MODULE 4 ML

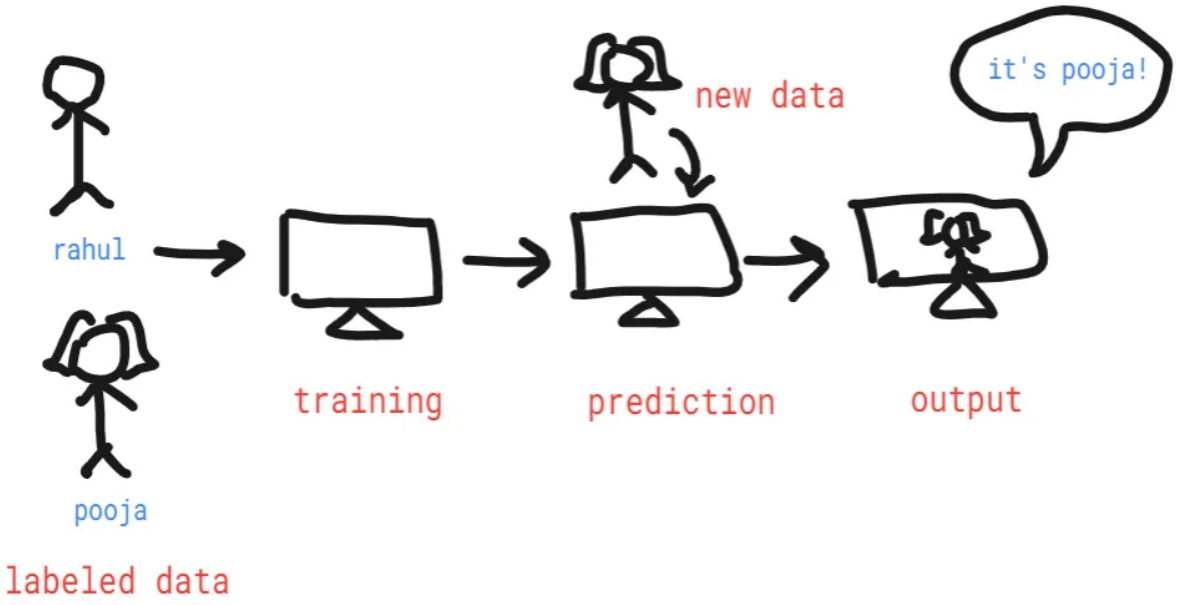
Q1.Support Vector Machines

ANS.

Support Vector Machine (SVM) stands out as a widely utilized **Supervised Learning** algorithm, serving both **Classification** and **Regression** tasks, although it is predominantly recognized for its effectiveness in classification within the realm of Machine Learning.

The central objective of SVM is to **craft an optimal line or decision boundary capable of partitioning an n-dimensional space into distinct classes**. This delineation ensures the accurate categorization of new data points in subsequent instances. The pivotal construct in this process is the **hyperplane**, representing the optimal decision boundary.

SVM identifies critical points, known as **support vectors**, strategically positioned to contribute to the formation of the hyperplane. The algorithm's nomenclature, Support Vector Machine, is derived from the emphasis on these significant support vectors in defining the **best possible decision boundary**.



The above image is a general depiction of how a support vector works

TYPES OF SVM

There are two types of SVMs:

1. **Linear SVM**: This type of SVM is used when input data is **linearly separable**, i.e, if a dataset can be classified into two classes by using a single straight line.
2. **Non-linear SVM**: This type of SVM is used when input data is **not linearly separable**, i.e, if a dataset cannot be classified by using a single straight line.

OTHER IMPORTANT TERMS

1. **Hyperplane**:

In an n-dimensional space, there exist numerous lines or decision boundaries capable of segregating classes. However, the objective is to identify the **optimal decision boundary**, referred to as the **hyperplane**, to effectively classify data points.

The configuration of the hyperplane is contingent upon the features within the dataset. For instance, in a scenario with two features (as illustrated in the image), the hyperplane manifests as a straight line. Conversely, with three features, the hyperplane transforms into a 2-dimensional plane.

