UNIT-I

Machine Learning (ML) is a subset of artificial intelligence (AI) that enables systems to learn from data, identify patterns, and make decisions with minimal human intervention. In simple terms, machine learning allows computers to automatically improve their performance over time through experience.

1. Types of Machine Learning

Machine learning is broadly categorized into three types:

a. Supervised Learning

- **Definition:** In supervised learning, the algorithm is trained on a labeled dataset, meaning that each input comes with the correct output (target). The model learns from the training data and tries to make predictions on new, unseen data.
- **Goal:** The goal is to learn a mapping from inputs to outputs based on examples.
- **Common Algorithms:**
  - Linear Regression
  - Decision Trees
  - Support Vector Machines (SVM)
  - Neural Networks
- **Examples:**
  - Predicting house prices based on features like size, location, etc.
  - Classifying emails as spam or not spam.

b. Unsupervised Learning

- **Definition:** In unsupervised learning, the algorithm is given data without explicit labels or outputs. The goal is to find hidden structures or patterns within the data.
- **Goal:** To discover the underlying structure of data or group similar data points together.
- **Common Algorithms:**
  - Clustering (e.g., K-Means, Hierarchical Clustering)
  - Principal Component Analysis (PCA)
  - Association Rules (e.g., Apriori)
- **Examples:**
  - Customer segmentation in marketing (grouping customers based on purchasing behavior).
  - Reducing the dimensionality of a dataset while preserving key information.

c. Reinforcement Learning

- **Definition:** In reinforcement learning, the model learns through trial and error by interacting with an environment. It receives feedback in the form of rewards or penalties and adjusts its actions to maximize the total reward.

2. Key Concepts in Machine Learning

a. Model

A **model** in machine learning is a mathematical representation that the algorithm uses to make predictions or decisions. It is the output of the learning process and is constructed by training the model on the dataset.

- **Example:** A linear regression model might predict house prices based on features like area and number of bedrooms.

b. Training

Training refers to the process of feeding data into the machine learning algorithm so it can learn the relationship between the inputs (features) and outputs (labels). The algorithm uses this data to adjust its parameters to minimize prediction errors.

c. Dataset

A **dataset** is a collection of data used to train and evaluate a machine learning model. A typical dataset consists of:

- **Features (Input Variables):** These are the attributes or independent variables used to predict the target.
- **Labels (Target Variables):** These are the outputs or dependent variables that the model is trying to predict (used only in supervised learning).

Example:

Area	Bedrooms	Price (Label)
1200	3	$300,000
1500	4	$350,000

d. Overfitting and Underfitting

- **Overfitting:** Occurs when a model is too complex and learns the noise in the training data, leading to poor performance on unseen data. The model "memorizes" rather than generalizes.
- **Underfitting:** Occurs when a model is too simple and fails to capture the underlying patterns in the data, resulting in poor performance on both the training and test data.

e. Bias-Variance Tradeoff

- **Bias:** Refers to the error introduced by simplifying assumptions made by the model. High bias can lead to underfitting.
- **Variance:** Refers to the model's sensitivity to small fluctuations in the training data. High variance can lead to overfitting.

The **bias-variance tradeoff** is a key concept in machine learning that describes the balance between these two sources of error.

f. Generalization

The ability of a machine learning model to perform well on unseen or new data is called **generalization**. A model that generalizes well captures the true patterns in the data rather than noise.

---

3. Steps in the Machine Learning Process

a. Data Collection

The first step in machine learning is gathering relevant data. This data can come from various sources like sensors, databases, or user interactions. The quality and quantity of data are crucial for training a good model.

b. Data Preprocessing

Raw data is often noisy, incomplete, or unstructured. Preprocessing steps include:

- **Cleaning:** Removing missing or erroneous data.
- **Normalization/Standardization:** Scaling data to a standard range (e.g., 0-1).
- **Encoding:** Converting categorical data into numerical form (e.g., one-hot encoding).
- **Splitting:** Dividing the dataset into training, validation, and test sets.

c. Feature Engineering

Feature engineering involves selecting and transforming raw data into meaningful inputs for the model. This may include:

- **Feature Selection:** Identifying the most relevant features for the task.
- **Feature Transformation:** Creating new features based on existing ones (e.g., adding polynomial terms).

d. Model Selection

Choosing the right algorithm or model is key. Different algorithms work better for different types of problems (e.g., linear regression for predicting continuous values, decision trees for classification).

e. Training the Model

The selected model is trained on the training data. During this process, the model learns to map input features to the correct output labels by minimizing a loss function.

f. Model Evaluation

After training, the model is evaluated on a separate test dataset to measure its performance. Common evaluation metrics include:

- **Accuracy:** Proportion of correct predictions (for classification).
- **Mean Squared Error (MSE):** Average squared difference between predicted and actual values (for regression).
- **Precision, Recall, F1-Score:** Metrics for classification tasks, particularly in imbalanced datasets.

g. Model Tuning

Once the model is trained, **hyperparameter tuning** may be done to improve performance. Techniques like grid search or random search can be used to find the best hyperparameters for the model.

4. Common Machine Learning Algorithms

a. Regression Algorithms (for continuous output):

- **Linear Regression:** Predicts a continuous value based on a linear relationship between input and output.
- **Ridge and Lasso Regression:** Variants of linear regression that use regularization to prevent overfitting.

b. Classification Algorithms (for categorical output):

- **Logistic Regression:** Predicts a binary outcome (e.g., yes/no) based on input features.
- **Decision Trees:** Splits data into branches based on feature values, making decisions at each node.
- **Random Forest:** An ensemble of decision trees that improves prediction accuracy and reduces overfitting.
- **Support Vector Machines (SVM):** Finds the hyperplane that best separates classes in feature space.
- **K-Nearest Neighbors (KNN):** Classifies a data point based on the majority class among its k nearest neighbors.

c. Clustering Algorithms (for unsupervised learning):

- **K-Means:** Divides data into k clusters based on similarity.
- **Hierarchical Clustering:** Builds a hierarchy of clusters using either an agglomerative or divisive approach.

d. Dimensionality Reduction:

- **Principal Component Analysis (PCA):** Reduces the dimensionality of the data while preserving as much variance as possible.
- **t-SNE (t-Distributed Stochastic Neighbor Embedding):** A non-linear dimensionality reduction technique often used for visualizing high-dimensional data.

---

5. Key Applications of Machine Learning

- **Healthcare:** Predicting diseases, personalized treatment plans.
- **Finance:** Fraud detection, stock price prediction.
- **Marketing:** Customer segmentation, recommendation systems.
- **Self-Driving Cars:** Real-time decision-making based on sensor data.
- **Natural Language Processing (NLP):** Sentiment analysis, language translation, chatbots.

Summary:

- **Machine Learning** allows computers to learn from data and make predictions or decisions.
- **Supervised, Unsupervised, and Reinforcement Learning** are the main types of ML.
- Key concepts include **features**, **labels**, **training**, **overfitting**, and **generalization**.
- ML involves steps like **data collection, preprocessing, model selection, training, and evaluation**.
- Common algorithms include **linear regression, decision trees, SVM, K-Means**, and **PCA**.

Supervised learning

Supervised learning is a type of machine learning where an algorithm is trained on labeled data. In this approach, the input data is paired with the correct output, allowing the model to learn the relationship between inputs and outputs. Once trained, the model can predict the output for new, unseen data based on what it has learned.

.

Common Algorithms in Supervised Learning:

- **Linear Regression**: Predicts a continuous output (e.g., predicting house prices).
- **Logistic Regression**: Used for binary classification problems (e.g., email spam detection).
- **Decision Trees**: A flowchart-like model for making decisions based on features.
- **Support Vector Machines (SVM)**: Finds a hyperplane to separate different classes of data.
- **K-Nearest Neighbors (KNN)**: Classifies data based on the majority class of its nearest neighbors.
- **Neural Networks**: Complex models inspired by the human brain, useful for tasks like image and speech recognition.

Example Workflow in Supervised Learning:

1. **Data Collection**: Gather labeled data.
2. **Data Preprocessing**: Clean and prepare the data for training (e.g., handling missing values, normalizing features).
3. **Model Selection**: Choose a suitable supervised learning algorithm.
4. **Training**: Feed the model with training data and tune its parameters.
5. **Evaluation**: Test the model on unseen data to assess its accuracy.
6. **Deployment**: Once a satisfactory performance is achieved, the model is deployed to make predictions on real-world data.

Applications of Supervised Learning:

- **Spam Detection**: Classifying emails as spam or not spam.
- **Image Classification**: Labeling objects in images (e.g., identifying animals in photos).
- **Sentiment Analysis**: Predicting the sentiment of text (e.g., classifying movie reviews as positive or negative).
- **Medical Diagnosis**: Predicting the likelihood of a disease based on patient data.

Supervised learning is widely used in various domains due to its ability to learn from examples and generalize to unseen data.

Distance-based methods in supervised machine learning are algorithms that use the distance between data points to make predictions. These methods rely on the assumption that points that are closer in feature space are more likely to share the same label (for classification) or have similar values (for regression).

Here are the **main distance-based methods specifically in supervised learning**:

---

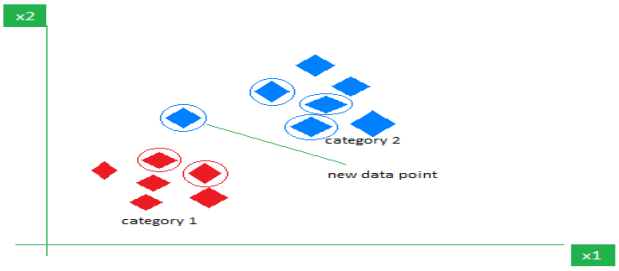
1. K-Nearest Neighbors (KNN)

- **Type**: Classification and Regression
- **Concept**: KNN is one of the simplest distance-based supervised learning algorithms. It assigns a label or value to a new data point based on the majority label (for classification)

What is the K-Nearest Neighbors Algorithm?

KNN is one of the most basic yet essential classification algorithms in machine learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining, and intrusion detection.

We are given some prior data (also called training data), which classifies coordinates into groups identified by an attribute.
As an example, consider the following table of data points containing two features:



KNN Algorithm working visualization

Now, given another set of data points (also called testing data), allocate these points to a group by analyzing the training set.

Why do we need a KNN algorithm?

(K-NN) algorithm is a versatile and widely used machine learning algorithm that is primarily used for its simplicity and ease of implementation.
It does not require any assumptions about the underlying data distribution. It can also handle both numerical and categorical data, making it a flexible choice for various types of datasets in classification and regression tasks. It is a non-parametric method that makes predictions based on the similarity of data points in a given dataset. K-NN is less sensitive to outliers compared to other algorithms.

The K-NN algorithm works by finding the K nearest neighbors to a given data point based on a distance metric, such as Euclidean distance. The class or value of the data point is then determined by the majority vote or average of the K neighbors. This approach allows the algorithm to adapt to different patterns and make predictions based on the local structure of the data.

Distance Metrics Used in KNN Algorithm

As we know that the KNN algorithm helps us identify the nearest points or the groups for a query point. But to determine the closest groups or the nearest points for a query point we need some metric. For this purpose, we use below distance metrics:

Euclidean Distance

This is nothing but the cartesian distance between the two points which are in the plane/ hyperplane. Euclidean distance can also be visualized as the length of the

straight line that joins the two points which are into consideration. This metric helps us calculate the net displacement done between the two states of an object.

$$\text{distance}(x, X_i) = \sqrt{\sum_{j=1}^{d} (x_j - X_{i_j})^2]}$$

Manhattan Distance

Manhattan Distance metric is generally used when we are interested in the total distance traveled by the object instead of the displacement. This metric is calculated by summing the absolute difference between the coordinates of the points in n-dimensions.

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

Minkowski Distance

We can say that the Euclidean, as well as the Manhattan distance, are special cases of the Minkowski distance.

$$d(x, y) = (\sum_{i=1}^{n} (x_i - y_i)^p)^{\frac{1}{p}}$$

From the formula above we can say that when p = 2 then it is the same as the formula for the Euclidean distance and when p = 1 then we obtain the formula for the Manhattan distance.

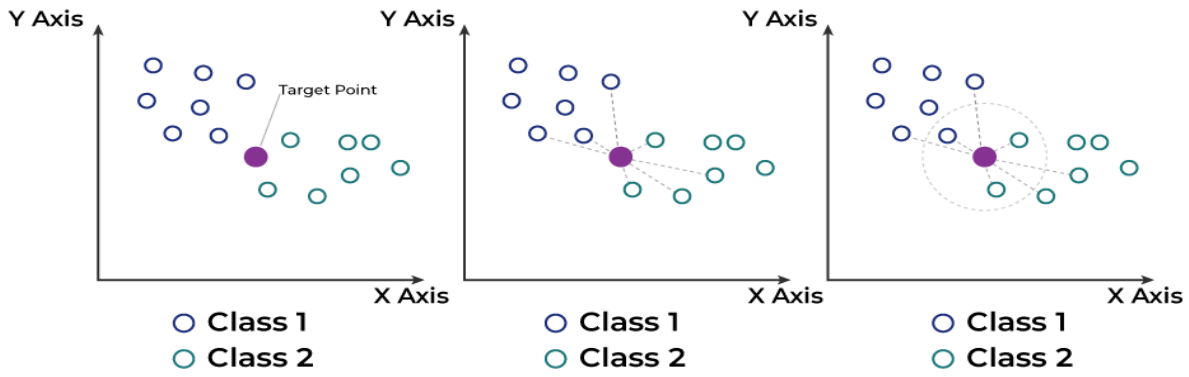
.

How to choose the value of k for KNN Algorithm?

The value of k is very crucial in the KNN algorithm to define the number of neighbors in the algorithm. The value of k in the k-nearest neighbors (k-NN) algorithm should be chosen based on the input data. If the input data has more outliers or noise, a higher value of k would be better. It is recommended to choose an odd value for k to avoid ties in classification.

Workings of KNN algorithm

The K-Nearest Neighbors (KNN) algorithm operates on the principle of similarity, where it predicts the label or value of a new data point by considering the labels or values of its K nearest neighbors in the training dataset.



Step-by-Step explanation of how KNN works is discussed below:

Step 1: Selecting the optimal value of K

- K represents the number of nearest neighbors that needs to be considered while making prediction.

Step 2: Calculating distance

- To measure the similarity between target and training data points, Euclidean distance is used. Distance is calculated between each of the data points in the dataset and target point.

Step 3: Finding Nearest Neighbors

- The k data points with the smallest distances to the target point are the nearest neighbors.

Step 4: Voting for Classification or Taking Average for Regression

- In the classification problem, the class labels of K-nearest neighbors are determined by performing majority voting. The class with the most occurrences among the neighbors becomes the predicted class for the target data point.
- In the regression problem, the class label is calculated by taking average of the target values of K nearest neighbors. The calculated average value becomes the predicted output for the target data point.

