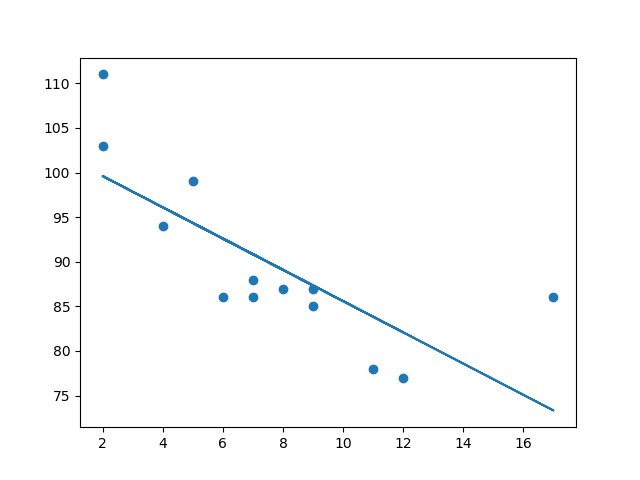
# Simple Linear Regression

The term regression is used when you try to find the relationship between variables. In Machine Learning, and in statistical modeling, that relationship is used to predict the outcome of future events.

Linear regression uses the relationship between the data-points to draw a straight line through all them.

This line can be used to predict future values.



**Linear regression** is also a type of [**supervised machine-learning algorithm**](https://www.geeksforgeeks.org/supervised-machine-learning/) that learns from the labelled datasets and maps the data points with most optimized linear functions which can be used for prediction on new datasets. It computes the linear relationship between the dependent variable and one or more independent features by fitting a linear equation with observed data.

The model’s equation offers **clear coefficients** that illustrate the **influence of each independent variable on the dependent variable**, enhancing our understanding of the underlying relationships. Its simplicity is a significant advantage; linear regression is transparent, easy to implement.

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.

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 performs the task to predict a dependent variable value (y) based on a given independent variable (x)).

The model gets the best regression fit line by finding the best θ1 and θ2 values.

* **θ1:** intercept
* **θ2:** coefficient of x (also known as slope)

Once we find the best θ1 and θ2 values, we get the best-fit line.

Simple Linear Regression

[Simple linear regression](https://www.geeksforgeeks.org/simple-linear-regression-in-python/) 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=β0+β1X*y*=*β*0​+*β*1​*X*  
where:

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

Multiple Linear Regression

[Multiple linear regression](https://www.geeksforgeeks.org/ml-multiple-linear-regression-using-python/) involves more than one independent variable and one dependent variable. The equation for multiple linear regression is:  
y=β0+β1X1+β2X2+………βnXn*y*=*β*0​+*β*1​*X*1+*β*2​*X*2+………*βn*​*Xn*  
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.**

In regression set of records are present with X and Y values and these values are used to learn a function so if you want to predict Y from an unknown X this learned function can be used. In regression we have to find the value of Y, So, a function is required that predicts continuous Y in the case of regression given X as independent features.

## Evaluation Metrics

**Mean Square Error (MSE)**

[Mean Squared Error (MSE)](https://www.geeksforgeeks.org/python-mean-squared-error/) is an evaluation metric that calculates the average of the squared differences between the actual and predicted values for all the data points. The difference is squared to ensure that negative and positive differences don’t cancel each other out.

MSE=1n∑i=1n(yi–yi^)2*MSE*=*n*1​∑*i*=1*n*​(*yi*​–*yi*​​)2

Here,

* n is the number of data points.
* yi is the actual or observed value for the ith data point.
* yi^*yi*​​ is the predicted value for the ith data point.

MSE is a way to quantify the accuracy of a model’s predictions. MSE is sensitive to outliers as large errors contribute significantly to the overall score.

**Mean Absolute Error (MAE)**

[Mean Absolute Error](https://www.geeksforgeeks.org/how-to-calculate-mean-absolute-error-in-python/)is an evaluation metric used to calculate the accuracy of a regression model. MAE measures the average absolute difference between the predicted values and actual values.

Mathematically, MAE is expressed as:

MAE=1n∑i=1n∣Yi–Yi^∣*MAE*=*n*1​∑*i*=1*n*​∣*Yi*​–*Yi*​​∣

Here,

* n is the number of observations
* Yi represents the actual values.
* Yi^*Yi*​​ represents the predicted values

Lower MAE value indicates better model performance. It is not sensitive to the outliers as we consider absolute differences.

**Root Mean Squared Error (RMSE)**

The square root of the residuals’ variance is the [Root Mean Squared Error](https://www.geeksforgeeks.org/root-mean-square-error-in-r-programming/). It describes how well the observed data points match the expected values, or the model’s absolute fit to the data.

