SVM

Used for:

- **Classification → Binary + Multiclass**
- **Regression**
- **Image processing**
- Works for both Linear + Non-linear

Drawback:

- Difficult to understand

> 💡 **Gives good results in almost any situation.**

Goal: to find a **hyperplane** that best separates the data into classes with the maximum possible margin.

How Does SVM Work?

SVM works by:

✔️ **Finding a hyperplane** that separates classes

✔️ **Maximizing the margin** (distance between the hyperplane and the nearest data points)

✔️ **Using support vectors** (points that determine the boundary)

Key Concepts

Hyperplane:

- A decision boundary that separates data points of different classes.
- In 2D, it's a line; in 3D, it's a plane; in higher dimensions, it's a hyperplane.

Support Vectors:

- The **closest data points** to the hyperplane.
- These **determine the position and orientation** of the hyperplane.
- The model **only depends on these points**.

Margin:

- The distance between the hyperplane and the nearest data points (support vectors).
- SVM aims to maximize this margin.

Types of SVM

1️⃣ **Linear SVM** → Used when data is **linearly separable**.

- Assumes data is linearly separable.
- Finds a straight line (or hyperplane) to separate classes.

2️⃣ **Non-Linear SVM (Kernel SVM)** → Used when data is **not linearly separable**.

- Uses **kernel functions** to transform data into a higher-dimensional space where it becomes linearly separable.
- Common kernels: Polynomial, Radial Basis Function (RBF), Sigmoid.

SVM Parameters

1. **C (Regularization Parameter)**:

- **C** controls the trade-off between achieving a low error on the training data and maintaining a large margin.
    - **High C**: SVM tries to classify all training points correctly, which may lead to overfitting (narrow margin).
    - **Low C**: SVM allows some misclassifications but aims for a wider margin, which might lead to underfitting.

2. **Kernel**:
    - Specifies the type of kernel used (e.g., **linear**, **RBF**, **polynomial**, etc.).
3. **Gamma** (for non-linear kernels like RBF):
    - **Gamma** defines how far the influence of a single training example reaches. A low gamma means a far-reaching influence, while a high gamma means a closer, more local influence.
        - **Low gamma**: The decision boundary is smoother.
        - **High gamma**: The decision boundary is more flexible and could overfit the data.
4. **Degree** (for polynomial kernel):
    - Specifies the degree of the polynomial kernel function (used only when kernel = polynomial).

Classification

1. **Hard Margin SVM → assumes perfect separation** (works only for clean datasets).
2. **Soft Margin SVM** → allows **some misclassification** using a penalty term **C**.

