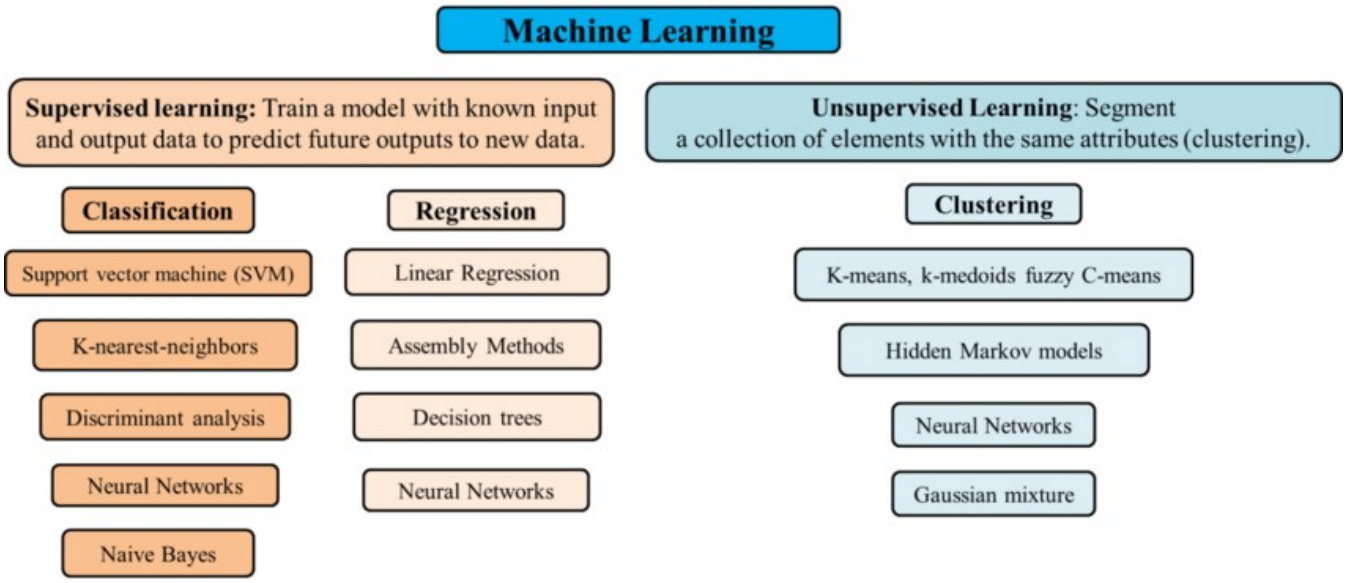
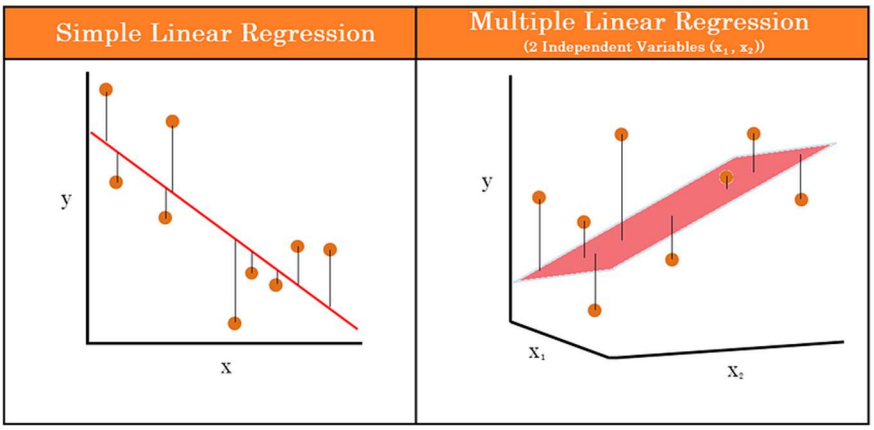
MACHINE LEARNING ALGORITHMS



Linear Regression

Linear regression is a foundational algorithm used for predicting a continuous output variable based on one or more input features. The model assumes a linear relationship between the input features and the output. The fundamental idea is to find the best-fitting line that minimizes the difference between the predicted and actual values.