Advantages of the KNN Algorithm

- **Easy to implement** as the complexity of the algorithm is not that high.
- **Adapts Easily** – As per the working of the KNN algorithm it stores all the data in memory storage and hence whenever a new example or data point is added then the algorithm adjusts itself as per that new example and has its contribution to the future predictions as well.
- **Few Hyperparameters** – The only parameters which are required in the training of a KNN algorithm are the value of k and the choice of the distance metric which we would like to choose from our evaluation metric.

Disadvantages of the KNN Algorithm

- **Does not scale** – As we have heard about this that the KNN algorithm is also considered a Lazy Algorithm. The main significance of this term is that this takes lots of computing power as well as data storage. This makes this algorithm both time-consuming and resource exhausting.
- **Curse of Dimensionality** – There is a term known as the peaking phenomenon according to this the KNN algorithm is affected by the curse of dimensionality which implies the algorithm faces a hard time classifying the data points properly when the dimensionality is too high.
- **Prone to Overfitting** – As the algorithm is affected due to the curse of dimensionality it is prone to the problem of overfitting as well. Hence generally feature selection as well as dimensionality reduction techniques are applied to deal with this problem.

- **Pros**:

- Simple and easy to implement.
- Non-parametric: No need to make assumptions about the data distribution.

Cons:

- Computationally expensive, especially for large datasets.
- Sensitive to irrelevant features and the curse of dimensionality.

Applications:

- Image recognition, recommendation systems, and text classification.

---

2. Distance-Weighted KNN

- **Type**: Classification and Regression
- **Concept**: A variation of KNN where neighbors closer to the test point are given more weight in determining the output.
- **How it works**:
    - Similar to KNN, but instead of simply taking a majority vote (for classification) or averaging (for regression), each neighbor's contribution is weighted by its distance.
    - The closer the neighbor, the greater its influence on the prediction.
    - Common weighting functions include inverse distance (1/d) or Gaussian decay.

Pros:

- Gives more importance to closer neighbors, improving predictions in some cases.
- Helps mitigate cases where outliers are in close proximity to the decision boundary.

Cons:

- Computational complexity similar to KNN.
- Requires a good choice of weighting function.

Applications:

- Similar to KNN, but especially useful where the importance of proximity is critical (e.g., medical diagnosis).

---

3. Support Vector Machines (SVM)

- **Type**: Classification (can also be extended to regression via Support Vector Regression)
- **Concept**: SVM is a supervised learning algorithm that finds the hyperplane that best separates data points from different classes. In its kernelized form, SVM can be considered distance-based, as it computes distances between data points using kernel functions.
- **How it works**:
    - SVM identifies the support vectors, which are the data points closest to the decision boundary (hyperplane).
    - The goal is to maximize the distance (margin) between the support vectors and the hyperplane.
    - If the data is not linearly separable, SVM uses a **kernel function** (e.g., radial basis function (RBF), polynomial) to map the data into a higher-dimensional space, where it computes distances to create a more flexible decision boundary.

Distance metrics:

- SVM with linear kernel uses the Euclidean distance to define the hyperplane.
- SVM with the RBF kernel computes distance in a transformed feature space.

Pros:

- Effective in high-dimensional spaces.
- Works well when there's a clear margin of separation between classes.

Cons:

- Sensitive to outliers.
- Choosing the right kernel function and hyperparameters can be complex.

Applications:

- Text categorization, image classification, and bioinformatics.

---

4. Learning Vector Quantization (LVQ)

- **Type**: Classification
- **Concept**: LVQ is a prototype-based distance-learning algorithm that assigns a class label based on the closest "prototype" or reference vector for each class.
- **How it works**:
    - During training, the algorithm learns a set of prototype vectors for each class.
    - When making predictions, the algorithm finds the nearest prototype to the input data and assigns the class label of that prototype.
    - The prototypes are adjusted during training using a distance metric (usually Euclidean distance).

Pros:

- Simple and interpretable, as it reduces the dataset to a smaller set of representative prototypes.
- Effective for classification tasks where the decision boundaries are non-linear.

Cons:

- Sensitive to the initial choice of prototypes.
- Not as widely used as other distance-based methods.

Applications:

- Pattern recognition, speech recognition, and image classification.

---

5. Radius-Based Nearest Neighbors (RNN)

- **Type**: Classification and Regression
- **Concept**: Instead of selecting a fixed number of neighbors KKK like in KNN, RNN selects all points within a fixed radius rrr of the query point.
- **How it works**:

- All points within the specified radius are considered as neighbors.
- For classification, a majority vote determines the class.
- For regression, the average value of the neighbors is used as the prediction.

Pros:

- More flexible than KNN when the data is unevenly distributed, as it adapts based on density.
- Avoids the problem of selecting an arbitrary number of neighbors KKK.

Cons:

- Choosing an appropriate radius rrr can be challenging, and results are sensitive to this parameter.
- Can have no neighbors or too many neighbors, depending on the radius.

Applications:

- Similar to KNN but useful when data density is variable across regions.

---

6. Mahalanobis Distance-Based Classification

- **Type**: Classification
- **Concept**: This method computes the Mahalanobis distance between points and uses it for classification, especially when features are correlated.
- **How it works**:
  - Mahalanobis distance takes into account the covariance of the data and is a multivariate generalization of Euclidean distance.
  - It measures the distance between a point and the mean of a distribution.
  - It is particularly useful when the data has different scales or when features are correlated.

Pros:

- Handles correlated and multi-dimensional data well.
- More accurate in cases where the assumption of independence between features is violated.

Cons:

- Requires computing the covariance matrix, which can be computationally expensive for large datasets.
- Sensitive to outliers, which can skew the covariance matrix.

Applications:

- Outlier detection, anomaly detection, and classification in multi-dimensional datasets (e.g., finance, biology).

Summary:

Distance-based methods in supervised learning are intuitive and rely on the idea that similar data points tend to have similar outputs. The performance of these algorithms is highly dependent on the choice of the distance metric, feature scaling, and the presence of irrelevant features. Common distance-based methods

include **K-Nearest Neighbors (KNN)**, **Support Vector Machines (SVM)**, **Distance-Weighted KNN**, **Learning Vector Quantization (LVQ)**, and **Mahalanobis Distance-based Classification**.

They are widely used in real-world applications such as recommendation systems, image and text classification, medical diagnosis, and anomaly detection.

5. Applications of Nearest Neighbors Algorithms:

1. Image Recognition:

- Nearest neighbors are commonly used in image recognition tasks, such as the classification of handwritten digits (e.g., MNIST dataset). The feature space might consist of pixel intensities, and images that are similar (closer) in this space are likely to share the same label.

2. Recommendation Systems:

- In collaborative filtering, KNN can be used to find users similar to a given user and recommend items based on the preferences of those users.

3. Anomaly Detection:

- Nearest neighbors can help detect outliers by flagging points that have no close neighbors, which could indicate they are anomalies in the dataset.

4. Medical Diagnosis:

- In medical applications, KNN can classify patients based on symptoms or test results, with the assumption that similar patients are likely to have the same diagnosis.

Choosing the Right Value of KKK:

- **Small KKK**: Leads to a more complex decision boundary (high variance, low bias) and is more sensitive to noise.
- **Large KKK**: Results in a smoother decision boundary (low variance, high bias) and is less sensitive to noise.
- **Best practice**: Use cross-validation to find the optimal value of KKK for your specific dataset.

Strengths of Nearest Neighbor Algorithms:

1. **Intuitive and easy to understand**: Simple algorithm with no complex training phase.
2. **Versatility**: Can be applied to both classification and regression problems.
3. **Non-parametric**: Makes no assumptions about the underlying distribution of the data.

Weaknesses of Nearest Neighbor Algorithms:

1. **Computational complexity**: Especially for large datasets, computing the distance between the test point and all training points can be slow.
2. **Sensitive to irrelevant features**: The inclusion of irrelevant or redundant features can skew the distance calculations, leading to poor predictions.
3. **Curse of dimensionality**: As the number of features increases, the distances between points become less meaningful, reducing the effectiveness of the algorithm.

Techniques to Improve Nearest Neighbors:

1. **Feature scaling**: Normalizing or standardizing features ensures that no single feature dominates the distance calculation.
2. **Dimensionality reduction**: Techniques like **Principal Component Analysis (PCA)** or **t-SNE** can reduce the number of dimensions, mitigating the curse of dimensionality.
3. **Approximate Nearest Neighbors (ANN)**: Instead of computing exact nearest neighbors, algorithms like **KD-trees** or **Ball Trees** can approximate the nearest neighbors to reduce computation time.

Conclusion:

Nearest neighbors methods, particularly KNN, are straightforward and effective for many supervised learning tasks. They make predictions based on the similarity (distance) between data points, making them particularly useful for classification and regression when the decision boundary is complex or when there is no clear mathematical model for the data. Despite their simplicity, nearest neighbor methods face challenges like computational cost and sensitivity to high-dimensional data, but these can be mitigated through techniques like feature scaling

Decision Tree in Machine Learning

A decision tree is a type of supervised learning algorithm that is commonly used in machine learning to model and predict outcomes based on input data.
It is a tree-like structure where **each internal node tests on attribute**,
each branch corresponds to attribute value and each leaf node represents the final decision or prediction.
The decision tree algorithm falls under the category of supervised learning.

They can be used to solve both **regression** and **classification problems**.

Decision Tree Terminologies

There are specialized terms associated with decision trees that denote various components and facets of the tree structure and decision-making procedure. :

- **Root Node:** A decision tree's root node, which represents the original choice or feature from which the tree branches, is the highest node.
- **Internal Nodes (Decision Nodes)**: Nodes in the tree whose choices are determined by the values of particular attributes. There are branches on these nodes that go to other nodes.

- **Leaf Nodes (Terminal Nodes)**: The branches' termini, when choices or forecasts are decided upon. There are no more branches on leaf nodes.
- **Branches (Edges)**: Links between nodes that show how decisions are made in response to particular circumstances.
- **Splitting**: The process of dividing a node into two or more sub-nodes based on a decision criterion. It involves selecting a feature and a threshold to create subsets of data.
- **Parent Node**: A node that is split into child nodes. The original node from which a split originates.
- **Child Node**: Nodes created as a result of a split from a parent node.
- **Decision Criterion**: The rule or condition used to determine how the data should be split at a decision node. It involves comparing feature values against a threshold.
- **Pruning**: The process of removing branches or nodes from a decision tree to improve its generalisation and prevent overfitting.

Understanding these terminologies is crucial for interpreting and working with decision trees in machine learning applications.

How Decision Tree is formed?

The process of forming a decision tree involves recursively partitioning the data based on the values of different attributes. The algorithm selects the best attribute to split the data at each internal node, based on certain criteria such as information gain or Gini impurity. This splitting process continues until a stopping criterion is met, such as reaching a maximum depth or having a minimum number of instances in a leaf node.

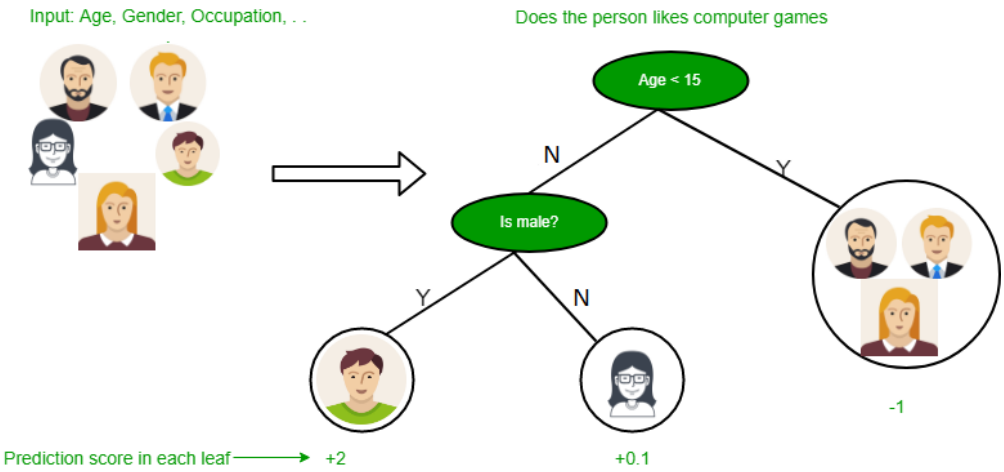
Why Decision Tree?

Decision trees are widely used in machine learning for a number of reasons:

- Decision trees are so versatile in simulating intricate decision-making processes, because of their interpretability and versatility.
- They provide comprehensible insights into the decision logic, decision trees are especially helpful for tasks involving categorisation and regression.
- They are proficient with both numerical and categorical data, and they can easily adapt to a variety of datasets thanks to their autonomous feature selection capability.
- Decision trees also provide simple visualization, which helps to comprehend and elucidate the underlying decision processes in a model.

Decision Tree Approach

Decision tree uses the tree representation to solve the problem in which each leaf node corresponds to a class label and attributes are represented on the internal node of the tree. We can represent any boolean function on discrete attributes using the decision tree.



Types of Decision Trees

- **Classification Trees**: Used when the target variable is categorical. The leaves of the tree represent class labels, and the internal nodes represent decisions based on features.
- **Regression Trees**: Used when the target variable is continuous. The leaves of the tree represent the predicted values (e.g., prices, temperatures), and internal nodes make decisions based on splitting continuous features.

Pruning the Tree

Pruning is a technique used to reduce the size of the tree and prevent **overfitting**:

- **Pre-pruning (Early Stopping)**: Stop splitting a node before it becomes a leaf node if the split does not result in a significant improvement (e.g., if a minimum number of samples per node is not met).
- **Post-pruning**: After the tree has been fully grown, remove nodes that add little value to the prediction. This can be done by collapsing nodes whose removal reduces prediction error (measured via cross-validation).

Advantages and Disadvantages of Decision Trees

Advantages:

1. **Simple to understand and interpret**: Decision trees mimic human decision-making and are easy to visualize.
2. **Non-parametric**: They do not require assumptions about the data distribution.
3. **Can handle both categorical and numerical features**: Decision trees are flexible and can work with a mix of feature types.
4. **Feature Importance**: They can compute feature importance scores based on how much each feature reduces impurity in the tree.

Disadvantages:

1. **Prone to overfitting**: Decision trees can grow deep and fit to the noise in the data, especially if they are not pruned.
2. **Instability**: Small changes in the data can result in drastically different trees.
3. **Greedy algorithm**: The algorithm makes local decisions at each node without considering future splits, which might not always lead to the globally optimal tree.

4. **Bias toward features with more levels**: Features with many distinct values (such as continuous features) tend to be selected more often for splits.

Popular Decision Tree Algorithms

1. ID3 (Iterative Dichotomiser 3)

- **Used for classification**.
- It uses **information gain** as the criterion to select the feature that best separates the data at each step.
- ID3 can only handle categorical features.