The emphasis in SVM is consistently placed on constructing a hyperplane with **maximum margin** — denoting the greatest distance between data points. This approach aims to enhance the robustness of classification.

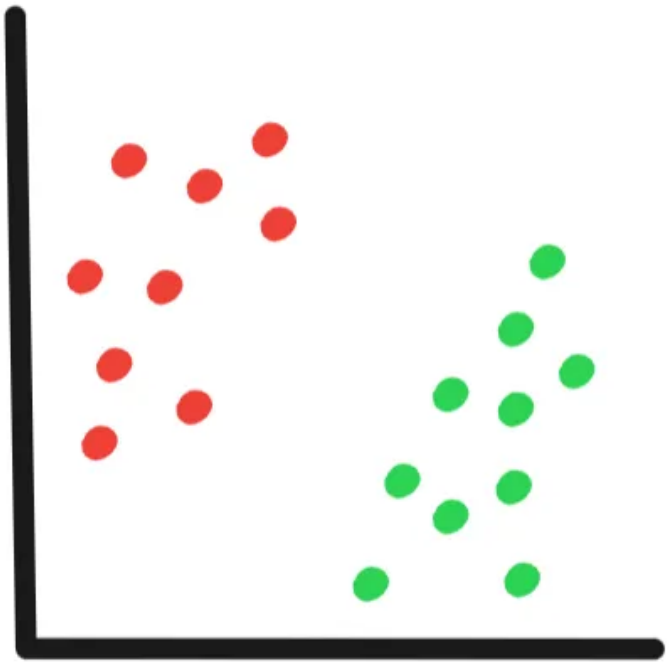
2. **Support Vector**:

The data points or vectors that are the **closest to the hyperplane,** which affect the position of the hyperplane are termed as support vectors. Since these vectors **support the hyperplane**, they are called support vectors.

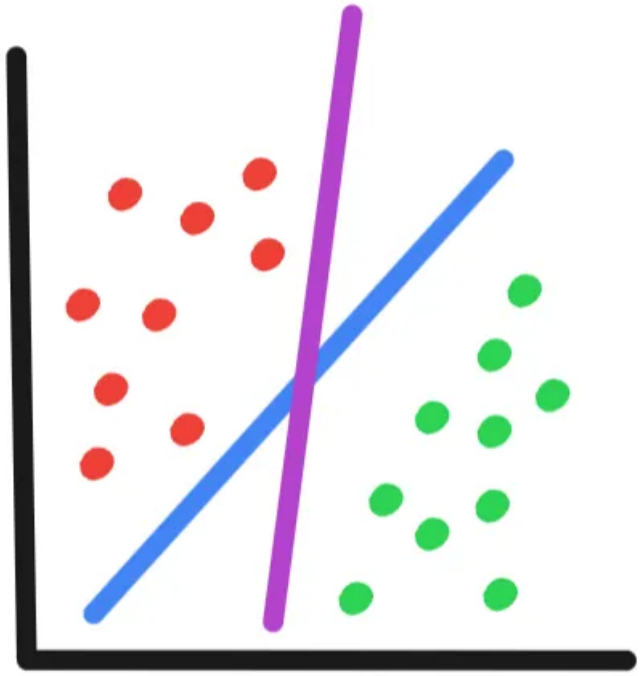
HOW SVM WORKS

Linear SVM:

Consider a dataset with two set of balls, red and green. The dataset has two features, p1 and p2. We want to classify the coordinate pair (p1,p2) into either red balls or green balls. The image is shown below:

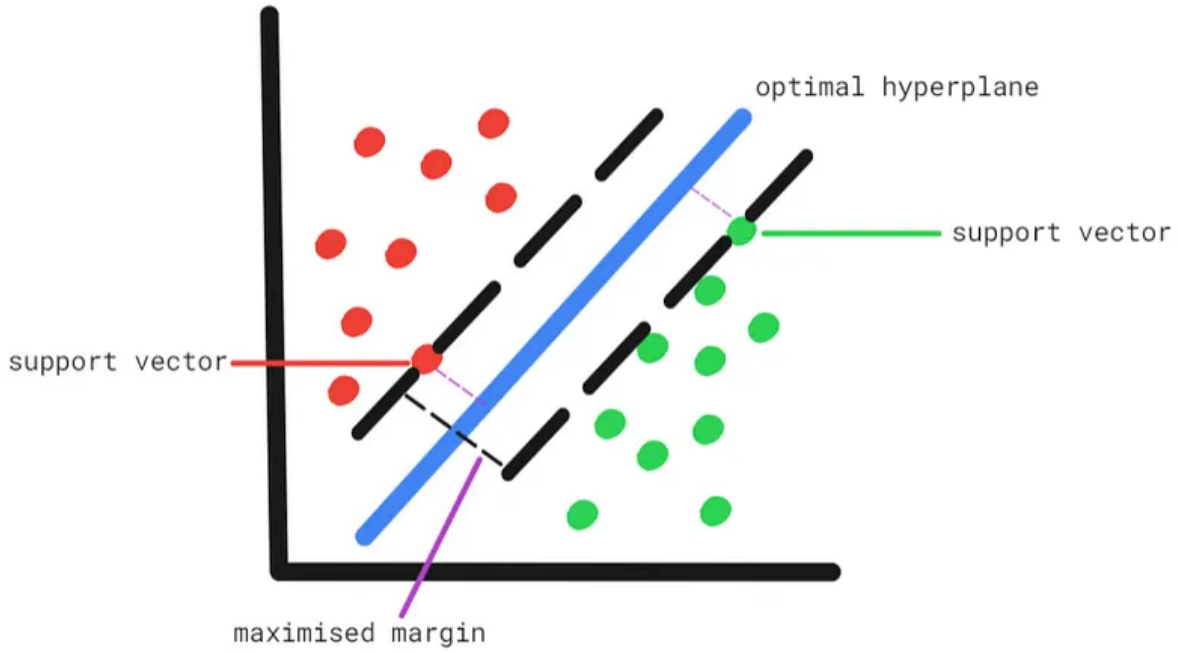


As it is a 2-dimensional space, we can easily separate these two classes by using a straight line. But there can be multiple lines that can separate these classes. Consider the below image:



Hence, the SVM algorithm helps to find the best line or decision boundary, called the **hyperplane**. SVM finds the closest point of the lines from both the classes. These points are called support vectors.

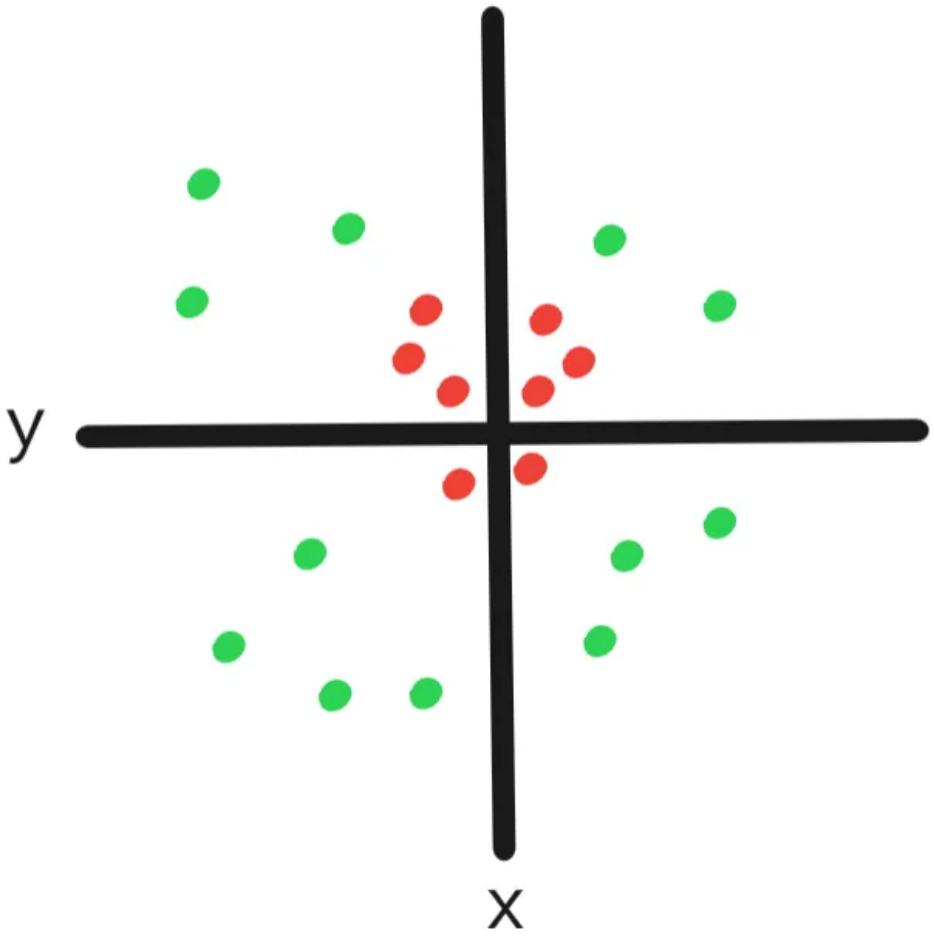
The distance between the vectors and the hyperplane is called as **margin**. And the goal of SVM is to **maximize this margin**. The **hyperplane** with maximum margin is called the **optimal hyperplane**.