- For a single feature: $h_\theta(x) = \theta_0 + \theta_1 x$
- For multiple features: $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$

Loss Function

Mean Squared Error (MSE)	$MSE = \frac{1}{N}\sum_{i}^{N}(Yi - \hat{Y}i)^2$
Root Mean Squared Error (RMSE)	$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$
Mean Absolute Error (MAE)	$MAE = \frac{1}{n}\sum_{i=1}^{n}\lvert y_i - \hat{y}_i \rvert$

The algorithm solves the minimization problem and is achieved using ***Gradient Descent.*** Gradient descent is a method of changing weights based on the loss function for each data point. We calculate the sum of squared errors at each input-output data point. We take a *partial derivative* of the *weight* and *bias* to get the *slope of the cost function* at each point.

Evaluation Metric: R-squared (R^2) Score

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \mu)^2}$$

- Also referred to as the coefficient of determination. It quantifies the percentage of the dependent variable's variation that the model's independent variables contribute to.
- R2 score is a useful statistic for evaluating the overall effectiveness and explanatory power of a regression model.

Assumptions of Linear Regression

- Linear Model
- No Multicollinearity in the data
- Homoscedasticity of Residuals or Equal Variances
- No Autocorrelation in residuals
- Number of observations greater than the number of predictors
- Each observation is unique
- Predictors are distributed Normally

- **Advantages:**
    - Simple and interpretable.
    - Efficient for linear relationships.
- **Disadvantages:**
    - Sensitive to outliers.
    - Assumes a linear relationship, which might not hold in complex data.

Logistic Regression

Logistic regression is a classification algorithm used for binary or multiclass classification problems. The model uses the logistic function to map a linear combination of input features to a probability between 0 and 1. To accomplish this, a logistic regression model uses sigmoid function as its hypothesis function.

Sigmoid Function

Sigmoid function is the hypothesis function for a Logistic Regression model.

$$Sigmoid\ Function: \quad g(z) = \frac{1}{1 + e^{(-z)}}$$

$$Hypothesis: \quad h_\theta(x) = \frac{1}{1 + e^{(-\theta^T x)}}$$

Loss Function

$$\text{Log Loss} = \sum_{(x,y)\in D} -y \log(y') - (1-y) \log(1-y')$$

where:

- $(x, y) \in D$ is the data set containing many labeled examples, which are (x, y) pairs.
- y is the label in a labeled example. Since this is logistic regression, every value of y must either be 0 or 1.
- y' is the predicted value (somewhere between 0 and 1), given the set of features in x.

Evaluation Metrics

There are four ways to check if the predictions are right or wrong:

1. **TN / True Negative**: the case was negative and predicted negative
2. **TP / True Positive**: the case was positive and predicted positive
3. **FN / False Negative**: the case was positive but predicted negative
4. **FP / False Positive**: the case was negative but predicted positive.

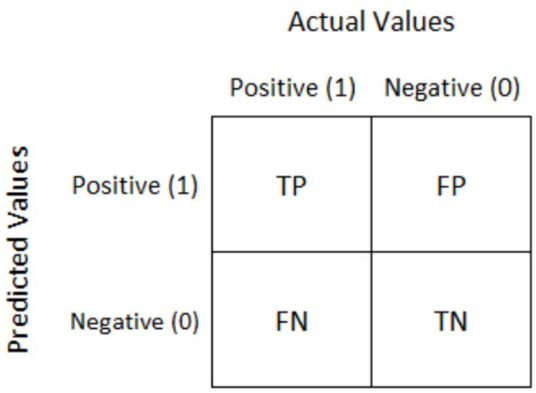
1) Accuracy: The ratio of number of correct predictions divided by the total number of predictions.