Support Vector Classifier

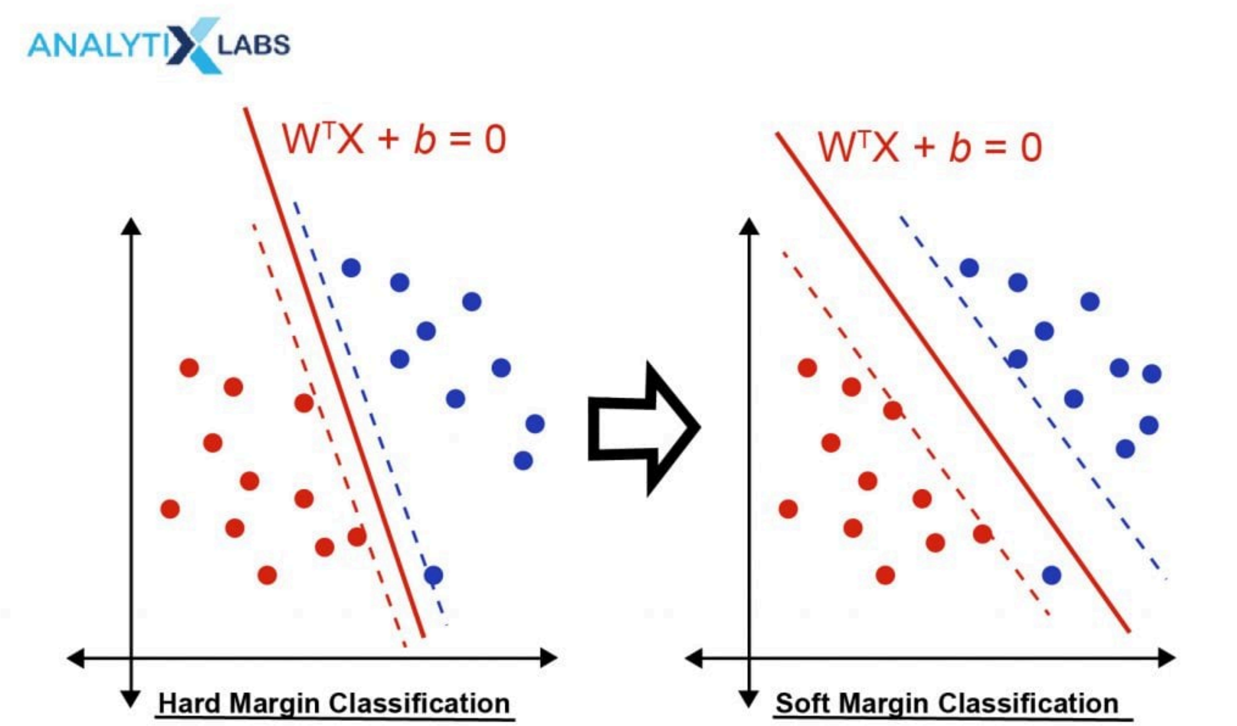
C (Regularization Parameter):

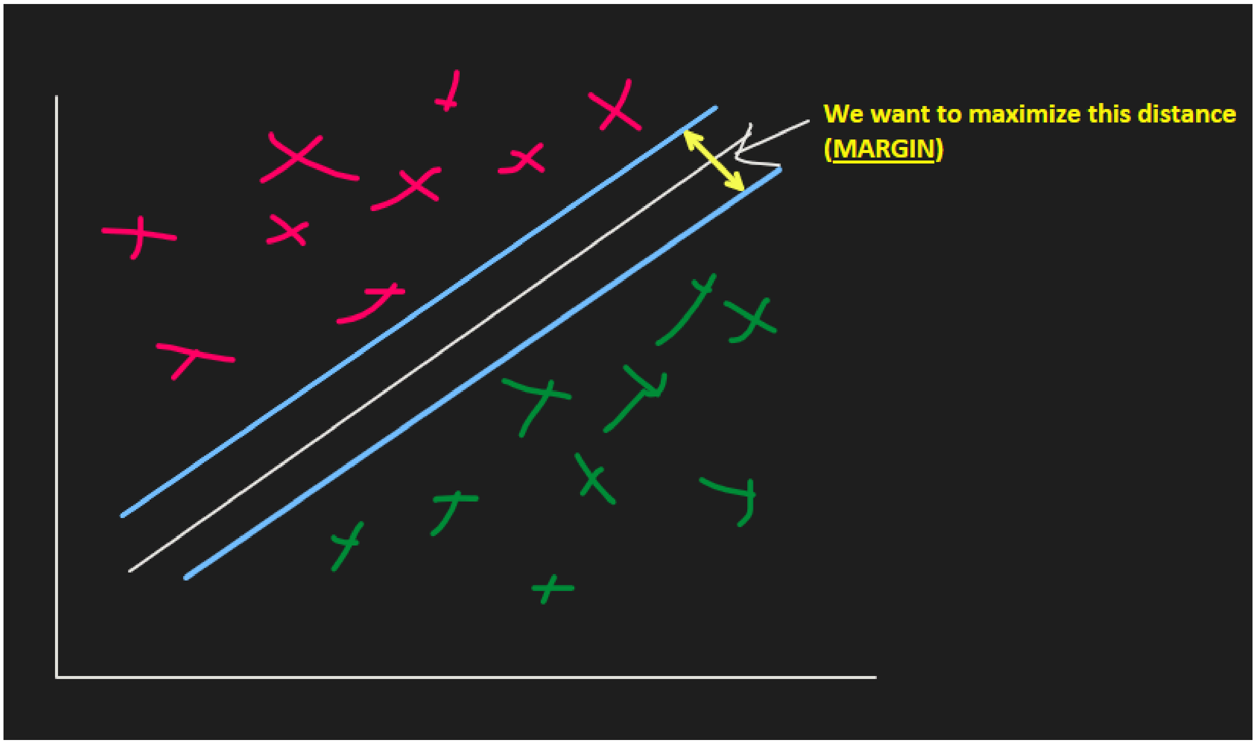
- **High C** → Tries to **classify every point correctly** (low bias, high variance).
- **Low C** → Allows **some misclassification** (high bias, low variance).

