Parul® University

Machine Learning
SUBJECT CODE: 203105515

Dr. Amit Vajpayee, Associate Professor (PIET)
Computer Science & Engineering

CHAPTER-2

Supervised Learning

Contents

- □ Supervised Learning
- □ Linear and Non-Linear Examples
- □ Multi-Class & Multi-Label Classification
- □ Linear Regression
- □ Multi-linear Regression
- □ Naïve Bayes Classifier
- □ Decision Trees
- □ ID3
- □ CART
- □ Error Bounds

Supervised Learning

- Supervised learning is a form of machine learning in which the model is taught using a dataset that has been labeled. This implies that every training instance in the dataset contains both the input characteristics and the corresponding accurate output (label).
- Supervised learning is the types of machine learning in which machines are trained using well "labeled" training data, and on basis of that data, machines predict the output.
- The labeled data means some input data is already tagged with the correct output.
- The goal of supervised learning is to understand how inputs are related to outputs in order to make accurate predictions for new, unseen inputs.

Supervised Learning

- In supervised learning, the training data provided to the machines work as the supervisor that teaches the machines to predict the output correctly. It applies the same concept as a student learns in the supervision of the teacher.
- Supervised learning is a process of providing input data as well as correct output data to the machine learning model. The aim of a supervised learning algorithm is to **find a mapping function to map the input variable(x) with the output variable(y).**
- In the real-world, supervised learning can be used for Risk Assessment, Image classification, Fraud Detection, spam filtering, etc.

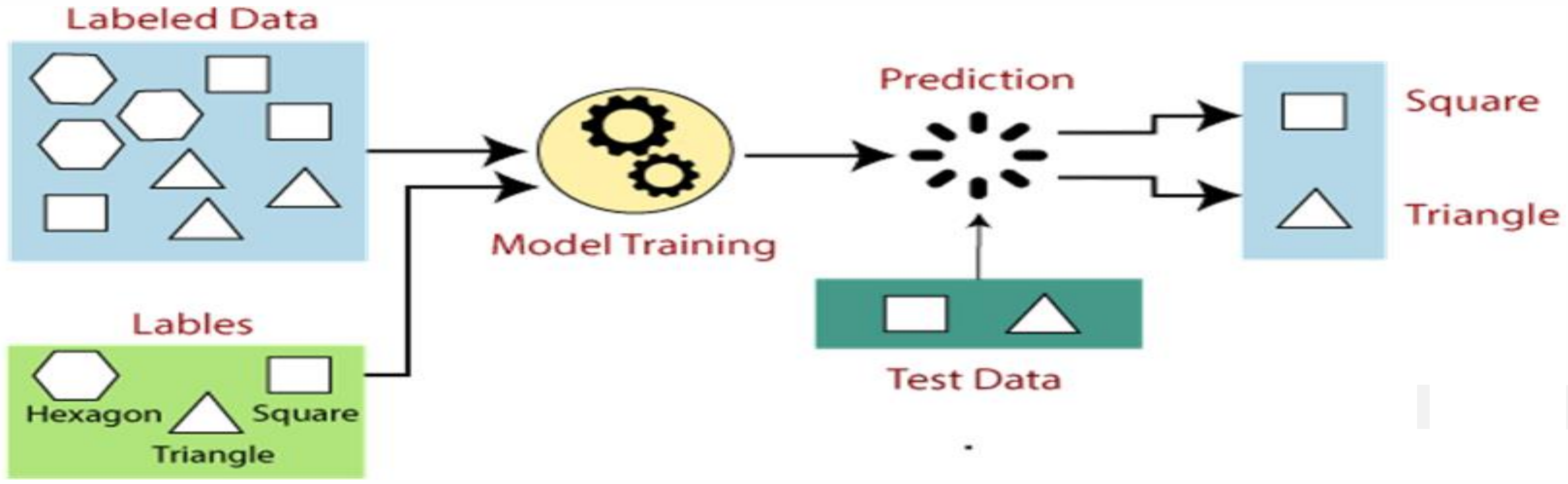
Key Concepts

- **Labeled Data:**
  - The dataset used in supervised learning consists of input-output pairs. The inputs are typically represented as feature vectors, and the outputs are the labels or target values.
- **Training Phase:**
  - During the training phase, the supervised learning algorithm analyzes the labeled data to learn patterns and relationships between the input features and the output labels. This learning process involves optimizing the model's parameters to minimize the error in predictions.
- **Prediction Phase:**
  - Once the model is trained, it can be used to predict the output for new, unseen input data. The model's performance is usually evaluated using a separate test set that was not seen during training.

How Supervised Learning Works?

In supervised learning, models are trained using labeled dataset, where the model learns about each type of data. Once the training process is completed, the model is tested on the basis of test data (a subset of the training set), and then it predicts the output.

The working of Supervised learning can be easily understood by the below example and diagram:



How Supervised Learning Works?

Suppose we have a dataset of different types of shapes which includes square, rectangle, triangle, and Polygon.

Now the first step is that we need to train the model for each shape.

- If the given shape has four sides, and all the sides are equal, then it will be labeled as a **Square**.
- If the given shape has three sides, then it will be labeled as a **triangle**.
- If the given shape has six equal sides then it will be labeled as **hexagon**.

Now, after training, we test our model using the test set, and the task of the model is to identify the shape.

The machine is already trained on all types of shapes, and when it finds a new shape, it classifies the shape on the bases of a number of sides, and predicts the output.

Steps in Supervised Learning

1. **Data Collection:**

   Gather a labeled dataset relevant to the problem you want to solve.

2. **Data Preprocessing:**

   Clean and preprocess the data, which may involve handling missing values, normalizing features, and encoding categorical variables.

3. **Train-Test Split:**

   Split the dataset into training and testing subsets. The training set is used to train the model, while the test set is used to evaluate its performance.

4. **Model Selection:**

   Choose an appropriate supervised learning algorithm based on the problem type (regression or classification) and the nature of the data.

Steps in Supervised Learning

5. **Training:**

   Train the model on the training data by feeding it the input-output pairs and allowing it to learn the mapping from inputs to outputs.

6. **Evaluation:**

   Evaluate the model's performance on the test set using relevant metrics, such as accuracy, precision, recall, F1 score (for classification), or mean squared error, R-squared (for regression).

7. **Hyper parameter Tuning:**

   Fine-tune the model's hyper parameters to improve its performance. This can be done using techniques like grid search or random search with cross-validation.

8. **Prediction:**

   Use the trained model to make predictions on new, unseen data.

Example

Consider a supervised learning task of predicting house prices based on features such as square footage, number of bedrooms, and location. The dataset consists of these features along with the corresponding house prices (labels).

1. **Data Collection:** Collect data on various houses, including their features and prices.
2. **Data Preprocessing:** Normalize the feature values and handle any missing data.
3. **Train-Test Split:** Split the data into a training set (e.g., 80%) and a test set (e.g., 20%).
4. **Model Selection:** Choose a regression algorithm, such as linear regression.