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made}$$

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

Accuracy is often used as the measure of classification performance because it is simple to compute and easy to interpret. However, it can turn out to be misleading in some cases. This is especially true when dealing with imbalanced data, a scenario when certain classes contain way more data points than the others.

2) Confusion Matrix: A confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known.



- **True Positive (TP):** when both the actual and predicted values are 1.
- **True Negative (TN):** when both the actual and predicted values are 0.
- **False Positive (FP):** when the actual value is 0 but the predicted value is 1.
- **False Negative (FN):** when the actual value is 1 but the predicted value is 0.

3) Precision: The ratio of number of true positives divided by the number of predicted positives.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

Precision is the ability of a classifier not to label an instance positive that is actually negative. It is useful in the cases where False Positive is a higher concern than False Negatives.

The importance of *Precision is in music or video recommendation systems, e-commerce websites, etc. where wrong results could lead to customer churn and this could be harmful to the business.*

4) Recall: The ratio of number of true positives divided by the total number of actual positives.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

Recall is the ability of a classifier to find all positive instances. It is a useful metric in cases where False Negative is of higher concern than False Positive.

It *is important in medical cases where it doesn't matter whether we raise a false alarm but the actual positive cases should not go undetected!*

5) F1 Score: Weighted Harmonic mean of precision and recall.

$$F1 = 2.\frac{Precision \times Recall}{Precision + Recall}$$

It gives a combined idea about Precision and Recall metrics. The best F1 score is 1 and the worst is 0. It is maximum when Precision is equal to Recall.

Precision and recall are more interpretable than F1-score, since precision measures the type-1 error and recall measures the type-2 error. However, F1-score measures the trade-off between these two. As a rule of thumb, the weighted average of F1 should be used to compare classifier models, not global accuracy.

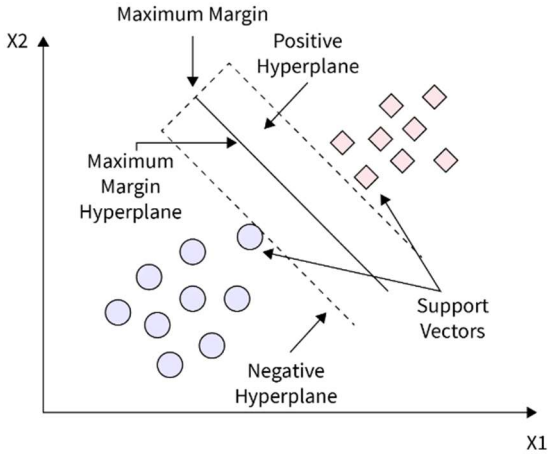
- **Advantages:**
  - Efficient for binary classification.
  - Provides probabilities for decision making.
- **Disadvantages:**
  - Assumes a linear relationship between features and log-odds.
  - May struggle with non-linear relationships.

Support Vector Machines

Support Vector Machine (SVM) is a supervised machine learning algorithm used for both classification and regression.

The main objective of the SVM algorithm is to find the optimal hyperplane in an N-dimensional space that can separate the data points in different classes in the feature space. 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. The hyperplane tries that the margin between the closest points of different classes should be as maximum as possible.

Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

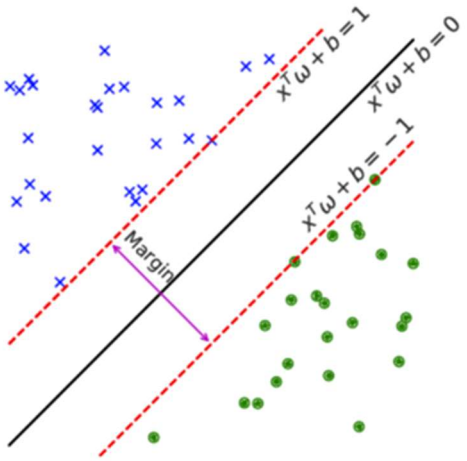


The SVM algorithm has the characteristics to ignore the outlier and finds the best hyperplane that maximizes the margin. Maximizing this margin is essential as it provides a measure of robustness against noise and aids in better generalization to unseen data. Two approaches to margins in SVMs are Hard Margin and Soft Margin.