3. **SVM Kernels → Non-linear**

4. **SVM for multi-class**
5. **SVR → Regression**

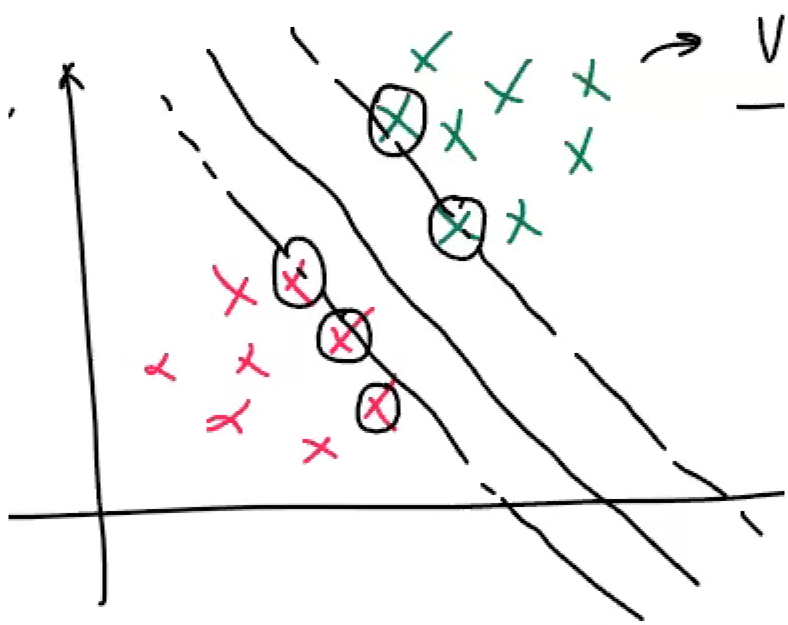
Hard Margin SVM (Maximal Margin Classifier)

