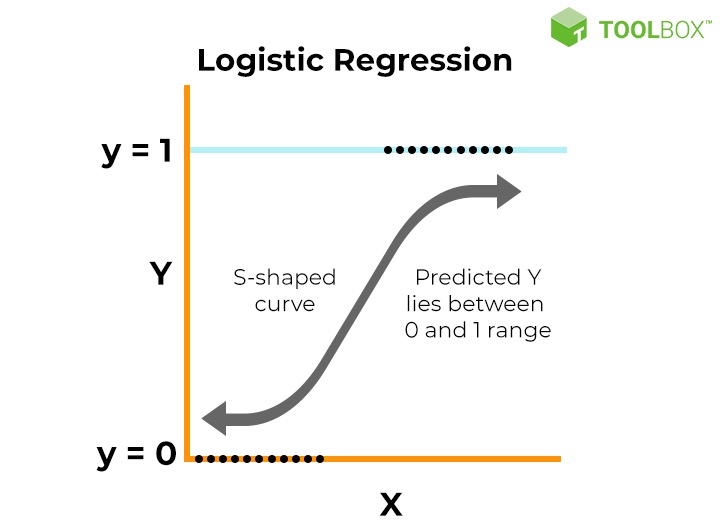
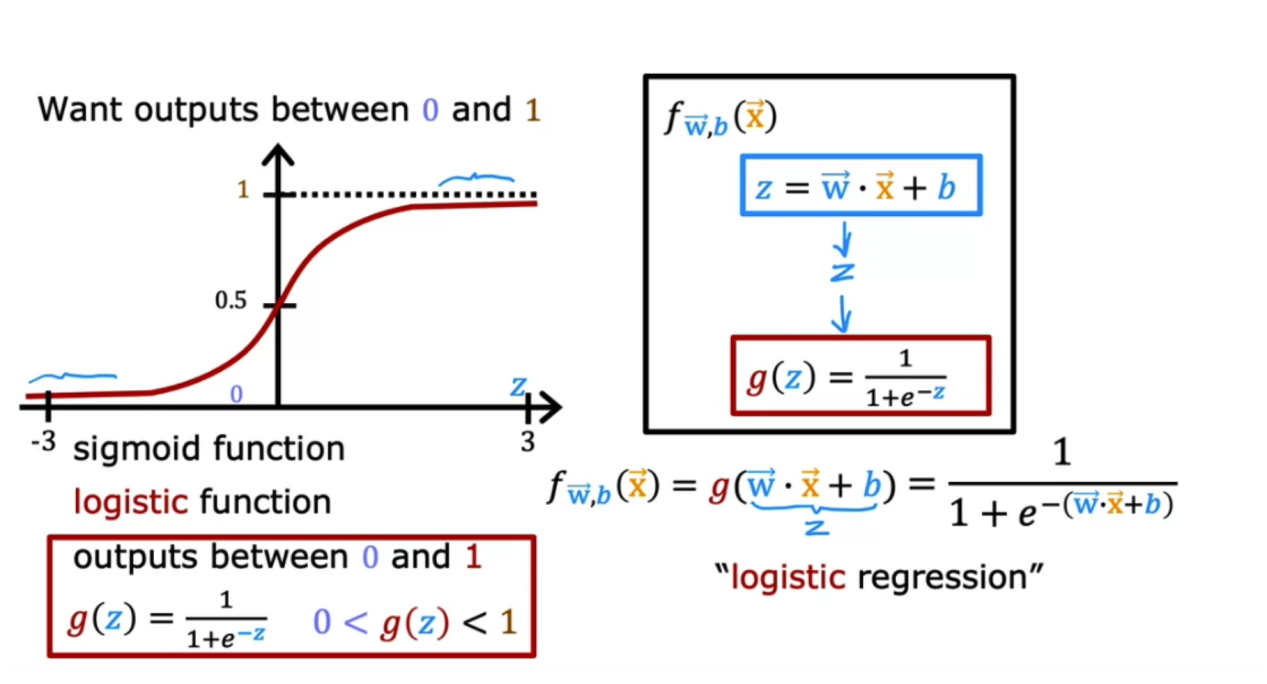
**Machine Learning Algorithms**

**Logistic Regression:**

* Logistic Regression is supervised machine learning algorithm.
* It is primarily used for binary classification problems, where the outcome variable is categorical and has two classes but It can be used for multiple classification problems as well.



**Model Formulation:**



**Training the Model:**

* Start with an initial guess for the line (decision boundary) represented by the coefficients wi and b.
* Use the logistic function to make predictions for each data point based on the current line.
* Assess how well the current line fits the observed data by comparing predictions to the actual outcomes.
* Shift the line (update wi and b) to improve the fit. This is done by finding the line that maximizes the likelihood of observing the given outcomes.
* Repeat Steps 2-4 until you find a line that maximizes the likelihood of the observed data.