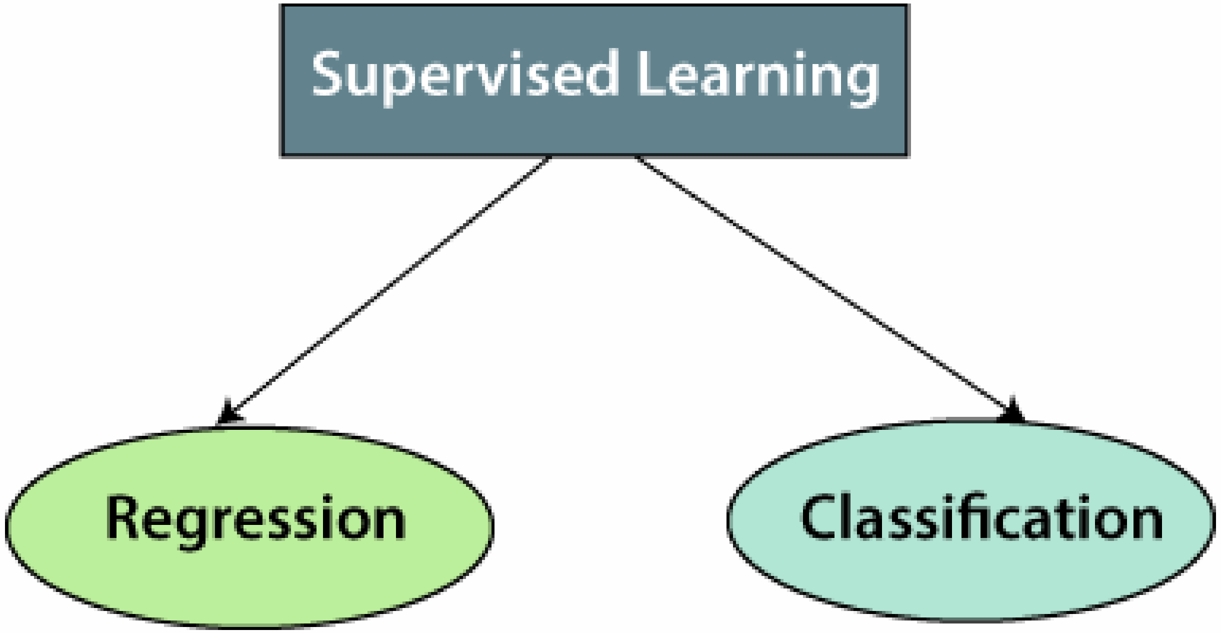
Example

5. **Training:** Train the linear regression model on the training data.
6. **Evaluation:** Evaluate the model's performance on the test set using metrics like mean squared error.
7. **Hyper parameter Tuning:** Adjust the model's hyperparameters if needed to improve performance.
8. **Prediction:** Use the trained model to predict house prices for new data.

Supervised learning is a fundamental and widely used approach in machine learning, applicable to a broad range of problems in various domains.

Types of supervised Machine learning Algorithms:

Supervised learning can be further classified into two main categories:



Regression

- Regression is used when the output variable is continuous. The goal is to predict a numerical value. Examples include predicting house prices, stock prices, or temperatures.
- **Common Algorithms:**
  - Linear Regression
  - Polynomial Regression
  - Support Vector Regression (SVR)
  - Decision Trees
  - Random Forests

Classification

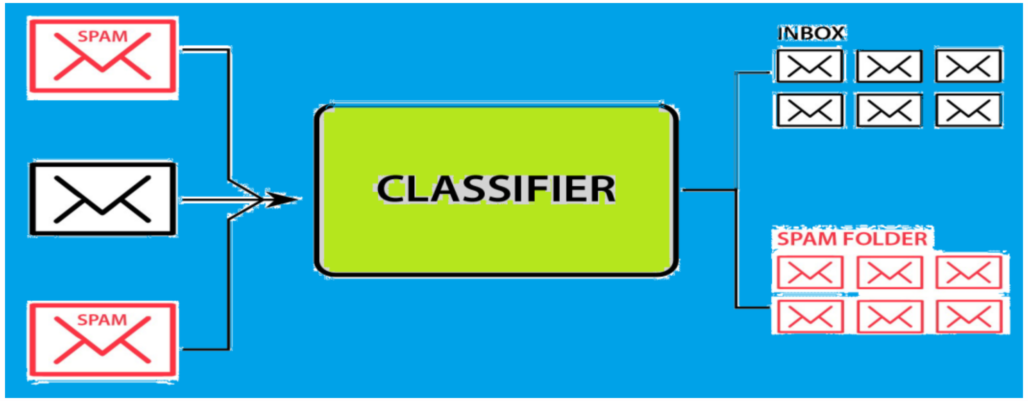
Classification is used when the output variable is categorical. The goal is to predict a class label. Examples include spam detection, image recognition, and medical diagnosis.

Common Algorithms:

- Logistic Regression
- k-Nearest Neighbors (k-NN)
- Support Vector Machines (SVM)
- Decision Trees
- Random Forests
- Neural Networks

What is Classification in Machine Learning?

- Classification is a supervised machine learning method where the model tries to predict the correct label of a given input data. In classification, the model is fully trained using the training data, and then it is evaluated on test data before being used to perform prediction on new unseen data.
- For instance, an algorithm can learn to predict whether a given email is spam or ham (no spam), as illustrated below.



Key Concepts

- **Supervised Learning**: Classification falls under supervised learning, meaning the model is trained on a labeled dataset, which includes input-output pairs. The model learns the mapping from input features to the output class labels.
- **Class Labels**: These are discrete values or categories that the model aims to predict. For example, in a spam detection system, emails can be classified into 'spam' or 'not spam'.
- **Features**: These are the input variables used to make predictions. Features can be numerical, categorical, or a mix of both. In the spam detection example, features might include the presence of certain keywords, email length, etc.
- **Training Phase**: During training, the model is provided with a dataset where the outcomes (class labels) are known. The model uses this data to learn patterns and relationships between the features and the class labels.
- **Prediction Phase**: Once trained, the model can predict the class labels of new, unseen data instances.

Example Use Cases

- **Spam Detection**: Classify emails as 'spam' or 'not spam'.
- **Medical Diagnosis**: Predict whether a patient has a particular disease based on their medical records.
- **Image Recognition**: Classify images into categories, such as identifying whether a picture contains a cat or a dog.
- **Sentiment Analysis**: Determine the sentiment of a piece of text, such as classifying a movie review as positive or negative.