2. CART (Classification and Regression Trees)

- Can be used for both **classification** and **regression** tasks.
- For classification, CART uses the **Gini impurity** as the splitting criterion.
- For regression, CART uses the **mean squared error (MSE)** to choose the best splits.
- CART produces binary trees (i.e., each node has exactly two child nodes).

3. C4.5

- An extension of ID3 that can handle both categorical and continuous features.
- It uses **gain ratio** (a normalized version of information gain) to decide the best splits.
- C4.5 can also handle missing values and allows post-pruning.

4. CHAID (Chi-squared Automatic Interaction Detection)

- Used for **classification**.
- CHAID uses a **Chi-squared test** to find the feature that is most significantly associated with the target variable.

Understanding Naive Bayes and Machine Learning

Naive Bayes is a powerful and simple algorithm for supervised learning, especially for tasks like text classification. It assumes conditional independence among features, which simplifies the calculations but may not hold in all cases. Despite its limitations, Naive Bayes remains popular due to its scalability, speed, and effectiveness in specific types of problems like spam filtering, sentiment analysis, and medical diagnosis

Machine learning falls into two categories:

- Supervised learning
- Unsupervised learning

Supervised learning falls into two categories:

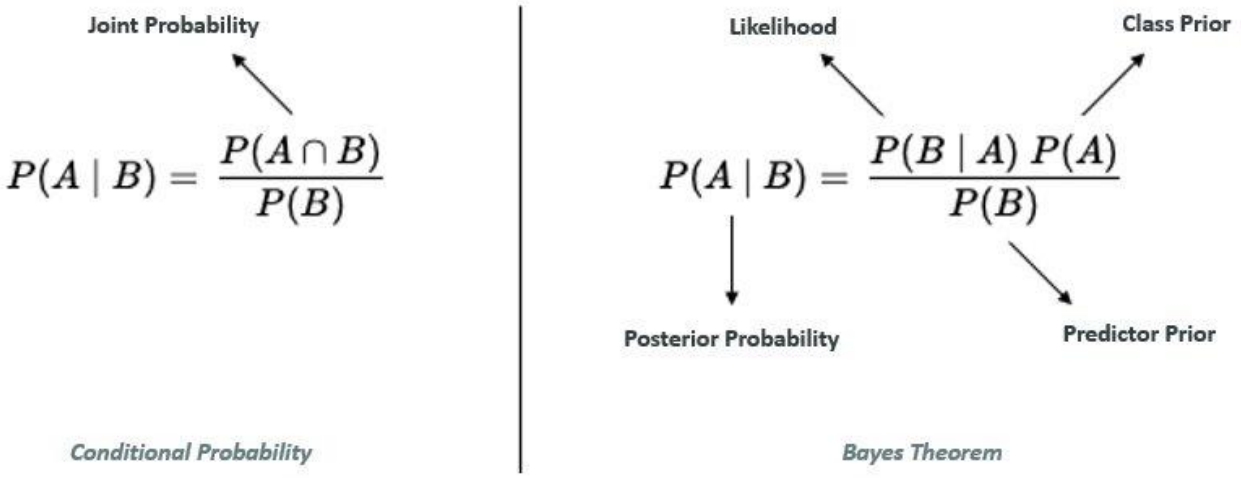
- Classification
- Regression

Naive Bayes algorithm falls under classification.

Bayes' Theorem describes the probability of an event based on prior knowledge of the conditions that might be related to the event.

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)}$$

This equation is derived from the formula of conditional probability given below:



In fact, Bayes' Theorem is nothing but an alternate way of calculating the conditional probability of an event. When the reverse conditional probability is easier to calculate than the joint probability, Bayes' Theorem is used.

You can use Bayes' theorem to build a learner ML model, from an existing set of attributes, that predicts the probability of the response variable belonging to some class, given a new set of attributes.

Consider the previous equation again. Now, assume that event A is the response variable and event B is the input attribute. So according to the equation,

- **P(A) or Class Prior** is the prior probability of the response variable
- **P(B) or Predictor Prior** is the evidence or the probability of training data
- **P(A|B) or Posterior Probability** is the conditional probability of the response variable being of a particular value given the input attributes
- **P(B|A) or Likelihood** is basically the reverse of the posterior probability or the likelihood of training data

Stepwise Bayes Theorem

Let's come back to the problem at hand. Looks like you're very serious with your resolution this time given that you have been keeping track of the weather outside for the past two weeks:

Step 1 – Collect raw data

Day	Outlook	Humidity	Wind	Run
D1	Sunny	High	Weak	No
D2	Overcast	High	Strong	No
D3	Rainy	High	Weak	Yes
D4	Rainy	Normal	Weak	No
D5	Sunny	Normal	Weak	Yes
D6	Sunny	High	Weak	Yes
D7	Sunny	High	Weak	Yes
D8	Rainy	Normal	Strong	No
D9	Overcast	High	Weak	Yes
D10	Sunny	High	Weak	Yes
D11	Rainy	High	Weak	No
D12	Overcast	Normal	Strong	No
D13	Overcast	High	Weak	Yes
D14	Sunny	High	Weak	Yes

Next, you need to create a frequency table for each attribute of your dataset.

Step 2 – Convert data to a frequency table(s)

Frequency Table		Run	
		Yes	No
Outlook	Sunny	5	1
	Overcast	2	2
	Rainy	1	3

Frequency Table		Run	
		Yes	No
Humidity	High	7	3
	Normal	1	3

Frequency Table		Run	
		Yes	No
Wind	Strong	0	3
	Weak	9	2

Then, for each frequency table, you will create a likelihood table.

Step 3 – Calculate prior probability and evidence

Likelihood Table		Run		
		Yes	No	
Outlook	Sunny	5/8	1/6	6/14
	Overcast	2/8	2/6	4/14
	Rainy	1/8	3/6	4/14
		8/14	6/14	

Likelihood Table for Outlook

Likelihood Table		Run		
		Yes	No	
Humidity	High	7/8	3/6	10/14
	Normal	1/8	3/6	4/14
		8/14	6/14	

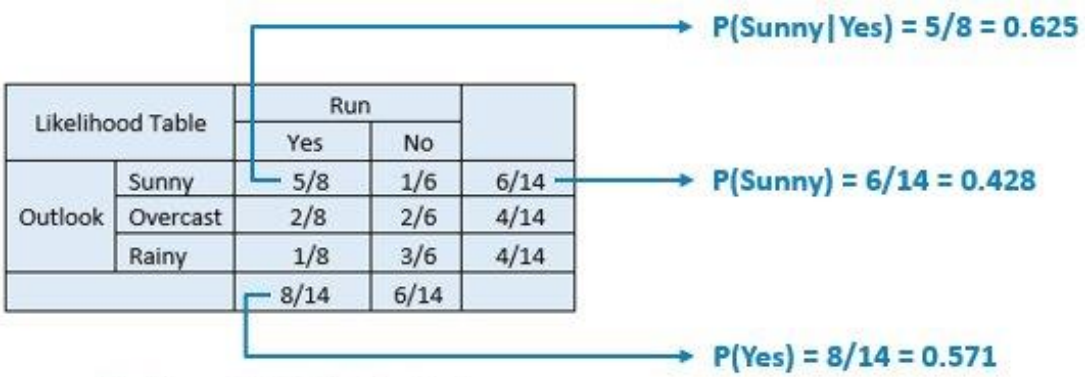
Likelihood Table for Humidity

Likelihood Table		Run		
		Yes	No	
Wind	Strong	0/9	3/5	3/14
	Weak	9/9	2/5	11/14
		9/14	5/14	

Likelihood Table for Wind

Step 4 – Apply probabilities to Bayes' Theorem equation

Let's say you want to focus on the likelihood that you go for a run given that it's sunny outside.



Likelihood Table		Run		
		Yes	No	
Outlook	Sunny	5/8	1/6	6/14
	Overcast	2/8	2/6	4/14
	Rainy	1/8	3/6	4/14
		8/14	6/14	

Likelihood Table for Outlook

P(Yes|Sunny) = P(Sunny|Yes) * P(Yes) / P(Sunny) = 0.625 * 0.571 / 0.428 = **0.834**

The Naïve Bayes Algorithm

Naïve Bayes assumes conditional independence over the training dataset. The classifier separates data into different classes according to the Bayes' Theorem. But assumes that the relationship between all input features in a class is independent. Hence, the model is called ***naïve***.

This helps in simplifying the calculations by dropping the denominator from the formula while assuming independence:

Bayes Theorem ⟶ $$P(A \mid x_1, \ldots, x_n) = \frac{P(x_1, \ldots, x_n \mid A)\, P(A)}{P(x_1, \ldots, x_n)}$$

Naïve Bayes ⟶ $$P(A \mid x_1, \ldots, x_n) = P(x_1 \mid A) \cdot P(x_2 \mid A) \cdot P(x_i \mid A)\, P(A)$$

Let's understand this through our running resolution example:

Say you want to predict if on the coming Wednesday, given the following weather conditions, should you go for a run or sleep in:

Outlook: Rainy

Humidity: Normal

Wind: Weak

Run: ?
Likelihood of 'Yes' on Wednesday:

P(Outlook = Rainy|Yes) * P(Humidity = Normal|Yes) * P(Wind = Weak|Yes) * P(Yes)

= 1/8 * 1/8 * 9/9 * 8/14 = **0.0089**
Likelihood of 'No' on Wednesday:

P(Outlook = Rainy|No) * P(Humidity = Normal|No) * P(Wind = Weak|No) * P(No)

= 3/6 * 3/6 * 2/5 * 6/14 = **0.0428**
Now, to determine the probability of going for a run on Wednesday, you just need to divide P(Yes) with the sum of the likelihoods of Yes and No.

P(Yes) = 0.0089 / (0.0089 + 0.0428) = 0.172

Similarly, P(No) = 0.0428 / (0.0089 + 0.0428) = 0.827

According to your model, it looks like there's an almost 83% probability that you're going to stay under the covers next Wednesday!

This was just a fun example. Although Naïve Bayes IS used for weather predictions, for advanced machine learning problems, the complexity of the Bayesian classifier needs to be reduced for it to be practical. This is where the naïve in Naïve Bayes comes in.

Industrial Applications of Naïve Bayes

- **Text Classification:** Naïve Bayes algorithm is almost always used as a classifier and is an excellent choice for spam filtering of your emails or news categorization on your smartphone.
- **Recommendation Systems:** Naïve Bayes is used with collaborative filtering to build recommendation systems for you. The '*Because you watched* ____' section on Netflix is exactly that.

Sentiment Analysis: Naïve Bayes is an effective algorithm for the identification of positive or negative sentiments of a target group (customers, audience, etc.). Think of feedback forms and IMDb reviews.

Types of Naïve Bayes Classifiers

Bernoulli Naïve Bayes

Bernoulli Naive Bayes (for binary data)

- This is a variant of Naive Bayes for **binary features** (features that can take on only two values, e.g., 0 or 1).
- For each feature, the model estimates the probability of the feature being present (1) or absent (0) in each class.
- **Application**: Used for tasks like document classification when features represent binary occurrences of words (i.e., whether a word appears or not in a document).

Multinomial Naïve Bayes

Multinomial Naive Bayes (for discrete, count-based data)

- This variant is used for **discrete data**, typically when features represent counts or frequencies, such as word counts in a text document.
- The likelihood P(X|C) is calculated based on the frequency of features in each class.
- **Application**: Commonly used for text classification problems, like spam detection, sentiment analysis, and document classification.

Gaussian Naïve Bayes

- Used when data is as per the Gaussian distribution
- Predictors are continuous variables
  - **Application**: Often used when the data contains continuous features (e.g., predicting age, height, etc.).
- (for continuous data

Advantages of Naïve Bayes

- Easy to work with when using binary or categorical input values.
- Require a small number of training data for estimating the parameters necessary for classification.
- Handles both continuous and discrete data.
- Fast and reliable for making real-time predictions.
  - **Simple and Fast**: Naive Bayes is highly efficient, both in terms of computational complexity and memory usage. It's easy to implement and fast to train and predict.
  - **Scalable**: It performs well on large datasets because the complexity is linear with respect to the number of features and instances.
  - **Works Well with High-Dimensional Data**: Naive Bayes can handle high-dimensional data (e.g., text data with thousands of words) quite well.
  - **Robust to Irrelevant Features**: The naive assumption of independence means that irrelevant features don't impact the model significantly.
  - **Performs Well with Small Datasets**: Naive Bayes can perform well even with smaller datasets, unlike more complex models like deep learning.
-

Limitations of Naïve Bayes

- Assumes that all the features are independent, which is highly unlikely in practical scenarios.
- Unsuitable for numerical data.
- The number of features must be equal to the number of attributes in the data for the algorithm to make correct predictions.
- Encounters 'Zero Frequency' problem: If a categorical variable has a category in the test dataset that wasn't included in the training dataset, the model will assign it a 0 probability and will be unable to

make a prediction. This problem can be resolved using smoothing techniques which are out of the scope of this article.

- Computationally expensive when used to classify a large number of items.

- **Assumption of Independence**: The major limitation is the naive assumption that features are independent. In many real-world applications, features are often correlated, and this assumption may not hold, potentially leading to poor performance.
- **Zero Frequency Problem**: If a category or feature value is not present in the training data for a given class, Naive Bayes will assign a probability of zero, which can be problematic. This can be handled by using techniques like **Laplace smoothing** (adding a small value to the counts).
- **Not Ideal for Complex Relationships**: Naive Bayes is less effective in datasets where complex relationships between features exist because the algorithm doesn't capture interactions between features.

Applications of Naive Bayes

Naive Bayes is widely used in applications where the independence assumption is reasonable or where speed and simplicity are more important than model accuracy. Some of the most common applications include:

1. Text Classification:

- **Spam filtering**: Classifying emails as spam or not spam based on the occurrence of words.
- **Sentiment analysis**: Determining whether a text expresses a positive or negative sentiment.
- **Document classification**: Categorizing text documents (e.g., news articles) into predefined categories.
- **Topic modeling**: Assigning topics to text based on the frequency of words in the document.

2. Recommendation Systems:

- Naive Bayes is used in collaborative filtering to recommend products, services, or content based on user preferences or behavior.

3. Medical Diagnosis:

- In medical applications, Naive Bayes can be used to predict diseases based on symptoms, where each symptom is treated as a feature.

4. Real-Time Predictions:

- Due to its simplicity and speed, Naive Bayes is well-suited for real-time prediction systems, such as predicting user preferences on websites or flagging fraudulent transactions in real-time.

Face Recognition

As a classifier, it is used to identify the faces or its other features, like nose, mouth, eyes, etc.

Weather Prediction

It can be used to predict if the weather will be good or bad.