**\*** The goal of maximum likelihood is to find the optimal way to fit a distribution to the data.

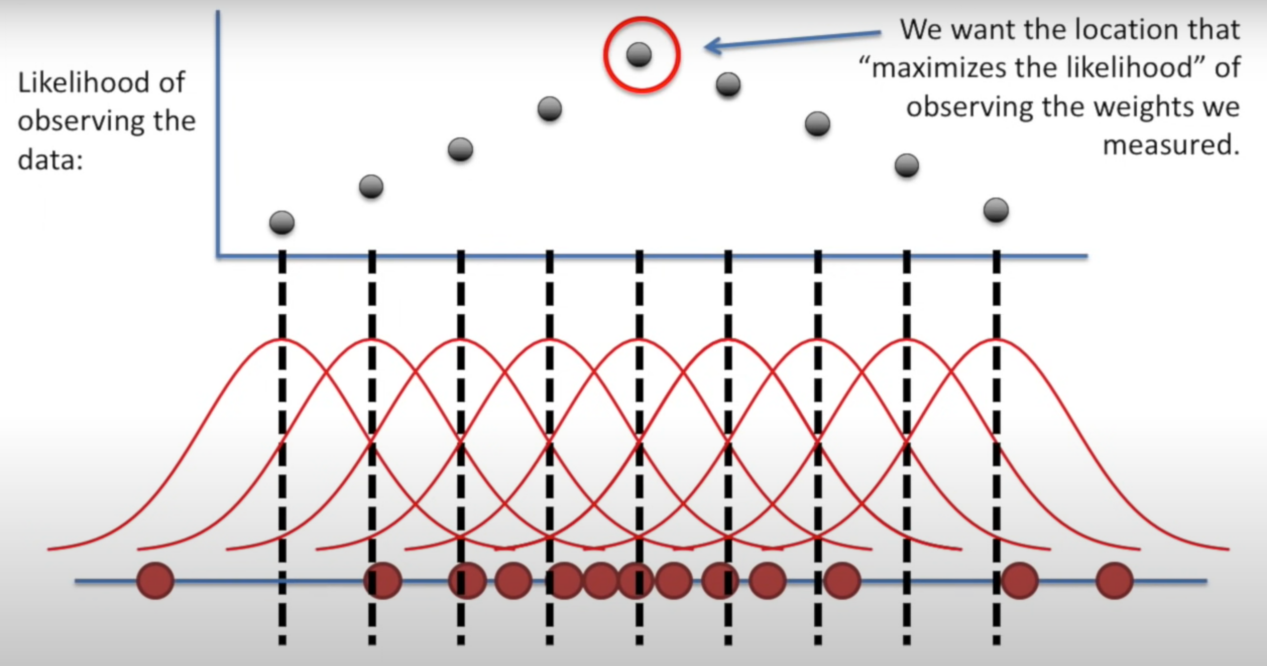
**Decision Boundary:**

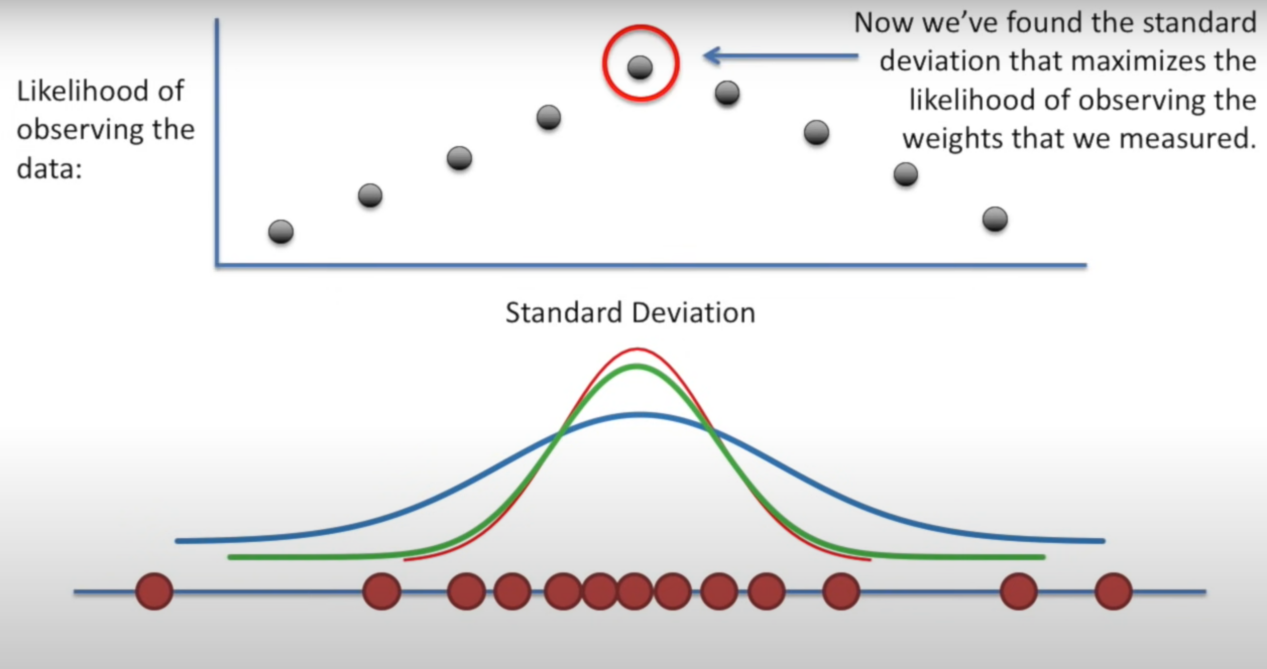
A decision boundary is established based on a threshold probability (commonly 0.5). If the predicted probability is greater than or equal to the threshold, the instance is classified as class 1; otherwise, it is classified as class 0.

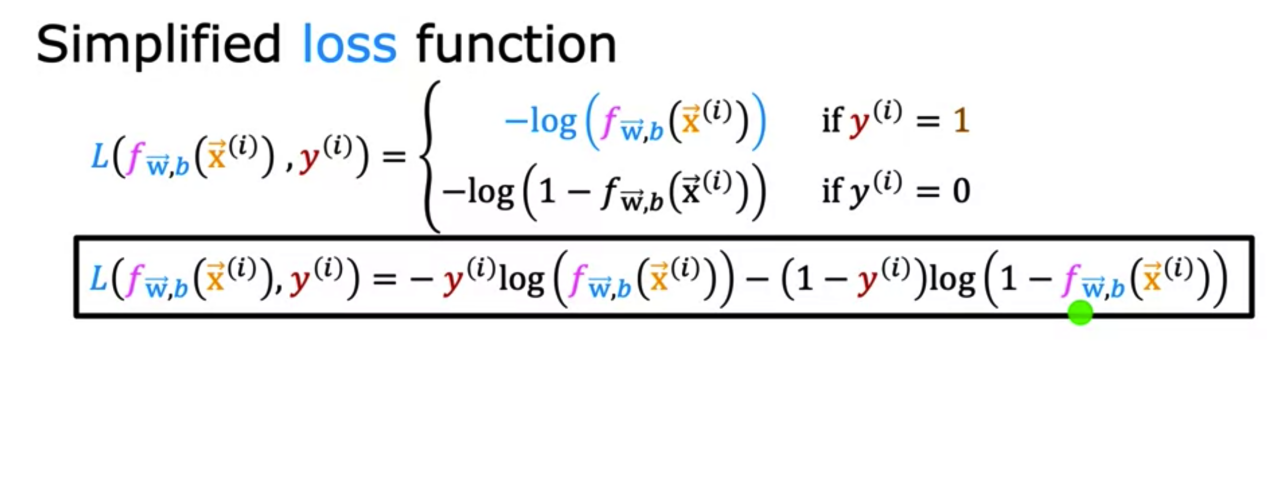
**Cost Function (Log Loss/Cross Entropy):**