In mathematical notation, it can be expressed as:  
RMSE=RSSn=∑i=2n(yiactual−yipredicted)2n*RMSE*=*nRSS*​​=*n*∑*i*=2*n*​(*yiactual*​−*yipredicted*​)2​​  
Rather than dividing the entire number of data points in the model by the number of degrees of freedom, one must divide the sum of the squared residuals to obtain an unbiased estimate. Then, this figure is referred to as the Residual Standard Error (RSE).

In mathematical notation, it can be expressed as:  
RMSE=RSSn=∑i=2n(yiactual−yipredicted)2(n−2)*RMSE*=*nRSS*​​=(*n*−2)∑*i*=2*n*​(*yiactual*​−*yipredicted*​)2​​

RSME is not as good of a metric as R-squared. Root Mean Squared Error can fluctuate when the units of the variables vary since its value is dependent on the variables’ units (it is not a normalized measure).

**Coefficient of Determination (R-squared)**

[R-Squared](https://www.geeksforgeeks.org/r-squared/) is a statistic that indicates how much variation the developed model can explain or capture. It is always in the range of 0 to 1. In general, the better the model matches the data, the greater the R-squared number.  
In mathematical notation, it can be expressed as:  
R2=1−(RSSTSS)*R*2=1−(*TSSRSS*​)

* [**Residual sum of Squares**](https://www.geeksforgeeks.org/residual-sum-of-squares/#:~:text=Residual%20sum%20of%20squares%20is%20used%20to%20calculate%20the%20variance,squares%2C%20the%20better%20the%20model.)**(RSS): The** sum of squares of the residual for each data point in the plot or data is known as the residual sum of squares, or RSS. It is a measurement of the difference between the output that was observed and what was anticipated.  
  RSS=∑i=2n(yi−b0−b1xi)2*RSS*=∑*i*=2*n*​(*yi*​−*b*0​−*b*1​*xi*​)2
* **Total Sum of Squares (TSS):**The sum of the data points’ errors from the answer variable’s mean is known as the total sum of squares, or TSS.  
  TSS=∑(y−yi‾)2*TSS*=∑​(*y*−*yi*​​)2

R squared metric is a measure of the proportion of variance in the dependent variable that is explained the independent variables in the model.

**Adjusted R-Squared Error**

Adjusted R2 measures the proportion of variance in the dependent variable that is explained by independent variables in a regression model. [Adjusted R-square](https://www.geeksforgeeks.org/ml-adjusted-r-square-in-regression-analysis/) accounts the number of predictors in the model and penalizes the model for including irrelevant predictors that don’t contribute significantly to explain the variance in the dependent variables.

Mathematically, adjusted R2 is expressed as:

AdjustedR2=1–((1−R2).(n−1)n−k−1)*AdjustedR*2=1–(*n*−*k*−1(1−*R*2).(*n*−1)​)

Here,

* n is the number of observations
* k is the number of predictors in the model
* R2 is coeeficient of determination

Adjusted R-square helps to prevent overfitting. It penalizes the model with additional predictors that do not contribute significantly to explain the variance in the dependent variable.

While evaluation metrics help us measure the performance of a model, regularization helps in improving that performance by addressing overfitting and enhancing generalization.

What qualifies as a **"good" score (R²), MSE, MAE, or RMSE** depends on the context, such as the dataset, problem domain, and business requirements. However, here are some general guidelines and benchmarks:

**1. R² Score (Coefficient of Determination)**

* **Range:** 0≤R2≤10 \leq R² \leq 1
* **Interpretation:**
  + **0.7 or higher**: Good fit (explains 70%+ of the variance).
  + **0.5–0.7**: Moderate fit (may need improvement).
  + **< 0.5**: Poor fit (model needs better features or tuning).
* **Example:**  
  For house price prediction, an R² of 0.8 is good, as the features like square footage, number of rooms, and location typically explain a large portion of price variation.

**2. Mean Squared Error (MSE)**

* **Units:** Same as the square of the target variable's units.
* **Lower is Better:** Ideally as close to 0 as possible.
* **Interpretation:** MSE penalizes large errors more than small ones, so it's sensitive to outliers.
* **Domain-Specific Consideration:**
  + If predicting house prices (e.g., $300,000), an MSE in the range of 108−10910^8 - 10^9 could be reasonable.
  + For smaller targets like exam scores (e.g., out of 100), an MSE below 50 might be acceptable.

