Regression in Machine learning

•

Introduction to Regression

Regression is a **supervised learning technique** used to model the relationship between a **dependent variable (target)** and one or more **independent variables (features)**.

- **Goal:** Predict or estimate the dependent variable based on independent variables.
- **Types:**
  - **Linear Regression:** Used for continuous outputs.
  - **Logistic Regression:** Used for categorical (binary or multinomial) outputs.

Linear Regration

Linear regression is a type of **supervised machine-learning algorithm** 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 assumes that there is a linear relationship between the input and output, meaning the output changes at a constant rate as the input changes. This relationship is represented by a straight line.

For example we want to predict a student's exam score based on how many hours they studied. We observe that as students study more hours, their scores go up. In the example of predicting exam scores based on hours studied. Here

- **Independent variable (input):** Hours studied because it's the factor we control or observe.
- **Dependent variable (output):** Exam score because it depends on how many hours were studied.

We use the independent variable to predict the dependent variable.

Why Linear Regression is Important?

Here's why linear regression is important:

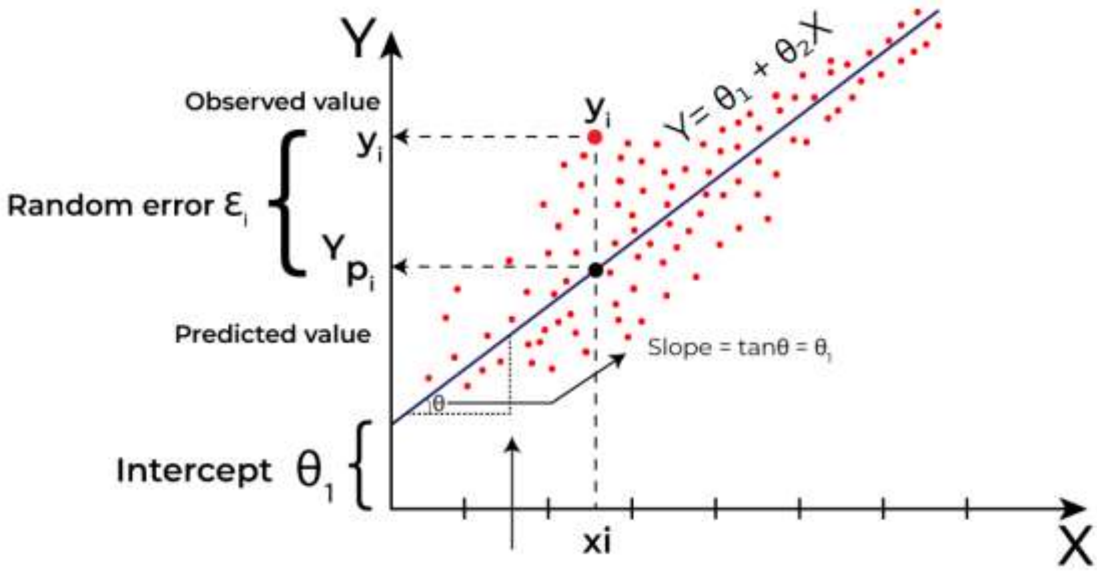
- **Simplicity and Interpretability:** It's easy to understand and interpret, making it a starting point for learning about machine learning.
- **Predictive Ability**: Helps predict future outcomes based on past data, making it useful in various fields like finance, healthcare and marketing.
- **Basis for Other Models:** Many advanced algorithms, like logistic regression or neural networks, build on the concepts of linear regression.
- **Efficiency:** It's computationally efficient and works well for problems with a linear relationship.
- **Widely Used:** It's one of the most widely used techniques in both statistics and machine learning for regression tasks.
- **Analysis:** It provides insights into relationships between variables (e.g., how much one variable influences another).

Best Fit Line in Linear Regression

In linear regression, the best-fit line is the straight line that most accurately represents the relationship between the independent variable (input) and the dependent variable (output). It is the line that minimizes the difference between the actual data points and the predicted values from the model.

1. Goal of the Best-Fit Line

The goal of linear regression is to find a straight line that minimizes the error (the difference) between the observed data points and the predicted values. This line helps us predict the dependent variable for new, unseen data.


Y
Observed value
y_i
Random error ε_i
Y_{p_i}
Predicted value
Intercept θ_1
y_i
$Y = \theta_1 + \theta_2 X$
Slope = $\tan\theta = \theta_1$
θ
xi
X

Linear Regression

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

2. Equation of the Best-Fit Line

For simple linear regression (with one independent variable), the best-fit line is represented by the equation

$y=mx+by=mx+b$

Where:

- **y** is the predicted value (dependent variable)
- **x** is the input (independent variable)
- **m** is the slope of the line (how much y changes when x changes)
- **b** is the intercept (the value of y when x = 0)

The best-fit line will be the one that optimizes the values of m (slope) and b (intercept) so that the predicted y values are as close as possible to the actual data points.

3. Minimizing the Error: The Least Squares Method

To find the best-fit line, we use a method called **Least Squares**. The idea behind this method is to minimize the sum of squared differences between the actual values (data points) and the predicted values from the line. These differences are called residuals. The formula for residuals is:

$Residual=y_i-\hat{y}_i \; Residual=y_i-\hat{y}_i$

Where:

- $y_i y_i$ is the actual observed value
- $y^{\wedge}_i y^{\wedge}_i$ is the predicted value from the line for that $x_i x_i$

The least squares method minimizes the sum of the squared residuals:
Sum ofsquarederrors(SSE)$=\Sigma(y_i-y^{\wedge}_i)^2$*Sumofsquarederrors(SSE)*$=\Sigma(y_i-y^{\wedge}_i)^2$

This method ensures that the line best represents the data where the sum of the squared differences between the predicted values and actual values is as small as possible.

4. Interpretation of the Best-Fit Line

- **Slope (m):** The slope of the best-fit line indicates how much the dependent variable (y) changes with each unit change in the independent variable (x). For example if the slope is 5, it means that for every 1-unit increase in x, the value of y increases by 5 units.
- **Intercept (b):** The intercept represents the predicted value of y when x = 0. It's the point where the line crosses the y-axis.