Challenges in Classification

- **Class Imbalance**: When the number of instances in different classes is significantly imbalanced, it can bias the model towards the majority class.
- **Overfitting**: A model that performs well on training data but poorly on unseen data due to its complexity and memorization of the training data instead of learning general patterns.
- **Feature Engineering**: The process of selecting, modifying, and creating new features can be crucial and challenging for model performance.

Types of classification

Classification in machine learning can be broadly categorized into several types based on the nature of the target variable and the complexity of the task. Here are the main types of classification:

1. Binary Classification
2. Multi-Class Classification
3. Multi-Label Classification
4. Imbalanced Classification
5. Ordinal Classification
6. Hierarchical Classification
7. Single-Class Classification (One-Class Classification or Anomaly Detection)
8. Streaming Classification

Binary Classification

Definition: In binary classification, the target variable has only two possible classes or categories.

Examples:

- Spam detection (spam vs. not spam)
- Medical diagnosis (disease vs. no disease)
- Sentiment analysis (positive vs. negative)

Common Algorithms:

- Logistic Regression
- Support Vector Machines (SVM)
- Decision Trees
- Naive Bayes
- Neural Networks

Multi-Class Classification

Definition: In multi-class classification, the target variable has more than two possible classes, and each instance belongs to exactly one class.

Examples:

Handwritten digit recognition (digits 0-9)
Animal classification (cat, dog, horse, etc.)
Document topic classification (sports, politics, technology, etc.)

Common Algorithms:

Logistic Regression (extended for multi-class)
Decision Trees
Random Forests
Neural Networks

One-vs-Rest (OvR) and One-vs-One (OvO) methods

Multi-Label Classification

Definition: In multi-label classification, each instance can belong to multiple classes simultaneously.

Examples:

- Text classification (a document tagged with multiple topics)
- Image classification (an image containing multiple objects like a person, dog, and tree)
- Music genre classification (a song tagged with genres like rock, blues, and pop)

Common Algorithms:

- Binary Relevance
- Classifier Chains
- Adapted Decision Trees
- Adapted k-Nearest Neighbors
- Neural Networks with multi-label output

Imbalanced Classification

Definition: Classification where the number of instances in different classes is highly imbalanced, often leading to challenges in model training and evaluation.

Examples:

- Fraud detection (fraudulent transactions vs. legitimate transactions)
- Rare disease detection (disease vs. healthy)

Common Techniques:

- Resampling techniques (oversampling the minority class, undersampling the majority class)
- Synthetic data generation (SMOTE)
- Cost-sensitive learning
- Ensemble methods designed for imbalance

Ordinal Classification

Definition: Classification where the classes have a natural order, but the intervals between the classes are not necessarily equal or known.

Examples:

Customer satisfaction (very unsatisfied, unsatisfied, neutral, satisfied, very satisfied)

Credit ratings (A, B, C, D)

Common Algorithms:

Ordinal Logistic Regression

Decision Trees with ordinal splitting

Neural Networks adapted for ordinal output

Hierarchical Classification

Definition: Classification where classes are organized in a hierarchical structure, and predictions are made at multiple levels of the hierarchy.

Examples:

- Document classification within a hierarchical taxonomy (e.g., science -> biology -> genetics)
- Animal classification with taxonomic hierarchy (e.g., animal -> mammal -> carnivore -> dog)

Common Algorithms:

- Hierarchical clustering combined with classification
- Tree-based methods
- Custom hierarchical models (neural networks designed for hierarchical classification)

Single-Class Classification

Definition: Also known as one-class classification or anomaly detection, it involves identifying whether a new instance belongs to the normal class or is an anomaly.

Examples:

- Network intrusion detection
- Industrial equipment failure detection

Common Algorithms:

- One-Class SVM
- Isolation Forest
- Autoencoders (neural networks for anomaly detection)

Streaming Classification

Definition: Classification where the data is continuously arriving over time, and the model needs to adapt incrementally.

Examples:

- Real-time spam filtering
- Stock market prediction
- Online recommendation systems

Common Algorithms:

- Incremental versions of traditional algorithms (e.g., incremental decision trees)
- Online learning algorithms (e.g., Hoeffding Tree)
- Adaptive algorithms designed for concept drift

Multi-Class Classification

- Multi-class classification is a type of supervised learning where the goal is to categorize instances into one of three or more classes. Unlike binary classification, which involves only two classes (e.g., spam vs. not spam), multi-class classification deals with multiple classes, requiring the model to identify which class an instance belongs to out of several possible options.

Key Concepts

- **Classes**: The distinct categories into which instances are classified. For example, in a multi-class classification problem to identify the type of fruit, classes could include "apple," "banana," "cherry," etc.
- **Features**: The attributes or characteristics of the instances that the model uses to make predictions. In the fruit example, features might include color, size, weight, etc.
- **Training Data**: A dataset consisting of instances that have been labeled with their corresponding classes. This data is used to train the model.
- **Model**: An algorithm that learns from the training data to make predictions. Common algorithms for multi-class classification include decision trees, random forests, support vector machines, neural networks, and logistic regression.
- **Prediction**: The process of using the trained model to classify new, unseen instances into one of the predefined classes.

Strategies for Multi-Class Classification

There are main two strategies to handle multi-class classification:

- **One-vs-Rest (OvR)**
- **One-vs-One (OvO)**

One-vs-Rest (OvR)

- **One-vs-Rest (OvR)**: Also known as One-vs-All, this strategy involves training a separate binary classifier for each class. Each classifier distinguishes between one class and all other classes. During prediction, the classifier with the highest confidence score is chosen.
    - **Example**: If there are three classes (A, B, C), three classifiers are trained:
        - Classifier 1: A vs. (B and C)
        - Classifier 2: B vs. (A and C)
        - Classifier 3: C vs. (A and B)

One-vs-One (OvO)

One-vs-One (OvO): This strategy involves training a binary classifier for every possible pair of classes. For nn classes, $n(n-1)22n(n-1)$ classifiers are needed. Each classifier distinguishes between two classes. During prediction, a voting scheme is used where each classifier casts a vote for one of the two classes it was trained on, and the class with the most votes wins.