Hard Margin SVM

In a hard margin SVM, the goal is to find the hyperplane that can perfectly separate the data into two classes without any misclassification. The objective function for hard margin is given by:

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}||\mathbf{w}||^2 \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad i = 1, ..., n \end{aligned}$$



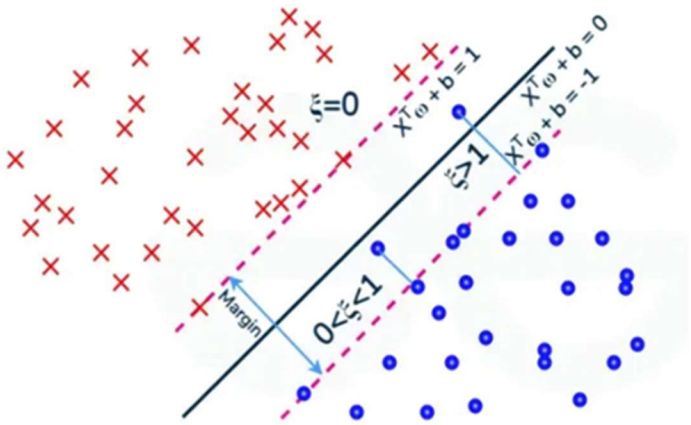
However, this is not always possible when the data is not linearly separable or contains outliers. In such cases, the hard margin SVM will fail to find a hyperplane that can perfectly separate the data, and the optimization problem will have no solution**.**

Soft Margin SVM

In a soft margin SVM, we allow some misclassification by introducing slack variables that allow some data points to be on the wrong side of the margin. The optimization problem in a soft margin SVM finds the hyperplane that maximizes the margin while minimizing the penalty for the misclassified data points.

The objective function of a soft margin SVM combines the margin maximization with a penalty term for margin violations is given by,

$$\begin{aligned} \min_{\xi,w,b} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i \\ s.t. \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, \quad i = 1, \cdots, m \\ & \xi_i \geq 0, \quad i = 1, \cdots, m \end{aligned}$$



where ξi are the slack variable, and C is a hyperparameter that controls the trade-off between maximizing the margin and minimizing the misclassification.

A larger value of C results in a narrow margin and fewer misclassifications, while a smaller value of C results in a wider margin but more misclassifications.