In linear regression some hypothesis are made to ensure reliability of the model's results.

Limitations

- ***Assumes Linearity:*** *The method assumes the relationship between the variables is linear. If the relationship is non-linear, linear regression might not work well.*
- ***Sensitivity to Outliers:*** *Outliers can significantly affect the slope and intercept, skewing the best-fit line.*

Types of Linear Regression

When there is only one independent feature it is known as Simple Linear Regression or Univariate Linear Regression and when there are more than one feature it is known as Multiple Linear Regression or Multivariate Regression.

1. Simple Linear Regression

Simple linear regression is used when we want to predict a target value (dependent variable) using only one input feature (independent variable). It assumes a straight-line relationship between the two.

Example:

Predicting a person's salary (y) based on their years of experience (x).

2. Multiple Linear Regression

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.

Use Case of Multiple Linear Regression

Multiple linear regression allows us to analyze relationship between multiple independent variables and a single dependent variable. Here are some use cases:

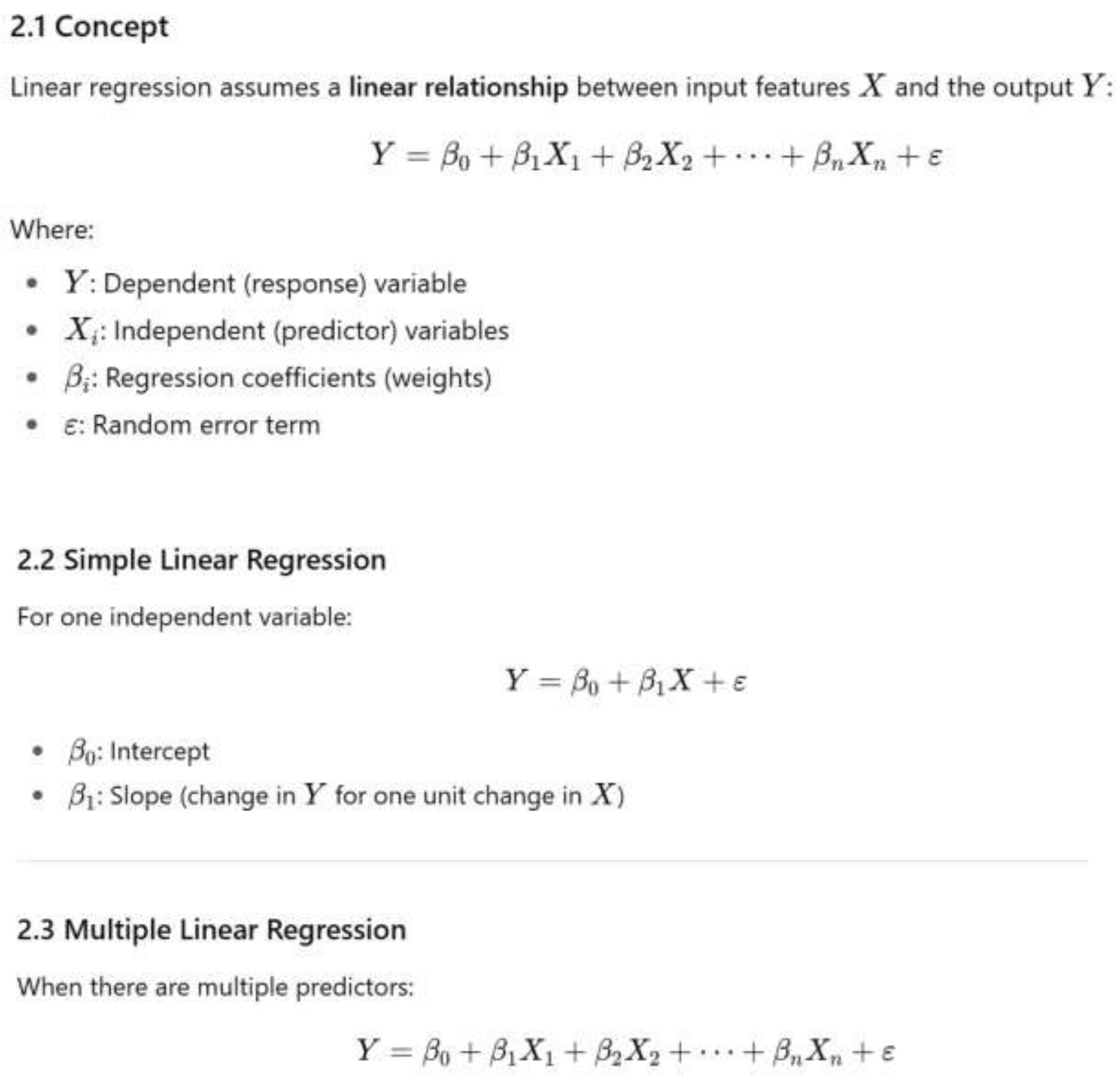
- **Real Estate Pricing:** In real estate MLR is used to predict property prices based on multiple factors such as location, size, number of bedrooms, etc. This helps buyers and sellers understand market trends and set competitive prices.
- **Financial Forecasting:** Financial analysts use MLR to predict stock prices or economic indicators based on multiple influencing factors such as interest rates, inflation rates and market trends. This enables better investment strategies and risk management24.

- **Agricultural Yield Prediction:** Farmers can use MLR to estimate crop yields based on several variables like rainfall, temperature, soil quality and fertilizer usage. This information helps in planning agricultural practices for optimal productivity
- **E-commerce Sales Analysis:** An e-commerce company can utilize MLR to assess how various factors such as product price, marketing promotions and seasonal trends impact sales.

Now that we have understood about linear regression, its assumption and its type now we will learn how to make a linear regression model.

