Support_Vector_Classifier

Prepared by Eng. Doaa Fahmy

Objective :

- What’s the SVC ?
- Types of SVC ?
- Applications
- How Algorthim work
- Adv. & Dis Adv.
- Mathematical Intuitions Behind SVM
- Hyper Parmeter
- -References
-

What is a Support Vector Machine (SVM)?

- **A Support Vector Machine (SVM)**
- is a machine learning algorithm used for classification and regression.
- aims to find the best line (or hyperplane) to separate data into groups, maximizing the distance between the closest points (support vectors) of each group.
- It can handle complex data using kernels to transform it into higher dimensions. In short, SVM helps classify data effectively.

- **SVM types :**
- Svregressor
- SV classifier

What is a Support Vector Machine (SVM)?

1.Hyperplane:

1. Acts as the decision boundary in the feature space, separating different classes.

2. In 2D, it appears as a line; in higher dimensions, it becomes a flat affine subspace.

Margin:

Represents the distance between the hyperplane and the closest data points of any class.

SVM aims to maximize this margin to ensure the widest possible separation between classes.

The goal is to create the largest possible “street” between classes without misclassification.

Support Vectors:

These are the data points closest to the hyperplane.

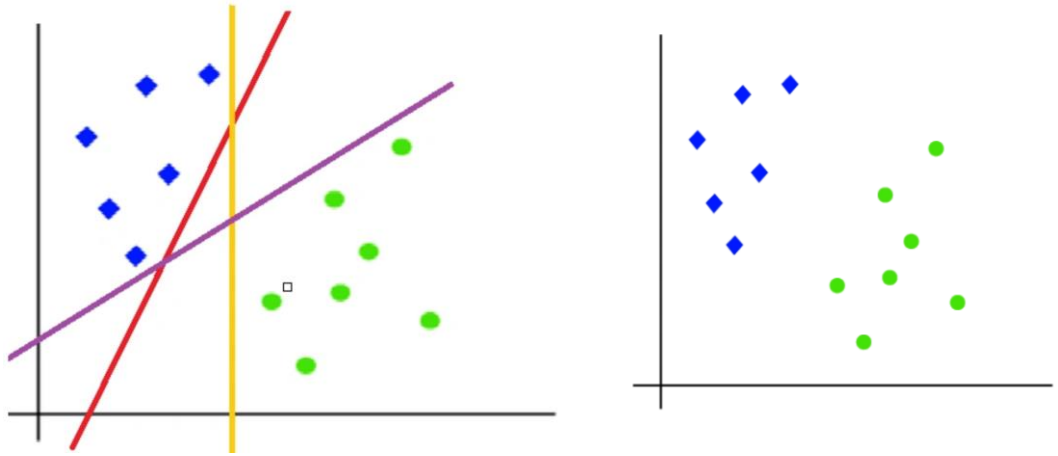
They are critical in determining the hyperplane’s position and orientation.

Support vectors directly influence the optimal hyperplane

Types of Support Vector Machine (SVM) Algorithms

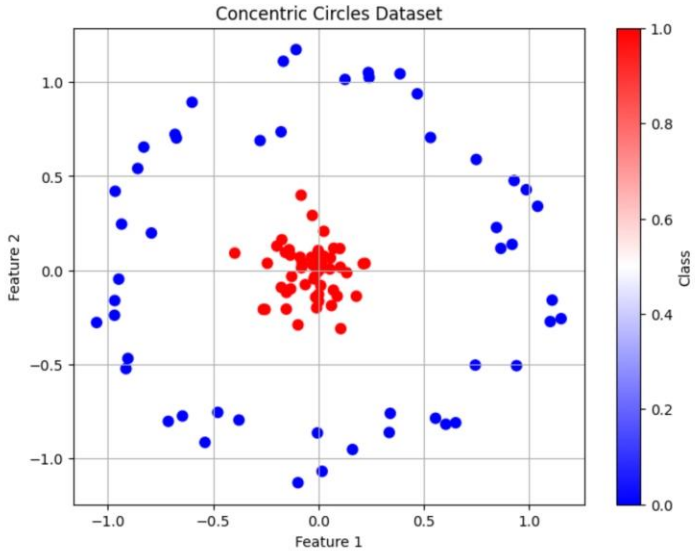
•Linear SVM:

• When the data is perfectly linearly separable only then we can use Linear SVM. Perfectly linearly separable means that the data points can be classified into 2 classes by using a single straight line(if 2D).