Non-Linear SVM:

If data is linearly arranged, then we can separate it by using linear SVM, but for non-linear data, we cannot draw a single straight line.

Consider the below image:

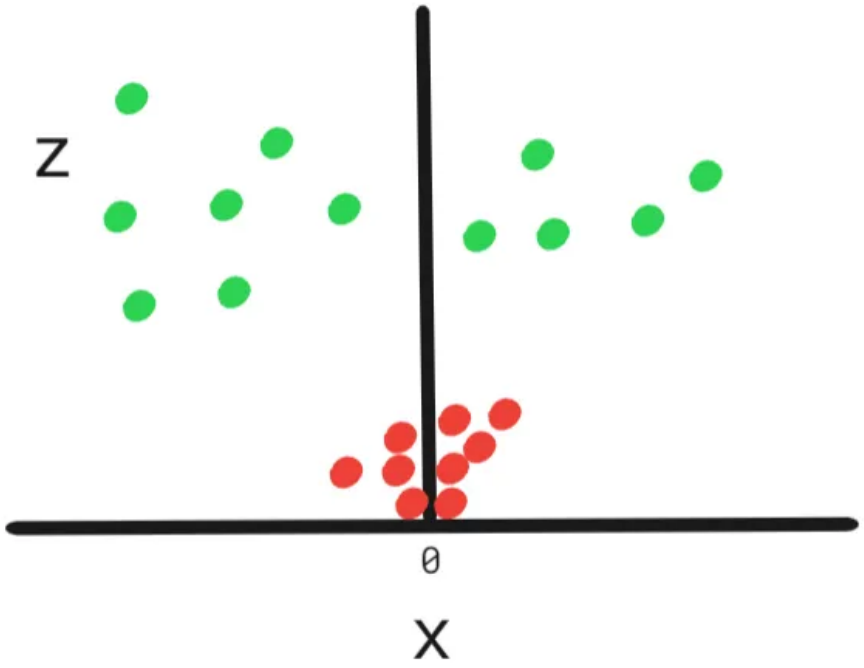


To separate these data points, we need to add one more dimension. For linear data, we used two dimensions x and y, so for non-linear data, we will add a third dimension z.