**3. Mean Absolute Error (MAE)**

* **Units:** Same as the target variable's units.
* **Lower is Better:** Ideally as close to 0 as possible.
* **Interpretation:** MAE gives the average magnitude of errors and is less sensitive to outliers than MSE.
* **Rule of Thumb:**
  + MAE should be **less than 10% of the average target value** for a good model.
  + Example: If predicting house prices with an average value of $300,000, a **MAE < $30,000** is reasonable.

**4. Root Mean Squared Error (RMSE)**

* **Units:** Same as the target variable's units.
* **Lower is Better:** Ideally close to 0.
* **Interpretation:** RMSE is similar to MSE but easier to interpret because it's in the same units as the target variable.
* **Domain-Specific Consideration:**
  + RMSE should ideally be close to the **standard deviation of the target variable**.
  + Example: For house prices, if the standard deviation of prices is $50,000, an RMSE close to $50,000 is reasonable.

**5. Relative Benchmarks (MAE/MSE/RMSE vs. Target)**

To get a better sense of whether your error metrics are good:

* **Relative Error:**  
  Relative MAE=MAEMean of Target×100%\text{Relative MAE} = \frac{\text{MAE}}{\text{Mean of Target}} \times 100 \%  
  Relative RMSE=RMSEMean of Target×100%\text{Relative RMSE} = \frac{\text{RMSE}}{\text{Mean of Target}} \times 100 \%
  + **Below 10%:** Great
  + **10–20%:** Acceptable
  + **Above 20%:** Poor

**For Your Case:**

Given your output:

* **R² Score (0.7567):** A score of ~0.76 is **good** for many regression problems. It means your model explains ~76% of the variance in the data.
* **MAE (4098.22):** If the average charges (target variable) is ~40,000, then the MAE is about **10% of the mean**, which is reasonable.
* **MSE (34,610,241):** High because it squares the errors. Focus more on RMSE or MAE for interpretability.
* **RMSE (5883.05):** Compare it with the standard deviation of charges. If it's close to or smaller than the standard deviation, it's acceptable.

**Improving the Metrics**

If you think the metrics could be better:

1. **Feature Engineering:** Add more relevant features or remove irrelevant ones.
2. **Regularization:** Use Lasso/Ridge to avoid overfitting.
3. **Hyperparameter Tuning:** Improve model performance by adjusting hyperparameters.
4. **Advanced Models:** Try boosting algorithms like XGBoost or LightGBM.

## Regularization Techniques

**Regularization techniques** are used in regression to **prevent overfitting**, where a model becomes too complex and starts memorizing the data instead of learning general patterns. These techniques add a penalty term to the loss function, discouraging the model from giving too much importance to any single feature.

### **1. Lasso Regression (L1 Regularization)**

* **What it does:** Adds a penalty proportional to the **absolute value of the coefficients**. This not only shrinks the coefficients but can force some to become exactly zero, effectively **selecting the most important features**.
* **Formula:**  
  Loss=MSE+λ∑i=1n∣βi∣Loss = \text{MSE} + \lambda \sum\_{i=1}^n |\beta\_i|Loss=MSE+λ∑i=1n​∣βi​∣
* **Real-Life Example:**  
  In the **house price prediction** scenario, if "fountain view" or "paint color" has negligible or no impact, Lasso will shrink their coefficients to zero, **removing them entirely** from the model. This simplifies the model and makes it interpretable.

### **2. Ridge Regression (L2 Regularization)**

* **What it does:** Adds a penalty proportional to the **square of the coefficients**. This shrinks the coefficients of less important features but doesn’t force them to zero.
* **Formula:**  
  Loss=MSE+λ∑i=1nβi2Loss = \text{MSE} + \lambda \sum\_{i=1}^n \beta\_i^2Loss=MSE+λ∑i=1n​βi2​  
  Where λ\lambdaλ is a tuning parameter (controls the strength of regularization), and βi\beta\_iβi​ are the feature coefficients.
* **Real-Life Example:**  
  Imagine you're predicting **house prices** based on 10 features (like area, number of rooms, location). If one feature, say "fountain view," is less important, Ridge will reduce its influence but not completely remove it. This keeps all features contributing but limits their impact.

### **3. Elastic Net (Combination of L1 & L2)**

* **What it does:** Combines the penalties of both Ridge (L2) and Lasso (L1). This is useful when you want the benefits of both approaches: feature selection (Lasso) and reduced multicollinearity (Ridge).
* **Formula:**  
  Loss=MSE+αλ∑i=1n∣βi∣+(1−α)λ∑i=1nβi2Loss = \text{MSE} + \alpha \lambda \sum\_{i=1}^n |\beta\_i| + (1 - \alpha) \lambda \sum\_{i=1}^n \beta\_i^2Loss=MSE+αλ∑i=1n​∣βi​∣+(1−α)λ∑i=1n​βi2​  
  α\alphaα: Balances the contribution of L1 and L2 regularization.
* **Real-Life Example:**  
  Elastic Net is great for datasets with many correlated features (e.g., predicting stock prices where multiple features are interrelated). It selects relevant features (like Lasso) while controlling for multicollinearity (like Ridge).