Medical Diagnosis

Doctors can diagnose patients by using the information that the classifier provides. Healthcare professionals can use Naive Bayes to indicate if a patient is at high risk for certain diseases and conditions, such as heart disease, cancer, and other ailments.

News Classification

With the help of a Naive Bayes classifier, Google News recognizes whether the news is political, world news, and so on.
As the Naive Bayes Classifier has so many applications, it's worth learning more about how it works.

Example of Naive Bayes Classification

Let's assume we have a simple dataset where we're classifying whether an email is "Spam" or "Not Spam" based on words present in the email (binary features).

- **Training data**:
  - Email 1: "Buy", "Discount", "Limited" → Spam
  - Email 2: "Meeting", "Tomorrow", "Agenda" → Not Spam
  - Email 3: "Buy", "Offer", "Limited" → Spam
- **Test email**: "Buy", "Limited", "Agenda"

Using Naive Bayes:

- Calculate P(Spam)P(Spam)P(Spam) and P(NotSpam)P(Not Spam)P(NotSpam).
- Estimate P(Buy|Spam)P(Buy|Spam)P(Buy|Spam), P(Limited|Spam)P(Limited|Spam)P(Limited|Spam), and P(Agenda|Spam)P(Agenda|Spam)P(Agenda|Spam), etc.
- Compute the posterior probabilities and classify the email as "Spam" or "Not Spam" based on the higher posterior probability.

Linear regression

Supervised learning has two types:

- **Classification**: It predicts the class of the dataset based on the independent input variable. Class is the categorical or discrete values. like the image of an animal is a cat or dog?
- **Regression**: It predicts the continuous output variables based on the independent input variable. like the prediction of house prices based on different parameters like house age, distance from the main road, location, area, etc.

Here, we will discuss one of the simplest types of regression i.e. **Linear Regression.**

What is Linear Regression?

Linear regression is a type of supervised machine learning algorithm that computes the linear relationship between the dependent variable and one or more independent features by fitting a linear equation to observed data.
When there is only one independent feature, it is known as Simple Linear Regression, and when there are more than one feature, it is known as Multiple Linear Regression.
Similarly, when there is only one dependent variable, it is considered Univariate Linear Regression, while when there are more than one dependent variables, it is known as Multivariate Regression.

Types of Linear Regression

There are two main types of linear regression:

Simple Linear Regression

This is the simplest form of linear regression, and it involves only one independent variable and one dependent variable. The equation for simple linear regression is:
$Y=\beta 0+\beta 1X$

where:

- Y is the dependent variable
- X is the independent variable
- β0 is the intercept
- β1 is the slope

Multiple Linear Regression

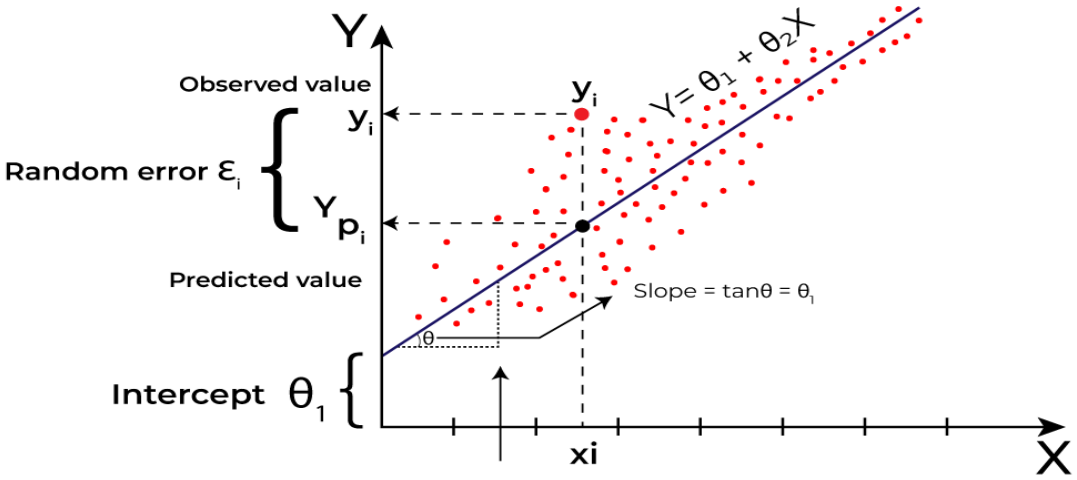
This involves more than one independent variable and one dependent variable. The equation for multiple linear regression is:
$y=\beta 0+\beta 1X1+\beta 2X2+\ldots\ldots\beta nXn y=\beta_0+\beta_1X1+\beta_2X2+\ldots\ldots\beta_nXn$
where:

- Y is the dependent variable
- X1, X2, …, Xn are the independent variables
- β0 is the intercept
- β1, β2, …, βn are the slopes

The goal of the algorithm is to find the best Fit Line equation that can predict the values based on the independent variables.

What is the best Fit Line?

Our primary objective while using linear regression is to locate the best-fit line, which implies that the error between the predicted and actual values should be kept to a minimum. There will be the least error in the best-fit line.
The best Fit Line equation provides a straight line that represents the relationship between the dependent and independent variables. The slope of the line indicates how much the dependent variable changes for a unit change in the independent variable(s).



Linear Regression

Here Y is called a dependent or target variable and X is called an independent variable also known as the predictor of Y. There are many types of functions or modules that can be used for regression. A linear function is the simplest type of function. Here, X may be a single feature or multiple features representing the problem.

Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x)). Hence, the name is Linear Regression. In the figure above, X (input) is the work experience and Y (output) is the salary of a person. The regression line is the best-fit line for our model.

We utilize the cost function to compute the best values in order to get the best fit line since different values for weights or the coefficient of lines result in different regression lines.

Hypothesis function in Linear Regression

As we have assumed earlier that our independent feature is the experience i.e X and the respective salary Y is the dependent variable. Let's assume there is a linear relationship between X and Y then the salary can be predicted using:

$\hat{Y}=\theta 1+\theta 2X$

OR

$\hat{y}_i=\theta 1+\theta 2x_i$

Here,

- $y_i \epsilon Y (i=1,2,\cdots,n)$ are labels to data (Supervised learning)
- $x_i \epsilon X (i=1,2,\cdots,n)$ are the input independent training data (univariate – one input variable(parameter))
- $\hat{y}_i \epsilon \hat{Y} (i=1,2,\cdots,n)$ are the predicted values.

The model gets the best regression fit line by finding the best θ_1 and θ_2 values.

- **θ_1:** intercept
- **θ_2:** coefficient of x

Once we find the best θ_1 and θ_2 values, we get the best-fit line. So when we are finally using our model for prediction, it will predict the value of y for the input value of x.

1. **Cost Function (Mean Squared Error)**: To find the best-fitting line, linear regression minimizes the error between the predicted values (y and the actual target values y The most common cost function used is **Mean Squared Error (MSE)**, which is defined as:

Hypothesis:	$h_\theta(x) = \theta_0 + \theta_1 x$
Parameters:	θ_0, θ_1
Cost Function:	$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$
Goal:	$\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

Given our simple linear equation $y = mx + b$, we can calculate MSE as:

$$MSE = \frac{1}{N} \sum_{i=1}^{n} (y_i - (mx_i + b))^2$$

2. **Gradient Descent**: In many cases, linear regression uses **Gradient Descent** to minimize the cost function and find the optimal weights. Gradient descent iteratively adjusts the weights in the direction that reduces the cost function:
3. **Normal Equation**: For small datasets, the optimal weights can also be computed analytically using the **Normal Equation** without requiring gradient descent. This equation minimizes the cost function in a single step:

Assumptions of Linear Regression:

1. **Linearity**: The relationship between the independent and dependent variables is linear.
2. **Independence**: The observations are independent of each other.
3. **Homoscedasticity**: The variance of residuals (errors) is constant across all levels of the independent variables.
4. **Normality**: The residuals (differences between actual and predicted values) are normally distributed.
5. **No multicollinearity**: The independent variables are not highly correlated with each other.

Applications of Linear Regression:

- **Predicting housing prices** based on features like size, location, and number of rooms.
- **Forecasting sales** based on historical data and economic indicators.
- **Risk analysis in finance**, predicting stock prices or company performance.
- **Medical prediction models**, such as estimating disease progression based on patient data.

Advantages:

- **Simplicity**: Linear regression is easy to implement and interpret.
- **Speed**: It can be computationally efficient for small to medium-sized datasets.
- **Interpretability**: The model provides insight into the relationships between features and the target.

Disadvantages:

- **Assumption of linearity**: It only works well if the relationship between the features and the target is approximately linear.
- **Sensitive to outliers**: Large outliers can have a disproportionate effect on the model.
- **Limited flexibility**: In complex datasets with nonlinear relationships, linear regression might not capture the full picture.

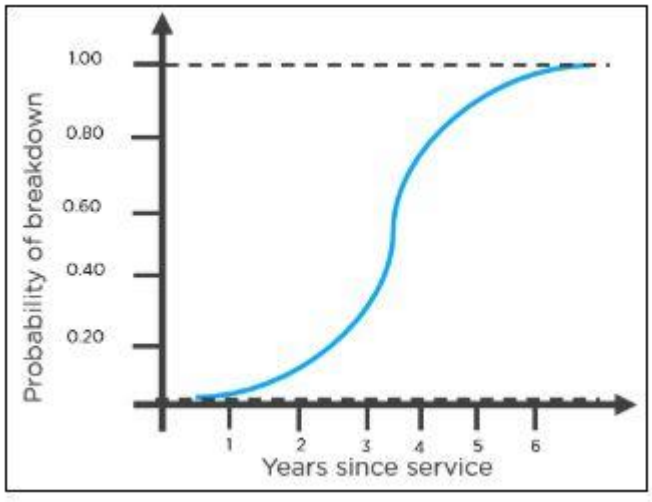
In summary, linear regression is a foundational tool in machine learning for regression tasks, providing a straightforward way to model relationships and make predictions, although it has limitations in cases where data relationships are nonlinear or complex.

What is Logistic Regression?

Logistic regression machine learning is a statistical method that is used for building machine learning models where the dependent variable is dichotomous: i.e. binary. Logistic regression is used to describe data and the relationship between one dependent variable and one or more independent variables. The independent variables can be nominal, ordinal, or of interval type.

The name “logistic regression” is **derived from the concept of the logistic function** that it uses. The logistic function is also known as **the sigmoid function**. The value of this logistic function lies between zero and one.

The following is an example of a logistic function we can use to find the probability of a vehicle breaking down, depending on how many years it has been since it was serviced last.



Here is how you can interpret the results from the graph to decide whether the vehicle will break down or not.

- Identify Key Metrics

- Understand Threshold Values
- Analyze Trend Patterns
- Compare Current Data to Baseline
- Look for Warning Signs
- Consider Historical Data
- Assess Overall Condition
- Make a Decision

Logistic Regression in Machine Learning

Logistic regression machine learning is a key classification technique. It can work with both numerical and categorical data, making it versatile for various applications. For example, it's commonly used to predict whether a customer will leave a service (churn), identify fraudulent transactions, or determine if a patient has a specific condition.

One of the main advantages of logistic regression is its simplicity. Logistic regression machine learning not only predicts outcomes but also helps understand which factors are most important for these predictions. This makes logistic regression a practical tool for solving classification problems while providing clear insights into the data. Its ease of use and interpretability make it popular in many machine-learning projects.

Logistic Function - Sigmoid Function

The sigmoid or logistic function is essential for converting predicted values into probabilities in logistic regression. This function maps any real number to a value between 0 and 1, ensuring that predictions remain within this probability range. Its "S" shaped curve helps translate raw scores into a more interpretable format.

A threshold value is used in logistic regression to make decisions based on these probabilities. For instance, if the predicted probability is above a certain threshold, such as 0.5, the result is 1. If it's below, it's classified as 0. This approach allows for clear and actionable outcomes, such as determining whether a customer will purchase a product or a patient has a particular condition based on the probability calculated by the sigmoid function.

Types of Logistic Regression

Logistic regression is a versatile machine learning algorithm used for binary and multi-class classification tasks. Depending on the nature of the dependent variable, logistic regression can be categorized into different types. The main types of logistic regression include Binary Logistic Regression, Multinomial Logistic Regression, and Ordinal Logistic Regression.

Binary Logistic Regression

Binary logistic regression is the most common type of logistic regression, where the dependent variable has only two possible outcomes or classes, typically represented as 0 and 1. It is used when the target variable is binary, such as yes/no, pass/fail, or true/false. The logistic function in binary logistic regression models the probability of an observation belonging to one of the two classes.

Multinomial Logistic Regression

Multinomial logistic regression, also known as softmax regression, is used when the dependent variable has more than two unordered categories. Unlike binary logistic regression, which deals with binary outcomes, multinomial logistic regression can handle multiple classes simultaneously. It models the probability of an observation belonging to each class using the softmax function, which ensures that the predicted probabilities sum up to one across all classes.

Ordinal Logistic Regression

Ordinal logistic regression is employed when the dependent variable has more than two ordered categories. In other words, the outcome variable has a natural ordering or hierarchy among its categories. Examples include ordinal scales like low, medium, and high, or Likert scale responses ranging from strongly disagree to strongly agree. Ordinal logistic regression models the cumulative probabilities of an observation falling into or below each category using the cumulative logistic distribution function.

Assumption in a Logistic Regression Algorithm

- In a binary logistic regression, the dependent variable must be binary
- For a binary regression, the factor level one of the dependent variables should represent the desired outcome
- Only meaningful variables should be included
- The independent variables should be independent of each other. This means the model should have little or no multicollinearity
- The independent variables are linearly related to the log odds
- Logistic regression requires quite large sample sizes

Logistic Regression Equation

The logistic regression equation is:

y= 11+ e(b0+b1x)

where x is the input value, y is the predicted probability, b0 is the intercept, and b1 is the coefficient for the input x. This equation models the probability of a binary outcome based on a linear combination of input features.

Properties of Logistic Regression Equation

Logistic regression comes with a few key characteristics that define how it works and how it's assessed:

- Bernoulli Distribution

In logistic regression, the predicted outcome is binary, meaning it follows a Bernoulli distribution. This simply means that the result can only be one of two possible values, like "yes" or "no," "success" or "failure." This fits perfectly with logistic regression's goal of classifying data into two categories.

- Maximum Likelihood Estimation

The maximum likelihood method is used to find the best-fit parameters for a logistic regression model. This technique identifies the parameter values that make the observed data most probable. In other words, it adjusts the model to match the data it has seen best, which helps make accurate predictions.

- Concordance for Model Fit

Instead of using R squared like in linear regression to measure how well the model fits the data, logistic regression algorithms use concordance. Concordance assesses how well the model ranks the predicted probabilities. It checks whether the model is good at ordering outcomes correctly rather than just fitting the data. This approach is more beneficial for classification tasks where the goal is to predict which category something belongs to.