•Non-Linear SVM:

• When the data is not linearly separable, we can use Non-Linear SVM. This happens when the data points cannot be separated into two classes using a straight line (if 2D). In such cases, we use advanced techniques like kernel tricks to classify them. In most real-world applications we do not find linearly separable datapoints hence we use kernel trick to solve them.



Advantages of Support Vector Machine (SVM)

1.High-Dimensional Performance:

2.SVM excels in high-dimensional spaces, making it suitable for image classification and gene expression analysis.

2.Nonlinear Capability:

3.Utilizing kernel functions like RBF and polynomial SVM effectively handles nonlinear relationships.

3.Outlier Resilience:

The soft margin feature allows SVM to ignore outliers, enhancing robustness in spam detection and anomaly detection.

4.Binary and Multiclass Support:

5. SVM is effective for both binary classification and multiclass classification suitable for applications in text classification.

5.Memory Efficiency:

6. It focuses on support vectors making it memory efficient compared to other algorithms.

Disadvantages of Support Vector Machine (SVM)

1.Slow Training:

SVM can be slow for large datasets, affecting performance in SVM in data mining tasks.

2.Parameter Tuning Difficulty:

Selecting the right kernel and adjusting parameters like C requires careful tuning, impacting SVM algorithms.

3.Noise Sensitivity

SVM struggles with noisy datasets and overlapping classes, limiting effectiveness in real-world scenarios.

4.Limited Interpretability:

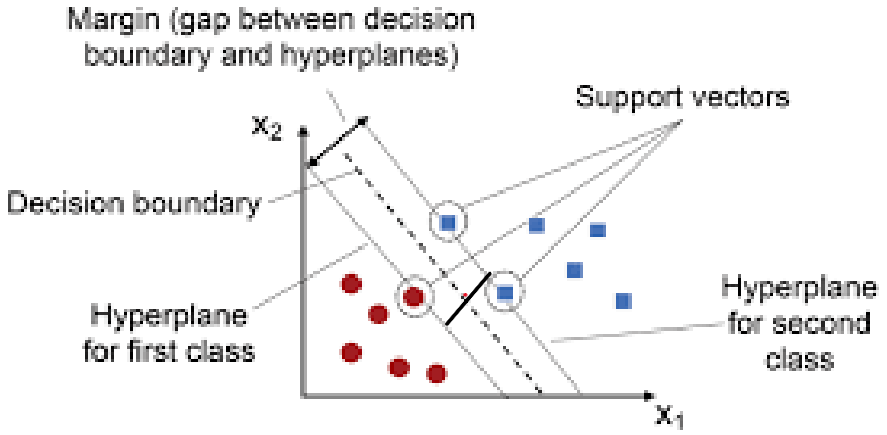
The complexity of the hyperplane in higher dimensions makes SVM less interpretable than other models.

5.Feature Scaling Sensitivity:

Proper feature scaling is essential, otherwise SVM models may perform poorly.

How does Support Vector Machine Algorithm Work?

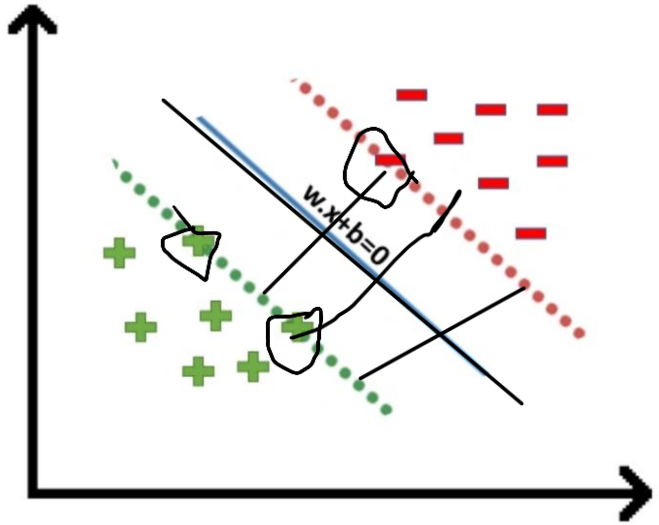
The key idea behind the SVM algorithm is to find the hyperplane that best separates two classes by maximizing the margin between them. This margin is the distance from the hyperplane to the nearest data points (support vectors) on each side.