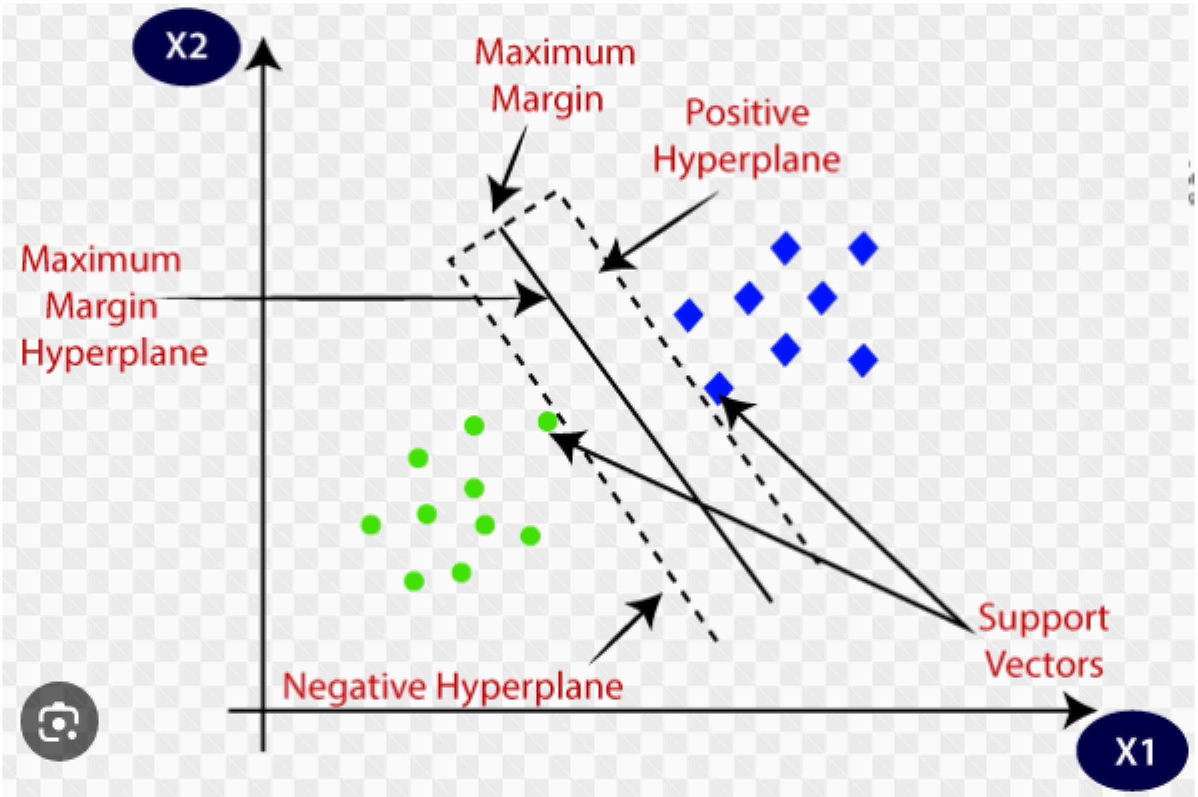




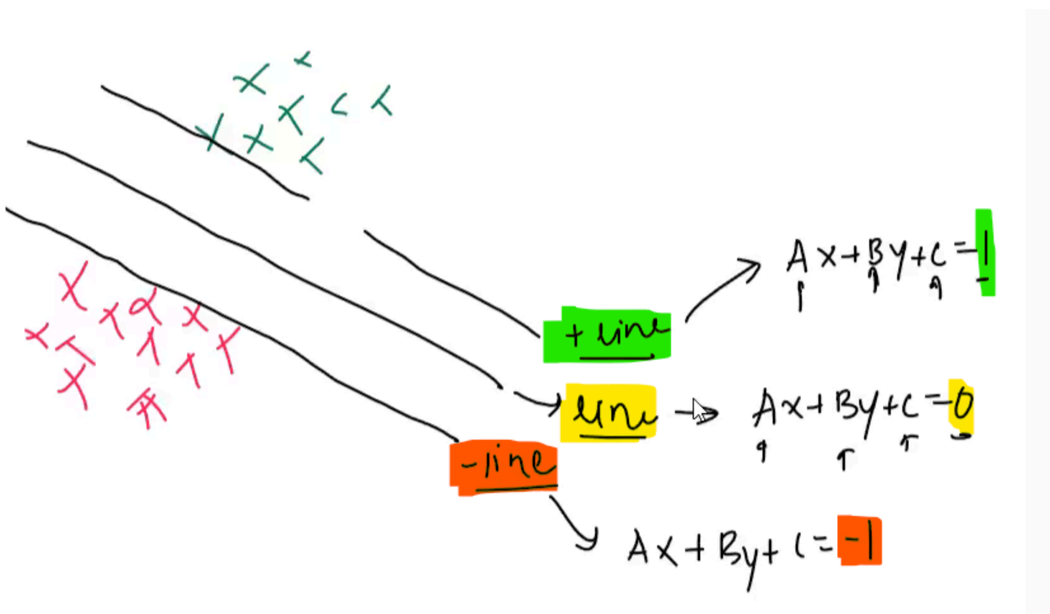
Support Vectors:

- Vectors which lie on the outer line





- In **Hard Margin SVM**, **we** do not want any points inside the support vector lines.