Key Terminologies of Logistic Regression

Apart from the properties of logistic regression, several key terms are crucial for understanding how the logistic regression machine learning model works:

- Independent Variables

These are the features or factors used to predict the model's outcome. They are the inputs that help determine the value of the dependent variable. For instance, independent variables might include age, income, and past buying behavior in a model predicting whether a customer will purchase a product.

- Dependent Variable

This is the outcome the model is trying to predict. In logistic regression, the dependent variable is binary, meaning it has two possible values, such as "yes" or "no," "spam" or "not spam." The goal is to estimate the probability of this variable being in one category versus the other.

- Logistic Function

The logistic function is a formula that converts the model’s input into a probability score between 0 and 1. This score indicates the likelihood of the dependent variable being 1. It’s what turns the raw predictions into meaningful probabilities that can be used for classification.

- Odds

Odds represent the ratio of the probability of an event happening to the probability of it not happening. For example, if there’s a 75% chance of an event occurring, the odds are 3 to 1. This concept helps to understand how likely an event is compared to it not happening.

- Log-Odds

Log-odds, or the logit function, is the natural logarithm of the odds. In logistic regression, the relationship between the independent variables and the dependent variable is expressed through log-odds. This helps model how changes in the independent variables affect the likelihood of the outcome.

- Coefficient

Coefficients are the values that show how each independent variable influences the dependent variable. They indicate the strength and direction of the relationship. For example, a positive coefficient means that as the independent variable increases, the likelihood of the dependent variable being 1 also increases.

- Intercept

The intercept is a constant term in the model representing the dependent variable's log odds when all the independent variables are zero. It provides a baseline level of the dependent variable’s probability before considering the effects of the independent variables.

- Maximum Likelihood Estimation

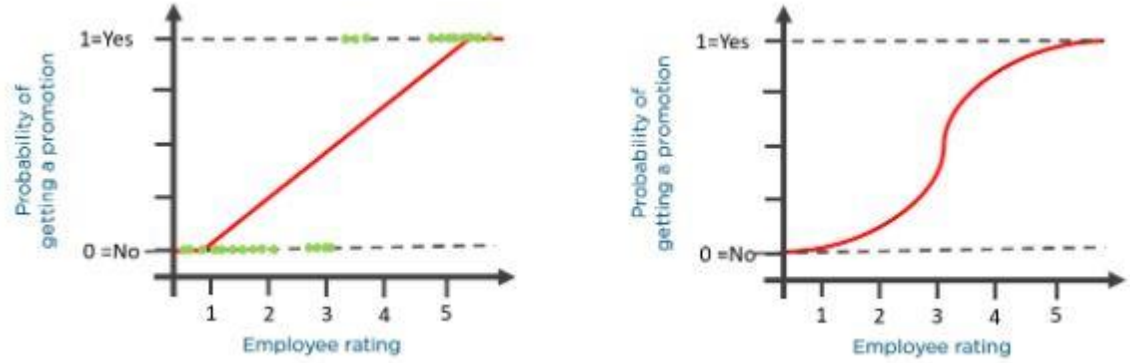
Maximum likelihood estimation (MLE) is the method used to find the best-fitting coefficients for the model. It determines the values that make the observed data most probable under the logistic regression framework, ensuring the model provides the most accurate predictions based on the given data.

How Does the Logistic Regression Algorithm Work?

Consider the following example: An organization wants to determine an employee's salary increase based on their performance.

For this purpose, a linear regression algorithm will help them decide. Plotting a regression line by considering the employee's performance as the independent variable, and the salary increase as the dependent variable will make their task easier.

Now, what if the organization wants to know whether an employee would get a promotion or not based on their performance? The above linear graph won't be suitable in this case. As such, we clip the line at zero and one, and convert it into a sigmoid curve (S curve).



Based on the threshold values, the organization can decide whether an employee will get a salary increase or not.

To understand logistic regression, let's go over the odds of success.

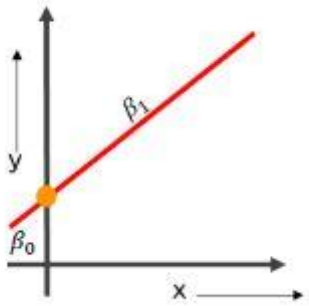
Odds (θ) = Probability of an event happening / Probability of an event not happening

θ = p / 1 - p

The values of odds range from zero to ∞ and the values of probability lies between zero and one.

Consider the equation of a straight line:

$y = \beta 0 + \beta 1 * x$



Here, $\beta 0$ is the y-intercept

$\beta 1$ is the slope of the line

x is the value of the x coordinate

y is the value of the prediction

Now to predict the odds of success, we use the following formula:

$$\log\left(\frac{p(x)}{1-P(x)}\right) = \beta_0 + \beta_1 x$$

Exponentiating both the sides, we have:

$$e^{ln}\left(\frac{p(x)}{1-p(x)}\right) = e^{\beta_0+\beta_1 x}$$

$$\left(\frac{p(x)}{1-p(x)}\right) = e^{\beta_0+\beta_1 x}$$

Let Y = e $\beta 0+\beta 1 * x$

Then p(x) / 1 - p(x) = Y

p(x) = Y(1 - p(x))

p(x) = Y - Y(p(x))

p(x) + Y(p(x)) = Y

p(x)(1+Y) = Y

p(x) = Y / 1+Y

$$p(x) = \frac{e^{\beta_0+\beta_1 x}}{1 + e^{\beta_0+\beta_1 x}}$$

The equation of the sigmoid function is:

$$p(x) = \frac{1}{1 + e^{-(\beta_0+\beta_1 x)}}$$

The sigmoid curve obtained from the above equation is as follows:

Now that you know more about logistic regression algorithms, let's look at the difference between linear regression and logistic regression.

Advantages of the Logistic Regression Algorithm

- Logistic regression performs better when the data is linearly separable
- It does not require too many computational resources as it's highly interpretable
- There is no problem scaling the input features—It does not require tuning
- It is easy to implement and train a model using logistic regression

- It gives a measure of how relevant a predictor (coefficient size) is, and its direction of association (positive or negative).

Code Implementation for Logistic Regression

Let's explore how logistic regression can be implemented for different scenarios. Depending on the nature of the target variable, logistic regression can be used for both binomial and multinomial classification problems.

- Binomial Logistic Regression

In a binomial logistic regression, the target variable has only two possible outcomes, such as "accepted" vs. "rejected," "approved" vs. "denied," or "positive" vs. "negative." A practical example is predicting whether a loan application will be approved based on various applicant features like income, credit score, and employment history.

To implement this, the necessary libraries are imported, and a dataset of loan applications is used to train the model. The data is split into training and testing sets to evaluate the model's performance. After training the logistic regression model on the training data, it predicts the outcomes for the test data. The accuracy of the model is then calculated to measure its effectiveness. This provides a clear understanding of how the model predicts loan approvals.

- Multinomial Logistic Regression

Multinomial logistic regression is used when the target variable can have three or more possible outcomes, and these outcomes are not in any specific order. For example, consider a situation where we are classifying diseases into three categories: "disease A," "disease B," and "disease C." Here, logistic regression can help predict the likelihood of each category.

In this example, the Digit Dataset is used to classify handwritten digits (0-9). Unlike binomial logistic regression, the data is split into training and testing sets. After training the model, predictions are made on the test data, and the model's accuracy is evaluated. In this case, the model achieved an impressive accuracy of 96.52%.

Python Implementation of Logistic Regression with Example

Here's how to use logistic regression in Python to predict SUV purchases based on user age and salary:

- Data Preparation

Start by loading your dataset, which includes user details like age and salary. From this dataset, you'll identify the columns representing the features (age and salary) and the target variable (whether the user will

purchase the SUV). The next step is to split this data into two parts: a training set to build the model and a test set to evaluate its performance. This separation ensures that the model is tested on new, unseen data.

- Training the Model

With your data prepared, you can now train the logistic regression model. Logistic regression is well-suited for binary outcomes, like predicting whether a user will buy the SUV (yes or no). During training, the model learns from the data by finding patterns in age and salary that help distinguish between users who are likely to purchase the SUV and those who are not.

- Making Predictions

Once the model is trained, it predicts outcomes on the test set. This means applying the model to new data to estimate the likelihood that each user will purchase the SUV. These predictions are then compared to actual outcomes to determine how accurate the model is.

- Evaluating Performance

To evaluate the model's accuracy, you'll create a confusion matrix. This matrix shows how many predictions were correct and how many were incorrect. It compares the predicted results with the purchase decisions, helping you understand how well the model performs.

- Visualizing Results

Finally, visualize the results to see how well the model has done. Plotting the data and predictions provides a clear picture of how the model separates users who are likely to buy the SUV from those who aren't. This visualization makes interpreting the model's performance and seeing its practical implications easier.

Evaluate Logistic Regression Model

Evaluating a logistic regression model involves several key metrics that assess its performance:

- Accuracy

Accuracy measures how often the model correctly classifies instances. It gives the proportion of correct predictions (both true positives and true negatives) out of all predictions made. High accuracy means the model is generally reliable at predicting the correct class.

- Precision

Precision focuses on the correctness of positive predictions. It tells you how many of the instances predicted as positive by the model are positive. This metric is especially important when the cost of false positives (incorrectly predicting a negative instance as positive) is high.

- Recall

Recall, also known as sensitivity or the true positive rate, evaluates how well the model identifies actual positive instances. It shows the proportion of actual positives that the model correctly predicts. High recall indicates that the model identifies positive cases, even if it means more false positives.

- F1 Score

The F1 Score is a combined measure of precision and recall, providing a single value that balances both metrics. It is useful when the trade-off between precision and recall must be balanced, especially in situations where both false positives and false negatives are important.

- Area Under the Receiver Operating Characteristic Curve (AUC-ROC)

The AUC-ROC assesses the model's ability to distinguish between positive and negative classes across various thresholds. It reflects how well the model performs overall in classifying different classes, with a higher score indicating better performance.

- Area Under the Precision-Recall Curve (AUC-PR)

The AUC-PR measures the model's precision and recall performance across different levels. It is handy for evaluating models on imbalanced datasets where one class is much more common than the other.

Precision-Recall Tradeoff in Logistic Regression Threshold Setting

Let's now look at how setting the decision threshold in logistic regression affects the balance between precision and recall.

- Low Precision/High Recall

When it's crucial to catch as many positive cases as possible, even if it means a higher number of false positives, a threshold that boosts recall is the way to go. For example, in medical tests for serious conditions like cancer, you want to identify every possible case, even if some healthy people are mistakenly diagnosed as having the disease. Using a lower threshold increases the chances of detecting all potential cases, which is vital for early intervention, despite the trade-off of some incorrect positive results.

- High Precision/Low Recall

On the other hand, if your priority is to avoid false positives and you can accept missing some true positives, you should choose a threshold that enhances precision. For instance, in targeted advertising, you want to be sure that those identified as likely to respond positively will actually do so. A higher threshold means that only those with a strong likelihood of responding are selected, reducing the risk of spending resources on people who are unlikely to engage, even though it might lead to missing some genuine positive responses.

Linear Regression vs. Logistic Regression

Linear Regression	Logistic Regression
Used to solve regression problems	Used to solve classification problems
The response variables are continuous in nature	The response variable is categorical in nature
It helps estimate the dependent variable when there is a change in the independent variable	It helps to calculate the possibility of a particular event taking place
It is a straight line	It is an S-curve (S = Sigmoid)

Logistic Regression Best Practices

Here are the key best practices for ensuring that logistic regression models are accurate and effective:

- Identify Dependent Variables to Ensure Model Consistency

In logistic regression, it's essential to have an inherently binary dependent variable. This means it should naturally fall into one of two categories, such as "yes/no," "disease/no disease," or "successful/unsuccessful."

For example, in healthcare research, outcomes like "cancerous/non-cancerous" fit well with logistic regression. However, it's crucial to avoid turning continuous variables (like income) into binary categories (e.g., "rich" versus "poor") without a strong justification. Doing so can lead to a significant loss of information and reduce the model's effectiveness, as this recoding oversimplifies complex data and might obscure important nuances.

- Discover the Technical Requirements of the Model

Logistic regression needs careful attention to several technical factors to work properly and efficiently:

- *Increase the Number of Observations*

More data generally leads to more reliable and stable estimates. Larger sample sizes help reduce the impact of multicollinearity (when independent variables are highly correlated), which can skew results.

- *Use Data Reduction Techniques*

Techniques like principal component analysis (PCA) can help manage multicollinearity by combining correlated variables into fewer synthetic measures. This reduces redundancy and improves model performance.

- *Monitor Sample Size*

Small sample sizes can lead to inaccurate and unstable estimates. Ensuring a sufficiently large sample size is crucial for obtaining reliable results.

- *Exclude Extreme Outliers*

Outliers can disproportionately affect the model's coefficients. Identifying and removing these outliers helps create a model that better represents most of the data and improves overall fit.

- Estimate the Model and Evaluate the Goodness of Fit

Accurate model estimation involves several key steps that ensure the robustness and reliability of the logistic regression model:

- *Transparency*

Report all relevant details about the model, including the software and data used. Access to the original data and computational scripts enhances replicability and allows others to verify the results.

- *Goodness-of-Fit Evaluation*

Assess how well the model fits the data by comparing it to a null model, which includes only the intercept and no predictors. This comparison helps determine if the logistic regression model significantly improves predictions over a simple baseline model. A model with a better fit will generally provide more accurate predictions and insights.

- Appropriately Interpret the Results

Interpreting logistic regression results requires understanding the coefficients expressed in terms of odds ratios rather than raw values. Here's how to interpret them:

- *Odds Ratios*

A coefficient represents how changes in an independent variable affect the odds of the dependent variable occurring. For example, a coefficient of 0.4 suggests that a one-unit increase in the independent variable corresponds to an increase of 0.4 in the log odds of the dependent variable.

- *Contextual Explanation*

Unlike linear regression, where coefficients directly show the impact on the dependent variable, logistic regression coefficients must be explained in terms of odds. This means understanding and describing how each predictor influences the likelihood of the outcome occurring.

- Validate Observed Results

Validation is crucial to ensure the model's findings are reliable and applicable. Here is how it can be effectively carried out:

- *Use a Subsample*

Test the model on a subsample of the original dataset to assess its performance. This helps confirm that the model's predictions are specific to the training data and can be generalized to other datasets.

- *External Validity*

This practice assesses whether the results can be applied to other populations or settings. It ensures that the model's findings are robust and not limited to the particular sample used for training. Validating with different data helps confirm that the model's predictions are accurate and generalizable.

Applications of Logistic Regression

Let's explore some of the most common applications of logistic regression across different industries:

- Optical Character Recognition (OCR)

Optical Character Recognition (OCR) is a process that converts handwritten or printed characters into digital text, making it readable by computers. Since OCR involves identifying specific characters from possible outcomes, it qualifies as a classification task in machine learning.