Example: For three classes (A, B, C), three classifiers are trained:

- Classifier 1: A vs. B
- Classifier 2: A vs. C
- Classifier 3: B vs. C

Evaluation Metrics

Evaluating the performance of a multi-class classification model involves various metrics that provide insights into different aspects of the model's predictive capabilities. To evaluate the performance of a multi-class classification model, the following metrics can be used:

- Accuracy
- Confusion Matrix
- Precision (for each class)
- Recall (for each class)
- F1-Score (for each class)
- Weighted Average
- ROC-AUC for Multi-Class
- Logarithmic Loss (Log Loss)

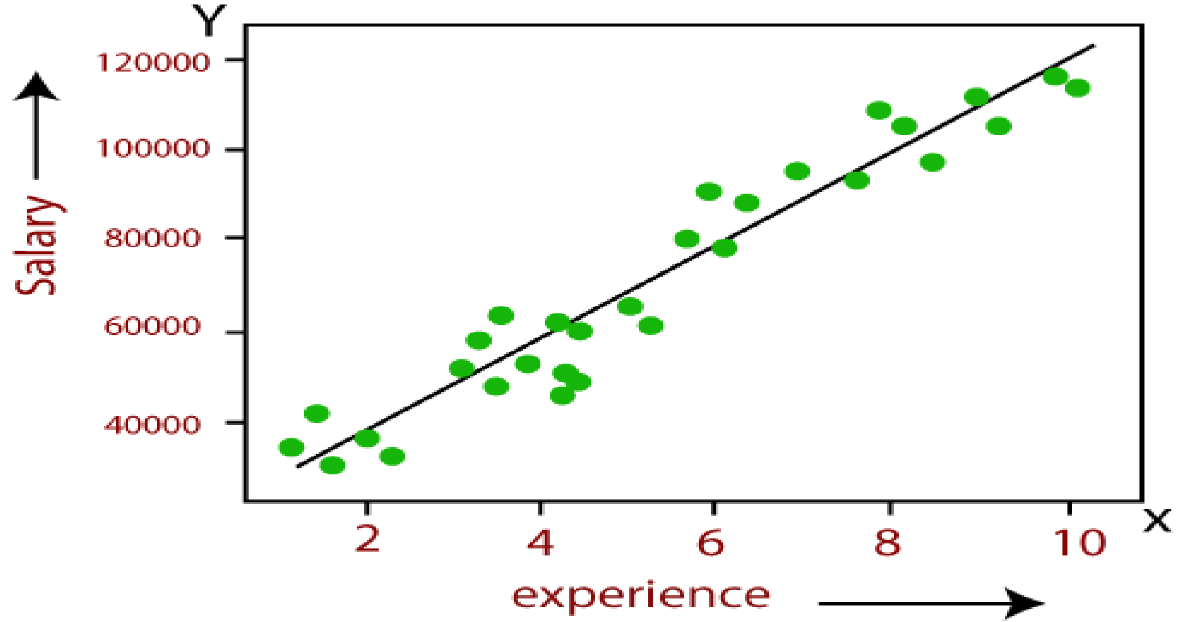
Linear Regression

- **Linear Regression** is a basic and widely used type of predictive analysis in statistics and machine learning. It models the relationship between a dependent variable and one independent variable by fitting a linear equation to the observed data.

Linear Regression

- Linear Regression is one of the most simple Machine learning algorithm that comes under Supervised Learning technique and used for solving regression problems.
- It is used for predicting the continuous dependent variable with the help of independent variables.
- The goal of the Linear regression is to find the best fit line that can accurately predict the output for the continuous dependent variable.
- If single independent variable is used for prediction then it is called Simple Linear Regression and if there are more than two independent variables then such regression is called as Multiple Linear Regression.
- By finding the best fit line, algorithm establish the relationship between dependent variable and independent variable. And the relationship should be of linear nature.
- The output for Linear regression should only be the continuous values such as price, age, salary, etc. The relationship between the dependent variable and independent variable can be shown in below image:

Linear Regression



In above image the dependent variable is on Y-axis (salary) and independent variable is on x-axis(experience).

The regression line can be written as: $y= a_0+a_1x+ \varepsilon$

Where, a_0 and a_1 are the coefficients and ε is the error term.

Multilinear Regression

Multilinear Regression (also known as Multiple Linear Regression) extends linear regression by modeling the relationship between a dependent variable and multiple independent variables.

Multiple Independent Variables: The equation involves more than one independent variable:

$$Y=\beta0+\beta1X1+\beta2X2+\cdots+\beta nXn+\epsilon$$

where:

Y is the dependent variable.

$X1,X2,\ldots,Xn$ are the independent variables.

$\beta0$ is the y-intercept.

$\beta1,\beta2,\ldots,\beta n$ are the coefficients (slopes) for each independent variable.

ϵ is the error term.

Multilinear Regression

Goal: Similar to simple linear regression, the goal is to find the coefficients (βs) that minimize the sum of the squared differences between the observed and predicted values.

Assumptions: The same as for simple linear regression, but applied to a multidimensional space:

- Linearity: The relationship between the dependent variable and each independent variable is linear.
- Independence: The residuals are independent.
- Homoscedasticity: Constant variance of residuals.
- Normality: The residuals are normally distributed.
- No multicollinearity: Independent variables should not be highly correlated with each other.

Differences Between Linear and Multilinear Regression

- **Number of Independent Variables**:
  - **Linear Regression**: Involves one independent variable.
  - **Multilinear Regression**: Involves multiple independent variables.
- **Complexity**:
  - **Linear Regression**: Simpler, easier to visualize and interpret.
  - **Multilinear Regression**: More complex, requires more data to estimate multiple parameters.
- **Use Cases**:
  - **Linear Regression**: Suitable for simple scenarios where the outcome is influenced by a single factor.
  - **Multilinear Regression**: Suitable for more complex scenarios where the outcome is influenced by multiple factors.

Applications

- **Linear Regression**: Predicting housing prices based on a single feature like square footage.
- **Multilinear Regression**: Predicting housing prices based on multiple features like square footage, number of bedrooms, location, age of the house, etc.

Naïve Bayes Classifier Algorithm

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solving classification problems.
- It is mainly used in *text classification* that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- **It is a probabilistic classifier, which means it predicts on the basis of the probability of an object**.
- Some popular examples of Naïve Bayes Algorithm are **spam filtration, Sentimental analysis, and classifying articles**.

Why is it called Naïve Bayes?

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

- **Naïve**: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.

- **Bayes**: It is called Bayes because it depends on the principle of Bayes' Theorem.

Bayes' Theorem:

Bayes' theorem is also known as **Bayes' Rule** or **Bayes' law**, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.

The formula for Bayes' theorem is given as:

$$P(A|B)= \frac{P(B|A)P(A)}{P(B)}$$

Where,

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

Working of Naïve Bayes' Classifier:

Working of Naïve Bayes' Classifier can be understood with the help of the below example:

Suppose we have a dataset of **weather conditions** and corresponding target variable "**Play**". So using this dataset we need to decide that whether we should play or not on a particular day according to the weather conditions. **So to solve this problem, we need to follow the below steps:**

1. Convert the given dataset into frequency tables.
2. Generate Likelihood table by finding the probabilities of given features.
3. Now, use Bayes theorem to calculate the posterior probability.

Problem: If the weather is sunny, then the Player should play or not?