- 👆We can **multiply** the equation by a **number more than 1** to **decrease**🔽 the margin.
- Multiply with **number less than 1** to **increase the distance**.

$$Ax + By + c \geq 1$$

$$Ax + By + c \leq 1$$

- **We want to maximize the distance between support vectors which satisfies the below equation:**

$$\begin{cases} \text{if } y_i = 1 & A x_{1i} + B x_{2i} + c \geq 1 \\ y_i = -1 & A x_{i1} + B x_{i2} + c \leq -1 \end{cases}$$

- We can write the above equation as:

$$\boxed{y_i \left(A x_{i1} + B x_{i2} + c \right) \geq 1} \quad \boxed{\begin{matrix} +ve \\ -ve \end{matrix}}$$

Loss Function:

You maximize this👇

$$\underset{A,B,C}{\operatorname{argmax}} \frac{2}{\sqrt{A^2+B^2}} \quad \text{given} \left\{ y_i \left(A x_{1i} + B x_{2i} + C \right) \geq 1 \right\}$$

$$\begin{array}{ccc} & x_{2i} & y_i \\ & \uparrow & \uparrow \\ X_1 & X_2 & \underline{Y} \\ x_{11} & x_{21} & y_1 \\ x_{12} & x_{22} & y_2 \\ x_{13} & x_{23} & y_3 \end{array}$$

- This is constrained optimization problem because we have to solve 2 things at a time.
- We solve this with quadratic programming (high level math we won’t get into)

💡 **You never use hard margin SVM because real world datasets are not perfectly separable.**

Drawbacks of Hard Margin SVM:

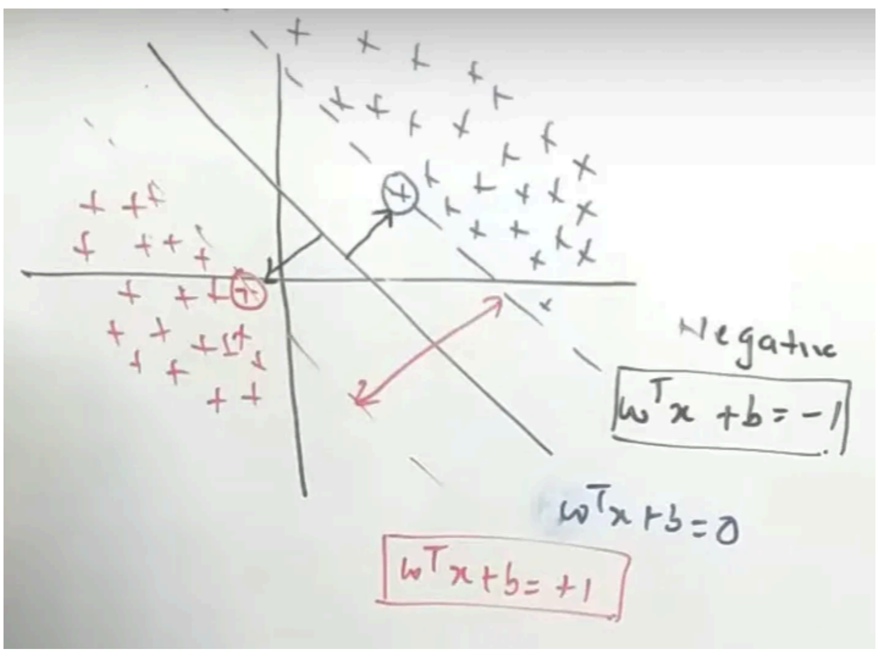
- Sensitive to outliers.
- **Limited to Linearly Separable Data:**
  - Hard margin SVMs can only be used when the data is perfectly linearly separable. In real-world scenarios, data is often complex and non-linearly separable, rendering hard margin SVMs impractical.
- Lack of Flexibility.

Soft Margin vs. Hard Margin

Feature	Hard Margin SVM	Soft Margin SVM
Misclassification Allowed?	❌ No	✅ Yes
Works for Noisy Data?	❌ No	✅ Yes
Overfitting Risk	✅ High	❌ Lower
Used in Real-World?	❌ Rarely	✅ Yes

Soft Margin SVM (SVC)

- **Soft Margin SVM** allows **some misclassification** using a penalty term **C**.
- Here, we **soften the constraint**.