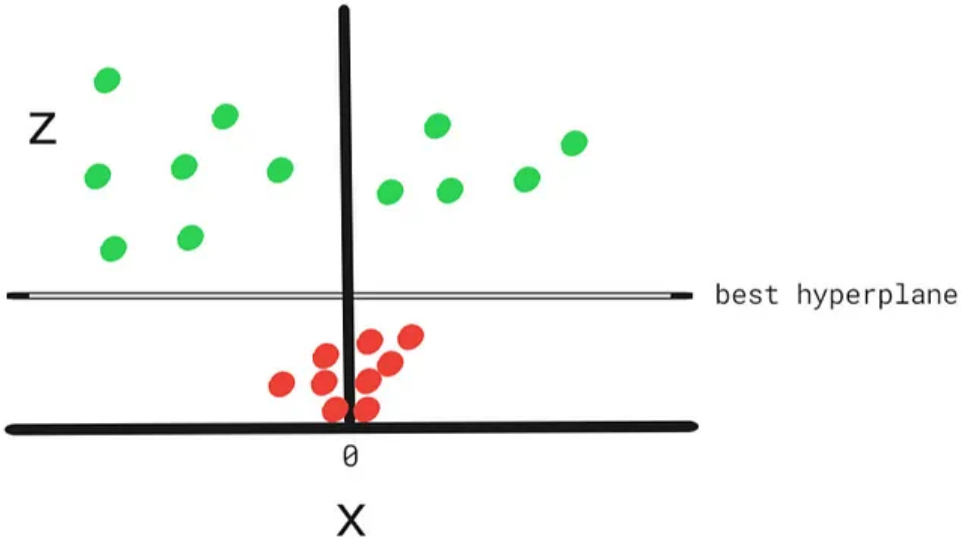
It can be calculated as:

```
z = x^2 + y^2
```

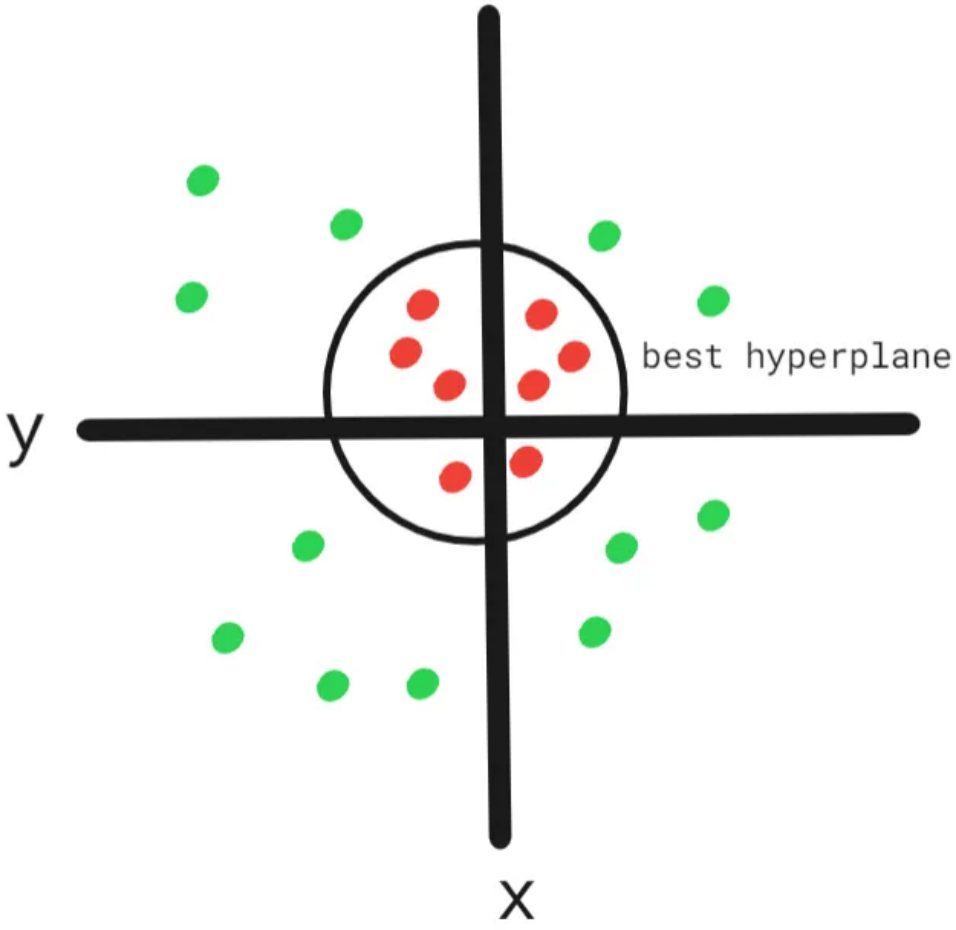
By adding the third dimension, the sample space will become as below image:



Now we will use SVM to divide the datasets into two classes in the following way.