**Comparison with Real-Life Analogy**

Think of these techniques as controlling how much attention you give to each feature:

**Ridge**: Like gently turning down the volume on features that are too noisy but still important.

**Lasso**: Like muting features that contribute nothing valuable.

**Elastic Net**: Like combining the two approaches – lower the noise and completely remove irrelevant features if needed.

# Logistic Regression

**Logistic regression** is a **supervised machine learning algorithm**used for **classification tasks** where the goal is to predict the probability that an instance belongs to a given class or not.

Logistic regression is used for binary [classification](https://www.geeksforgeeks.org/getting-started-with-classification/) where we use [sigmoid function](https://www.geeksforgeeks.org/derivative-of-the-sigmoid-function/), that takes input as independent variables and produces a probability value between 0 and 1.

For example, we have two classes Class 0 and Class 1 if the value of the logistic function for an input is greater than 0.5 (threshold value) then it belongs to Class 1 otherwise it belongs to Class 0.

It’s referred to as regression because it is the extension of[linear regression](https://www.geeksforgeeks.org/ml-linear-regression/) but is mainly used for classification problems.

In Logistic regression, instead of fitting a regression line, we fit an “S” shaped logistic function, which predicts two maximum values (0 or 1).

**Types of Logistic Regression**

On the basis of the categories, Logistic Regression can be classified into three types:

1. **Binomial:** In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.
2. **Multinomial:** In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as “cat”, “dogs”, or “sheep”
3. **Ordinal:**In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as “low”, “Medium”, or “High”.

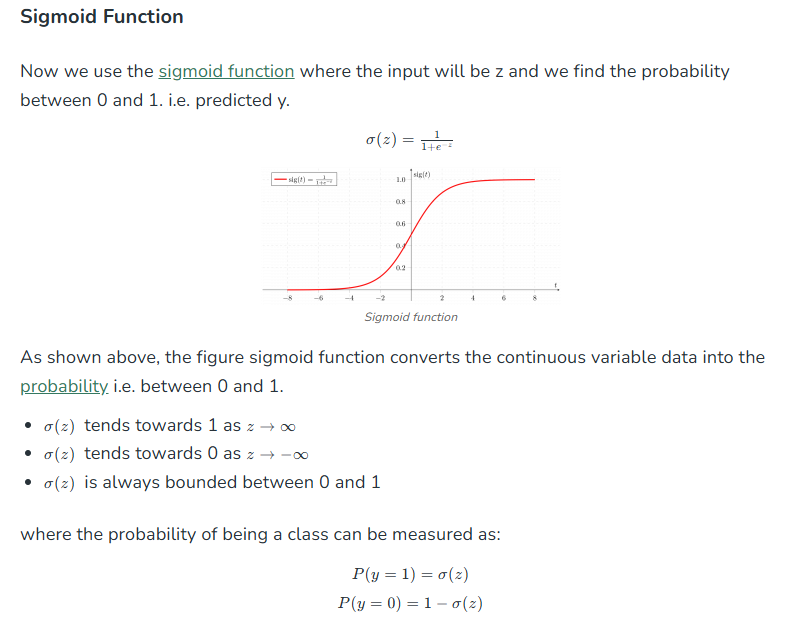
**Understanding Sigmoid Function**

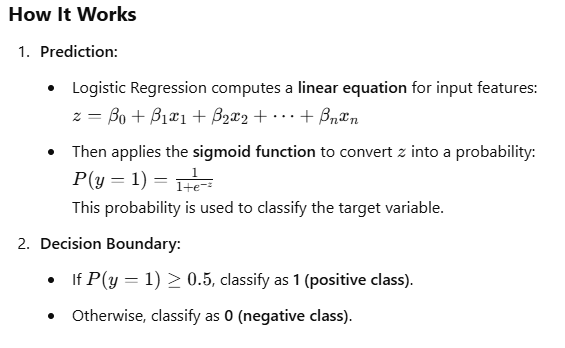
So far, we’ve covered the basics of logistic regression, but now let’s focus on the most important function that forms the core of logistic regression**.**

* The **sigmoid function**is a mathematical function used to map the predicted values to probabilities.
* It maps any real value into another value within a range of**0 and 1**. The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the “**S**” form.
* The S-form curve is called the **Sigmoid function or the logistic function.**
* In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

## **How does Logistic Regression work?**

The logistic regression model transforms the [linear regression](https://www.geeksforgeeks.org/ml-linear-regression/) function continuous value output into categorical value output using a sigmoid function, which maps any real-valued set of independent variables input into a value between 0 and 1. This function is known as the logistic function.