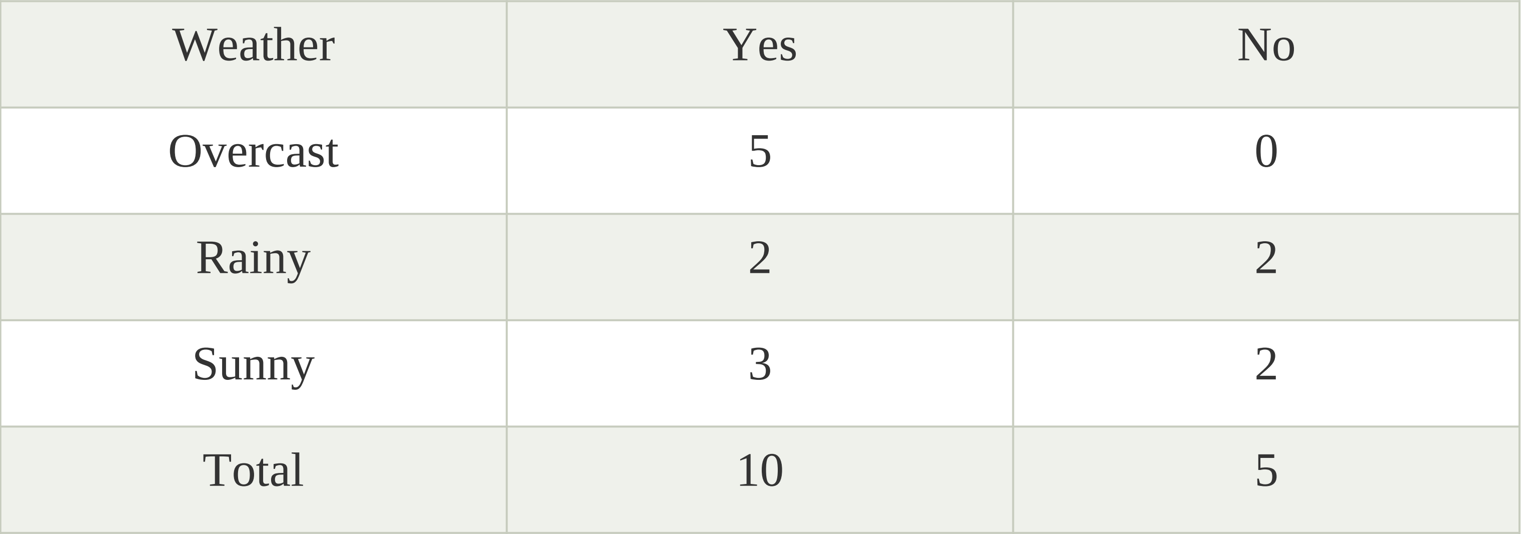
Working of Naïve Bayes' Classifier:

Solution: To solve this, first consider the given dataset: ->

	Outlook	Play
0	Rainy	Yes
1	Sunny	Yes
2	Overcast	Yes
3	Overcast	Yes
4	Sunny	No
5	Rainy	Yes
6	Sunny	Yes
7	Overcast	Yes
8	Rainy	No
9	Sunny	No
10	Sunny	Yes
11	Rainy	No
12	Overcast	Yes
13	Overcast	Yes

Working of Naïve Bayes' Classifier:

Frequency table for the Weather Conditions:



Weather	Yes	No
Overcast	5	0
Rainy	2	2
Sunny	3	2
Total	10	5

Working of Naïve Bayes' Classifier:

Likelihood table weather condition:

Weather	No	Yes	
Overcast	0	5	5/14= 0.35
Rainy	2	2	4/14=0.29
Sunny	2	3	5/14=0.35
All	4/14=0.29	10/14=0.71	

Working of Naïve Bayes' Classifier:

Applying Bayes'theorem:

P(Yes|Sunny)= P(Sunny|Yes)*P(Yes)/P(Sunny)

P(Sunny|Yes)= 3/10= 0.3

P(Sunny)= 0.35

P(Yes)=0.71

So P(Yes|Sunny) = 0.3*0.71/0.35= **0.60**

P(No|Sunny)= P(Sunny|No)*P(No)/P(Sunny)

P(Sunny|NO)= 2/4=0.5

P(No)= 0.29

P(Sunny)= 0.35

So P(No|Sunny)= 0.5*0.29/0.35 = **0.41**

So as we can see from the above calculation that **P(Yes|Sunny)>P(No|Sunny)**

Hence on a Sunny day, Player can play the game.

Advantages of Naïve Bayes Classifier::

- Naïve Bayes is one of the fast and easy ML algorithms to predict a class of datasets.
- It can be used for Binary as well as Multi-class Classifications.
- It performs well in Multi-class predictions as compared to the other Algorithms.
- It is the most popular choice for **text classification problems**.

Disadvantages of Naïve Bayes Classifier:

- Naive Bayes assumes that all features are independent or unrelated, so it cannot learn the relationship between features.

Applications of Naïve Bayes Classifier

- It is used for Credit Scoring.
- It is used in medical data classification.
- It can be used in real-time predictions because Naïve Bayes Classifier is an eager learner.
- It is used in Text classification such as Spam filtering and Sentiment analysis.

Types of Naïve Bayes Model:

There are three types of Naive Bayes Model, which are given below:

- **Gaussian**: The Gaussian model assumes that features follow a normal distribution. This means if predictors take continuous values instead of discrete, then the model assumes that these values are sampled from the Gaussian distribution.
- **Multinomial**: The Multinomial Naïve Bayes classifier is used when the data is multinomial distributed. It is primarily used for document classification problems, it means a particular document belongs to which category such as Sports, Politics, education, etc.
  The classifier uses the frequency of words for the predictors.
- **Bernoulli**: The Bernoulli classifier works similar to the Multinomial classifier, but the predictor variables are the independent Booleans variables. Such as if a particular word is present or not in a document. This model is also famous for document classification tasks.

Decision Tree Classification Algorithm:

- Decision Tree is a **Supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems.
- It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**
- In a Decision tree, there are two nodes, which are the **Decision Node** and **Leaf Node.**
- **Decision nodes** are used to make any decision and have multiple branches.
- **Leaf nodes** are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.