Since we are in 3-dimensional space, it looks like a plane parallel to the x-axis. If we convert it in 2-dimensional space with z=1, then it can be depicted as:



Support Vector Machines (SVM) are a powerful classification technique that relies on finding the hyperplane that best separates different classes. The concept of "margin" is crucial in SVM, and it's categorized into two types: **Hard Margin** and **Soft Margin**.

1. Hard Margin SVM

- **Definition**: Hard margin SVM assumes that the data is perfectly linearly separable. It attempts to find the maximum margin hyperplane that separates all data points without any errors.

- **Constraints**:yi(w·xi+b)≥1,for all i

  yi(w·xi+b)≥1,for all iy_i (\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad \text{for all } i

  where yiy_iyi is the class label, w\mathbf{w}w is the weight vector, and bbb is the bias.

- **Limitations**:
  - **Sensitive to outliers**: A single outlier can drastically affect the position of the hyperplane.
  - **Applicable only to linearly separable data**: If the data is not linearly separable, no solution exists.
- **Diagram**:
  - A simple 2D plot where two classes are perfectly separable by a straight line, with no data points on the wrong side of the margin.

2. Soft Margin SVM

- **Definition**: Soft margin SVM allows for some misclassifications to achieve a better generalization for non-separable data. It introduces slack variables ξi\xi_iξi that permit some data points to be within the margin or even misclassified.
- **Objective Function**:min21∥w∥2+Ci=1∑nξiyi(w·xi+b)≥1−ξi,ξi≥0,for all i

  min12∥w∥2+C∑i=1nξi\min \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \xi_i

  Subject to:

  yi(w·xi+b)≥1−ξi,ξi≥0,for all iy_i (\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \text{for all } i

  Here, CCC is a regularization parameter that controls the trade-off between maximizing the margin and minimizing the classification error.

- **Advantages**:
  - **Robust to outliers**: Can handle noisy data by allowing some misclassifications.

    - **Applicable to non-linearly separable data**: Works even when classes overlap.
- **Diagram**:
    - A 2D plot where the data points are not perfectly separable. Some points might lie within the margin or on the wrong side of the hyperplane.

Linear vs. Non-Linear vs. Kernelized SVM

1. Linear SVM

- **Definition**: Linear SVM is used when data is linearly separable, meaning the classes can be separated by a straight line (or a hyperplane in higher dimensions).
- **Characteristics**:
    - The decision boundary is a linear hyperplane.
    - Efficient and simple for linearly separable data.
- **Diagram**:
    - A 2D plot where two classes are separated by a straight line.

2. Non-Linear SVM

- **Definition**: Non-linear SVM is used when the data is not linearly separable in the original feature space.
- **Characteristics**:
    - Uses a non-linear decision boundary.
    - Often achieved by using the kernel trick (but not always).
- **Diagram**:
    - A 2D plot where two classes are separated by a non-linear boundary, such as a curve.

3. Kernelized SVM

- **Definition**: Kernelized SVM transforms the original feature space into a higher-dimensional space using a kernel function, where the data becomes linearly separable.
- **Kernel Functions**:
  - **Linear Kernel**: K(xi,xj)=xi·xj

    K(xi,xj)=xi·xjK(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j
  - **Polynomial Kernel**: K(xi,xj)=(xi·xj+c)d

    K(xi,xj)=(xi·xj+c)dK(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + c)^d
  - **RBF (Gaussian) Kernel**: K(xi,xj)=exp(−γ∥xi−xj∥2)

    K(xi,xj)=exp(−γ∥xi−xj∥2)K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2)
- **Advantages**:
  - Can model complex decision boundaries.
  - Very powerful and flexible due to the ability to choose different kernels.
- **Diagram**:
  - A 2D plot where a linear decision boundary in a higher-dimensional space translates to a non-linear boundary in the original space.