**Use Cases**

* **Binary Classification:** Spam detection, credit card fraud, disease diagnosis (e.g., predicting diabetes: Yes/No).
* **Multiclass Classification (Extension):** Logistic Regression can be extended using **One-vs-Rest (OvR)** or **Softmax Regression** for problems with multiple classes.

**Key Metrics**

Logistic Regression focuses on classification metrics rather than error metrics:

1. **Accuracy:** Percentage of correct predictions.
2. **Precision:** True Positives / (True Positives + False Positives). Focuses on minimizing false alarms.
3. **Recall (Sensitivity):** True Positives / (True Positives + False Negatives). Focuses on finding all positive cases.
4. **F1-Score:** Harmonic mean of precision and recall.
5. **Confusion Matrix:** Summarizes predictions as True Positives, True Negatives, False Positives, False Negatives.

# Decision Trees

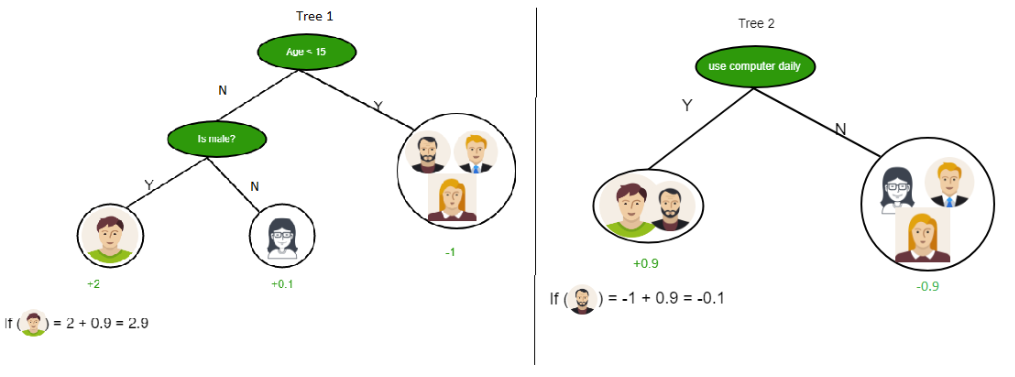
A decision tree is a [supervised learning](https://www.geeksforgeeks.org/supervised-machine-learning/) algorithm used for both [classification](https://www.geeksforgeeks.org/getting-started-with-classification/) and [regression](https://www.geeksforgeeks.org/regression-in-machine-learning/) tasks. It models decisions as a **tree-like structure** where internal nodes represent **attribute tests**, branches represent **attribute values**, and leaf nodes represent **final decisions or predictions**. Decision trees are versatile, interpretable, and widely used in machine learning for predictive modeling.

**Intuition behind the Decision Tree**

Here’s an example to make it simple to understand the intuition of decision tree:

Imagine you’re deciding whether to buy an umbrella:

1. **Step 1 – Ask a Question (Root Node):**  
   *Is it raining?*  
   If yes, you might decide to buy an **umbrella**. If no, you move to the next question.
2. **Step 2 – More Questions (Internal Nodes):**  
   If it’s not raining, you might ask:  
   *Is it likely to rain later?*  
   If yes, you buy an umbrella; if no, you don’t.
3. **Step 3 – Decision (Leaf Node):**  
   Based on your answers, you either buy or skip the umbrella