Hypothesis function in Linear Regression

In linear regression, the hypothesis function is the equation used to make predictions about the dependent variable based on the independent variables. It represents the relationship between the input features and the target output.
For a simple case with one independent variable, the hypothesis function is:



2.1 Concept

Linear regression assumes a **linear relationship** between input features X and the output Y:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n + \varepsilon$$

Where:

- Y: Dependent (response) variable
- X_i: Independent (predictor) variables
- β_i: Regression coefficients (weights)
- ε: Random error term

2.2 Simple Linear Regression

For one independent variable:

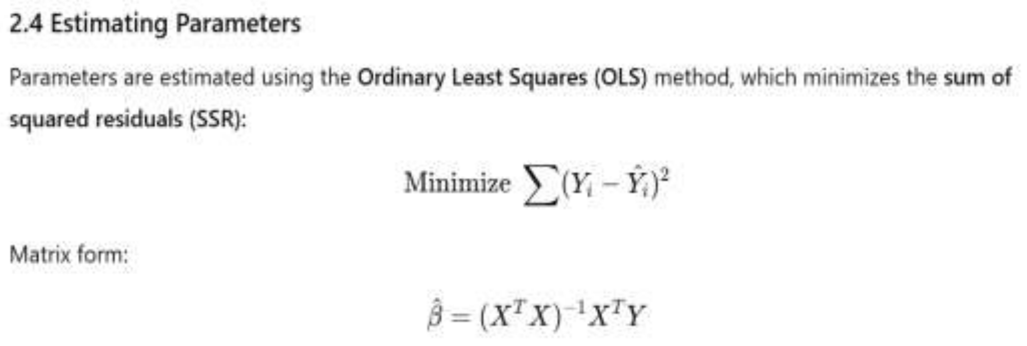
$$Y = \beta_0 + \beta_1 X + \varepsilon$$

- β_0: Intercept
- β_1: Slope (change in Y for one unit change in X)

2.3 Multiple Linear Regression

When there are multiple predictors:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n + \varepsilon$$



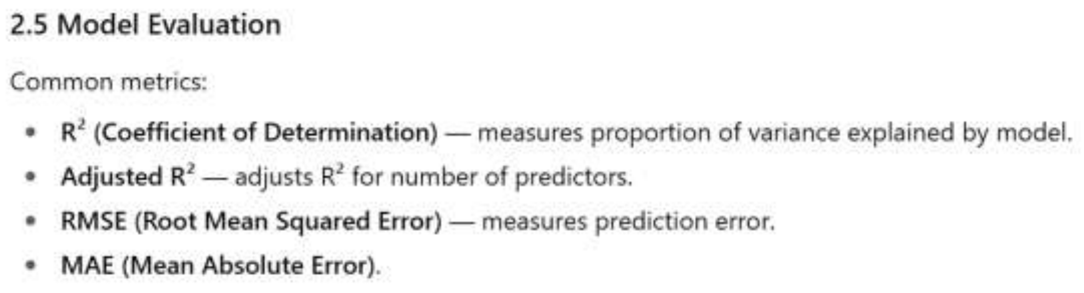
2.4 Estimating Parameters

Parameters are estimated using the **Ordinary Least Squares (OLS)** method, which minimizes the **sum of squared residuals (SSR)**:

$$\text{Minimize} \sum (Y_i - \hat{Y}_i)^2$$

Matrix form:

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$



2.5 Model Evaluation

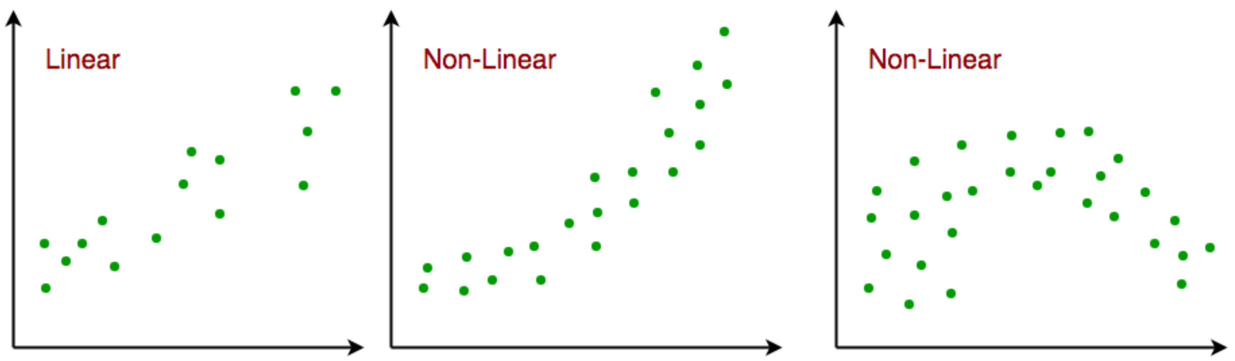
Common metrics:

- **R^2 (Coefficient of Determination)** — measures proportion of variance explained by model.
- **Adjusted R^2** — adjusts R^2 for number of predictors.
- **RMSE (Root Mean Squared Error)** — measures prediction error.
- **MAE (Mean Absolute Error).**

2.6 Assumptions of Linear Regression

1. **Linearity:** Relationship between X and Y is linear.
2. **Independence:** Observations are independent.
3. **Homoscedasticity:** Constant variance of errors.
4. **Normality:** Errors are normally distributed.
5. **No multicollinearity:** Predictors are not highly correlated.