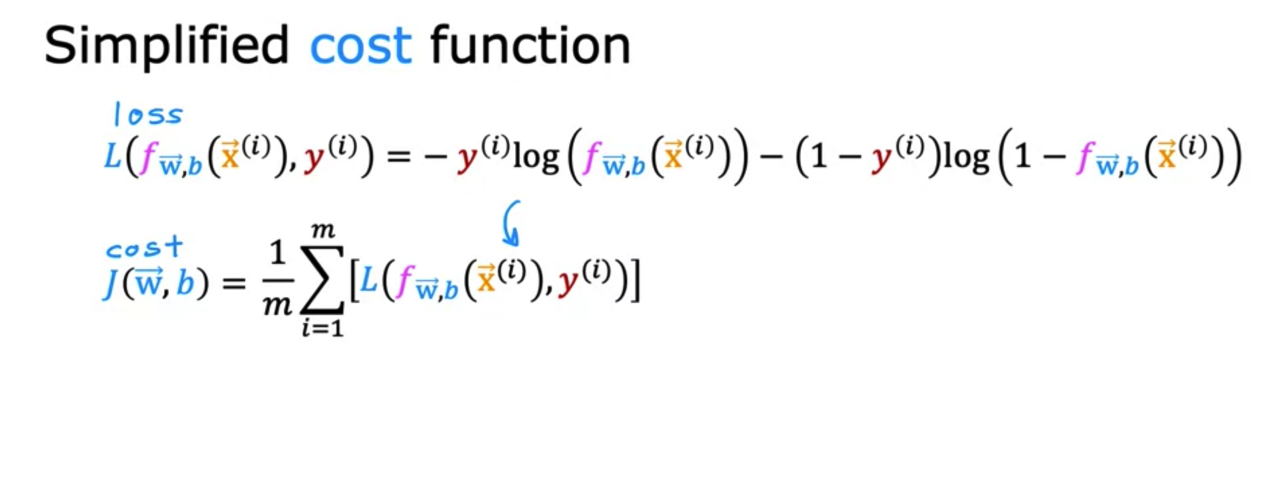
* Log loss uses the maximum likelihood concept.
* The goal of the maximum likelihood is to find the optimal way to fit the distribution to the data.

**Maximum Likehood Example:**





****

****

**Applications:**

Logistic Regression is widely used in various fields for binary classification tasks, such as spam detection, credit scoring, and medical diagnosis.

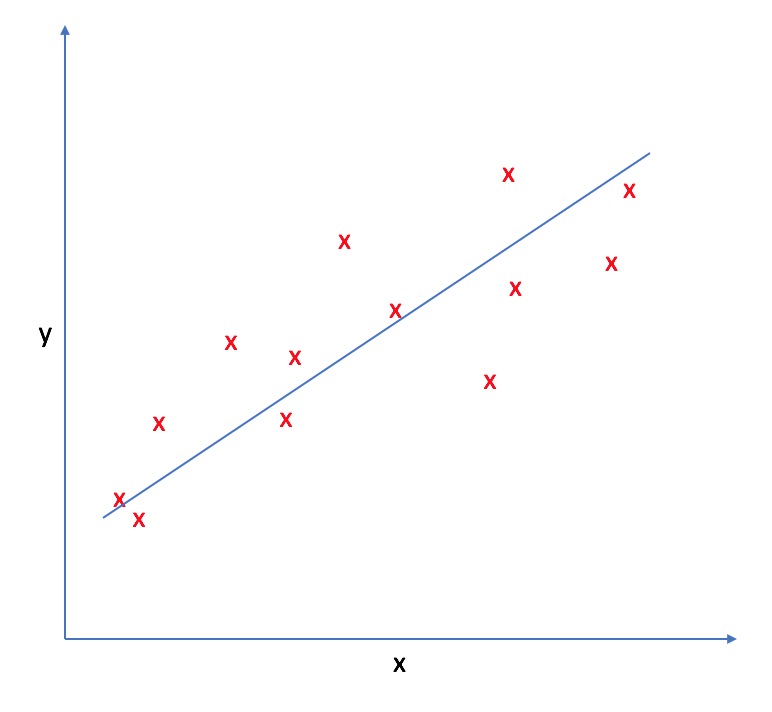
**Logistic regression for multi class classification:**

Yes, logistic regression can be extended for multi class classification problems. There are two main approaches to achieve this:

* **One-vs-Rest (OvR):** Also known as One-vs-Rest, this approach involves training a separate binary classifier for each class. Each classifier predicts whether an instance belongs to a specific class or not. During prediction, the class with the highest confidence score (probability) is chosen as the final class. This method is simple and works well when the number of classes is not too large.
* **Softmax Regression (Multinomial Logistic Regression):** This approach generalizes logistic regression to multiple classes directly. Instead of using the logistic (sigmoid) function, it uses the Softmax function to model the probabilities of each class. The Softmax function ensures that the sum of the predicted probabilities for all classes is equal to 1. This method is more efficient and often performs better for multi class classification tasks.

**Linear Regression:**

* Linear Regression is supervised machine learning algorithm.
* It is used when the relationship between the dependent variable (output) and the independent variable(s) (input) is assumed to be linear.
* It is used for regression problems.



**Working:**

A linear regression model predicts a dependent variable Y based on one or more independent variables X.

The relationship is assumed to be linear and is represented by the equation:

**Y = b0+b 1X**

**Initialize Parameters:**

Start with initial guesses for the parameters b0 (intercept) and b1(slope). These can be zeros or random values.

b0 = 0

b1 = 0

**Calculate Predictions:**

For each data point, compute the predicted value

**Yi = b0+b1Xi**

​

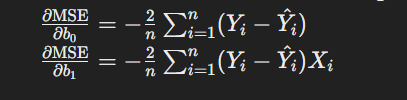
**Compute the Cost Function**

The most common cost function used is the Mean Squared Error (MSE), which measures the average of the squares of the errors (the difference between the actual and predicted values).

Screenshot 2024-07-22 215624

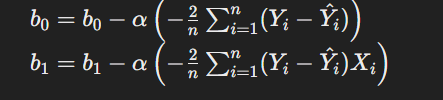
**Calculate Gradients**

Compute the gradients of the cost function with respect to the parameters. These gradients indicate the direction and rate of change of the cost function.



**Update Parameters:**

Adjust the parameters in the direction that reduces the cost function. This is done using the gradients and a learning rate α (a small positive number that controls the step size).