The Basic Components of SVC：

1- Decision Boundary

The decision boundary in SVM is the line (or called "hyperplane" in higher dimensions) that the algorithm determines to best separate different classes of data



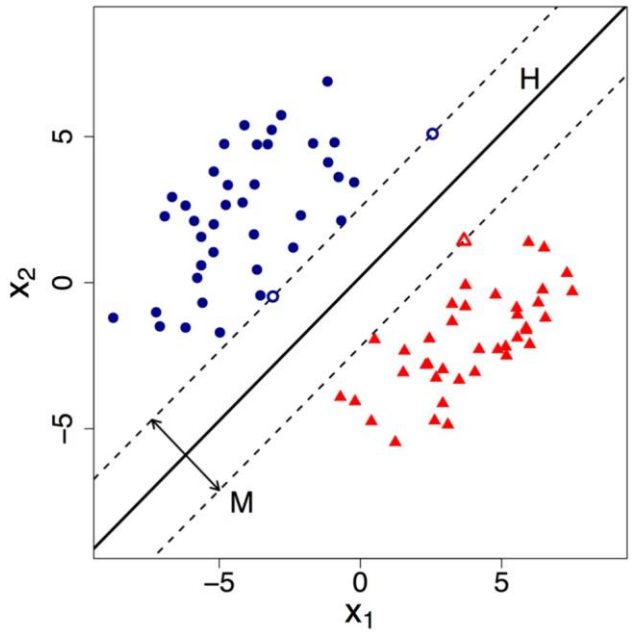
2- Linear separability

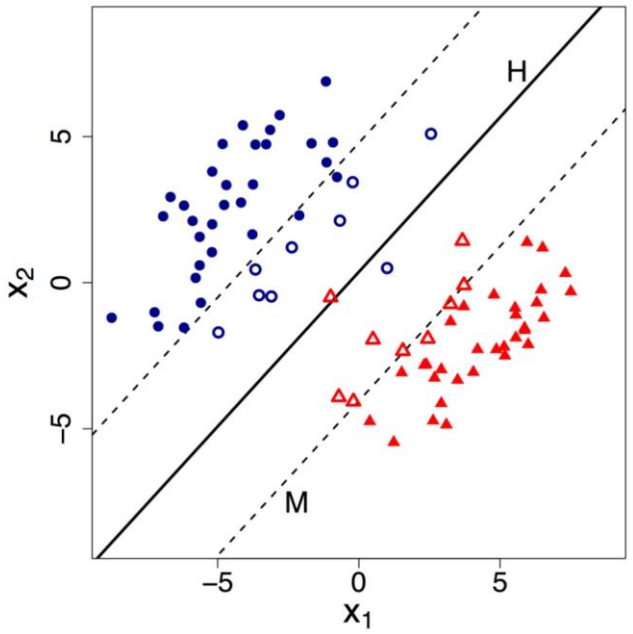
Linear separability refers to whether we can draw a straight line that perfectly separates two classes of data points.

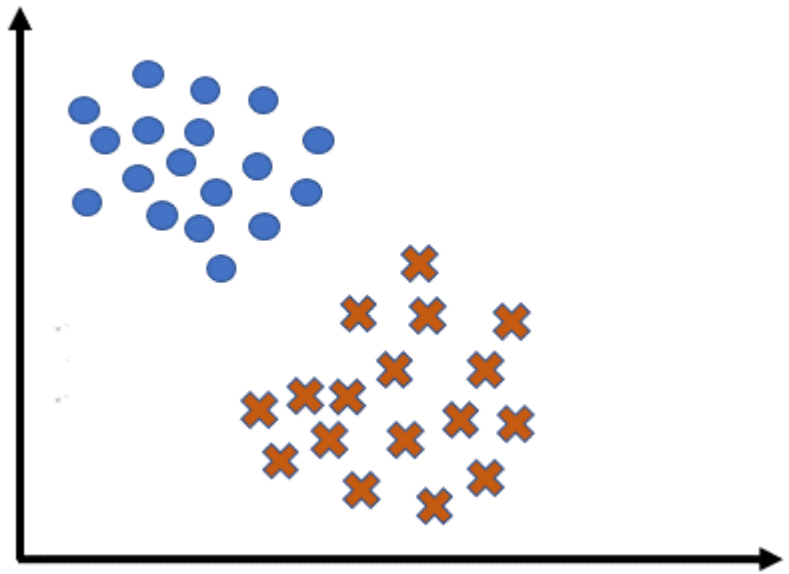
3- Margin

The margin in SVM is the distance between the decision boundary and the closest data points from each class. These closest points are called support vectors.