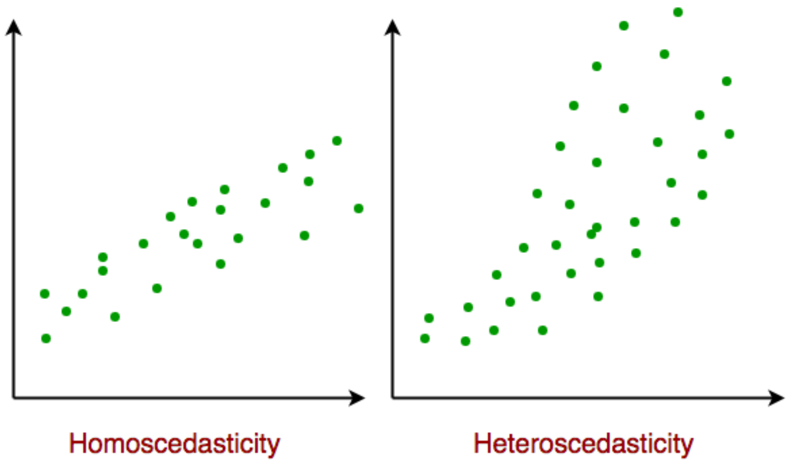
1. Linearity: The relationship between inputs (X) and the output (Y) is a straight line.



Linearity

2. Independence of Errors: The errors in predictions should not affect each other.

3. Constant Variance (Homoscedasticity): The errors should have equal spread across all values of the input. If the spread changes (like fans out or shrinks), it's called heteroscedasticity and it's a problem for the model.



Homoscedasticity
4. Normality of Errors: The errors should follow a normal (bell-shaped) distribution.
5. No Multicollinearity*(*for multiple regression)**: Input variables shouldn't be too closely related to each other.
6. No Autocorrelation: Errors shouldn't show repeating patterns, especially in time-based data.
7. Additivity: The total effect on Y is just the sum of effects from each X, no mixing or interaction between them.'
To understand Multicollinearityin detail refer to article: ***Multicollinearity***.

Cost function for Linear Regression

As we have discussed earlier about best fit line in linear regression, its not easy to get it easily in real life cases so we need to calculate errors that affects it. These errors need to be calculated to mitigate them. The difference between the predicted value and the true value Y and it is called cost function or the loss function.
In Linear Regression, the Mean Squared Error (MSE) cost function is employed, which calculates the average of the squared errors between the predicted values and the actual values. The purpose is to determine the optimal values for the intercept and the coefficient of the input feature to providing the best-fit line for the given data points.
Utilizing the MSE function, the iterative process of gradient descent is applied to update the values . This ensures that the MSE value converges to the global minima, signifying the most accurate fit of the linear regression line to the dataset.
This process involves continuously adjusting the parameters bita based on the gradients calculated from the MSE. The final result is a linear regression line that minimizes the overall squared differences between the predicted and actual values, providing an optimal representation of the underlying relationship in the data.
Now we have calculated loss function we need to optimize model to mtigate this error and it is done through gradient descent.

Gradient Descent for Linear Regression

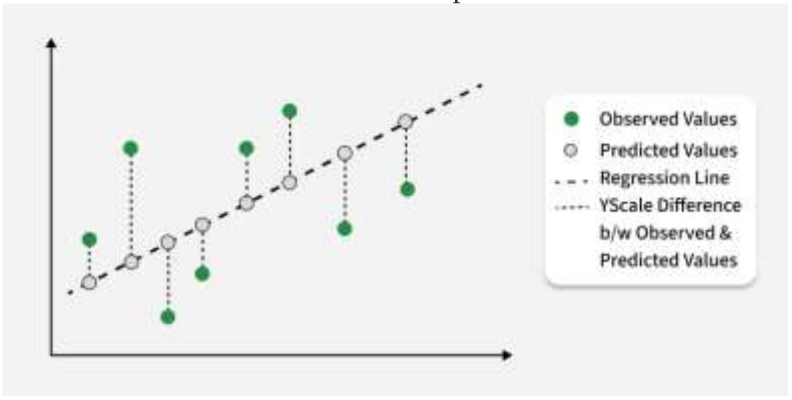
Gradient descent is an optimization technique used to train a linear regression model by minimizing the prediction error. It works by starting with random model parameters and repeatedly adjusting them to reduce the difference between predicted and actual values.

Gradient Descent
How it works:
- Start with random values for slope and intercept.
- Calculate the error between predicted and actual values.
- Find how much each parameter contributes to the error (gradient).
- Update the parameters in the direction that reduces the error.
- Repeat until the error is as small as possible.

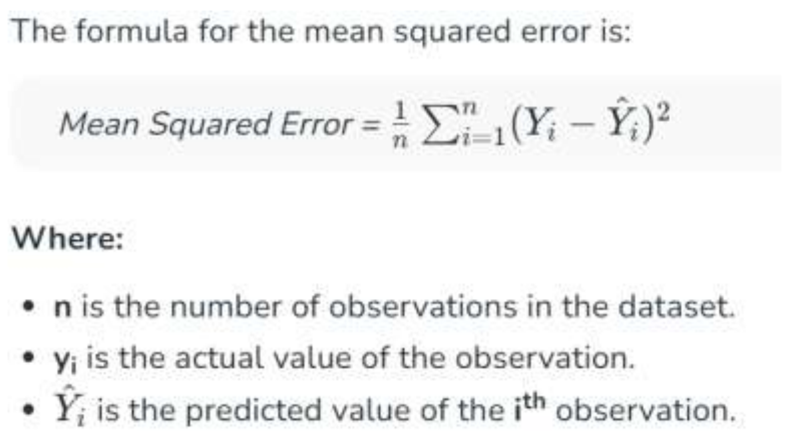
This helps the model find the best-fit line for the data.
For more details you can refer to: Gradient Descent in Linear Regression

Evaluation Metrics for Linear Regression
A variety of evaluation measures can be used to determine the strength of any linear regression model. These assessment metrics often give an indication of how well the model is producing the observed outputs.
The most common measurements are:
1. Mean Square Error (MSE)
Mean Squared Error (MSE) 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.

Mean Squared Error (MSE) is a fundamental concept in statistics and machine learning, playing a crucial role in assessing the accuracy of predictive models.
- The MSE value provides a way to analyze the accuracy of the model.
- It measures the average squared difference between predicted values and the actual values in the dataset.
- It is calculated by taking the average of the squared residuals, where the residual is the difference between the predicted value and the actual value for each data point.