Slack Variable

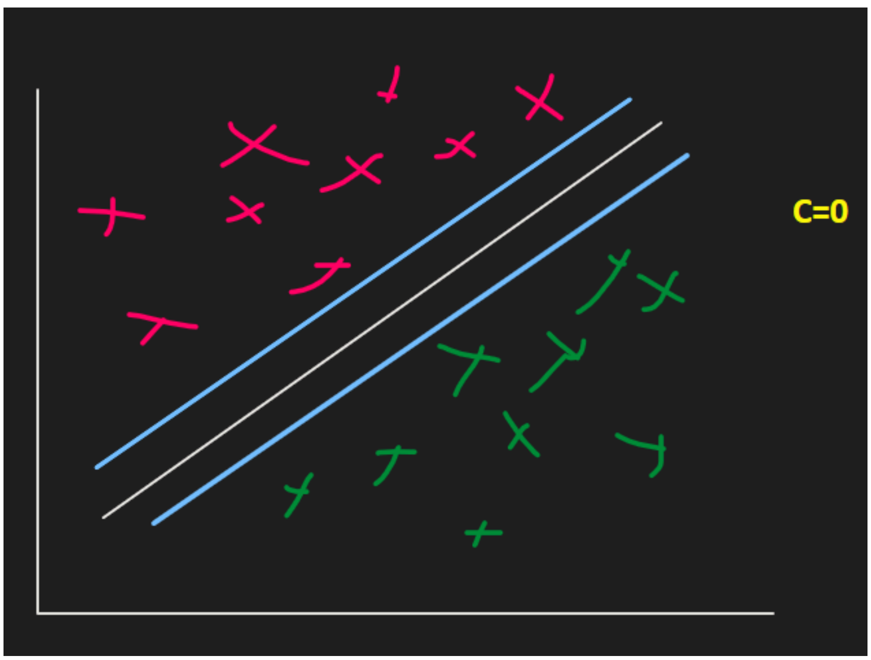
- Slack variables (often denoted as **ξ** or **ζ**) are introduced to relax the strict requirement of perfect separation.
- They allow some data points to violate the margin or even be misclassified.

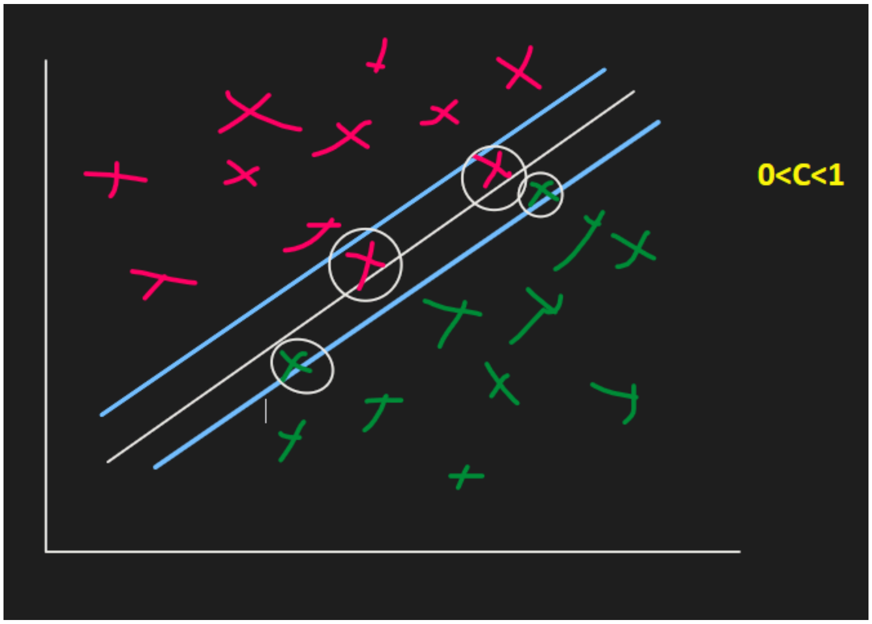
- Each data point is associated with a slack variable, representing the degree to which it violates the margin.