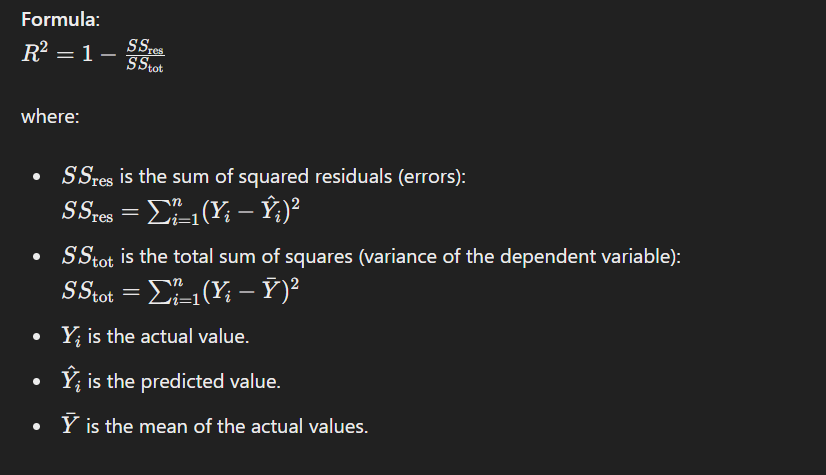
**Repeat Until Convergence:**

Repeat steps 2 to 5 until the parameters converge (i.e., the change in the cost function is very small) or for a fixed number of iterations.

**R-Squared (Coefficient of Determination):**

R-squared is a statistical measure that represents the proportion of the variance in the dependent variable (Y) that is predictable from the independent variable(s) (X).

It provides an indication of the goodness of fit of the regression model.



**Interpretation:**

* R2 ranges from 0 to 1.
* An R2 value of 0 indicates that the model does not explain any of the variability of the response data around its mean.
* An R2 value of 1 indicates that the model explains all the variability of the response data around its mean.
* A higher R2 value indicates a better fit of the model to the data.

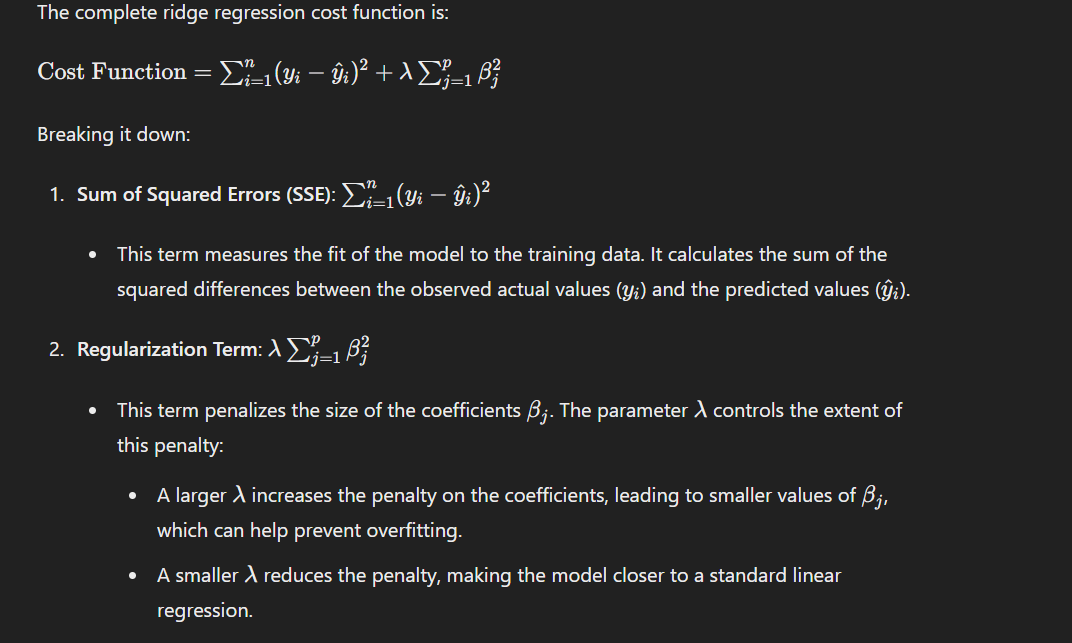
**Applications:**

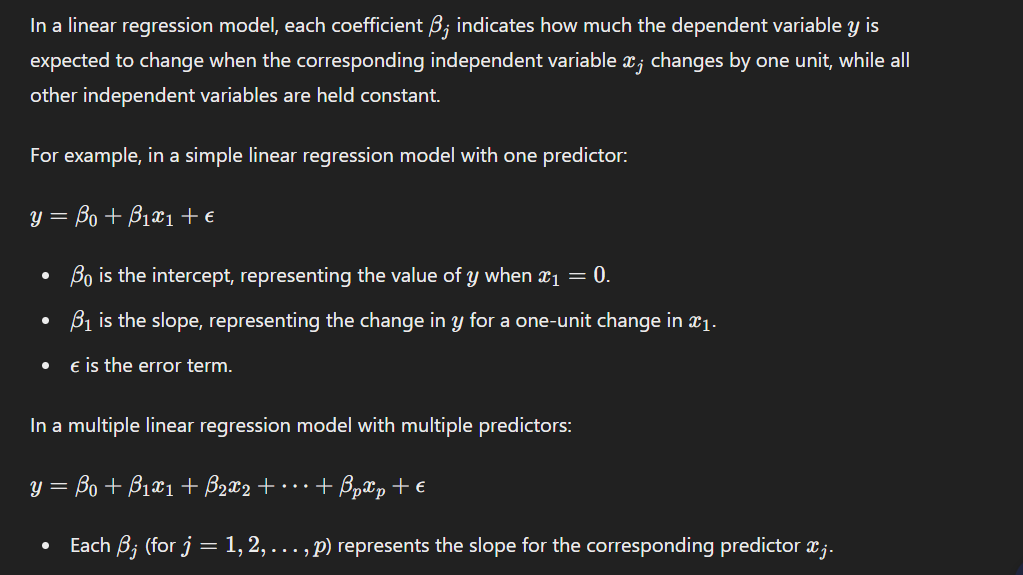
Linear Regression is widely used in various fields for tasks such as predicting house prices, stock prices, sales, and many other situations where understanding and predicting a continuous outcome are essential.

**Ridge regression (L2):**

Ridge regression is a type of linear regression that includes a regularization term in the cost function. This regularization term is the sum of the squared values of the coefficients (also known as L2 regularization).