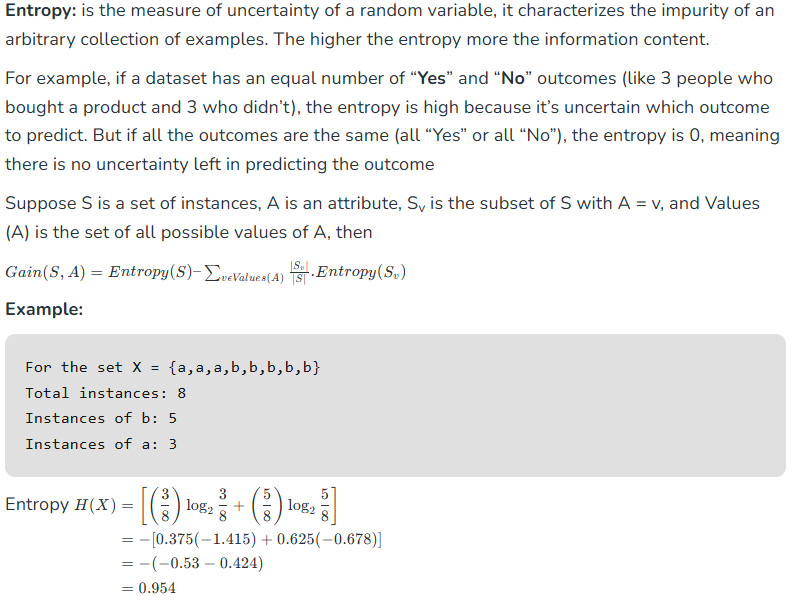
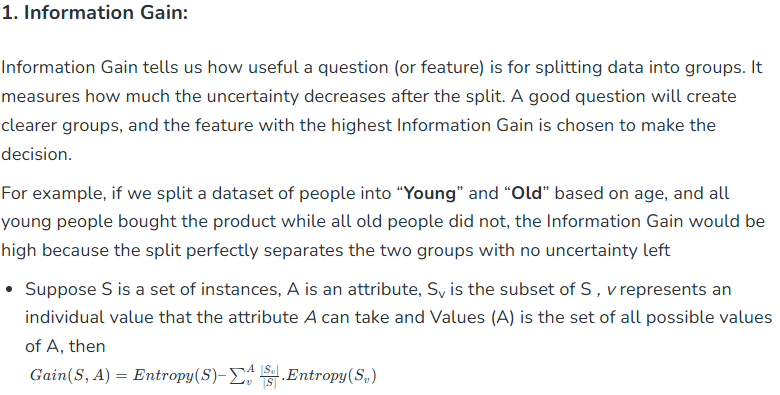
Examples shown above include **Predicting Whether a Person Likes Computer Games, Predicting Whether a Person Likes Computer Games Using Two Decision Trees**

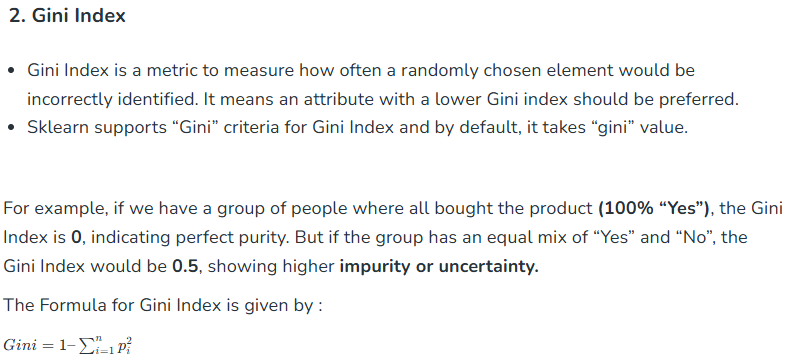
**Combining Trees: Final Prediction**

The final prediction score is the sum of scores from both trees

We have two popular attribute selection measures used:

1. **1. Information Gain**
2. **2. Gini Index**



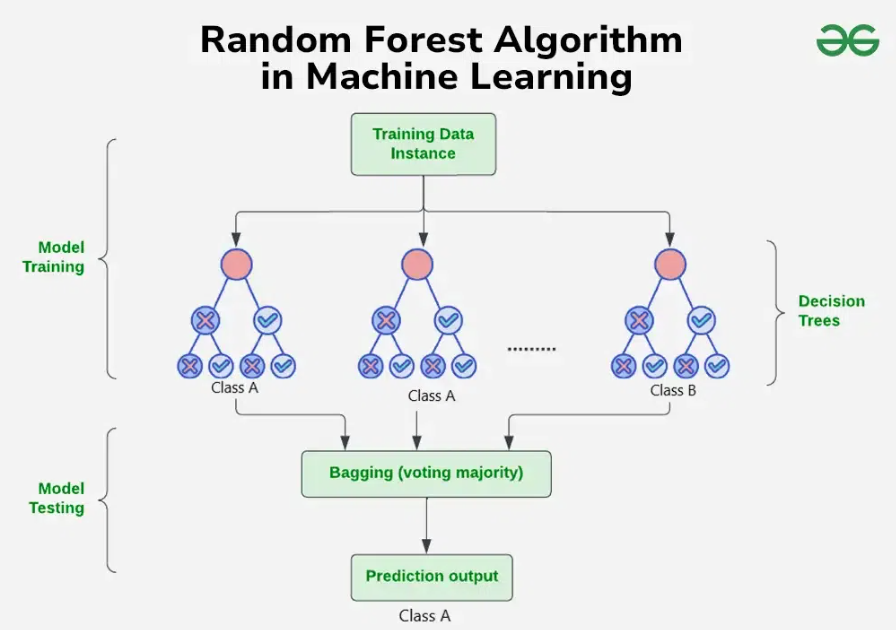


# Random Forest

Random Forest algorithm is a powerful tree learning technique in Machine Learning to make predictions and **then we do voting of all the tress to make prediction**. They are widely used for classification and regression task.