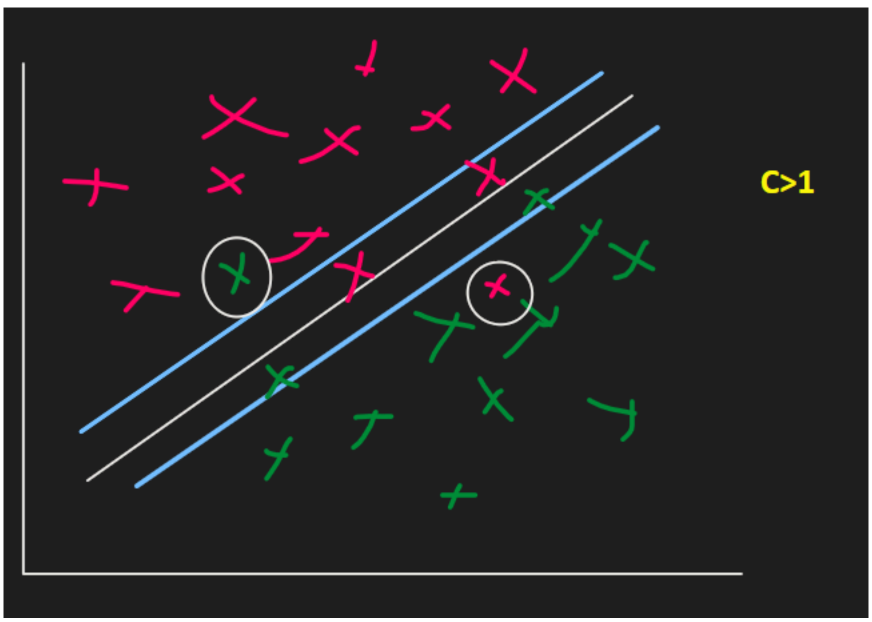
How Slack Variables Work:

- When a data point is correctly classified and lies outside the margin, its slack variable is zero.
- When a data point lies within the margin but is correctly classified, its slack variable is greater than zero but less than one.
- When a data point is misclassified, its slack variable is greater than one.

This balance is controlled by a regularization parameter (often denoted as C)







- **ξ(C/Zeta)** is Misclassification score.
  - Also called as Hinge loss in ML.

Mathematical Formulation of Soft Margin SVM:

$$\min_{w,b,\xi} \frac{1}{2}||w||^2 + C\sum_{i=1}^{N}\xi_i$$

Where:

- w: weight vector perpendicular to the hyperplane.
- b: bias term of the hyperplane.
- ξ_i: slack variable for the i-th data point.
- C: regularization parameter controlling the penalty for misclassification .

$$y_i(Ax_{1i} + Bx_{2i} + c) \geq 1 - \xi_i$$

such that

$$\boxed{\xi_i \geq 0}$$

$$\boxed{y_i(Ax_{1i} + Bx_{2i} + C) \geq 1 - \xi_i}$$

→ it is allowing all the points

→ this is no more a constraint

→ All true condition

- **This allowed all the points.**
- **It’s no more a constraint.**
- So, we add this 👇thing to the above formula:

$$\sum_{i=1}^{N} \xi_i$$

Objective Function:

$$\min_{w,b,\xi} \frac{1}{2}||w||^2 + C\sum_{i=1}^{N}\xi_i$$

Where:

- w: weight vector perpendicular to the hyperplane.
- b: bias term of the hyperplane.
- ξ_i: slack variable for the i-th data point.
- C: regularization parameter controlling the penalty for misclassification.