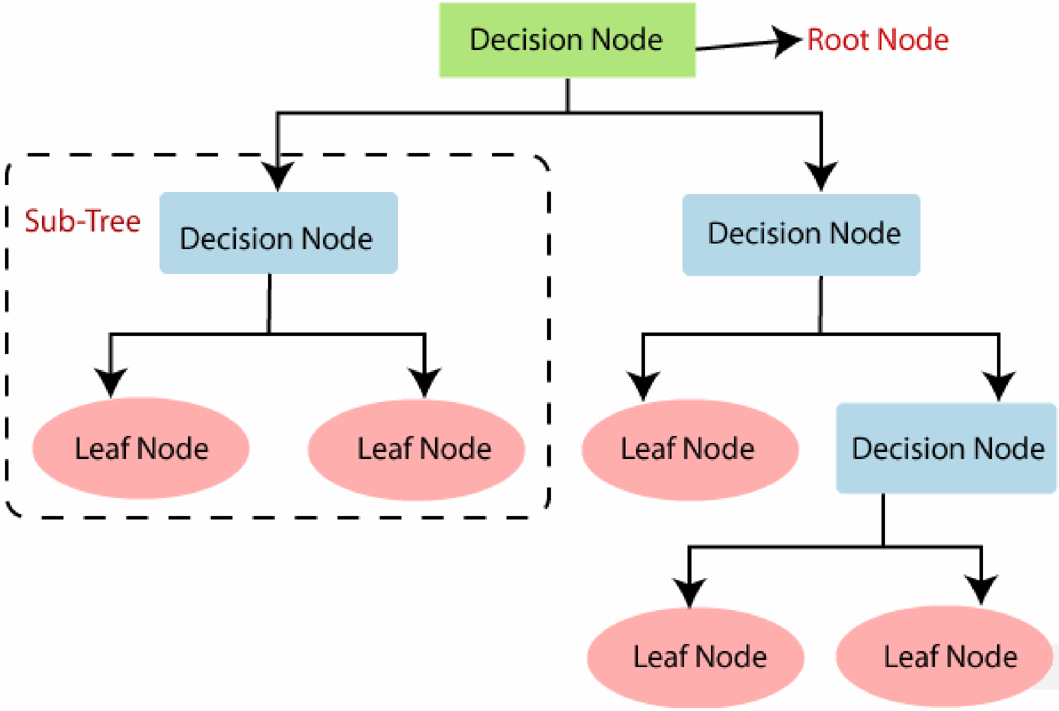
It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.

Decision Tree Classification Algorithm:

- It is called a decision tree because it is similar to a tree.
- It starts with the **root node**, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, we use the **ID3 And CART algorithm,** which stands for **Iterative Dichotomiser 3 / Classification and Regression Tree algorithm.**
- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.

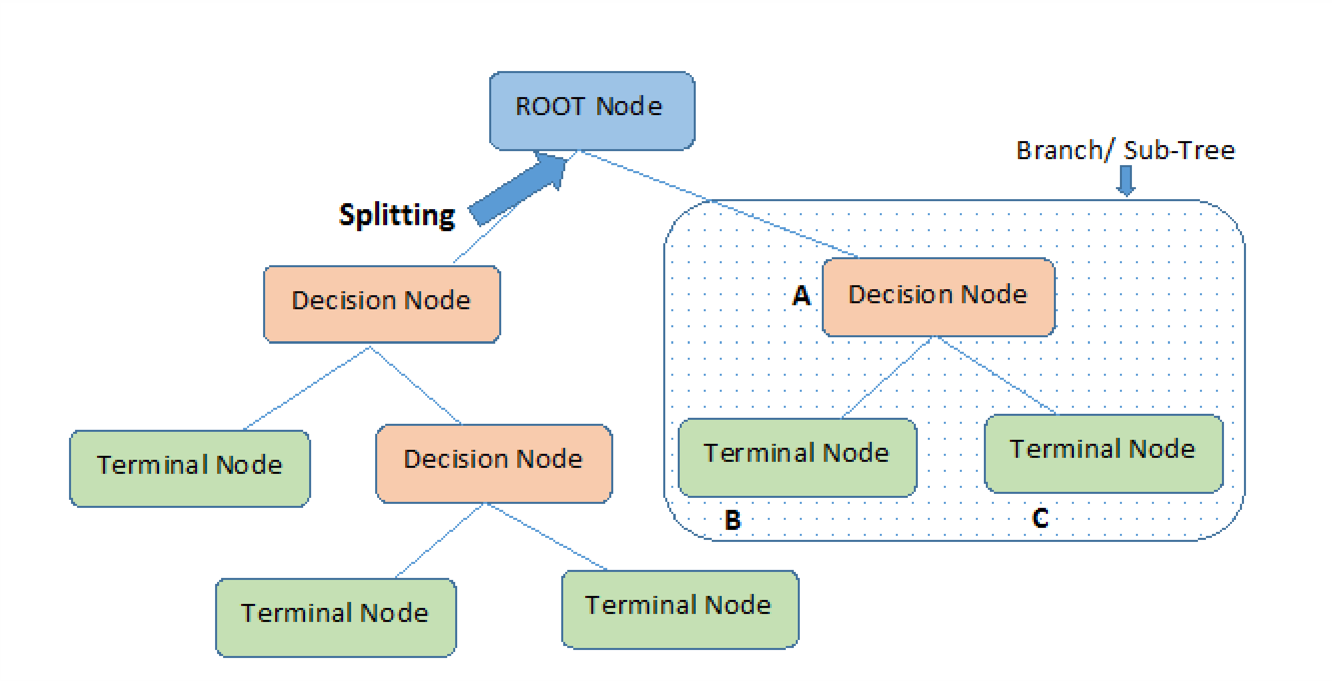
Decision Tree Classification Algorithm:

Below diagram explains the general structure of a decision tree:



Decision Tree Classification Algorithm:

Below diagram explains the general structure of a decision tree:



Why use Decision Trees?:

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:

- □ Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
- □ The logic behind the decision tree can be easily understood because it shows a tree-like structure.

Decision Tree Terminologies:

Root Node: Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.

Leaf Node: Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.

Splitting: Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.

Branch/Sub Tree: A tree formed by splitting the tree.

Pruning: Pruning is the process of removing the unwanted branches from the tree.

Parent/Child node: The root node of the tree is called the parent node, and other nodes are called the child nodes.

How does the Decision Tree algorithm Work?:

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree.

How does the Decision Tree algorithm Work?:

complete process can be better understood using the below algorithm:

Step-1: Begin the tree with the root node, says S, which contains the complete dataset.

Step-2: Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**

Step-3: Divide the S into subsets that contains possible values for the best attributes.

Step-4: Generate the decision tree node, which contains the best attribute.

Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

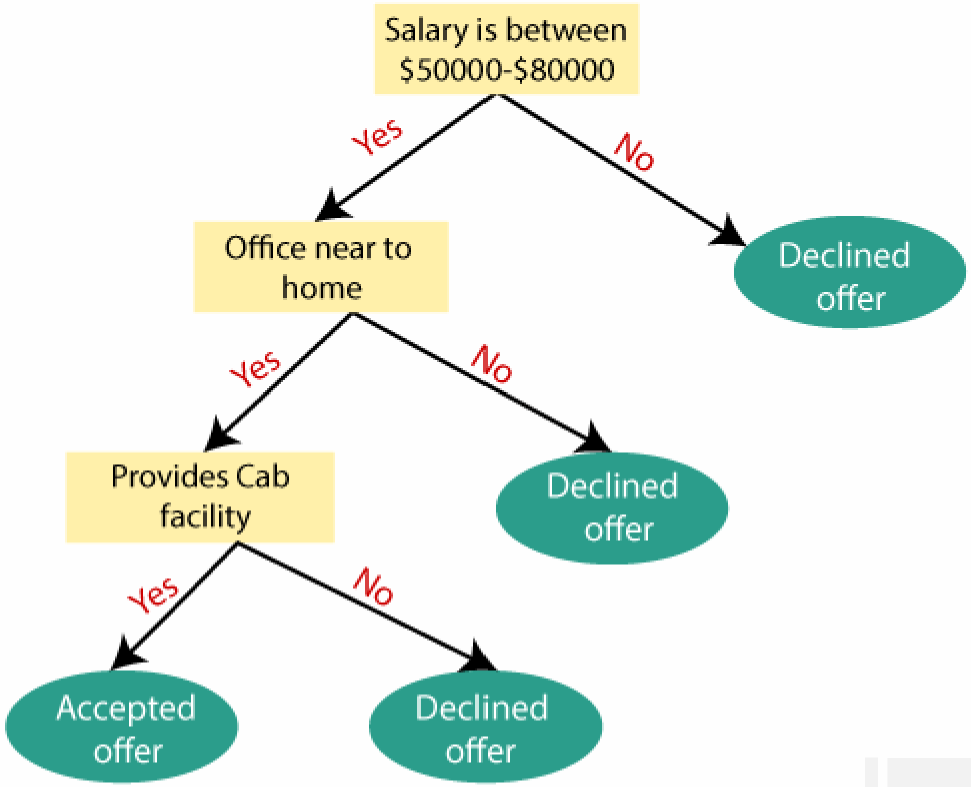
How does the Decision Tree algorithm Work?:

Example :

- ❑ Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM).
- ❑ The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels.
- ❑ The next decision node further gets split into one decision node (Cab facility) and one leaf node.
- ❑ Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). Consider the below diagram:

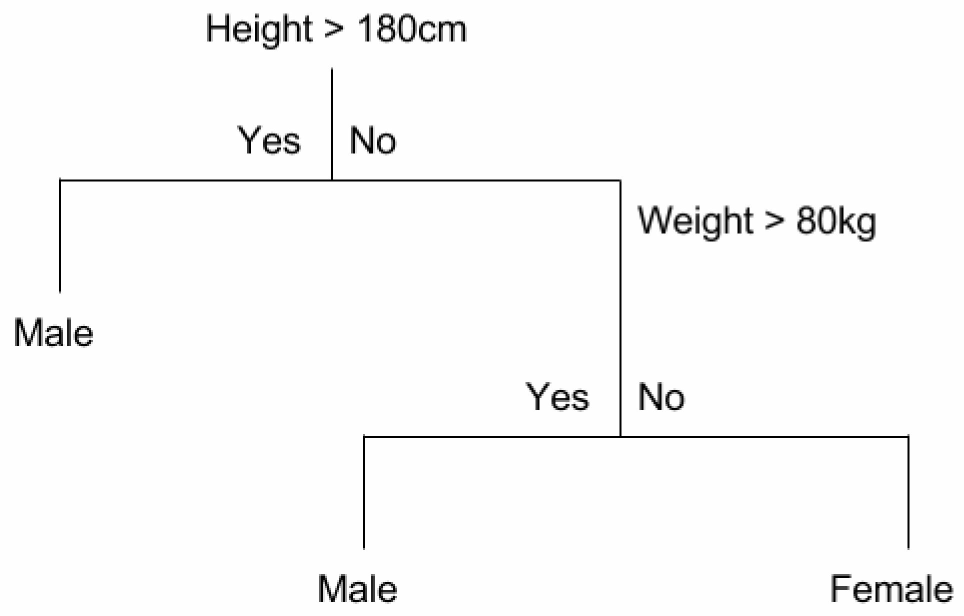
How does the Decision Tree algorithm Work?:

Consider the below diagram:



How does the Decision Tree algorithm Work?:

Example 2:



Attribute Selection Measures

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as Attribute selection measure or ASM. By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

❑ **Information Gain**

❑ **Gini Index**

Attribute Selection Measures

1. Information Gain:

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.
- According to the value of information gain, we split the node and build the decision tree.
- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first.
- It can be calculated using the below formula**:**

Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)

Attribute Selection Measures

Entropy: Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)**Where,**

S= Total number of samples

P(yes)= probability of yes

P(no)= probability of no

Attribute Selection Measures

2. Gini Index:

Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.

An attribute with the low Gini index should be preferred as compared to the high Gini index.

It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.

Gini index can be calculated using the below formula:

$$Gini = 1 - \sum_{i=1}^{j} P(i)^2$$

Where j represents the no. of classes in the target variable

P(i) represents the ratio of Pass/Total no. of observations in node

Pruning: Getting an Optimal Decision tree

- Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree.
- A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning.

There are mainly two types of tree **pruning** technology used:

□ **Cost Complexity Pruning**

□ **Reduced Error Pruning.**

Advantages of the Decision Tree

❑ It is simple to understand as it follows the same process which a human follow while making any decision in real-life.

❑ It can be very useful for solving decision-related problems.

❑ It helps to think about all the possible outcomes for a problem.

❑ There is less requirement of data cleaning compared to other algorithm

Disadvantages of the Decision Tree

- The decision tree contains lots of layers, which makes it complex.
- It may have an overfitting issue, which can be resolved using the Random Forest algorithm.
- For more class labels, the computational complexity of the decision tree may increase.

Disadvantages of the Decision Tree

- The decision tree contains lots of layers, which makes it complex.
- It may have an overfitting issue, which can be resolved using the Random Forest algorithm.
- For more class labels, the computational complexity of the decision tree may increase.

ID3 (Iterative Dichotomiser 3)

- ID3 algorithm, stands for Iterative Dichotomiser 3
- It is a classification algorithm that follows a greedy approach by selecting a best attribute that yields maximum Information Gain(IG) or minimum Entropy(H).

ID3 (Iterative Dichotomiser 3)

Procedure of ID3 Algorithm

1. Calculate entropy for the dataset.
2. For each node

- Calculate entropy for all its categorical values.
- Calculate information gain for the node.

3. Find the node with highest information gain at a particular level.
4. Repeat steps from 1 to 3 till we reach the leaf node and have created our decision tree.

ID3 (Iterative Dichotomiser 3)

Procedure of ID3 Algorithm

1. Calculate entropy for the dataset.
2. For each node

- Calculate entropy for all its categorical values.
- Calculate information gain for the node.

3. Find the node with highest information gain at a particular level.
4. Repeat steps from 1 to 3 till we reach the leaf node and have created our decision tree.

Creating a Decision Tree using the ID3 algorithm

Example 1: https://studygyaan.com/data-science-ml/creating-a-decision-tree-using-the-id3-algorithm

Example 2: https://iq.opengenus.org/id3-algorithm/

Limitations of ID3 Algorithm

There are some disadvantages of ID3 Algorithm :-

- ❑ Over fitting of Data for small dataset.
- ❑ One attribute is considered at the time of making decisions.
- ❑ Continuous data classification can be computationally expensive.