Mean Squared Error Formula


The formula for the mean squared error is:

$$\text{Mean Squared Error} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Where:

- **n** is the number of observations in the dataset.
- $\mathbf{y_i}$ is the actual value of the observation.
- $\hat{Y}_i$ is the predicted value of the $\mathbf{i^{th}}$ observation.

Interpretation of Mean Squared Error

The Interpreting MSE involves understanding the magnitude of the error and its implications for the model's performance.

- A lower MSE indicates that the model's predictions are closer to the actual values, signifying better accuracy.
- Conversely, a higher MSE suggests that the model's predictions deviate further from the true value, indicating poorer performance.

Significance of Mean Squared Error

The Mean Squared Error is widely used in various fields, including statistics, machine learning, and econometrics, due to its several important properties:

- It provides the quantitative measure of the accuracy of the predictive models.
- It penalizes large errors more heavily than small errors, making it sensitive to the outliers.
- It is mathematically convenient and easy to interpret, making it a preferred choice for evaluating model performance.

Applications of Mean Squared Error

The Mean Squared Error is extensively used in various applications, including:

- **Regression analysis**: Assessing the goodness of fit of the regression models.
- **Model evaluation**: Comparing the performance of the different machine learning algorithms.
- **Optimization**: Minimizing MSE during the model training to improve predictive accuracy.
- **Predictive modeling**: Evaluating the accuracy of the regression and forecasting models.
- **Image processing**: Assessing the quality of the image reconstruction and restoration algorithms.
- **Financial modeling**: Analyzing the performance of the investment strategies and risk models.

How to Minimize Mean Squared Error in Model Training

To minimize Mean Squared Error during the model training, several strategies can be employed, including:

- **Feature selection:** Choosing relevant features that contribute most to reducing prediction errors.
- **Model selection:** Experimenting with the different algorithms and model architectures to identify the best-performing model.
- **Hyperparameter tuning:** The Optimizing model hyperparameters such as the learning rate, regularization strength, and network depth to improve predictive accuracy.

Example problems on Mean Squared Error
Example: Suppose we have a dataset consisting of the actual and predicted values for the regression problem

- Actual Values: [10, 20, 30, 40, 50]
- Predicted Values: [12, 18, 32, 38, 48]

Solution:
To calculate MSE we first compute the squared differences between the each actual and predicted value:
Squared Differences: [(10-12)^2, (20-18)^2, (30-32)^2, (40-38)^2, (50-48)^2]
= [4, 4, 4, 4, 4]
Next, we take the average of these squared differences to the obtain the MSE:
MSE = (4 + 4 + 4 + 4 + 4) / 5
= 20 / 5
= 4
Therefore, the MSE for this regression model is 4.

Root Mean Square Error
The Root Mean Squared Error (RMSE) is a variant of MSE that calculates the square root of the average squared difference between actual and predicted values. It is often preferred over MSE as it provides an interpretable measure of the error in the same units as the original data.
RMSE Formula
RMSE = √(MSE)

Example of Root Mean Square Error
Example: Given the actual and predicted values for the regression problem, calculate the MSE and RMSE.

- Actual Values: [15, 25, 35, 45, 55]
- Predicted Values: [18, 22, 38, 42, 52]

Solution:
The Calculate the squared differences between the actual and predicted values:
Squared Differences: [(15-18)2, (25-22)2, (35-38)2, (45-42)2, (55-52)2]
= [9, 9, 9, 9, 9]
Compute the MSE
MSE = (9 + 9 + 9 + 9 + 9) / 5
= 45 / 5
= 9
Calculate the RMSE:
RMSE = √(9)
= 3
MSE vs RMSE
Mean Squared Error is often compared with other error metrics, such as the Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), to evaluate model performance. While MAE measures the average absolute difference between predicted and actual values, RMSE measures the square root of the average squared difference. The MSE and RMSE penalize large errors more heavily than MAE, making them more sensitive to the outliers.

Significance of Mean Squared Error
The Mean Squared Error is widely used in various fields, including statistics, machine learning, and econometrics, due to its several important properties:

- It provides the quantitative measure of the accuracy of the predictive models.
- It penalizes large errors more heavily than small errors, making it sensitive to the outliers.
- It is mathematically convenient and easy to interpret, making it a preferred choice for evaluating model performance.

Applications of Mean Squared Error

The Mean Squared Error is extensively used in various applications, including:

- **Regression analysis**: Assessing the goodness of fit of the regression models.
- **Model evaluation**: Comparing the performance of the different machine learning algorithms.
- **Optimization**: Minimizing MSE during the model training to improve predictive accuracy.
- **Predictive modeling**: Evaluating the accuracy of the regression and forecasting models.
- **Image processing**: Assessing the quality of the image reconstruction and restoration algorithms.
- **Financial modeling**: Analyzing the performance of the investment strategies and risk models.

How to Minimize Mean Squared Error in Model Training

To minimize Mean Squared Error during the model training, several strategies can be employed, including:

- **Feature selection:** Choosing relevant features that contribute most to reducing prediction errors.
- **Model selection:** Experimenting with the different algorithms and model architectures to identify the best-performing model.
- **Hyperparameter tuning:** The Optimizing model hyperparameters such as the learning rate, regularization strength, and network depth to improve predictive accuracy.

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=2n(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).

4. Coefficient of Determination (R-squared)

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)R2=1−(TSSRSS)

- **Residual sum of Squares**(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=1n(yi−b0−b1xi)2RSS=∑i=1n(yi−b0−b1xi)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=∑i=1n(y−yi¯)2TSS=∑i=1n(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.