Soft & Hard Margin







Mathematical Intuition Behind SVC

For all the Red points $\vec{w}.\vec{X}+b \leq -1$

For all the Green points $\vec{w}.\vec{X}+b \geq 1$

$$\Rightarrow \frac{(1-b)-(-b-1)}{\|w\|}$$

$$\Rightarrow \frac{1-b+b+1}{\|w\|} = \frac{2}{\|w\|} = d$$

for positive point $y = 1$

$$\Rightarrow 1 \times (\vec{w}.x1 + b) = 1$$

$$\Rightarrow \vec{w}.x1 = 1-b \quad -----(2)$$

Similarly for negative point $y = -1$

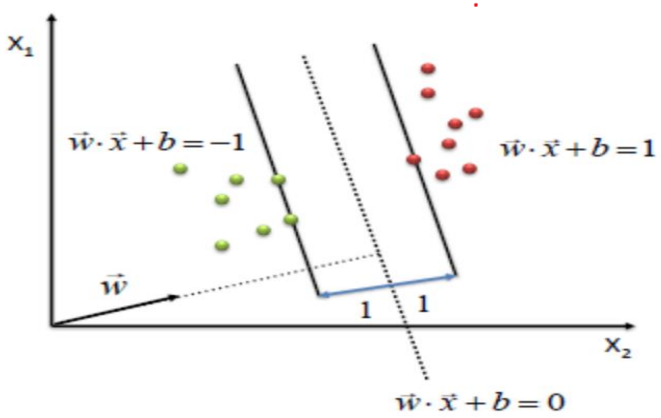
$$\Rightarrow -1 \times (\vec{w}.x2 + b) = 1$$

$$\Rightarrow \vec{w}.x2 = -b-1 \quad -----(3)$$

Putting equations (2) and (3) in equation (1) we get:

$$\Rightarrow \frac{(1-b)-(-b-1)}{\|w\|}$$

$$\Rightarrow \frac{1-b+b+1}{\|w\|} = \frac{2}{\|w\|} = d$$



$$\max \frac{2}{\|w\|}$$

s.t.

$(w \cdot x+b) \geq 1, \forall x$ of class 1

$(w \cdot x+b) \leq -1, \forall x$ of class 2

Optimization Function and its Constraints

$$\text{argmax}(w^*, b^*) \frac{2}{\|w\|} \text{ such that } y_i\left(\vec{w}.\vec{X}+b\right) \geq 1$$

We know that *max[f(x)]* can also be written as *min[1/f(x)]*, it is common practice to minimize a cost function for optimization problems; therefore, we can invert the function.

$$\min \frac{1}{2}\|\mathbf{w}\|^2 \qquad \text{s.t.} \qquad y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \quad \forall_i$$

In order to cater for the constraints in this minimization, we need to allocate them Lagrange multipliers $\boldsymbol{\alpha}$, where $\alpha_i \geq 0 \; \forall_i$:

$$\begin{aligned} L_P &\equiv \frac{1}{2}\|\mathbf{w}\|^2 - \boldsymbol{\alpha}\left[y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \; \forall_i\right] \\ &\equiv \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{L} \alpha_i \left[y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1\right] \\ &\equiv \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{L} \alpha_i y_i(\mathbf{x}_i \cdot \mathbf{w} + b) + \sum_{i=1}^{L} \alpha_i \end{aligned}$$