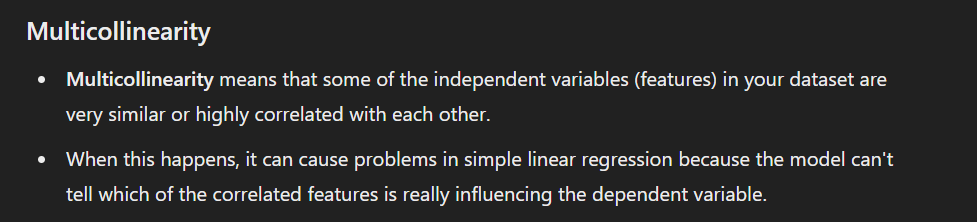
The ridge regression cost function is given by:





**Why Do We Need Ridge Regression if We Have Simple Linear Regression?**

**Multicollinearity:** In datasets where independent variables are highly correlated, simple linear regression can produce large variance in the coefficient estimates. Ridge regression helps by adding a penalty on the size of the coefficients, thus shrinking them and making the estimates more stable.



**Over fitting:** Simple linear regression can over fit the training data, especially when the number of predictors is large relative to the number of observations. Ridge regression reduces the risk of over fitting by imposing a penalty on the size of the coefficients, which helps in achieving better generalization to new data.

**Complexity Control:** Ridge regression controls the complexity of the model by penalizing large coefficients. This can result in models that are simpler and easier to interpret compared to models produced by simple linear regression.

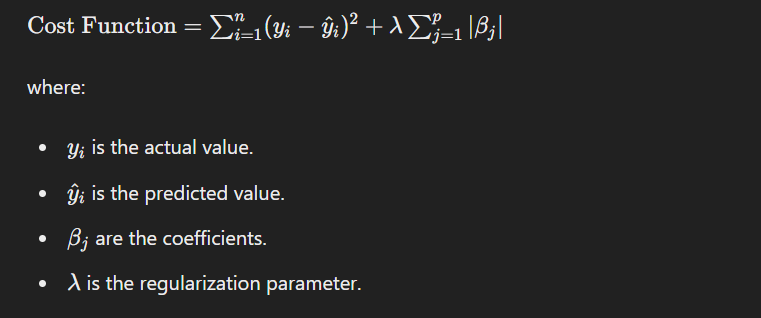
**Bias-Variance Trade-off:** By adding a regularization term, ridge regression introduces some bias into the model, but this can lead to a reduction in variance. This trade-off can result in a model that performs better on new, unseen data compared to a simple linear regression model that has low bias but high variance.

**Lasso regression (L1 Regularization):**

Lasso regression is another type of linear regression that, like ridge regression, includes a regularization term. However, unlike ridge regression, which uses an L2 penalty (sum of squared coefficients), lasso regression uses an L1 penalty (sum of absolute values of coefficients).

**Lasso Regression Formula:**

The cost function for lasso regression is:



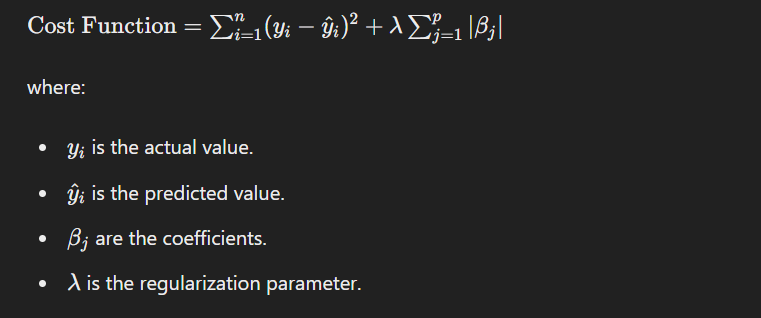
**Key Characteristics of Lasso Regression:**

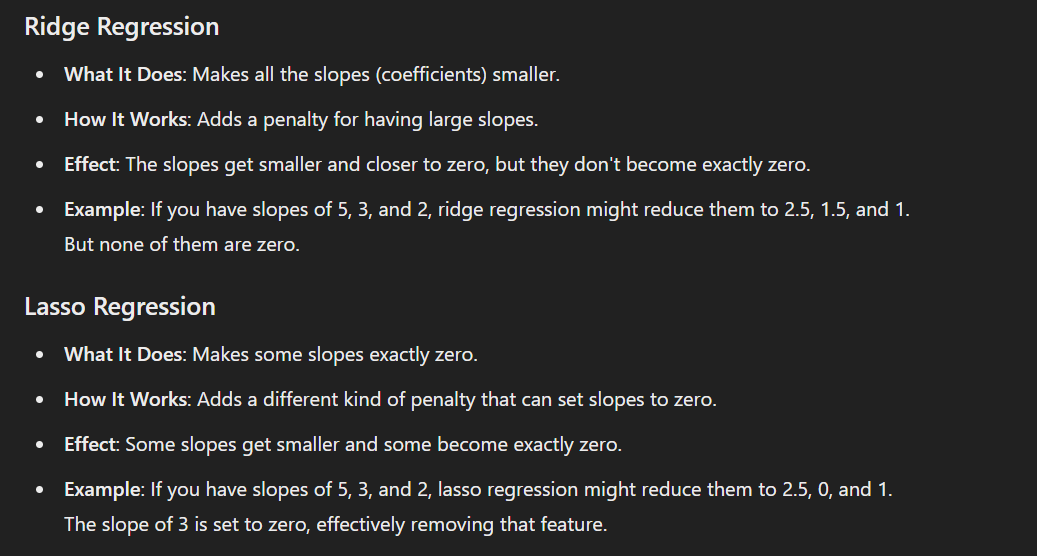
**Feature Selection:** One of the most notable properties of lasso regression is that it can set some coefficients to exactly zero. This means it can effectively select a simpler model by including only a subset of the features. This makes lasso particularly useful for feature selection in high-dimensional datasets.

**Bias and Variance:** By adding the L1 penalty, lasso regression introduces some bias into the model, which can help to reduce variance. This trade-off can lead to better generalization on unseen data.

**Handling Multicollinearity:** Like ridge regression, lasso can handle multicollinearity by shrinking the coefficients. However, because it can set some coefficients to zero, it can also help to identify which variables are most important.