5. Adjusted R-Squared Error

Adjusted R^2 measures the proportion of variance in the dependent variable that is explained by independent variables in a regression model. Adjusted R-square 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 R^2 is expressed as:

AdjustedR^2=1−((1−R2).(n−1)n−k−1)AdjustedR2=1−(n−k−1(1−R2).(n−1))

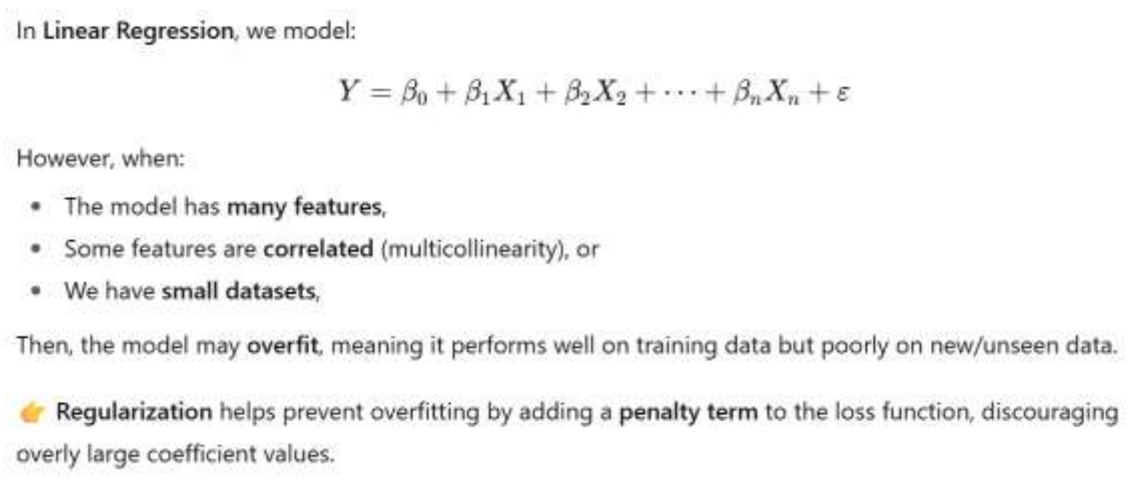
Here,

- 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.

Regularization Techniques for Linear Regration Models



In **Linear Regression**, we model:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n + \varepsilon$$

However, when:

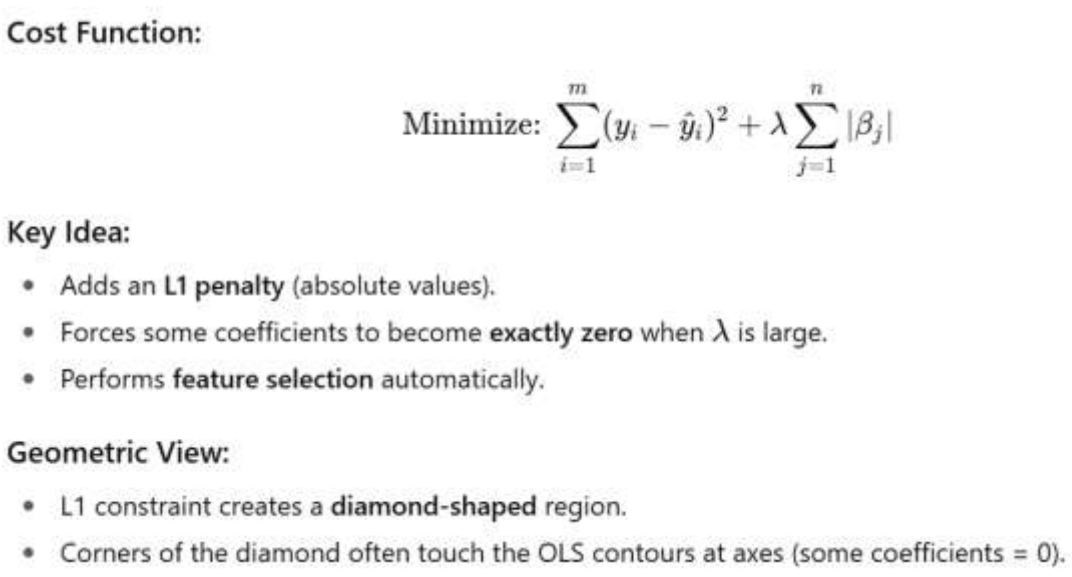
- The model has **many features**,
- Some features are **correlated** (multicollinearity), or
- We have **small datasets**,

Then, the model may **overfit**, meaning it performs well on training data but poorly on new/unseen data.

👉 **Regularization** helps prevent overfitting by adding a **penalty term** to the loss function, discouraging overly large coefficient values.

1. Lasso Regression (L1 Regularization)

Lasso Regression is a technique used for regularizing a linear regression model, it adds a penalty term to the linear regression objective function to prevent overfitting.


Cost Function:

$$\text{Minimize: } \sum_{i=1}^{m}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{n} |\beta_j|$$

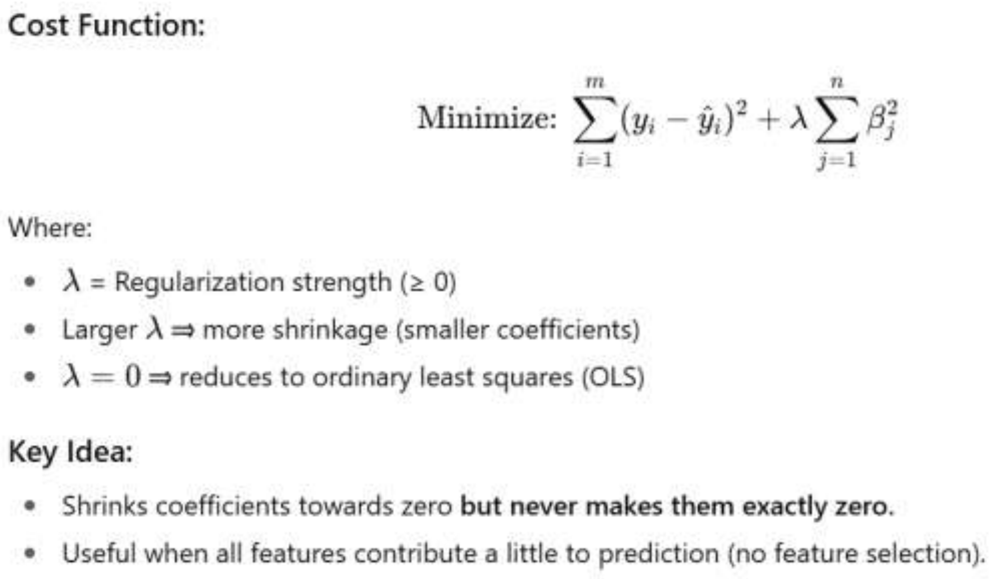
Key Idea:

- Adds an **L1 penalty** (absolute values).
- Forces some coefficients to become **exactly zero** when λ is large.
- Performs **feature selection** automatically.