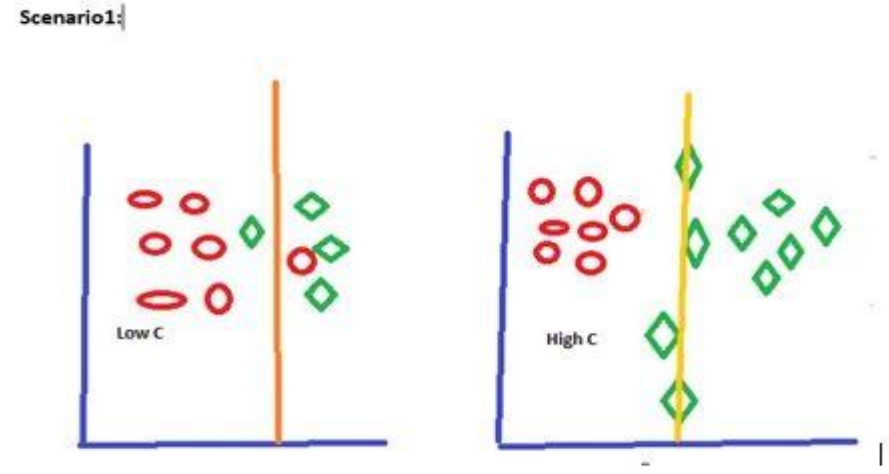
We wish to find the **w** and **b** which minimizes, and the **α** which maximizes LP(whilst keeping αi ≥ 0 ∀i). We can do this by differentiating LP with respect to w and b and setting the derivatives to zero:

$$\frac{\partial L_P}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{L} \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_{i=1}^{L} \alpha_i y_i = 0$$

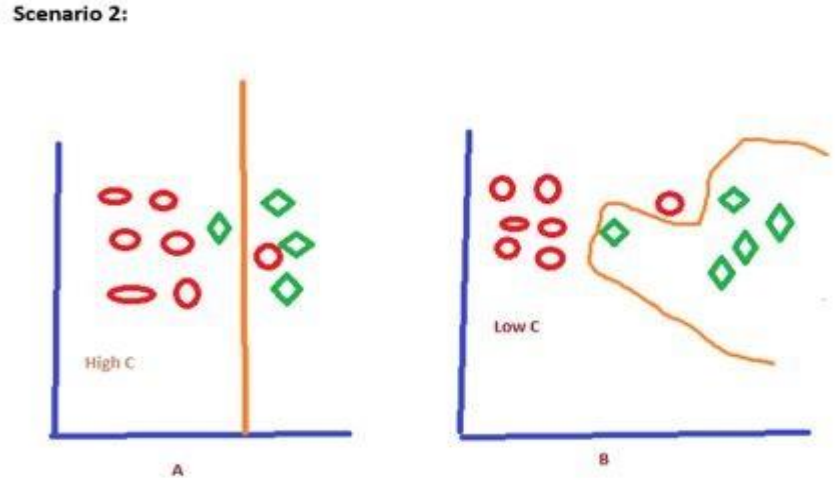
Regularization Parameter (C)

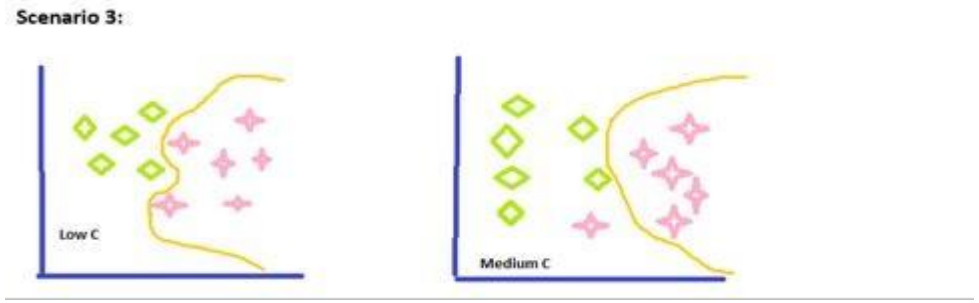
This parameter controls the trade-off between achieving a low training error and minimizing the norm of the weights. A higher value of C allows for more flexibility in the decision boundary, potentially leading to overfitting, while a lower value of C imposes a smoother decision boundary and may lead to underfitting.



Gamma Parameter (γ)

Gamma is a parameter for non-linear hyperplanes. It defines how far the influence of a single training example reaches, with low values meaning ‘far’ and high values meaning ‘close.’ A higher gamma value will result in more complex decision boundaries, which may lead to





Non-Linear Separable Classes

Kernal Trick:

As we have seen so far, no matter how we set up the hyperplane, we never could make a perfect separation between the two classes. There are actually some "trick" that we can do to make it separable… even though it is not linearly anymore

A kernel is a function that computes the similarity between two data points, implicitly representing them in a higher-dimensional space (the feature space).