**Comparison with Ridge Regression:**





**Principal Component Analysis (PCA):**

Principal Component Analysis (PCA) is a technique used to reduce the number of features (dimensions) in a dataset while retaining as much information as possible. Here's a simple explanation of PCA and how it reduces features:

**What is PCA?**

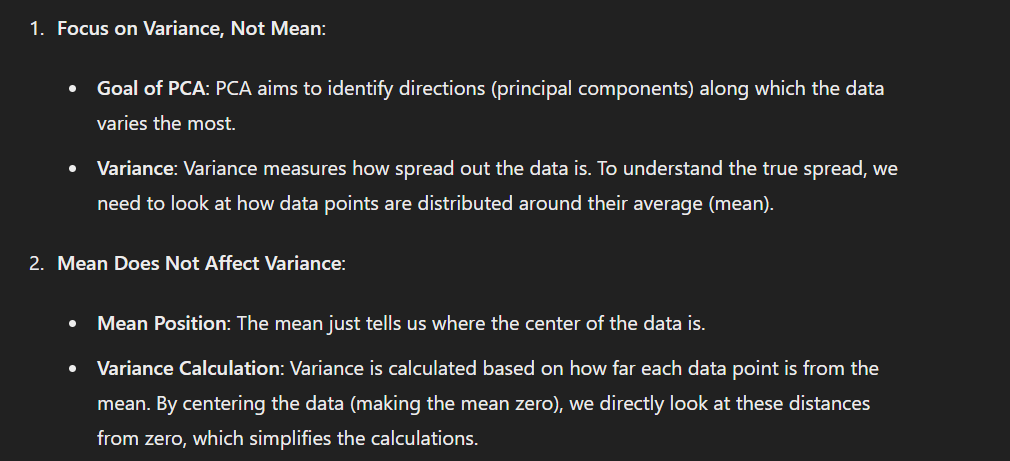
PCA is a dimensionality reduction technique that transforms a dataset into a new coordinate system. In this new system:

The first axis (principal component) captures the most variance (spread) in the data.

The second axis captures the second most variance, and so on.

**How PCA Works:**

1. **Center the Data:** Subtract the mean of each feature from the dataset so that the data is centered around the origin. This step ensures that the PCA analysis is not biased by the scale or mean of the data.



1. **Calculate the Covariance Matrix:** This matrix measures how features in the dataset vary with each other. The covariance matrix helps in understanding the relationship between different features.

Covariance measures how two features vary together.

**Positive Covariance:** If one feature increases as the other feature increases, the covariance is positive.

**Negative Covariance:** If one feature increases while the other feature decreases, the covariance is negative.

**Zero Covariance:** No relationship between the features.

1. **Compute the Eigenvalues and Eigenvectors:**

In PCA, eigenvalues and eigenvectors are mathematical concepts that help identify the directions (principal components) in which the data varies the most.

**1. Eigenvectors**

Eigenvectors are special directions in the feature space. When you project your data onto these directions, the variance (spread) of the data along these directions is maximized.

In PCA: Each eigenvector represents a principal component, which is a direction in the transformed space where the data shows the most variance.

**2. Eigenvalues**

Eigenvalues measure the amount of variance captured by each eigenvector. Essentially, they tell you how important each direction (eigenvector) is.

In PCA: Each eigenvalue corresponds to an eigenvector and indicates how much variance is captured in the direction of that eigenvector. Larger eigenvalues mean more variance.

1. **Sort and Select Principal Components:**

Sort the eigenvectors by their corresponding eigenvalues in descending order.

The eigenvectors with the largest eigenvalues capture the most variance.

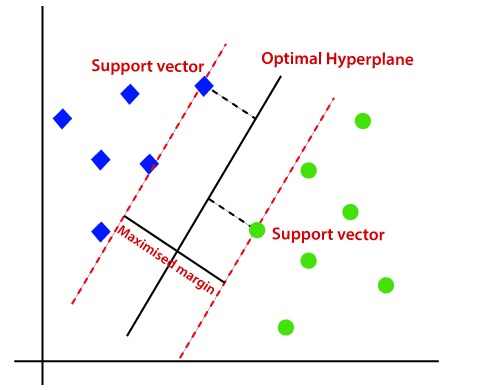
Choose the top k eigenvectors that correspond to the largest eigenvalues.

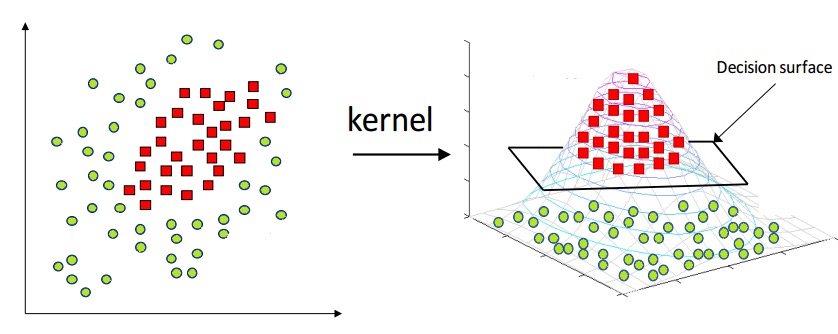
These top k eigenvectors form the new feature space.

1. **Transform the Data:** Project the original data onto the new feature space defined by the selected eigenvectors. This reduces the number of dimensions from p to k.

**Support Vector Machine (SVM):**

* A Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression tasks.
* It's particularly effective in high-dimensional spaces and is widely used in various applications, including image classification, text classification, and bioinformatics.





**Key Concepts:**

**Linear Separation:**

SVM aims to find the optimal hyperplane that best separates data points of different classes in the feature space.

**Hyperplane:**

In a two-dimensional space, the hyperplane is a line; in three-dimensional space, it's a plane, and so on.

**Support Vectors:**

Support vectors are the data points that lie closest to the decision boundary (hyperplane). They are crucial in defining the hyperplane and optimizing its position.

**Margin:**

The margin is the distance between the hyperplane and the nearest data point from either class. SVM aims to maximize this margin, which helps improve the model's generalization to unseen data.

**Kernel Trick:**

SVM can handle non-linear decision boundaries through the kernel trick. Kernels transform the input features into a higher-dimensional space, making it possible to find a linear hyperplane in that space.

**Types of SVM:**

**Linear SVM:** It is used for linearly separable data, which means if a dataset can be classified into two classes by single straight line.

**Non-linear SVM:** Non-linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line.

**Advantages:**

* Effective in high-dimensional spaces.
* Memory efficient (uses only a subset of training points as support vectors).
* Versatile through the use of different kernel functions.

**Limitations:**

* Sensitivity to the choice of the kernel (linear, polynomial ans so on) and hyperparameters.
* Computationally intensive, especially for large datasets.

**Decision Tree:**

* A Decision Tree is a versatile supervised machine learning algorithm used for both classification and regression tasks.
* It's a tree-like structure where each node represents a decision based on a specific feature, leading to subsequent nodes (branches) that represent possible outcomes.
* The leaves of the tree represent the final decisions or predictions.