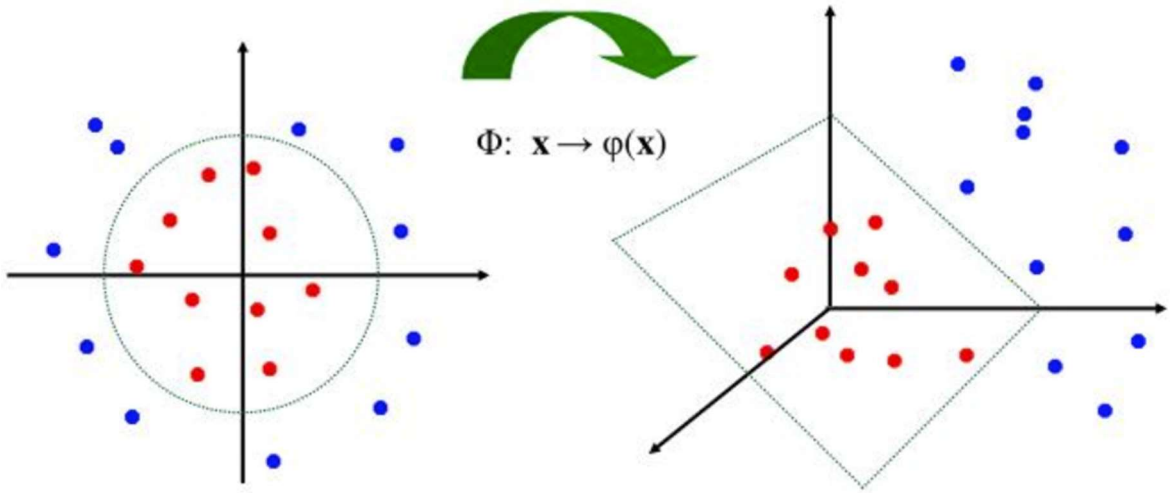
Hard Margin SVM vs Soft Margin SVM

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

Non – Linear SVM / Kernel SVM

Nonlinear SVM (Support Vector Machine) is necessary when the data cannot be effectively separated by a linear decision boundary in the original feature space. Nonlinear SVM addresses this limitation by utilizing Kernel functions to map the original input data points into high-dimensional feature spaces, allowing the hyperplane to be located even when the data points are not linearly separable in the original input space.

Kernal Function is given by, $K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$



The objective function of Non-Linear Function is given by,

$$\text{maximize} \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j)$$

$$\text{such that} \quad 0 \le \alpha_i \le C$$

$$\sum_{i=1}^{n}\alpha_i y_i = 0$$

The frequently used kernel functions in SVM are:

- linear $k(\mathbf{x}_1,\mathbf{x}_2) = \mathbf{x}_1 \cdot \mathbf{x}_2$
- polynomial $k(\mathbf{x}_1,\mathbf{x}_2) = (\gamma\, \mathbf{x}_1 \cdot \mathbf{x}_2 + c)^d$
- Gaussian or radial basis $k(\mathbf{x}_1,\mathbf{x}_2) = \exp\left(-\gamma\|\mathbf{x}_1 - \mathbf{x}_2\|^2\right)$
- sigmoid $k(\mathbf{x}_1,\mathbf{x}_2) = \tanh(\gamma\, \mathbf{x}_1 \cdot \mathbf{x}_2 + c)$

Linear SVM vs Non-Linear SVM

Linear SVM	Non-Linear SVM
The data points are separated using a single line	The data points are hard to separate, so other shapes are used as a decision boundary.
Data is classified with the help of a hyperplane.	Kernels are used to classify data points.
Difficult to classify more than two labels.	Used for classifying more than two labels.

Advantages of SVM

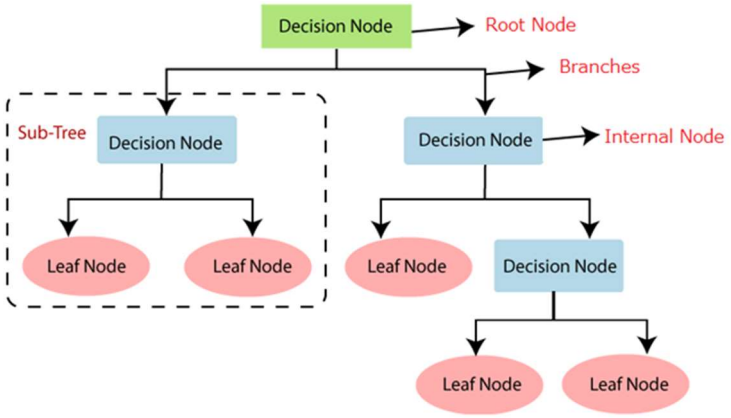
- SVM works relatively well when there is a clear margin of separation between classes.
- SVM is more effective in high dimensional spaces.
- SVM is effective in cases where the number of dimensions is greater than the number of samples.
- SVM is relatively memory efficient

Disadvantages of SVM

- SVM algorithm is not suitable for large data sets.
- SVM does not perform very well when the data set has more noise i.e. target classes are overlapping.
- In cases where the number of features for each data point exceeds the number of training data samples, the SVM will underperform.
- As the support vector classifier works by putting data points, above and below the classifying hyperplane there is no probabilistic explanation for the classification.

Decision Tree Algorithm

A decision tree is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks. It has a hierarchical, tree structure, which consists of a root node, branches, internal nodes and leaf nodes.



