **Machines**

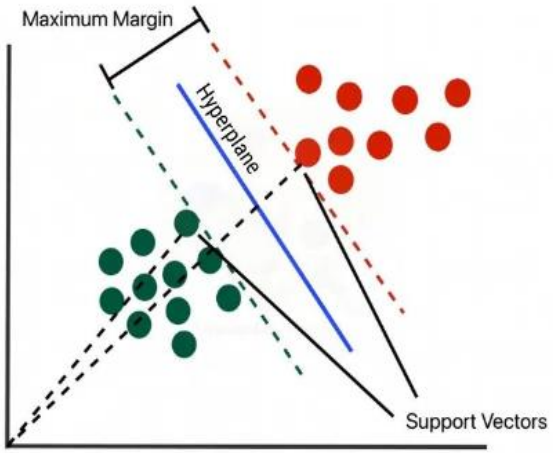
Explained with Code



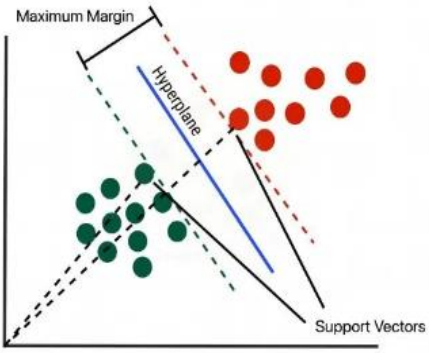
Swipe to see in detail

What are Support Vector Machines (SVM)

A Support Vector Machine (SVM) is a machine learning algorithm used for classification, regression, and outlier detection. **It works by finding an optimal hyperplane that maximizes the margin between different classes in the feature space.** SVMs are widely applied in various fields like healthcare, natural language processing, and image recognition.



What are Support Vector Machines (SVM)



In simple terms, imagine you have labeled data points (e.g., red and green dots) on a plane, and you want to find a line (hyperplane) that best separates them.

The SVM aims to maximize the distance (margin) between the line and the closest points of each class.

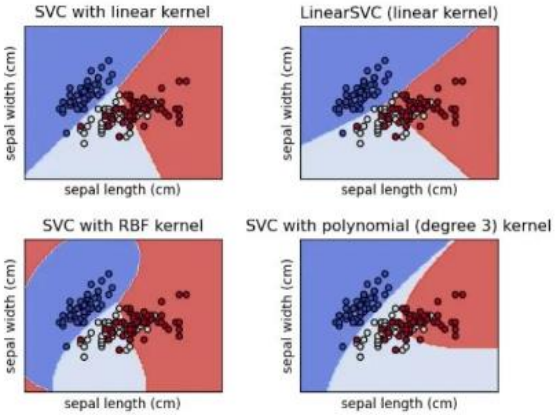
Now, the term "**support vectors**" refers to the data points that are crucial in determining the optimal hyperplane. These **are the points closest to the decision boundary** and play a significant role in defining the separation between classes. Essentially, support vectors are like the key players that help shape the decision-making line.

Types of SVMs

For linearly separable data, a straight line suffices. However, for non-linear data, SVM uses a technique called the kernel trick, which efficiently maps the data to higher dimensions for better separation. This involves transforming the data without explicitly calculating the higher-dimensional coordinates.

There are two main types of SVMs:

1. **Linear SVM:** Used for linearly separable data, where a single straight line can classify different classes.
2. **Kernel SVM:** Applied to non-linear data, it uses kernel functions to map data into higher dimensions, making it easier to separate classes.



Linear SVM Example

```python
from sklearn import svm
import numpy as np
import matplotlib.pyplot as plt

# Create a simple linearly separable dataset
X = np.array([[1, 2], [2, 3], [3, 3], [2, 1], [3, 2]])
y = np.array([1, 1, 1, 0, 0])

# Create a linear SVM model
model = svm.SVC(kernel='linear')

# Train the model
model.fit(X, y)

# Plot the decision boundary
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Paired)
ax = plt.gca()
xlim = ax.get_xlim()
ylim = ax.get_ylim()

# Create grid to evaluate model
xx, yy = np.meshgrid(np.linspace(xlim[0], xlim[1], 50),
np.linspace(ylim[0], ylim[1], 50))
Z = model.decision_function(np.c_[xx.ravel(), yy.ravel()])

# Plot decision boundary and margins
Z = Z.reshape(xx.shape)
plt.contour(xx, yy, Z, colors='k', levels=[-1, 0, 1], alpha=0.5,
linestyles=['--', '-', '--'])
plt.scatter(model.support_vectors_[:, 0], model.support_vectors_[:,
1], s=100, facecolors='none', edgecolors='k')
plt.title('Linear SVM Example')
plt.show()
```