Feature	Hard Margin SVM	Soft Margin SVM
Definition	Assumes the data is perfectly linearly separable.	Allows some misclassification for better generalization.
Margin	Maximizes the margin with no data points inside the margin.	Maximizes the margin while allowing some points inside it.
Handling Outliers	Sensitive to outliers; a single outlier can shift the hyperplane.	Robust to outliers; outliers can be within the margin or misclassified.

Feature	Hard Margin SVM	Soft Margin SVM
Applicability	Only works when data is perfectly separable.	Applicable even when data is not linearly separable.
Objective Function	Minimizes under the constraint that all data points are perfectly classified.	Minimizes where slack variables allowing some misclassifications.

Hard Margin vs Soft Margin in SVM

Criteria	Hard Margin	Soft Margin
Objective Function	Maximize margin.	Maximize margin, minimize margin violations.
Handling Noise	Sensitive, requires **perfectly linearly separable data**	Robust, handles noisy data with margin violations.
Regularization	Not applicable, no regularization parameter	Controlled by regularization parameter C.
Complexity	Simple, computationally efficient	May require more computational resources

Feature	Linear SVM	Non-Linear SVM	Kernelized SVM
Definition	Uses a linear hyperplane to separate data.	Uses a non-linear decision boundary in the original feature space.	Transforms the original space into a higher dimension using a kernel function to make data linearly separable.
Applicability	Suitable for linearly separable data.	Suitable for data that is not linearly separable.	Suitable for complex datasets where linear boundaries in

Feature	Linear SVM	Non-Linear SVM	Kernelized SVM
			the original space are insufficient.
Kernel Usage	No kernel function used; works directly in the original space.	No explicit kernel, but works in the original feature space.	Uses various kernels (e.g., RBF, Polynomial) to transform data into a higher-dimensional space.
Decision Boundary	A straight line or hyperplane.	A curved or complex boundary in the original feature space.	A linear hyperplane in a higher-dimensional space, which corresponds to a non-linear boundary in the original space.
Examples	Spam detection with a linear separator between spam and not spam.	XOR problem where a simple linear boundary cannot separate the classes.	Image classification where pixels are mapped to higher dimensions to find a separating hyperplane.

Q.

1. Introduction to ID3

- **ID3 (Iterative Dichotomiser 3)** is a decision tree algorithm developed by Ross Quinlan in the 1980s. It was designed to create a decision tree by recursively splitting the dataset based on the attribute that provides the maximum Information Gain.
- **Working Mechanism**:
    - **Information Gain**: At each node, ID3 calculates the Information Gain for each attribute, which measures how well an attribute separates the

training examples according to their class labels. The attribute with the highest Information Gain is chosen as the splitting criterion.

    - The algorithm repeats this process recursively until all data is perfectly classified or no further splits can be made.

2. Limitations of ID3

- **Handling Continuous Attributes**:
    - ID3 cannot directly handle continuous data. It requires discretization, meaning continuous variables must be divided into categorical bins, which can lead to a loss of information and may reduce the model's accuracy.
- **Overfitting**:
    - ID3 builds a tree that perfectly classifies the training data. While this might seem ideal, it leads to **overfitting**, especially when the training data is noisy or contains anomalies. An overfitted model performs poorly on unseen data, as it captures noise rather than the underlying pattern.
- **Bias Towards Attributes with Many Values**:
    - Information Gain tends to prefer attributes with many distinct values. For example, an ID or a timestamp might have a high Information Gain just because it uniquely identifies each record, leading to an unnecessarily complex and less generalizable tree.
- **No Pruning Mechanism**:
    - ID3 does not include any pruning mechanism. Once the tree is built, it remains as is, regardless of whether some branches contribute little to the overall accuracy. This often results in large, complex trees that are difficult to interpret and prone to overfitting.
- **Handling Missing Data**:
    - ID3 is not well-equipped to handle missing data. It either ignores instances with missing values or fills them with default values, both of which can introduce bias and degrade model performance.