Classification and Regression Trees (CART) algorithm

- The CART algorithm is a type of classification algorithm that is required to build a decision tree on the basis of Gini’s impurity index.
- This algorithm can be used for both classification & regression.
- CART algorithm uses Gini Index criterion to split a node to a sub-node.
- Carts create binary tree
- It start with the training set as a root node, after successfully splitting the root node in two, it splits the subsets using the same logic & again split the sub-subsets, recursively until it finds further splitting will not give any pure sub-nodes or maximum number of leaves in a growing tree or termed it as a Tree pruning.

Key Concepts CART algorithm

- **Decision Tree:** A flowchart-like structure where internal nodes represent features or attributes, branches represent decisions, and leaf nodes represent class labels or target values.
- **Splitting:** The process of dividing a node into two or more sub-nodes based on a certain feature and its threshold value.
- **Impurity:** A measure of the disorder or uncertainty in a node. Common impurity measures include Gini index and entropy.
- **Pruning:** The process of reducing the size of a tree by removing unnecessary branches to improve generalization.

CART Algorithm for Classification

Here is the approach for most decision tree algorithms at their most simplest.

The tree will be constructed in a top-down approach as follows:

- **Step 1:** Start at the root node with all training instances
- **Step 2:** Select an attribute on the basis of splitting criteria (Gini Index or other impurity metrics)
- **Step 3:** Partition instances according to selected attribute recursively

Partitioning stops when:

- There are no examples left
- All examples for a given node belong to the same class
- There are no remaining attributes for further partitioning – majority class is the leaf

CART Algorithm for Classification

```
//CART Algorithm
INPUT: Dataset D
        1. Tree = {}
        2. MinLoss = 0
        3. for all Attribute k in D do:
                3.1. loss = GiniIndex(k, d)
                3.2. if loss<MinLoss then
                        3.2.1. MinLoss = loss
                        3.2.2. Tree' = {k}
        4. Partition(Tree, Tree')
        5. until all partitions procressed
        6. return Tree
OUTPUT: Optimal Decision Tree
```

Gini Index for CART Algorithm

Gini index is a metric for classification tasks in CART. It stores sum of squared probabilities of each class. We can formulate it as illustrated below.

$$\textbf{Gini} = 1 - \Sigma\, (Pi)^2 \textbf{ for } i=1 \textbf{ to number of classes}$$

Classification and Regression Trees (CART) algorithm

- **Example 1:** https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/.

Advantages and Limitations CART

Advantages :

1. Simplicity and interpretability.
2. Ability to handle both categorical and numerical features.
3. Robustness against outliers and missing values.

Limitations :

1. Tendency to overfit if not properly regularized.
2. Difficulty handling irrelevant features.
3. Lack of smoothness in the decision boundaries.

Real-World Applications CART

1. **Credit Scoring:** Predicting creditworthiness based on customer attributes.
2. **Disease Diagnosis:** Identifying diseases based on symptoms and patient characteristics.
3. **Customer Churn Prediction:** Predicting whether a customer is likely to cancel a subscription or leave a service.
4. **Stock Market Analysis:** Forecasting stock prices based on historical data.

Error Bounds

- Error bounds are estimates that quantify the uncertainty or potential error in the predictions or estimates made by a model. They provide a range within which the true value is expected to lie, giving a measure of the reliability and accuracy of the model. Error bounds are crucial in both theoretical and practical applications of statistics and machine learning.

Types of Error Bounds

- Confidence Intervals
- Prediction Intervals
- Chebyshev's Inequality
- Hoeffding's Inequality
- PAC (Probably Approximately Correct) Bounds

Confidence Intervals

- **Definition**: A confidence interval provides a range of values, derived from the sample data, that is likely to contain the true value of an unknown population parameter.
- **Example**: If a 95% confidence interval for a population mean is (5, 10), we are 95% confident that the true mean lies between 5 and 10.
- **Calculation**: For a sample mean *x*¯*x*¯ with standard deviation *ss*, the confidence interval is given by:

$$\bar{x} \pm z\frac{s}{\sqrt{n}}$$

where *zz* is the z-score corresponding to the desired confidence level, and *nn* is the sample size.

Prediction Intervals

- **Definition**: A prediction interval provides a range within which a single new observation is expected to fall with a certain probability.
- **Example**: A 95% prediction interval for the next value in a series might be (8, 12), meaning there's a 95% chance the next observed value will lie between 8 and 12.
- **Calculation**: For a predicted value *y*^ with standard deviation of the residuals *sese*, the prediction interval is:

$$\hat{y} \pm t \cdot s_e \sqrt{1 + \frac{1}{n}}$$

where *tt* is the t-score for the desired confidence level.

Chebyshev's Inequality

- **Definition**: Provides a bound on the probability that the value of a random variable deviates from its mean by more than a certain number of standard deviations, applicable to any distribution with a finite mean and variance.
- **Example**: For any random variable XX with mean $\mu\mu$ and standard deviation $\sigma\sigma$, Chebyshev's inequality states that:

$$P(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}$$

- This means that at least $1 - \frac{1}{k^2}$ of the values lie within k standard deviations of the mean.

Hoeffding's Inequality

- **Definition**: Provides a bound on the probability that the sum of bounded independent random variables deviates from its expected value.
- **Example**: For independent random variables $X1,X2,\ldots,XnX1,X2,\ldots,Xn$ each bounded by the interval $[ai,bi][ai,bi]$, the inequality states that:

$$P\left(\left|\frac{1}{n}\sum_{i=1}^{n}X_i-\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}X_i\right]\right|\geq t\right)\leq 2\exp\left(-\frac{2n^2t^2}{\sum_{i=1}^{n}(b_i-a_i)^2}\right)$$

This helps in determining the deviation from the expected mean.

PAC (Probably Approximately Correct) Bounds

- **Definition**: Used in machine learning to provide a bound on the probability that a learned hypothesis is approximately correct within a certain margin of error.
- **Example**: For a hypothesis h learned from a training set of size m, with confidence $1−\delta$, the PAC bound is:

$$P\left(\left|\text{error}(h) - \widehat{\text{error}}(h)\right| > \epsilon\right) \leq \delta$$

where error(h) is the true error, error^(h) is the empirical error, ϵ is the error margin, and δ is the confidence parameter.

Importance of Error Bounds

1. **Uncertainty Quantification**: Error bounds provide a way to quantify the uncertainty in model predictions, allowing for more informed decision-making.
2. **Model Evaluation**: They help in evaluating the performance and reliability of models, ensuring that predictions are within acceptable limits.
3. **Risk Management**: In critical applications like finance or healthcare, error bounds help manage risks by providing worst-case scenarios.
4. **Research and Development**: In research, error bounds help validate the findings and ensure that the results are not due to random chance.

Thanks