Geometric View:

- L1 constraint creates a **diamond-shaped** region.
- Corners of the diamond often touch the OLS contours at axes (some coefficients = 0).

2. Ridge Regression (L2 Regularization)
Ridge regression is a linear regression technique that adds a regularization term to the standard linear objective. Again, the goal is to prevent overfitting by penalizing large coefficient in linear regression equation. It useful when the dataset has multicollinearity where predictor variables are highly correlated.
The objective function after applying ridge regression is:


Cost Function:

$$\text{Minimize: } \sum_{i=1}^{m}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{n} \beta_j^2$$

Where:

- λ = Regularization strength (≥ 0)
- Larger $\lambda \Rightarrow$ more shrinkage (smaller coefficients)
- $\lambda = 0 \Rightarrow$ reduces to ordinary least squares (OLS)

Key Idea:

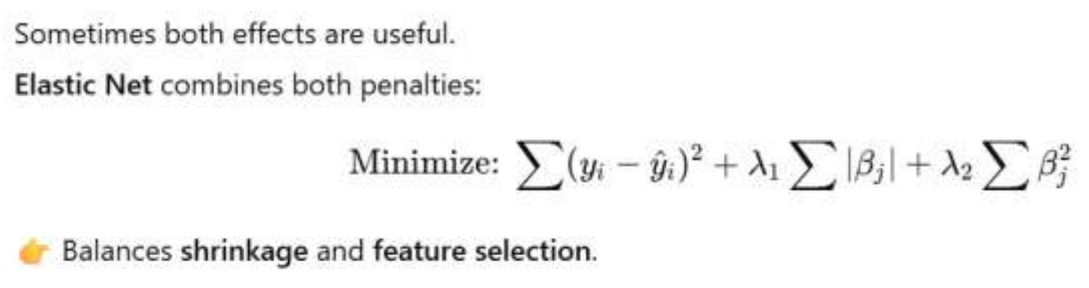
- Shrinks coefficients towards zero **but never makes them exactly zero.**
- Useful when all features contribute a little to prediction (no feature selection).

Geometric View:

- L2 constraint creates a **circular** region around zero.
- Intersection with OLS contours occurs within this circle, producing smaller coefficients.

3. Elastic Net Regression (Hybrid of Ridge + Lasso)

Elastic Net Regression is a hybrid regularization technique that combines the power of both L1 and L2 regularization in linear regression objective.



Sometimes both effects are useful.

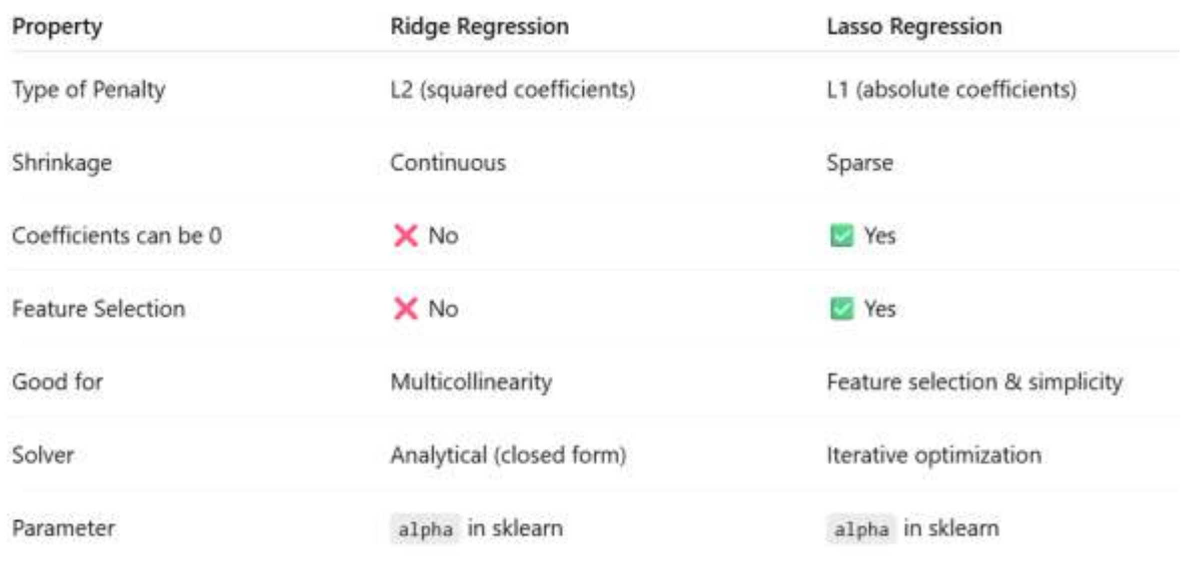
Elastic Net combines both penalties:

$$\text{Minimize: } \sum (y_i - \hat{y}_i)^2 + \lambda_1 \sum |\beta_j| + \lambda_2 \sum \beta_j^2$$

👉 Balances **shrinkage** and **feature selection**.

- the first term is least square loss.
- the second term is L1 regularization and third is ridge regression.
- λ λis the overall regularization strength. We Can tune λ\lambdaλ (called **alpha** in scikit-learn) using **cross-validation**:
- controls the mix between L1 and L2 regularization.