Logistic regression machine learning is instrumental in this context, where it helps classify characters as present or absent in an image. Features such as lines, curves, and edges extracted from the image serve as input variables, while logistic regression estimates the likelihood of character presence. By applying the model to new images, accurate character recognition becomes possible.

- Fraud Detection

Fraud detection identifies and prevents deceptive activities, particularly in finance, insurance, and e-commerce. Logistic regression is a powerful tool for detecting fraudulent transactions by classifying them as either legitimate or fraudulent. The model uses independent variables such as transaction value, location, time, and user information to predict the likelihood of fraud. Organizations can significantly improve their ability to spot and stop fraudulent activities by combining logistic regression with other methods like anomaly detection.

- Disease Spread Prediction

Predicting the spread of diseases can also be approached as a classification problem, where the goal is to determine whether an individual is likely to contract a disease. Logistic regression helps model the relationship between population demographics, health conditions, environmental factors, medical resource availability, and the probability of disease transmission. Using historical data, logistic regression can predict disease spread patterns, helping public health officials respond more effectively. Combining logistic

regression with time series analysis and clustering techniques can further enhance the accuracy of predictions.

- Illness Mortality Prediction

In healthcare, logistic regression is often used to predict mortality in patients suffering from specific illnesses. The model is trained using data on patient demographics, health status, and clinical indicators such as age, gender, and vital signs. By analyzing these variables, logistic regression can estimate the probability of a patient dying from the illness. This enables medical professionals to make informed decisions about patient care, improving outcomes and resource allocation.

- Churn Prediction

Churn prediction identifies customers likely to stop using a product or service. Logistic regression models customer churn by analyzing demographic information, usage patterns, and behavior. The model assigns a probability to each customer's likelihood of churning, enabling businesses to take proactive measures to retain them. Interventions such as targeted marketing campaigns, personalized offers, and enhanced customer support can be deployed based on these predictions, helping reduce churn rates and improve customer loyalty.

Generalized Linear Models (GLMs) are a flexible generalization of ordinary linear regression models in machine learning and statistics. They are used for modeling relationships between a response variable (dependent variable) and one or more predictor variables (independent variables). GLMs extend linear models by allowing for response variables that have error distributions other than a normal distribution.

Here's an overview of key concepts in GLMs and their application in machine learning:

Key Components of GLMs:

1. **Linear Predictor**:
   A linear combination of input features. The GLM starts by assuming that the target variable can be modeled as a linear function of the input features:

   $\eta = X\beta$

   where X is the matrix of input features, β is a vector of model coefficients, and η\eta is the linear predictor.

2. **Link Function**:
   The link function $g(\cdot)$ transforms the linear predictor η\eta to map it to the mean of the response variable, ensuring that predictions stay within valid ranges. The link function connects the linear predictor to the expected value of the response:

   $g(E(y)) = \eta$

Common link functions include:

- **Identity link**: Used in linear regression (no transformation).
- **Logit link**: Used in logistic regression, useful for binary outcomes.
- **Log link**: Used in Poisson regression, useful for count data.

3. **Response Distribution**:
   Unlike linear regression, which assumes that the target variable follows a normal distribution, GLMs allow for various types of distributions in the exponential family. Common choices include:
   - **Normal distribution**: For continuous data (standard linear regression).
   - **Binomial distribution**: For binary or categorical outcomes (logistic regression).
   - **Poisson distribution**: For count data.

Common Types of GLMs:

1. **Linear Regression** (Normal distribution, Identity link):
   - Predicts a continuous outcome using a linear relationship with input features.
2. **Logistic Regression** (Binomial distribution, Logit link):
   - Predicts binary or categorical outcomes (0 or 1, or multiple classes with extensions like softmax).
3. **Poisson Regression** (Poisson distribution, Log link):
   - Used for modeling count data where the outcome is non-negative integers.
4. **Gamma Regression** (Gamma distribution, Log link):
   - Used for modeling skewed continuous data, often in survival analysis or insurance.

GLMs in Machine Learning:

In machine learning, GLMs are often used as interpretable models that serve as the foundation for more complex algorithms. They offer a straightforward way to understand the relationship between features and the response variable. While more flexible models like decision trees, random forests, or neural networks can often achieve better predictive performance, GLMs are valued for their simplicity and interpretability.

Some typical applications of GLMs in machine learning include:

- **Classification**: Logistic regression is a widely used algorithm for binary classification tasks.
- **Regression analysis**: Linear regression is a core technique for modeling relationships in continuous data.
- **Survival analysis**: GLMs with the appropriate distribution and link functions (e.g., gamma distribution) are used for time-to-event modeling.

Advantages:

- **Interpretability**: The coefficients in a GLM model are easy to interpret.
- **Flexibility**: By using different link functions and distributions, GLMs can model a wide variety of data types.
- **Efficiency**: They are relatively easy to train and computationally efficient compared to more complex models.

Limitations:

- **Assumptions**: GLMs assume a specific form for the error distribution and linear relationships, which may not hold for all data.
- **Limited Non-linearity**: Although flexible, GLMs may not capture complex non-linear patterns in data like neural networks or tree-based models can.

Extensions of GLMs:

- **Generalized Additive Models (GAMs)**: GAMs allow for non-linear relationships by applying smoothing functions to each predictor while still maintaining the additive structure of a GLM.
- **Regularized GLMs**: Lasso (L1 regularization) and Ridge (L2 regularization) are common techniques used to extend GLMs, preventing overfitting and improving generalization.

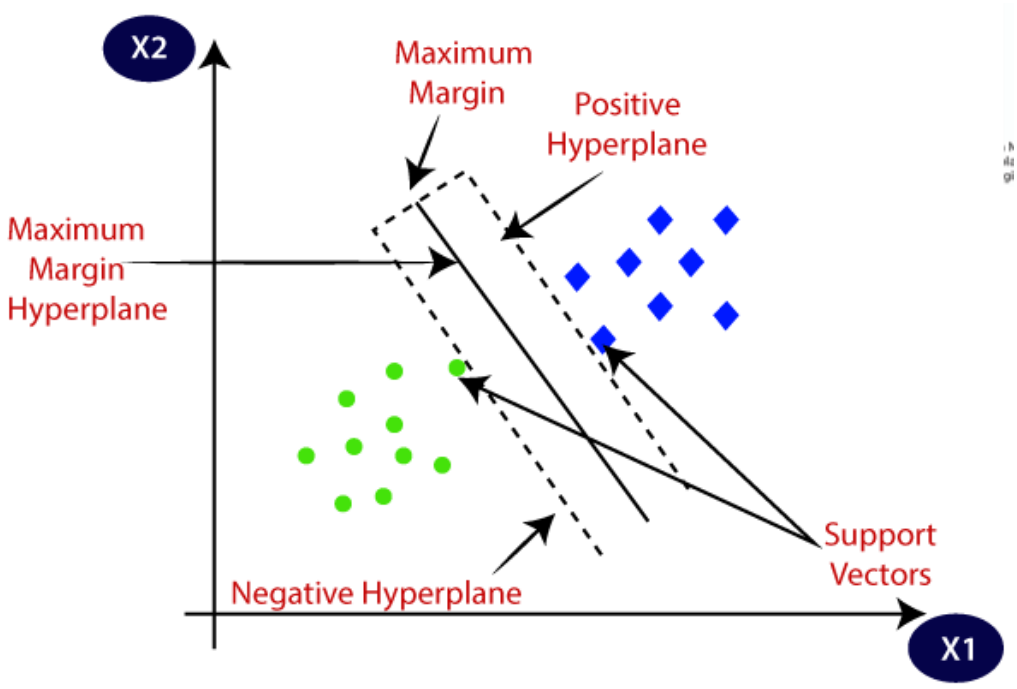
GLMs provide a solid, interpretable starting point in many machine learning tasks, particularly in regression and classification problems.

Support Vector Machine Algorithm

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



Support Vector Machine Terminology

1. **Hyperplane:** Hyperplane is the decision boundary that is used to separate the data points of different classes in a feature space. In the case of linear classifications, it will be a linear equation i.e. wx+b = 0.
2. **Support Vectors:** Support vectors are the closest data points to the hyperplane, which makes a critical role in deciding the hyperplane and margin.

3. **Margin**: Margin is the distance between the support vector and hyperplane. The main objective of the support vector machine algorithm is to maximize the margin. The wider margin indicates better classification performance.
4. **Kernel**: Kernel is the mathematical function, which is used in SVM to map the original input data points into high-dimensional feature spaces, so, that the hyperplane can be easily found out even if the data points are not linearly separable in the original input space. Some of the common kernel functions are linear, polynomial, radial basis function(RBF), and sigmoid.
5. **Hard Margin:** The maximum-margin hyperplane or the hard margin hyperplane is a hyperplane that properly separates the data points of different categories without any misclassifications.
6. **Soft Margin:** When the data is not perfectly separable or contains outliers, SVM permits a soft margin technique. Each data point has a slack variable introduced by the soft-margin SVM formulation, which softens the strict margin requirement and permits certain misclassifications or violations. It discovers a compromise between increasing the margin and reducing violations.
7. **C:** Margin maximisation and misclassification fines are balanced by the regularisation parameter C in SVM. The penalty for going over the margin or misclassifying data items is decided by it. A stricter penalty is imposed with a greater value of C, which results in a smaller margin and perhaps fewer misclassifications.
8. **Hinge Loss:** A typical loss function in SVMs is hinge loss. It punishes incorrect classifications or margin violations. The objective function in SVM is frequently formed by combining it with the regularisation term.

SVM algorithm can be used for **Face detection, image classification, text categorization,** etc.

Types of SVM

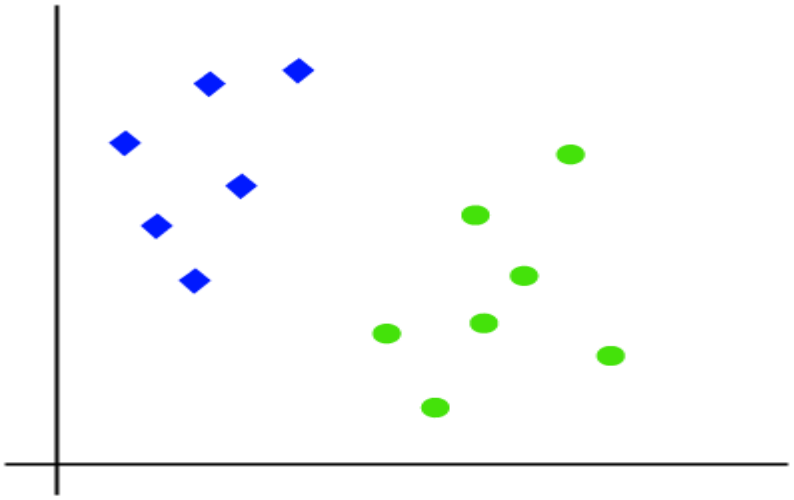
SVM can be of two types:

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

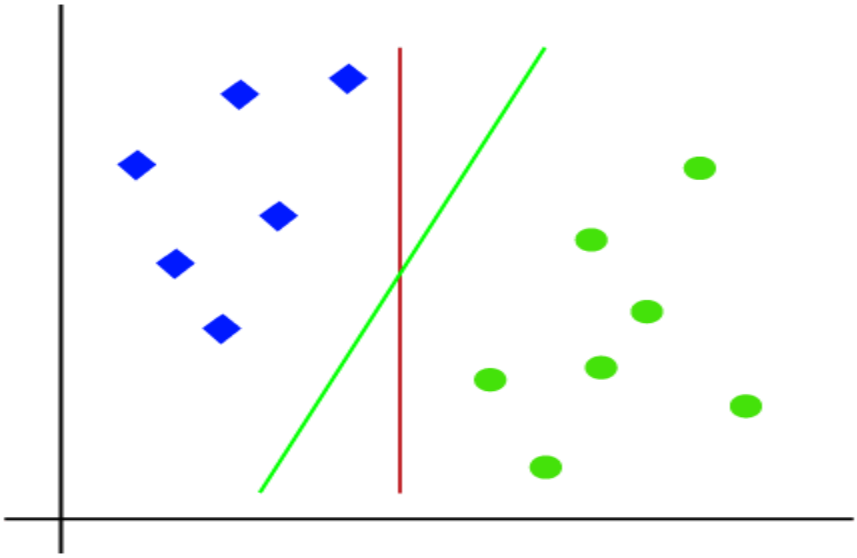
How does SVM works?

Linear SVM:

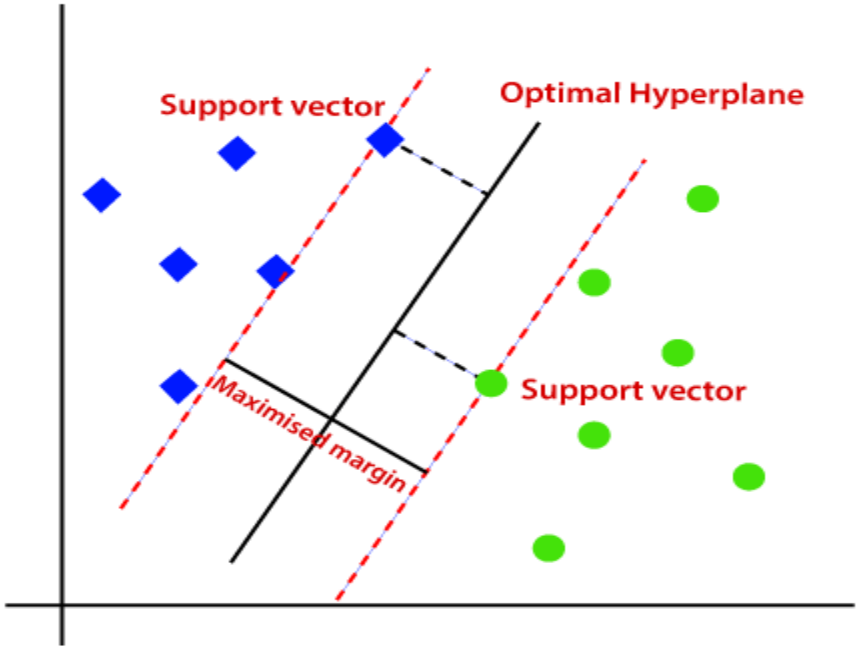
The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the below image:



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

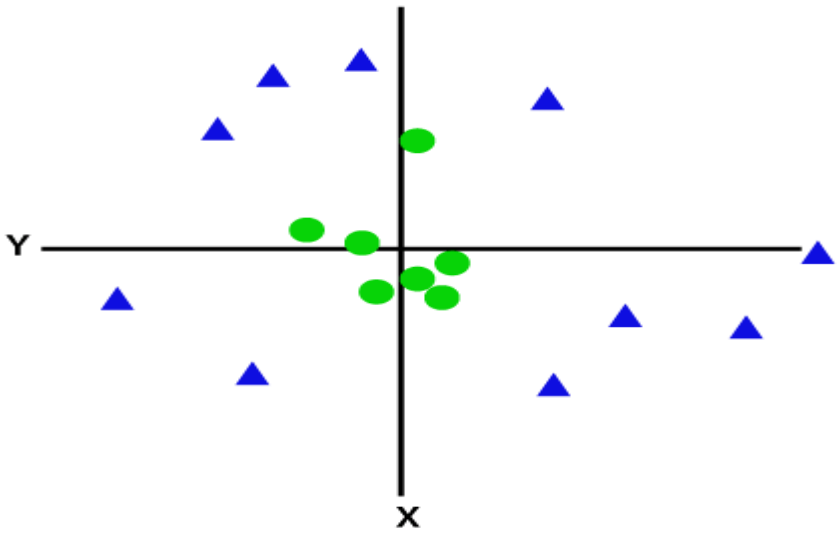


Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane 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:

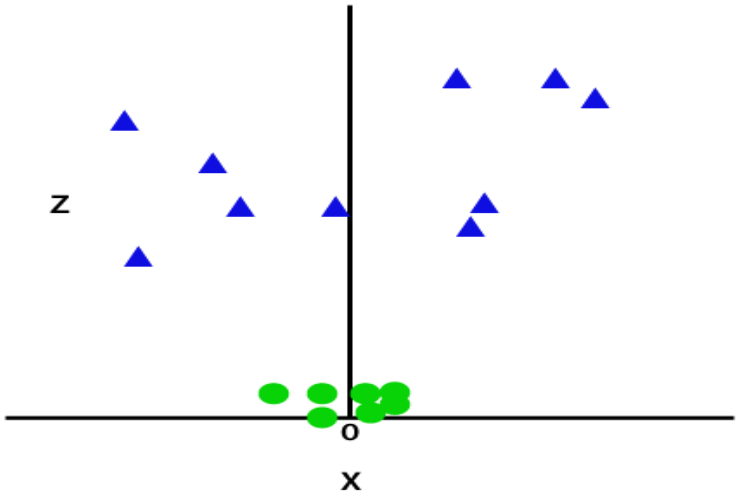
If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:



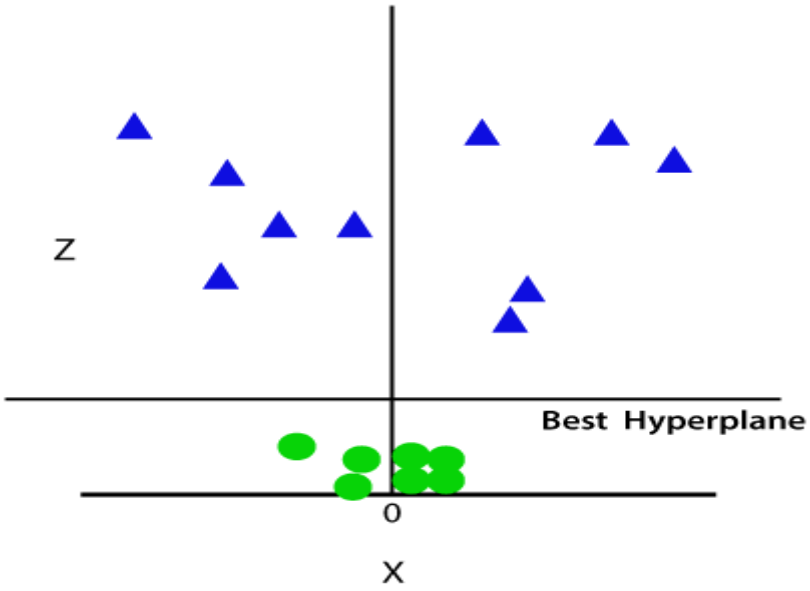
So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as:

$$z=x^2+y^2$$

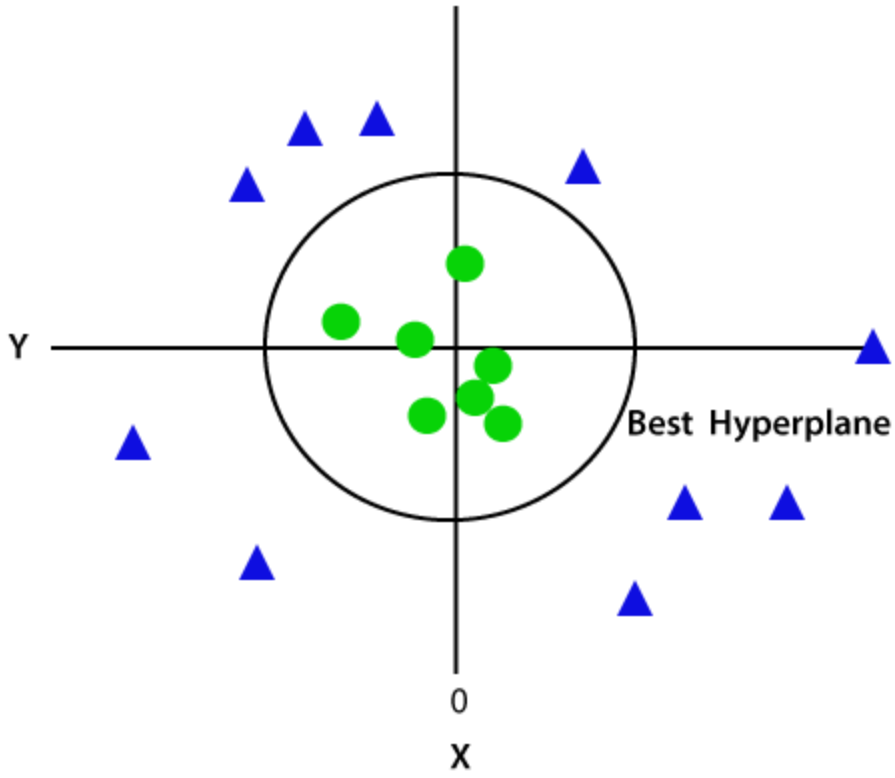
By adding the third dimension, the sample space will become as below image:



So now, SVM will divide the datasets into classes in the following way. Consider the below image:



Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with z=1, then it will become as:



Advantages of SVM

- Effective in high-dimensional cases.
- Its memory is efficient as it uses a subset of training points in the decision function called support vectors.
- Different kernel functions can be specified for the decision functions and its possible to specify custom kernels.

Machine Learning's Non-Linearity

More intricate correlations between input feature and output can be captured by non-linear models. These models can represent complex patterns using non-linear transformations or activation functions and do not presuppose a straight-line relationship.

Non-Linear Model Examples

1. Decision Trees: These models provide a tree-like structure that can represent non-linear relationships by dividing data into branches based on feature values.

2. Gradient Boosting Machines (GBMs) and Random Forests are two ensemble techniques that integrate several decision trees to improve prediction performance.

3. Non-Linear Kernels in Support Vector Machines (SVMs): SVMs can project data into higher-dimensional spaces, where a linear separator can be identified, by using kernel functions.

4. Neural Networks: Capable of modeling extremely complicated interactions, these networks are made up of layers of interconnected nodes, or neurons, with non-linear activation functions.

Benefits of Non-Linear Modeling

- Capability to Recap Complex Patterns: Non-linear models are able to manage complex feature interactions and linkages.

- Flexibility: They are appropriate for a range of applications since they can adjust to a large variety of data patterns.

The Drawbacks Non-Linear Models

- Complexity and Interpretability: Compared to linear models, non-linear models are frequently more complex and challenging to understand.

- Computationally Intense: They demand greater processing power and more time for training.

- Risk of Overfitting: Non-linear models are more likely to overfitting because of their flexibility, particularly when there is a lack of data.

Selecting Linear versus Non-Linear Models

Using a linear or non-linear model is determined by a number of factors, including:

- Nature of the Data: A linear model might be adequate if there is a roughly linear relationship between the inputs and the goal variable. Non-linear models are better suited for more intricate interactions.

- Conditions for Interpretability: Linear models are favored if interpretability of the model is important. Non-linear models could be preferable in situations when interpretability is less important and performance is more important.

- Computing Capabilities: Computationally, linear models are less demanding. Linear models could be more practical for real-time applications or huge datasets.

- Danger of Overfitting Because they are less likely to overfit, linear models are safer when there is less data. To prevent, non-linear models need to be carefully regularized and validated overfitting.

In conclusion

Comprehending linearity and non-linearity is essential for machine learning. The efficiency, interpretability, and simplicity of linear models make them useful in a wide range of applications. But they might not be able to capture the complexity of real-world data sufficiently. Non-linear models provide the flexibility needed to depict complex relationships, although using more resources and being more complex. Selecting between linear and non-linear models should take into account the specific requirements and constraints of the work. Experts may develop machine learning models that perform better and draw more insightful conclusions by comprehending these concepts.

Major Kernel Function in Support Vector Machine

Kernel Function is a method used to take data as input and transform it into the required form of processing data. “Kernel” is used due to a set of mathematical functions used in Support Vector Machine providing the window to manipulate the data. So, Kernel Function generally transforms the training set of data so that a non-linear decision surface is able to transform to a linear equation in a higher number of dimension spaces. Basically, It returns the inner product between two points in a standard feature dimension.

In Support Vector Machines (SVMs), there are several types of kernel functions that can be used to map the input data into a higher-dimensional feature space. The choice of kernel function depends on the specific problem and the characteristics of the data.

Here are some most commonly used kernel functions in SVMs:

Linear Kernel

A linear kernel is a type of kernel function used in machine learning, including in SVMs (Support Vector Machines). It is the simplest and most commonly used kernel function, and it defines the dot product between the input vectors in the original feature space.

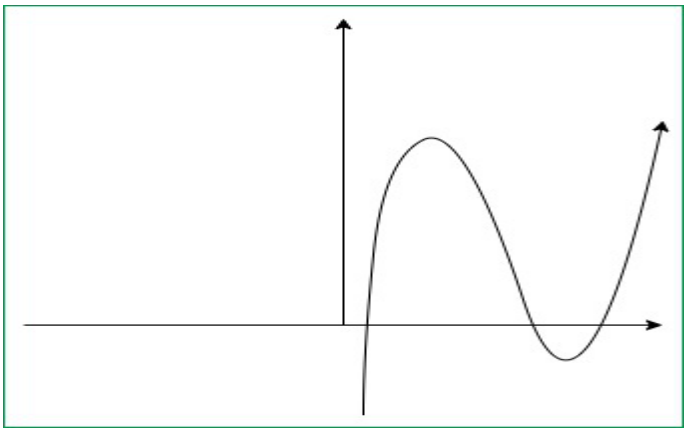
The linear kernel can be defined as:

1. K(x, y) = x .y
   Where x and y are the input feature vectors. The dot product of the input vectors is a measure of their similarity or distance in the original feature space.

When using a linear kernel in an SVM, the decision boundary is a linear hyperplane that separates the different classes in the feature space. This linear boundary can be useful when the data is already separable by a linear decision boundary or when dealing with high-dimensional data, where the use of more complex kernel functions may lead to overfitting.

Polynomial Kernel

A particular kind of kernel function utilised in machine learning, such as in SVMs, is a polynomial kernel (Support Vector Machines). It is a nonlinear kernel function that employs polynomial functions to transfer the input data into a higher-dimensional feature space.



One definition of the polynomial kernel is:

Where x and y are the input feature vectors, c is a constant term, and d is the degree of the polynomial, K(x, y) = (x. y + c)d. The constant term is added to, and the dot product of the input vectors elevated to the degree of the polynomial.

The decision boundary of an SVM with a polynomial kernel might capture more intricate correlations between the input characteristics because it is a nonlinear hyperplane.

The degree of nonlinearity in the decision boundary is determined by the degree of the polynomial.

The polynomial kernel has the benefit of being able to detect both linear and nonlinear correlations in the data. It can be difficult to select the proper degree of the polynomial, though, as a larger degree can result in overfitting while a lower degree cannot adequately represent the underlying relationships in the data.

In general, the polynomial kernel is an effective tool for converting the input data into a higher-dimensional feature space in order to capture nonlinear correlations between the input characteristics.

Gaussian (RBF) Kernel

It is used to perform transformation when there is no prior knowledge about data.

The Gaussian kernel, also known as the radial basis function (RBF) kernel, is a popular kernel function used in machine learning, particularly in SVMs (Support Vector Machines). It is a nonlinear kernel function that maps the input data into a higher-dimensional feature space using a Gaussian function.

The Gaussian kernel can be defined as:

1. K(x, y) = exp(-gamma * ||x - y||^2)
   Where x and y are the input feature vectors, gamma is a parameter that controls the width of the Gaussian function, and ||x - y||^2 is the squared Euclidean distance between the input vectors.

When using a Gaussian kernel in an SVM, the decision boundary is a nonlinear hyper plane that can capture complex nonlinear relationships between the input features. The width of the Gaussian function, controlled by the gamma parameter, determines the degree of nonlinearity in the decision boundary.

One advantage of the Gaussian kernel is its ability to capture complex relationships in the data without the need for explicit feature engineering. However, the choice of the gamma parameter can be challenging, as a smaller value may result in under fitting, while a larger value may result in over fitting.

Laplace Kernel

The Laplacian kernel, also known as the Laplace kernel or the exponential kernel, is a type of kernel function used in machine learning, including in SVMs (Support Vector Machines). It is a non-parametric kernel that can be used to measure the similarity or distance between two input feature vectors.

The Laplacian kernel can be defined as:

1. K(x, y) = exp(-gamma * ||x - y||)
   Where x and y are the input feature vectors, gamma is a parameter that controls the width of the Laplacian function, and ||x - y|| is the L1 norm or Manhattan distance between the input vectors.

When using a Laplacian kernel in an SVM, the decision boundary is a nonlinear hyperplane that can capture complex relationships between the input features. The width of the Laplacian function, controlled by the gamma parameter, determines the degree of nonlinearity in the decision boundary.

What is Machine Learning?

Machine learning is the science of teaching and educating the computer i.e. a machine to behave and act like a human and improve itself over time. This is done by feeding the machine with data and information in the form of real-world interactions, it can be done through coding and feeding the machine with the desired data.

Through Machine learning algorithms, the device learns from the data provided and acts accordingly in the situation provided. It is basically a part of artificial intelligence that provides computers the ability to learn through data and observations.

Supervised Machine Learning

Supervised machine learning is a type of machine learning where a specifically known dataset is provided to make predictions. In the dataset, there are two types of variables, input variable(X), output variable(Y).

In this, a supervised learning algorithm builds a model where the response variable is used over the known dataset, to check the accuracy of the model.

As a part of supervised machine learning, classification has achieved a speculations rise.

Definition of Classification

In machine learning, Classification, as the name suggests, classifies data into different parts/classes/groups. It is used to predict from which dataset the input data belongs to.