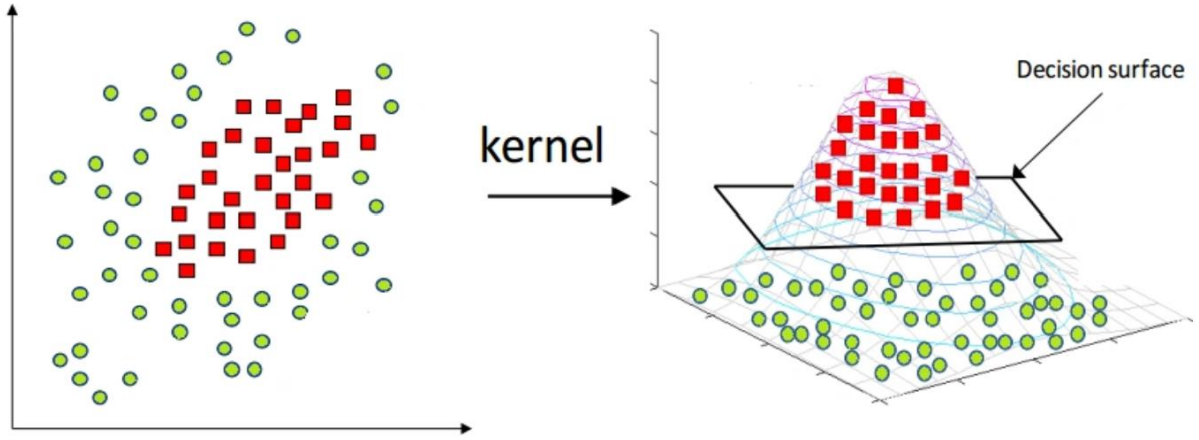


TABLE I. DIFFERENT KERNEL FUNCTIONS OF SVM

	Formula	Parameters	Merits
Linear	$K(x, x_i) = x \cdot x_i$	/	It is only used when the sample is separable in low dimensional space.
Polynomial	$K(x, x_i) = [\gamma * (x \cdot x_i) + coef]^d$	γ, *coef*, d	global kernels
RBF	$K(x, x_i) = \exp(-\gamma * \|x - x_i\|^2)$	γ.	good local performance
Sigmoid	$K(x, x_i) = \tanh(\gamma(x \cdot x_i) + coef)$	γ, *coef*	needs to meet certain conditions

How to Choose the Right Kernel?

I am well aware of the fact that you must be having this doubt about how to decide which kernel function will work efficiently for your dataset. It is necessary to choose a good kernel function because the performance of the model depends on it.

Here are the Points to choose the right Kernal:

•Kernel selection depends on the dataset type.

•For linearly separable data, use a linear kernel:

- It is simple and has lower complexity compared to other kernels.
- Start by assuming your data is linearly separable and try the linear kernel first.

•Move to more complex kernels if needed.

•Commonly used kernels:

- Linear and RBF (Radial Basis Function) are widely used.
- Polynomial kernels are rarely used due to poor efficiency.

•If both linear and RBF kernels give similar results:

- Choose the simpler option, which is the linear kernel.

Advantages of Using Kernel Functions

- **Handling Non-linearity**: Kernels enable SVMs to perform well on non-linearly separable data by implicitly mapping them into higher-dimensional spaces where linear separation is possible.
- **Computational Efficiency**: Thanks to the kernel trick, SVMs can operate in high-dimensional spaces without explicitly computing the transformed features, reducing computational overhead.