Subject to Constraints:

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i \quad \text{for} \quad i = 1, 2, \ldots, N$$

$$\xi_i \geq 0 \quad \text{for} \quad i = 1, 2, \ldots, N$$

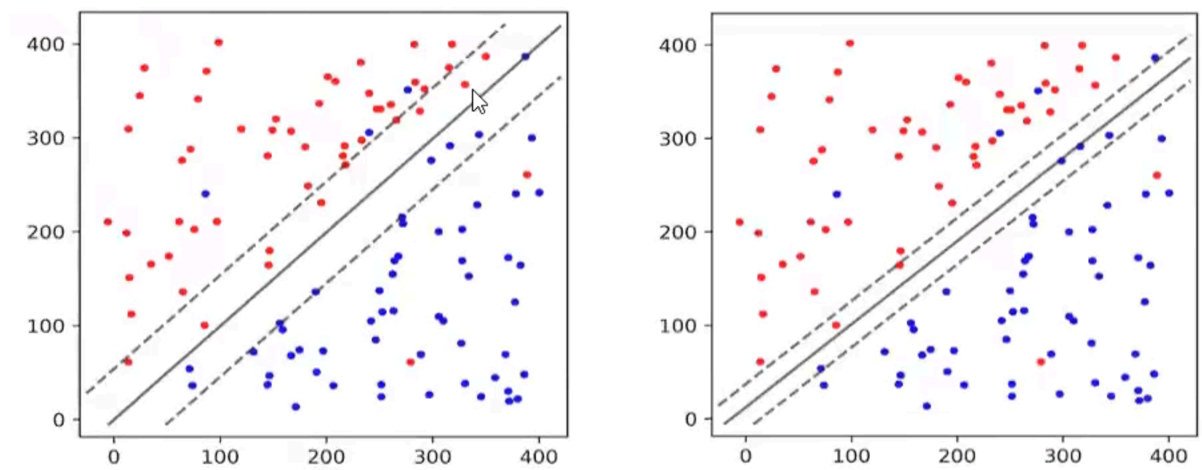
Where:

- $y_i \in \{-1, 1\}$: class label of the i-th data point.
- x_i: feature vector of the i-th data point.
- ξ_i: slack variable.

ξ_i **= Slack variable (allows misclassification)**

C = Regularization parameter (controls trade-off)

- **High C** → Tries to **classify every point correctly** (**low bias, high variance**).
- **Low C** → Allows **some misclassification** (**high bias, low variance**).



C = 1   C = 100

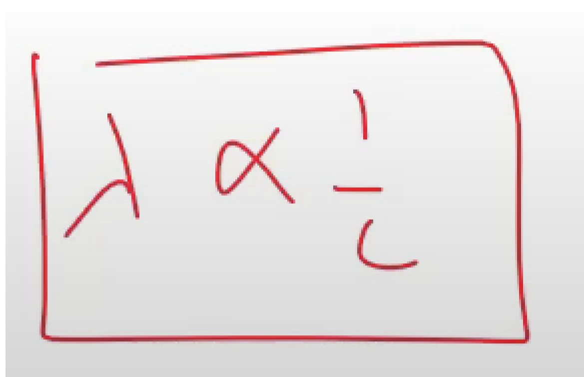
Python Code

```
# Import necessary libraries
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Load the Iris dataset
iris = datasets.load_iris()
X = iris.data
y = iris.target

# Only take two classes (binary classification for simplicity)
X = X[y != 2]
y = y[y != 2]

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
# Initialize the SVM classifier with a linear kernel (soft margin)
svm_model = SVC(kernel='linear', C=1)  # C is the regularization parameter

# Train the model
svm_model.fit(X_train, y_train)

# Make predictions
y_pred = svm_model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")
```

Accuracy: 100.00%

Relationship with Logistic Regression



C (Regularization Parameter) & ξ (Slack Variable)

C (Regularization Parameter)

C is a **hyperparameter** that you choose before training the model

- Controls the trade-off between **maximizing the margin** and **minimizing misclassification.**
- **High C** → **Less misclassification**, smaller margin, risk of **overfitting**.
- **Low C** → **More misclassification allowed**, larger margin, better **generalization**.

ξ (Slack Variable)

ξ are **variables** **created for each data point** during the model training process

- Measures how much a point **violates** the margin.
- If **ξ = 0**, the point is correctly classified and outside the margin.
- If **0 < ξ ≤ 1**, the point is inside the margin but correctly classified.
- If **ξ > 1**, the point is misclassified.
- **ξ is not something you choose; it's computed during training as part of the optimization process.**