**Advantages:**

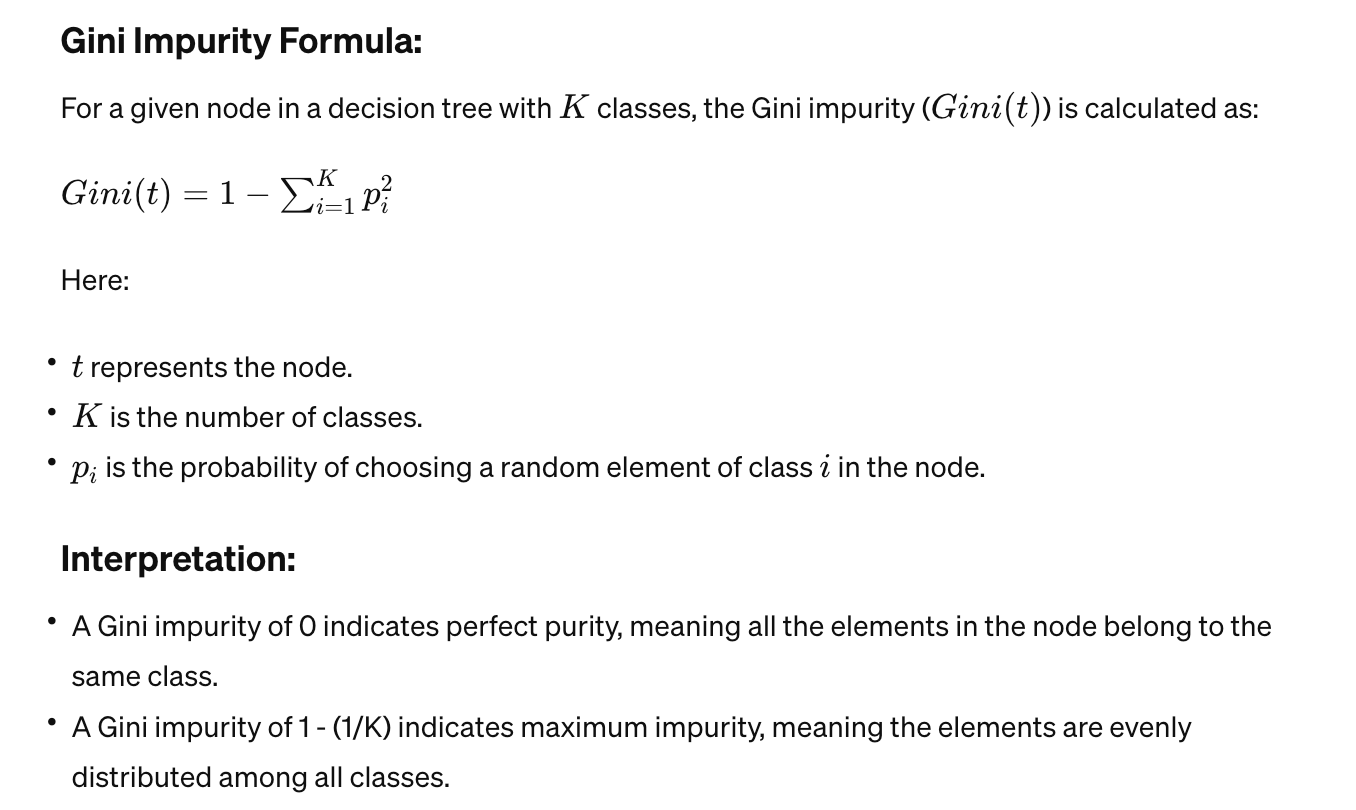
* Decision trees are easy to interpret and visualize, making them useful for explaining decisions.
* They can model complex relationships and interactions between features.

**Limitations:**

* Decision trees are prone to overfitting, especially when the tree depth is not appropriately controlled.
* Small variations in the data can lead to different tree structures.

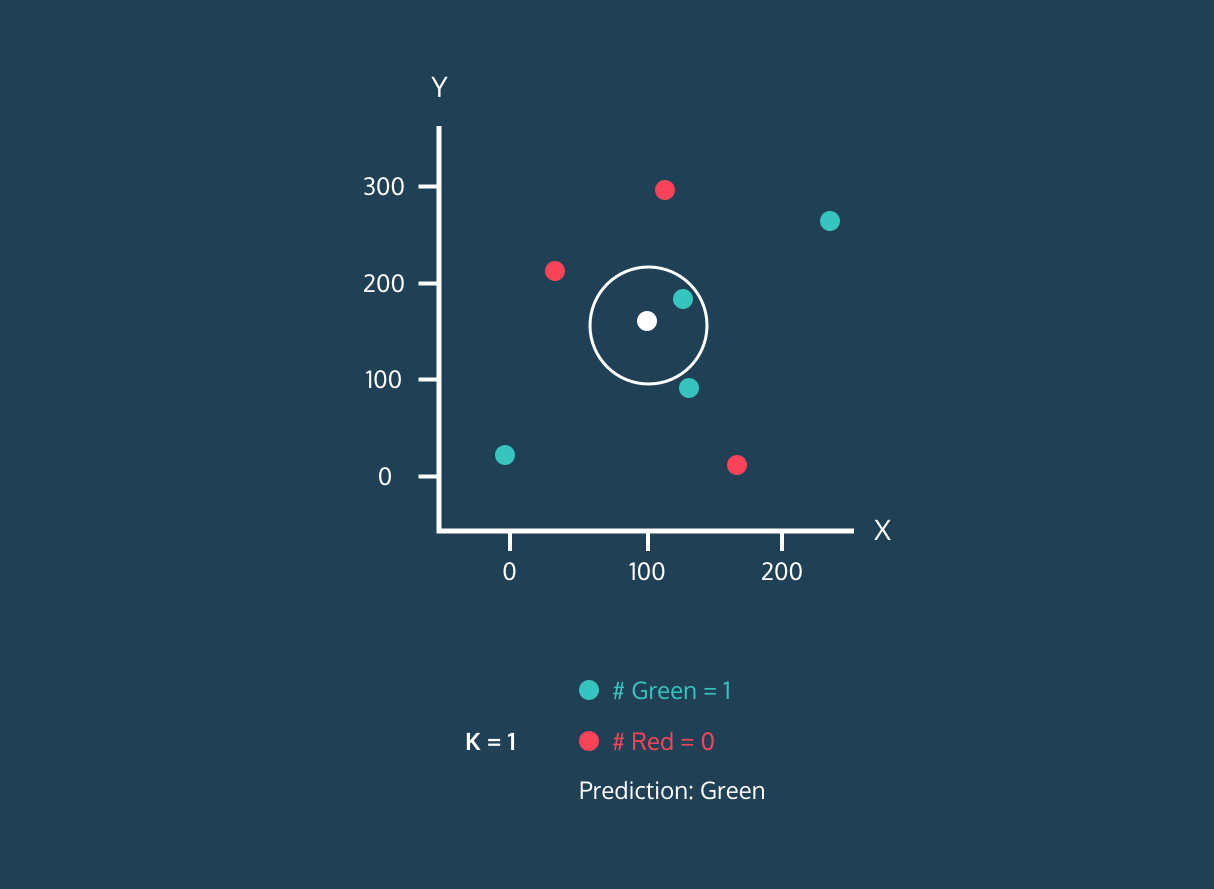
**Gini Impurity:**

Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the set.