For example, if we are taking a dataset of scores of a cricketer in the past few matches, along with average, strike rate, not outs etc, we can classify him as “in form” or “out of form”.

Classification is the process of assigning new input variables (X) to the class they most likely belong to, based on a classification model, as constructed from previously labeled training data.

Data with labels is used to train a classifier such that it can perform well on data without labels (not yet labeled). This process of continuous classification, of previously known classes, trains a machine. If the classes are discrete, it can be difficult to perform classification tasks.

Types of Classification

There are two types of classifications;

Binary classification

- Multi-class classification

Binary Classification

It is a process or task of classification, in which a given data is being classified into two classes. It’s basically a kind of prediction about which of two groups the thing belongs to.

Let us suppose, two emails are sent to you, one is sent by an insurance company that keeps sending their ads, and the other is from your bank regarding your credit card bill. The email service provider will classify the two emails, the first one will be sent to the spam folder and the second one will be kept in the primary one.

This process is known as binary classification, as there are two discrete classes, one is spam and the other is primary. So, this is a problem of binary classification.

Binary classification uses some algorithms to do the task, some of the most common algorithms used by binary classification are .

- **Logistic Regression**
- **k-Nearest Neighbors**
- **Decision Trees**
- **Support Vector Machine**
- **Naive Bayes**

Term Related to binary classification

1. ***PRECISION***

Precision in binary classification (Yes/No) refers to a model's ability to correctly interpret positive observations. In other words, how often does a positive value forecast turn out to be correct? We may manipulate this metric by only returning positive for the single observation in which we have the most confidence.

2. ***RECALL***

The recall is also known as sensitivity. In binary classification (Yes/No) recall is used to measure how “sensitive” the classifier is to detecting positive cases. To put it another way, how many real findings did we “catch” in our sample? We may manipulate this metric by classifying both results as positive.

3. ***F1 SCORE***

The F1 score can be thought of as a weighted average of precision and recall, with the best value being 1 and the worst being 0. Precision and recall also make an equal contribution to the F1 ranking.

Multiclass Classification

Multi-class classification is the task of classifying elements into different classes. Unlike binary, it doesn't restrict itself to any number of classes.

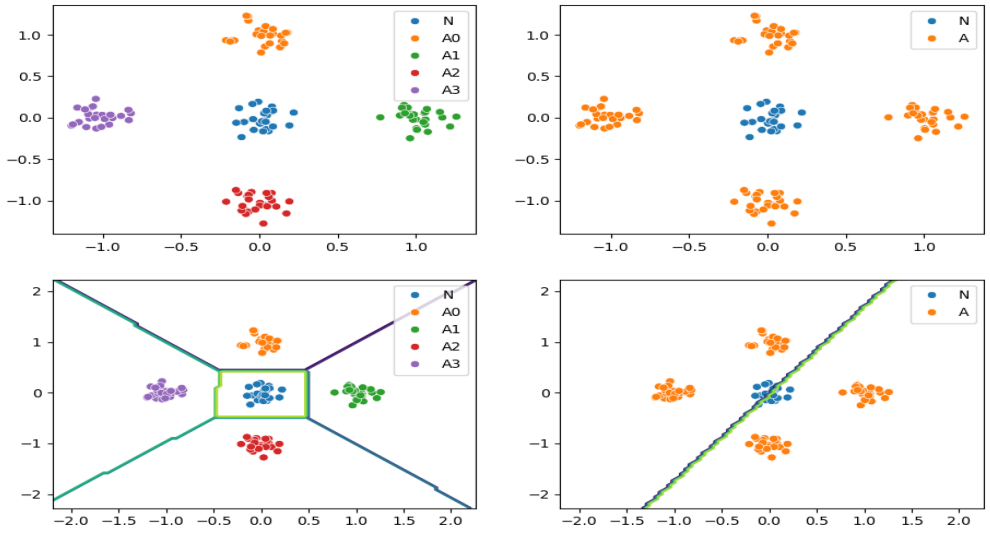
Examples of multi-class classification are

- classification of news in different categories,
- classifying books according to the subject,
- classifying students according to their streams etc.

In these, there are different classes for the response variable to be classified in and thus according to the name, it is a Multi-class classification.

Can a classification possess both binary or multi-class?

Let us suppose we have to do sentiment analysis of a person, if the classes are just "positive" and "negative", then it will be a problem of binary class. But if the classes are "sadness", happiness", "disgusting", "depressed", then it will be called a problem of Multi-class classification.



The multiclass model can find a good decision boundary, whereas the binary classifier is completely lost as the data cannot be separated linearly. I suppose this was what you were thinking.

This proves that multiclass classifier can sometimes work better than binary classifier. I don't think this is in general true though. Couple of things to consider come to mind:

- The multiclass model is more complex. If the anomalies could be separated e.g. by linear decision boundary, the multiclass approach would add unnecessary complexity and make the model prone to overfitting.
- Related to this, maybe some of the anomaly groups contain only couple of samples. In this case some of the decision boundaries would be very poorly estimated by the multiclass classifier.
- Instead of using a multiclass classifier, you can just use a more complex model and solve the binary classification problem directly.

The only way to know which approach will work best for your particular problem is usually just to test different models. The idea of first fitting multiclass model and then reducing the output to binary might work, or not. If nothing else, the output contains more information; maybe someone cares which type of anomaly it is.

Binary vs Multiclass Classification

Parameters	Binary classification	Multi-class classification
No. of classes	It is a classification of two groups, i.e. classifies objects in at most two classes.	There can be any number of classes in it, i.e., classifies the object into more than two classes.
Algorithms used	The most popular algorithms used by the binary classification are-	Popular algorithms that can be used for multi-class classification include: k-Nearest Neighbors

	Logistic Regression k-Nearest Neighbors Decision Trees Support Vector Machine Naive Bayes	Decision Trees Naive Bayes Random Forest. Gradient Boosting
Examples	Examples of binary classification include- Email spam detection (spam or not). Churn prediction (churn or not). Conversion prediction (buy or not).	Examples of multi-class classification include: Face classification. Plant species classification. Optical character recognition.

Ranking Algorithms & Types: Concepts & Examples

Ranking algorithms are used to rank items in a dataset according to some criterion. Ranking algorithms can be divided into two categories: deterministic and probabilistic. Ranking algorithms are used in search engines to rank webpages according to their relevance to a user's search query. In this article, we will discuss the different types of ranking algorithms

What is a Ranking Algorithm?

A ranking algorithm is a procedure that ranks items in a dataset according to some criterion. Ranking algorithms are used in many different applications, such as web search, recommender systems, and machine learning.
A ranking algorithm is a procedure used to rank items in a dataset according to some criterion. Ranking algorithms can be divided into two categories: deterministic and probabilistic.

- **Deterministic ranking algorithms**: A deterministic ranking algorithm is one in which the order of the items in the ranked list is fixed and does not change, regardless of the input data. An example of a deterministic ranking algorithm is the rank-by-feature algorithm. In this algorithm, each item is assigned a rank based on its feature value. The item with the highest feature value is assigned a rank of 1, and the item with the lowest feature value is assigned a rank of N, where N is the number of items in the dataset. One real-world application of a deterministic ranking algorithm is the ordering of items in a grocery store. The items in a grocery store are usually organized by department, such as produce, meat, dairy, etc. Within each department, the items are usually organized alphabetically. This type of organization is an example of a deterministic ranking algorithm.
- **Sorting algorithms** are used in deterministic ranking algorithms to order the items in the ranked list. There are many different types of sorting algorithms, each with its own set of advantages and disadvantages. Some of the most common sorting algorithms are **insertion sort, merge sort, and quicksort**.
- **Probabilistic ranking algorithms**: In a probabilistic ranking algorithm, the order of the items in the ranked list may vary, depending on the input data. An example of a probabilistic ranking algorithm is the rank-by-confidence algorithm. In this algorithm, each item is assigned a rank based on its confidence value. The item with the highest confidence value is assigned a rank of 1, and the item with the lowest confidence value is assigned a rank of N, where N is the number of items in the dataset. Another example of a probabilistic ranking algorithm is the Bayesian spam filter. In this algorithm, each email is assigned a probability of being spam. The emails with the highest probabilities are ranked first, and the emails with the lowest probabilities are ranked last. Probabilistic ranking algorithms can be used in web search engines to rank webpages

according to their relevance to a user's search query. The ranking algorithm uses the input data, such as the number of links to the webpage from other websites and the number of times the keyword appears on the page, to calculate the page's relevance score. The higher the relevance score, the higher the page is ranked in the search results. The probabilistic ranking algorithms can as well be used in machine learning algorithms to rank items in a dataset according to their likelihood of being a positive example. The ranking algorithm uses the input data, such as the number of features that are common to both positive and negative examples, to calculate the item's relevance score. The higher the relevance score, the more likely it is that the item is a positive example. There are many different types of probabilistic ranking algorithms, each with its own set of advantages and disadvantages. Some common types of probabilistic ranking algorithms are:

- **Bayesian Ranking Algorithm**: A Bayesian ranking algorithm is a probabilistic ranking algorithm that uses a Bayesian network to calculate the item's relevance score. The Bayesian network is a graphical model that represents a set of random variables and their conditional dependencies. The Bayesian ranking algorithm uses the input data, such as the number of features that are common to both positive and negative examples, to calculate the item's relevance score. The higher the relevance score, the more likely it is that the item is a positive example.
- **Log-linear Model Ranking Algorithm**: A log-linear model ranking algorithm is a probabilistic ranking algorithm that uses a log-linear model to calculate the item's relevance score. The log-linear model is a mathematical model that describes the relationship between two or more variables in terms of a linear combination of the logarithms of the variables.

One of the most common applications of ranking algorithms is in search engines. Search engines use ranking algorithms to determine which webpages are most relevant to a user's search query. Ranking algorithms are also used in recommendation systems to recommend items that a user may be interested in. The following is a quick overview on ranking algorithm used by popular search engines:

- **Google Ranking Algorithm**: Google's ranking algorithm is a secret, but we know that it is a probabilistic ranking algorithm. Google uses a variety of factors to rank webpages, including the number of links to a page, the page's PageRank, and the relevance of the search query to the page. Google's PageRank algorithm is a probabilistic ranking algorithm that uses the number of links to a webpage as a measure of its importance. The higher the PageRank of a webpage, the more likely it is to be ranked higher in the search results.
- **Amazon Ranking Algorithm**: Amazon's ranking algorithm is also a probabilistic ranking algorithm. Amazon uses a variety of factors to rank items, including the number of reviews an item has, the average rating of an item, and the price of an item. Amazon's algorithm is designed to recommend items that are relevant to a user's search query and that are popular with other users.
- **Facebook Ranking Algorithm**: Facebook's ranking algorithm is a secret, but we know that it is a probabilistic ranking algorithm. Facebook uses a variety of factors to rank news stories, including the number of likes, shares, and comments a story has, the story's PageRank, and the relevance of the story to the user's News Feed. Facebook's algorithm is designed to show users the stories that are most relevant to them and that are being talked about by their friends.
- **Twitter Ranking Algorithm**: Twitter's ranking algorithm is also a probabilistic ranking algorithm. Twitter uses a variety of factors to rank tweets, including the number of retweets, favorites, and replies a tweet has, the tweeter's PageRank, and the relevance of the tweet to the user's timeline. Twitter's algorithm is designed to show users the tweets that are most relevant to them and that are being talked about by their friends.

Types of Ranking Algorithms

There are many different types of ranking algorithms, each with its own set of advantages and disadvantages. Some of the most common types of ranking algorithms are:

- **Binary Ranking Algorithms**: Binary ranking algorithms are the simplest type of ranking algorithm. A binary ranking algorithm ranks items in a dataset according to their relative importance. The two most common types of binary ranking algorithms are the rank-by-feature

and the rank-by-frequency algorithms. Rank-by-feature algorithms rank items by the number of features that they have in common with the reference item. The reference item is the item that is used to calculate the similarity value for each of the other items in the dataset. Rank-by-frequency algorithms rank items by the number of times that they occur in the dataset. Both rank-by-feature and rank-by-frequency algorithms have their own set of advantages and disadvantages. Rank-by-feature algorithms are more accurate than rank-by-frequency algorithms, but they are also more computationally expensive. Rank-by-frequency algorithms are faster than rank-by-feature algorithms, but they are less accurate.

- **Ranking by Similarity**: Ranking by similarity is a type of probabilistic ranking algorithm that ranks items in a dataset according to their similarity to a reference item. The reference item is the item that is used to calculate the similarity value for each of the other items in the dataset. The ranking algorithm uses the input data, such as the number of features that are common to both positive and negative examples, to calculate the item's relevance score. The higher the relevance score, the more similar the item is to the reference item. There are many different types of ranking by similarity algorithms, each with its own set of advantages and disadvantages. Some common types of ranking by similarity algorithms are clustering ranking algorithm, vector space ranking algorithm, etc.
- **Ranking by Distance**: Ranking by distance algorithms are a type of probabilistic ranking algorithm that rank items in a dataset according to their distance from a reference item. The reference item is the item that is used to calculate the distance value for each of the other items in the dataset. The ranking algorithm uses the input data, such as the number of features that are common to both positive and negative examples, to calculate the item's relevance score. The higher the relevance score, the more distant the item is from the reference item. There are many different types of ranking by distance algorithms, each with its own set of advantages and disadvantages. Some common types of ranking by distance algorithms are Euclidean distance algorithm, Mahalanobis distance algorithm, etc.
- **Ranking by Preference**: Preferential ranking algorithms are a type of probabilistic ranking algorithm that rank items in a dataset according to their preference for a reference item. The reference item is the item that is used to calculate the preference value for each of the other items in the dataset. The ranking algorithm uses the input data, such as the number of features that are common to both positive and negative examples, to calculate the item's relevance score. The higher the relevance score, the more preferred the item is for the reference item.
- **Ranking by Probability**: Ranking by probability is a type of probabilistic ranking algorithm that ranks items in a dataset according to their probability of being a positive example. The ranking algorithm uses the input data, such as the number of features that are common to both positive and negative examples, to calculate the item's relevance score. The higher the relevance score, the more likely the item is to be a positive example. Ranking by probability is different from other types of ranking algorithms because it takes into account the uncertainty of the data. This makes it more accurate than other types of ranking algorithms. There are many different types of ranking by probability algorithms, each with its own set of advantages and disadvantages. Some common types of ranking by probability algorithms are Bayesian Ranking Algorithm, AUC Ranking Algorithm, etc.

Conclusion

Ranking algorithms are used to rank items in a dataset according to some criterion. There are many different types of ranking algorithms, each with its own set of advantages and disadvantages. Ranking by similarity, distance, preference, and probability are the most common types of ranking algorithms. Ranking by probability is the most accurate type of ranking algorithm because it takes into account the uncertainty of the data. If you would like to learn more about ranking algorithms, please drop a comment below.