* It is a type of classifier that uses many decision trees to make predictions.
* It takes different random parts of the dataset to train each tree and then it combines the results by averaging them. This approach helps improve the accuracy of predictions. Random Forest is based on [ensemble learning](https://www.geeksforgeeks.org/a-comprehensive-guide-to-ensemble-learning/).

Imagine asking a group of friends for advice on where to go for vacation. Each friend gives their recommendation based on their unique perspective and preferences (decision trees trained on different subsets of data). You then make your final decision by considering the majority opinion or averaging their suggestions (ensemble prediction).



**As explained in image:**Process starts with a dataset with rows and their corresponding class labels (columns).

* Then - Multiple Decision Trees are created from the training data. Each tree is trained on a random subset of the data (with replacement) and a random subset of features. This process is known as **bagging** or **bootstrap aggregating**.
* Each Decision Tree in the ensemble learns to make predictions independently.
* When presented with a new, unseen instance, each Decision Tree in the ensemble makes a prediction.

The final prediction is made by combining the predictions of all the Decision Trees. This is typically done through a majority vote (for classification) or averaging (for regression).

**Key Features of Random Forest**

* **Handles Missing Data**: Automatically handles missing values during training, eliminating the need for manual imputation.
* Algorithm ranks **features based on their importance in making predictions** offering valuable insights for feature selection and interpretability.
* **Scales Well with Large and Complex Data** without significant performance degradation.
* Algorithm is versatile and can be applied to both classification tasks (e.g., predicting categories) and regression tasks (e.g., predicting continuous values).

# Support Vector Machine

Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression tasks. While it can handle regression problems, SVM is particularly well-suited for classification tasks.

SVM aims to find the optimal hyperplane in an N-dimensional space to separate data points into different classes. The algorithm maximizes the margin between the closest points of different classes.

**Support Vector Machine (SVM) Terminology**

* **Hyperplane**: A decision boundary separating different classes in feature space, represented by the equation**wx + b = 0** in linear classification.
* **Support Vectors**: The closest data points to the hyperplane, crucial for determining the hyperplane and margin in SVM.
* **Margin**: The distance between the hyperplane and the support vectors. SVM aims to maximize this margin for better classification performance.
* **Kernel**: A function that maps data to a higher-dimensional space, enabling SVM to handle non-linearly separable data.
* **Hard Margin**: A maximum-margin hyperplane that perfectly separates the data without misclassifications.
* **Soft Margin**: Allows some misclassifications by introducing slack variables, balancing margin maximization and misclassification penalties when data is not perfectly separable.
* **C**: A regularization term balancing margin maximization and misclassification penalties. A higher C value enforces a stricter penalty for misclassifications.
* **Hinge Loss**: A loss function penalizing misclassified points or margin violations, combined with regularization in SVM.
* **Dual Problem**: Involves solving for Lagrange multipliers associated with support vectors, facilitating the kernel trick and efficient computation.

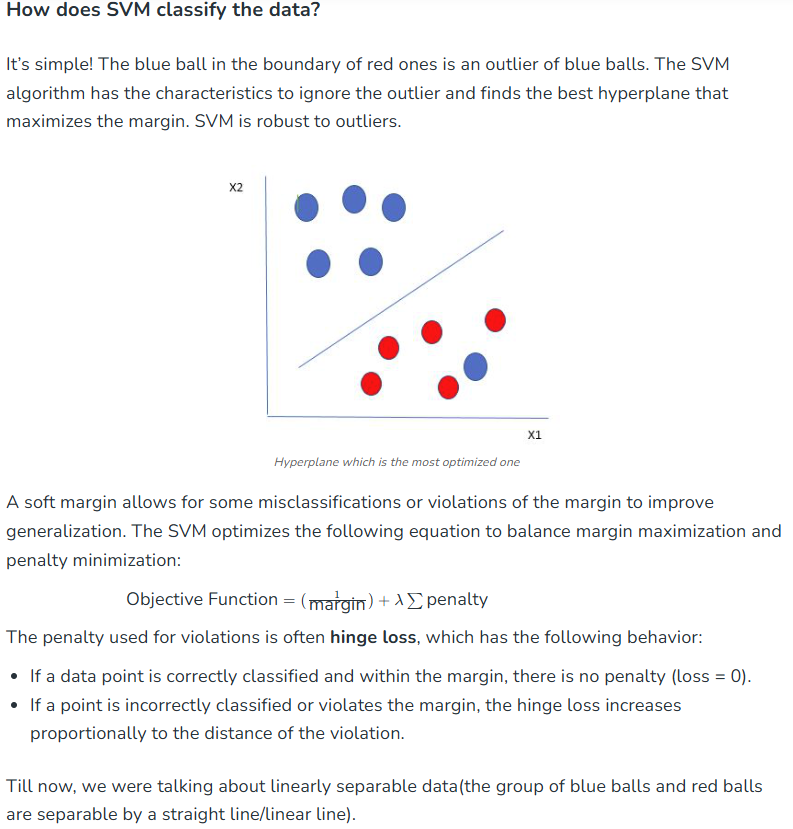
## How does Support Vector Machine Algorithm Work?