**K-Nearest Neighbors (KNN):**

* It is a simple and effective supervised machine learning algorithm used for both classification and regression tasks.
* It's a type of instance-based learning, where the model makes predictions based on the majority class (for classification) or average (for regression) of the K-nearest data points in the feature space.

****

**Key Concepts:**

**K-Nearest Neighbors:**

* "K" in KNN represents the number of nearest neighbors to consider when making a prediction.
* K is a hyperparameter that needs to be specified before training the model.
* A smaller K can lead to noisy predictions, while a larger K can smooth out the predictions but might miss local patterns.

k = sqrt(n)

n = Number of instances in training data

**Distance Metric:**

The choice of distance metric (e.g., Euclidean distance, Manhattan distance) is crucial in determining the proximity of data points.

**Decision Rule:**

* For classification, the decision is often made by a majority vote among the K-nearest neighbors.
* For regression, the decision is based on the average (or weighted average) of the values of the K-nearest neighbors.

**Training Phase:**

KNN is a lazy learner; it doesn't build a model during the training phase. Instead, it memorizes the training data.

**Noisy Data:**

KNN can be sensitive to noisy data or irrelevant features.

**K-Means Clustering:**

* K-Means clustering is an unsupervised machine learning algorithm used for partitioning a dataset into K distinct, non-overlapping subsets (clusters).
* The goal of K-Means is to assign each data point to one of K clusters in a way that minimizes the sum of squared distances between the data points and the centroid of their assigned cluster.

**Key Concepts:**

**Centroid:**

Each cluster is characterized by a centroid, which is the mean of all the data points in that cluster.

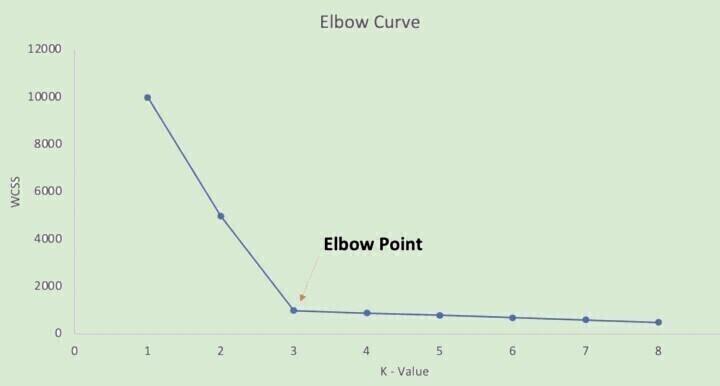
**Objective Function:**

The algorithm minimizes an objective function, often referred to as the "within-cluster sum of squares" or "inertia."

**How K-Means Clustering Works:**

1. **Choose the Number of Clusters (k):**

Decide on the number of clusters k you want to create. This is a parameter you need to specify before running the algorithm.



**Elbow Method for choosing value of k:**

* Run K-Means for Different Values of k
* Compute the Sum of Squared Distances (SSD)
* Create a plot with the number of clusters k on the x-axis and the SSD on the y-axis.
* Identify the Elbow Point: Look for a point on the plot where the SSD starts to decrease at a slower rate. This point is called the "elbow." The idea is that adding more clusters beyond this point yields only a marginal reduction in SSD, indicating that the additional clusters do not significantly improve the clustering.

1. **Initialize Cluster Centers:**

Randomly select k data points from the dataset to serve as the initial centroids (center points of the clusters).

1. **Assign Data Points to Clusters:**

Assign each data point to the nearest centroid based on the distance (often Euclidean distance). This step creates k clusters.

1. **Update Centroids:**

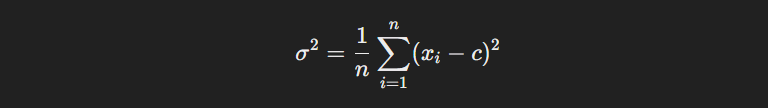
Recalculate the centroids of the clusters. The new centroid of each cluster is the mean of all data points assigned to that cluster.

1. **Repeat:**

Repeat steps 3 and 4 until the centroids no longer change significantly or until a pre-defined number of iterations is reached. This means the clusters are stable.

**Variance:**

Variance measures the spread of data points within a cluster. It is the average of the squared distances of all points in the cluster from the centroid.



**Explanation:** Variance is the mean of these squared distances. It gives an idea of how spread out the points are around the centroid.

**Ensemble learning:**

"Ensemble learning" refers to a technique in machine learning where multiple models are combined to improve the overall performance. This approach often results in better predictive performance than could be obtained from any of the constituent models alone. There are several types of ensemble learning methods, including:

**1.Bagging (Bootstrap Aggregating):** It involves training multiple instances of the same learning algorithm on different subsets of the training data and then averaging the predictions.

**2. Boosting:** This technique sequentially trains models, with each subsequent model focusing on the mistakes made by the previous ones. Popular algorithms like AdaBoost and Gradient Boosting Machines (GBM) fall into this category.

**3. Stacking:** Stacking combines the predictions of multiple models, known as base learners, using another model, often referred to as a meta-learner or blender, to make the final prediction.

**4. Voting:** In this method, predictions from multiple models are combined through a simple majority vote or weighted average.

Ensemble learning can be applied to various types of machine learning models, including decision trees, neural networks, support vector machines, and more. It's a powerful technique commonly used to boost the performance and robustness of predictive models in various domains.

**IMPORTANT!**

**Linear Models:** Suitable for linear relationships in data.

**Decision Trees:** Effective for non-linear relationships and interpretability.

**Ensemble Methods (Random Forest, Gradient Boosting):** Often improve performance by combining multiple models.

**Support Vector Machines (SVMs):** Useful for complex classification tasks.

**k-Nearest Neighbors (KNN):** Suitable for small to medium-sized datasets.

**Neural Networks:** Powerful for deep learning tasks, especially when large amounts of data are available.