3. Introduction of C4.5 and its Improvements

- **C4.5** is an extension of ID3, also developed by Ross Quinlan, designed to overcome the shortcomings of ID3. It introduces several enhancements that make it more robust, versatile, and widely applicable.
- **Key Improvements**:
    1. **Handling Continuous Attributes**:
        - Unlike ID3, C4.5 can directly handle continuous attributes by determining the best split points based on the data. This makes the algorithm applicable to a broader range of problems and improves accuracy.
    2. **Pruning**:
        - C4.5 includes a **post-pruning** step, where branches that do not contribute significantly to the predictive power of the model are pruned. This reduces overfitting and leads to more generalizable models.
    3. **Gain Ratio**:
        - To address ID3's bias towards attributes with many distinct values, C4.5 uses **Gain Ratio** instead of Information Gain. Gain Ratio adjusts the Information Gain by considering the number of distinct values in the attribute, reducing the tendency to choose attributes with many unique values.
    4. **Handling Missing Values**:
        - C4.5 handles missing data more gracefully by assigning probabilities to different possible outcomes rather than ignoring them or using default values. This reduces bias and allows the algorithm to make better use of all available data.
    5. **Improved Tree Structure**:
        - While ID3 generates a tree with multi-way splits, C4.5 can handle both binary and multi-way splits, depending on what is most appropriate for the data. Additionally, the pruning process ensures that the final tree is more compact and interpretable.

4. Conclusion: Why C4.5 is Preferred

- **Versatility**: C4.5 is capable of handling both continuous and categorical attributes, making it more versatile than ID3, which is limited to categorical attributes.
- **Robustness**: The post-pruning mechanism in C4.5 makes it more robust to overfitting, leading to models that perform better on unseen data.
- **Accuracy**: By using Gain Ratio and handling missing data effectively, C4.5 often produces more accurate models than ID3.
- **Efficiency**: Although more complex, C4.5's efficiency in producing more compact and interpretable trees outweighs the potential computational overhead compared to ID3.
- **Generalization**: C4.5's enhancements lead to better generalization to new, unseen data, making it more suitable for real-world applications where data is often noisy and incomplete.

C4.5 is an evolutionary step from ID3, addressing its key limitations and extending its applicability to a wider range of problems, making it a more preferred choice in many scenarios.

Q. gini index

ANS

The Gini Index is a proportion of impurity or inequality in statistical and monetary settings. In machine learning, it is utilized as an impurity measure in decision tree algorithms for classification tasks. The Gini Index measures the probability of a haphazardly picked test being misclassified by a decision tree algorithm, and its value goes from 0 (perfectly pure) to 1 (perfectly impure).

Steps to Determine the Root Node Using Gini Index (CART):

1. **Calculate Gini impurity for each attribute:**Gini=1−i=1∑n(pi)2

Gini Impurity Formula:

Gini=1−∑i=1n(pi)2Gini = 1 - \sum_{i=1}^{n} (p_i)^2

where pip_ipi is the proportion of class iii in the node.

Let's compute the Gini impurity for each attribute:

The attribute with least gini ndex is selected as root.

Real-World Applications of Gini Index

The Gini Index has been utilized in different applications in machine learning, for example, extortion location, credit scoring, and client division. For example, in extortion discovery, the Gini Index can be utilized to distinguish designs in exchange data and recognize bizarre way of behaving. In credit scoring, the Gini Index can be utilized to foresee the probability of default in view of variables like income, relationship of outstanding debt to take home pay, and record of loan repayment. In client division, the Gini Index can be utilized to bunch clients in view of their way of behaving and inclinations.

Information gain:

Information gain is an action used to assess the nature of a split while building a decision tree. The objective of a decision tree is to split the data into subsets that are basically as homogeneous as conceivable as for the objective variable, so the subsequent tree can be utilized to make exact expectations on new data. Information gain measures the decrease in entropy or impurity accomplished by a split. The feature with the most noteworthy information gain is chosen as the best feature to split on at every node of the decision tree.

Information gain is a normally involved measure for assessing the nature of splits in decision trees, yet it isn't the one to focus on. Different measures, for example, the Gini index or misclassification rate, can likewise be utilized. The decision of

splitting basis relies upon the main issue and the attributes of the dataset being utilized.