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



*Multiple hyperplanes separate the data from two classes*

The best hyperplane, also known as the **“hard margin,”** is the one that maximizes the distance between the hyperplane and the nearest data points from both classes. This ensures a clear separation between the classes. So, from the above figure, we choose L2 as hard margin.



**What to do if data are not linearly separable?**

When data is not linearly separable (i.e., it can’t be divided by a straight line), SVM uses a technique called **kernels** to map the data into a higher-dimensional space where it becomes separable. This transformation helps SVM find a decision boundary even for non-linear data.



*Original 1D dataset for classification*

A **kernel** is a function that maps data points into a higher-dimensional space without explicitly computing the coordinates in that space. This allows SVM to work efficiently with non-linear data by implicitly performing the mapping.

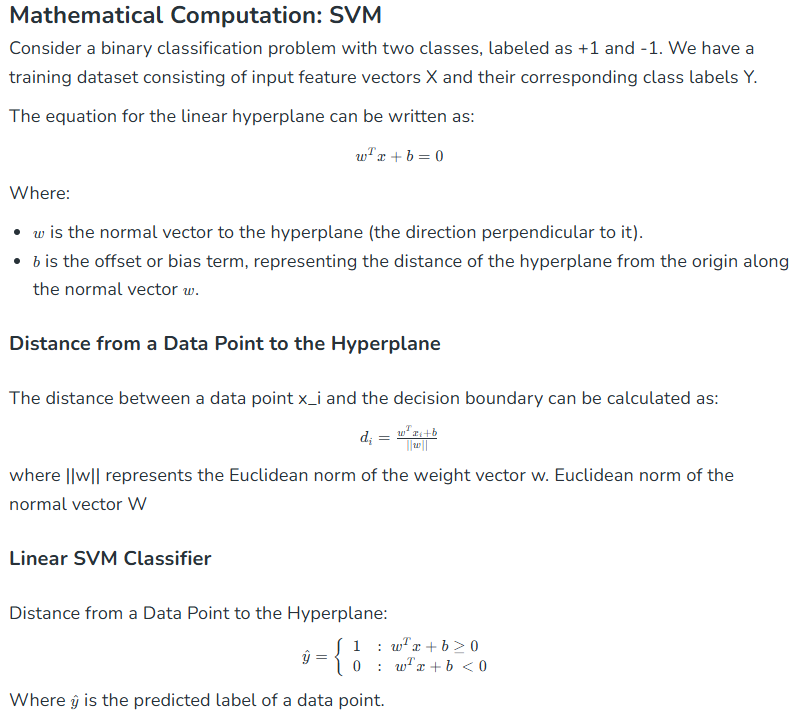
For example, consider data points that are not linearly separable. By applying a kernel function, SVM transforms the data points into a higher-dimensional space where they become linearly separable.

* **Linear Kernel**: For linear separability.
* **Polynomial Kernel**: Maps data into a polynomial space.
* **Radial Basis Function (RBF) Kernel**: Transforms data into a space based on distances between data points.



*Mapping 1D data to 2D to become able to separate the two classes*

In this case, the new variable y is created as a function of distance from the origin.



**Types of Support Vector Machine**

Based on the nature of the decision boundary, Support Vector Machines (SVM) can be divided into two main parts:

* **Linear SVM:**Linear SVMs use a linear decision boundary to separate the data points of different classes. When the data can be precisely linearly separated, linear SVMs are very suitable. This means that a single straight line (in 2D) or a hyperplane (in higher dimensions) can entirely divide the data points into their respective classes. A hyperplane that maximizes the margin between the classes is the decision boundary.
* **Non-Linear SVM:** Non-Linear SVM can be used to classify data when it cannot be separated into two classes by a straight line (in the case of 2D). By using kernel functions, nonlinear SVMs can handle nonlinearly separable data. The original input data is transformed by these kernel functions into a higher-dimensional feature space, where the data points can be linearly separated. A linear SVM is used to locate a nonlinear decision boundary in this modified space.