The nodes represent the input features, the branches represent the decisions based on those features, and the leaves represent the outcomes or predictions.

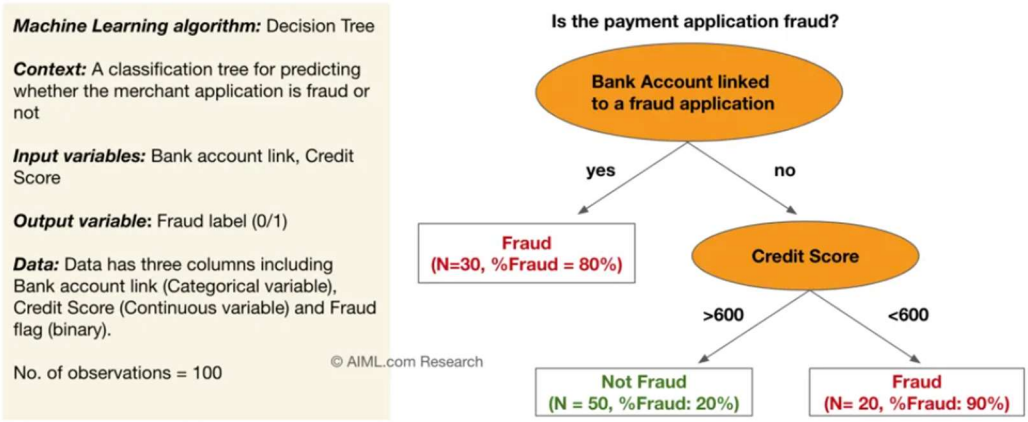
Root Nodes: A root node is the beginning of the decision tree where the decision tree starts. This is the first feature that starts splitting the data.

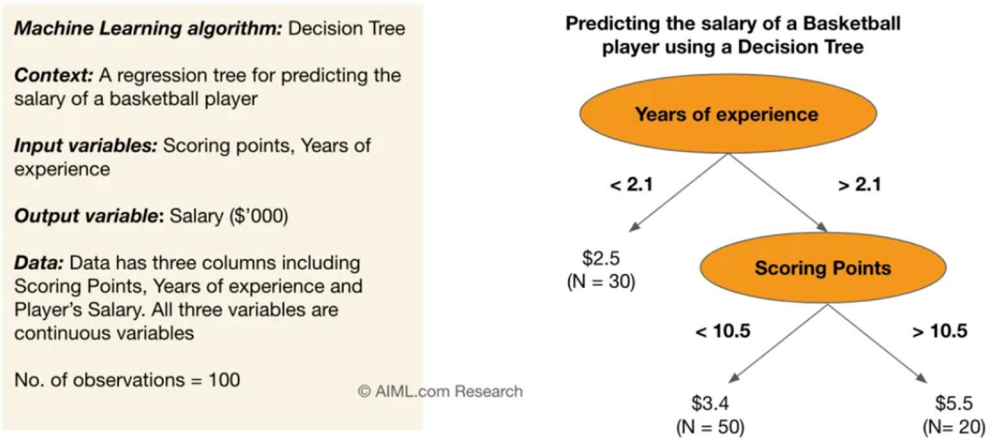
Decision Nodes: The nodes after splitting based on the root node are further split based on the decision node.

Leaf Nodes: These are the final nodes where the decision tree makes final decisions and no further splitting is possible.

How to predict using a Decision Tree model?

To predict the target variable for a new observation, the decision tree traverses the tree based on the values of the features for the new observation, starting from the root node and following the path that satisfies the conditions until a leaf node is reached. The prediction is then based on the majority class (classification) or average value of the samples (regression) in that leaf node.





Assumptions we make while using Decision tree are:

- In the beginning, the whole training set is considered as the **root.**
- Feature values are preferred to be categorical. If the values are continuous then they are discretized prior to building the model.
- Records are **distributed recursively** on the basis of attribute values.
- Order to placing attributes as root or internal node of the tree is done by using some statistical approach.