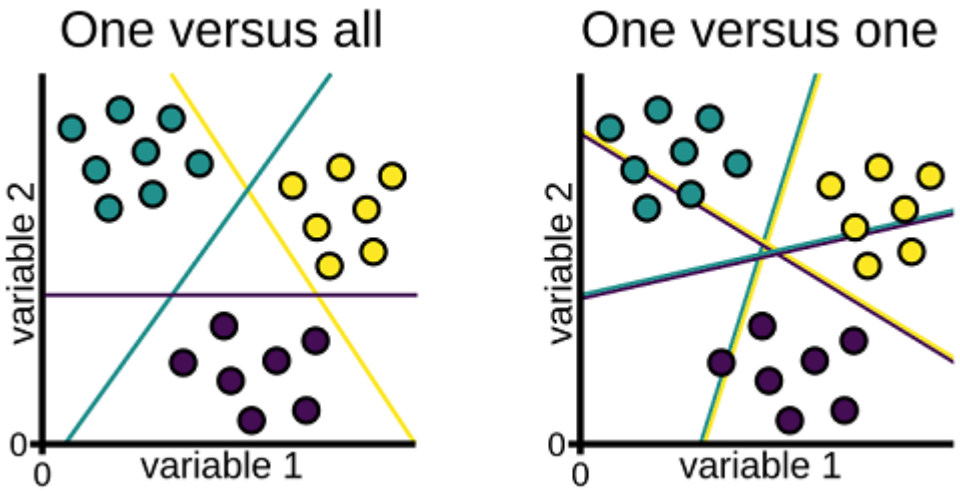
Considerations and Best Practices

1. **Choice of Kernel**: The choice of kernel and its parameters significantly impact the model's performance. It's essential to experiment with different kernels and perform hyperparameter tuning to find the optimal settings.
2. **Overfitting**: High-degree polynomial kernels or very small values of γγ in RBF kernels can lead to overfitting. Regularization parameters (like C in SVM) should be adjusted to balance the trade-off between margin maximization and classification error.
3. **Interpretability**: Models using non-linear kernels are often less interpretable compared to linear models. Ensure that model interpretability aligns with your application requirements.

Multiclass_ Classification in SVC

One-vs-One (OvO) :

This method is particularly advantageous when the number of classes is relatively small, as it allows for a focused comparison between pairs of classes. Each classifier is responsible for distinguishing between two specific classes, effectively ignoring the others.



One-vs-All (OvA) :

also known as One-vs-Rest, involves training a single binary classifier for each class. In this method, each class is treated as the positive class, while all other classes are grouped together as the negative class. This approach is straightforward and scales linearly with the number of classes.

How to Evaluate the classifications Models

Confusion matrix

is a simple table used to measure how well a classification model is performing. It compares the predictions made by the model with the actual results and shows where the model was right or wrong. This helps you understand where the model is making mistakes so you can improve it. It breaks down the predictions into four categories:

- **True Positive (TP):**

- The model correctly predicted a positive outcome i.e the actual outcome was positive.

- **True Negative (TN):**

- The model correctly predicted a negative outcome i.e the actual outcome was negative.

- **False Positive (FP):**

- The model incorrectly predicted a positive outcome i.e the actual outcome was negative. It is also known as a Type I error.

- **False Negative (FN):**

- The model incorrectly predicted a negative outcome i.e the actual outcome was positive. It is also known as a Type II error.

Metrics based on Confusion Matrix Data

1. Accuracy

Accuracy shows how many predictions the model got right out of all the predictions. It gives idea of overall performance, but it can be misleading when one class is more dominant over the other. For example, a model that predicts the majority class correctly most of the time might have high accuracy but still fail to capture important details about other classes. It can be calculated using the below formula:

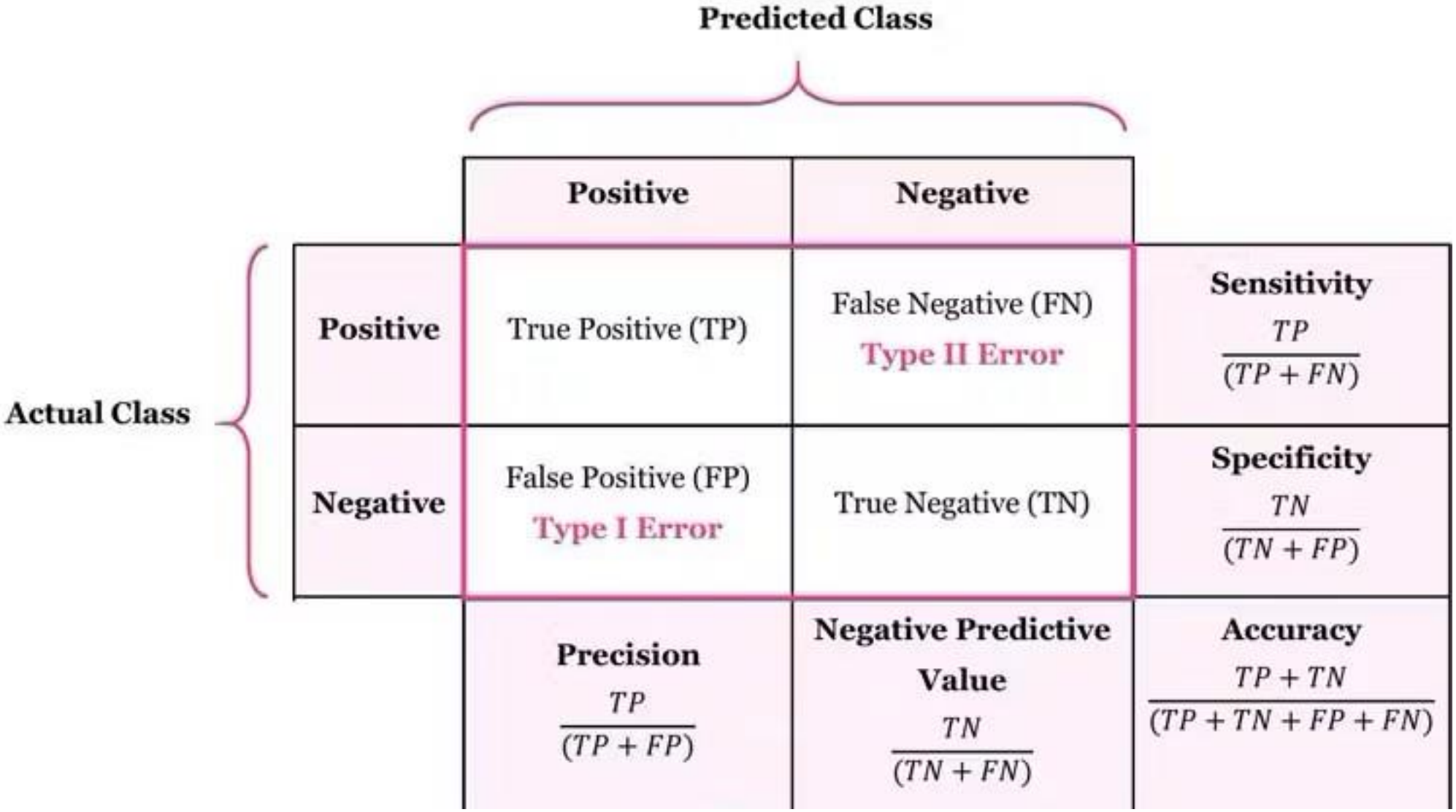
$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

2. Precision

Precision focus on the quality of the model's positive predictions. It tells us how many of the "positive" predictions were actually correct. It is important in situations where false positives need to be minimized such as detecting spam emails or fraud. The formula of precision is:

$$\text{Precision} = \frac{TP}{TP+FP}$$

Confusion Matrix



3. Recall

Recall measures how good the model is at predicting positives. It shows the proportion of true positives detected out of all the actual positive instances. High recall is essential when missing positive cases has significant consequences like in medical tests.

$$\text{Recall} = \frac{TP}{TP+FN}$$

4. F1-Score

F1-score combines precision and recall into a single metric to balance their trade-off. It provides a better sense of a model's overall performance particularly for imbalanced datasets. It is helpful when both false positives and false negatives are important though it assumes precision and recall are equally important but, in some situations, one might matter more than the other.

$$\text{F1-Score} = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

5. Specificity

Specificity is another important metric in the evaluation of classification models particularly in binary classification. It measures the ability of a model to correctly identify negative instances. Specificity is also known as the True Negative Rate Formula is given by:

$$\text{Specificity} = \frac{TN}{TN+FP}$$

Type 1 and Type 2 error

•**Type 1 error**: It occurs when the model incorrectly predicts a positive instance, but the actual instance is negative. This is also known as a **false positive**. Type 1 Errors affect the **precision** of a model which measures the accuracy of positive predictions.

$$\text{Type 1 Error} = \frac{\text{FP}}{\text{FP}+\text{TN}}$$

•**Type 2 error**: This occurs when the model fails to predict a positive instance even though it is actually positive. This is also known as a **false negative**. Type 2 Errors impact the **recall** of a model which measures how well the model identifies all actual positive cases.
Example: A diagnostic test is used to detect a particular disease in patients.

$$\text{Type 2 Error} = \frac{FN}{TP+FN}$$

References :

1. Introduction of Machine Learning book

2. sml-book- draft

3- https://scikit-learn.org/stable/modules/svm.html#svm-kernels

4- https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/

Thank you