How is a Decision Tree model built?

The decision tree model is built by recursively splitting the dataset into subsets based on the feature that provides the most **information gain** (for classification tasks) or the **highest reduction in variance** (for regression tasks). The information gain and variance reduction are calculated based on the impurity of the subsets where impurity measures how mixed the classes or values are in the subset. The most common impurity measures used for classification tasks are **Gini impurity and Entropy**, while **Mean squared error (MSE)** is used for regression tasks.

Once the dataset is split, the process is repeated for each subset until a stopping criterion is reached. The stopping criterion could be a maximum depth of the tree, a minimum number of samples required to split a node, or a minimum impurity decrease required to split a node. The depth of a decision tree refers to the length of the longest path from the root node to a leaf node.

Entropy: A metric to measure the impurity in a given attribute. It specifies randomness in data. The higher the entropy, the harder it is to draw any conclusions from that information.

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

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.

Information gain is a decrease in entropy. It computes the difference between entropy before split and average entropy after split of the dataset based on given attribute values.

$$Information\ Gain = Entropy(before) - \sum_{j=1}^{K} Entropy(j, after)$$

where “before” is the dataset before the split, K is the number of subsets generated by the split, and (j, after) is subset j after the split.

- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute that returns the highest information gain and the smallest entropy is split first.

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.
- Higher value of Gini index implies higher inequality, higher heterogeneity.

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

Pruning

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 trees pruning technology used:

- **Cost Complexity Pruning**
- **Reduced Error Pruning.**

Advantages of DT

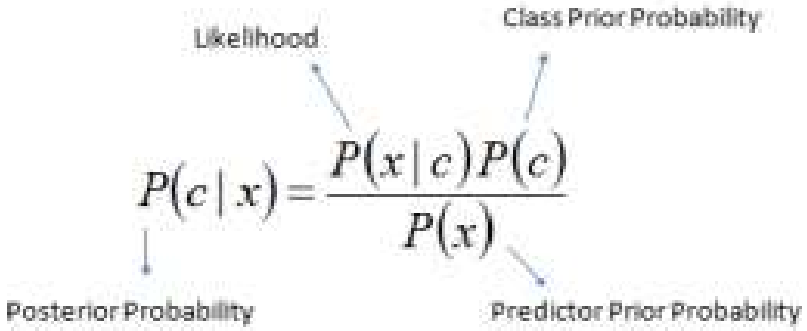
- No feature scaling is required.
- The non-linearity relationship between the dependent and independent variables does not affect the performance of the decision tree algorithms.
- A decision tree algorithm can handle outliers automatically.
- Handles missing values automatically.

Disadvantages of DT

- The decision tree contains lots of layers, which makes it complex.
- Decision tree algorithms can be unstable easily. A little bit of noise in the data or adding one extra piece of data may make the whole tree unstable or may build the tree all over again.
- A decision tree algorithm is not suitable for a large dataset. If the dataset is too large, the algorithm may become too complex and may lead to an overfitting issue.

Naïve Bayes Algorithm

Naive Bayes classifier is a probabilistic machine learning model based on Bayes' theorem. It assumes independence between features and calculates the probability of a given input belonging to a particular class. It's widely used in text classification, spam filtering, and recommendation systems.



$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

How Do Naive Bayes Algorithms Work?

1. Convert the data set into a frequency table
2. Create Likelihood table by finding the probabilities
3. Use Naive Bayesian equation to calculate the posterior probability

The class with the highest posterior probability is the outcome of the prediction.

Advantages of NB

- It is easy and fast to predict class of test data set. It also performs well in multi class prediction
- When assumption of independence holds, the classifier performs better compared to other machine learning models like logistic regression or decision tree, and requires less training data.
- It performs well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

Disadvantages of NB

- If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as "Zero Frequency". To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.
- On the other side, Naive Bayes is also known as a bad estimator, so the probability outputs from predict_proba are not to be taken too seriously.
- Another limitation of this algorithm is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.