Comparision



Property	Ridge Regression	Lasso Regression
Type of Penalty	L2 (squared coefficients)	L1 (absolute coefficients)
Shrinkage	Continuous	Sparse
Coefficients can be 0	❌ No	✅ Yes
Feature Selection	❌ No	✅ Yes
Good for	Multicollinearity	Feature selection & simplicity
Solver	Analytical (closed form)	Iterative optimization
Parameter	`alpha` in sklearn	`alpha` in sklearn

Key Takeaways:

- **Ridge Regression: Reduces model complexity and multicollinearity; all predictors retained.**
- **Lasso Regression: Performs both shrinkage and feature selection by zeroing unimportant coefficients.**
- **Both improve generalization and prevent overfitting.**
- **You can combine both using Elastic Net.**

Python Implementation of Linear Regression

1. Import the necessary libraries:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.axes as ax
from matplotlib.animation import FuncAnimation
```

2. Load the dataset and separate input and Target variables

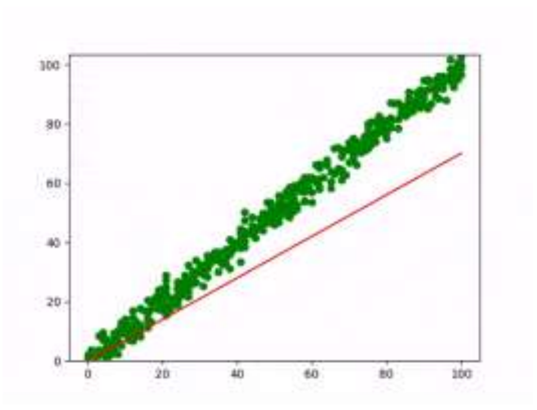
Here is the link for dataset: Dataset Link

3. Build the Linear Regression Model and Plot the regression line

In forward propagation Linear regression function Y=mx+c$Y=mx+c$ is applied by initially assigning random value of parameter (m and c).

The we have written the function to finding the cost function i.e the mean

4. Plot



The linear regression line provides valuable insights into the relationship between the two variables. It represents the best-fitting line that captures the overall trend of how a dependent variable (Y) changes in response to variations in an independent variable (X).

- **Positive Linear Regression Line**: A positive linear regression line indicates a direct relationship between the independent variable (XX) and the dependent variable (YY). This means that as the value of X increases, the value of Y also increases. The slope of a positive linear regression line is positive, meaning that the line slants upward from left to right.
- **Negative Linear Regression Line**: A negative linear regression line indicates an inverse relationship between the independent variable (XX) and the dependent variable (YY). This means that as the value of X increases, the value of Y decreases. The slope of a negative linear regression line is negative, meaning that the line slants downward from left to right.

4. Trained the model and Final Prediction

5. Output:

Applications of Linear Regression

Linear regression is used in many different fields including finance, economics and psychology to understand and predict the behavior of a particular variable.

For example linear regression is widely used in finance to analyze relationships and make predictions. It can model how a company's earnings per share (EPS) influence its stock price. If the model shows that a $1 increase in EPS results in a $15 rise in stock price, investors gain insights into the company's valuation. Similarly, linear regression can

forecast currency values by analyzing historical exchange rates and economic indicators, helping financial professionals make informed decisions and manage risks effectively.
Also read - Linear Regression - In Simple Words, with real-life Examples

Advantages and Disadvantages of Linear Regression

Advantages of Linear Regression

- Linear regression is a relatively simple algorithm, making it easy to understand and implement. The coefficients of the linear regression model can be interpreted as the change in the dependent variable for a one-unit change in the independent variable, providing insights into the relationships between variables.
- Linear regression is computationally efficient and can handle large datasets effectively. It can be trained quickly on large datasets, making it suitable for real-time applications.
- Linear regression is relatively robust to outliers compared to other machine learning algorithms. Outliers may have a smaller impact on the overall model performance.
- Linear regression often serves as a good baseline model for comparison with more complex machine learning algorithms.
- Linear regression is a well-established algorithm with a rich history and is widely available in various machine learning libraries and software packages.

Disadvantages of Linear Regression

- Linear regression assumes a linear relationship between the dependent and independent variables. If the relationship is not linear, the model may not perform well.
- Linear regression is sensitive to multicollinearity, which occurs when there is a high correlation between independent variables. Multicollinearity can inflate the variance of the coefficients and lead to unstable model predictions.
- Linear regression assumes that the features are already in a suitable form for the model. Feature engineering may be required to transform features into a format that can be effectively used by the model.
- Linear regression is susceptible to both overfitting and underfitting. Overfitting occurs when the model learns the training data too well and fails to generalize to unseen data. Underfitting occurs when the model is too simple to capture the underlying relationships in the data.
- Linear regression provides limited explanatory power for complex relationships between variables. More advanced machine learning techniques may be necessary for deeper insights.

3. Logistic Regression

3.1 Concept

Used when the **dependent variable is categorical**, usually **binary** (0 or 1).

Example: Predicting whether an email is *spam (1)* or *not spam (0)*.

3.2 Why Not Linear Regression?

- Linear regression can predict values <0 or >1 — not suitable for probabilities.
- Residuals are not normally distributed.
- Variance is not constant.

3.3 Logistic Function (Sigmoid Function)

To map predictions to a range between 0 and 1, logistic regression uses the **sigmoid function**:

$$P(Y = 1|X) = \frac{1}{1 + e^{-z}}$$

where

$$z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n$$

3.4 Decision Boundary

If $P(Y = 1|X) > 0.5$, predict **1**, else predict **0**.
(Threshold can be changed depending on application.)

3.5 Log-Odds and the Logistic Model

Taking the **logit (log-odds)** form:

$$\ln\left(\frac{P(Y = 1|X)}{1 - P(Y = 1|X)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_n X_n$$

This transforms the nonlinear sigmoid into a **linear relationship** between log-odds and predictors.

3.6 Parameter Estimation

Coefficients are estimated using **Maximum Likelihood Estimation (MLE)**, not OLS.

We find parameters β that **maximize the likelihood** of observing the given data.

3.7 Model Evaluation Metrics

For classification problems:

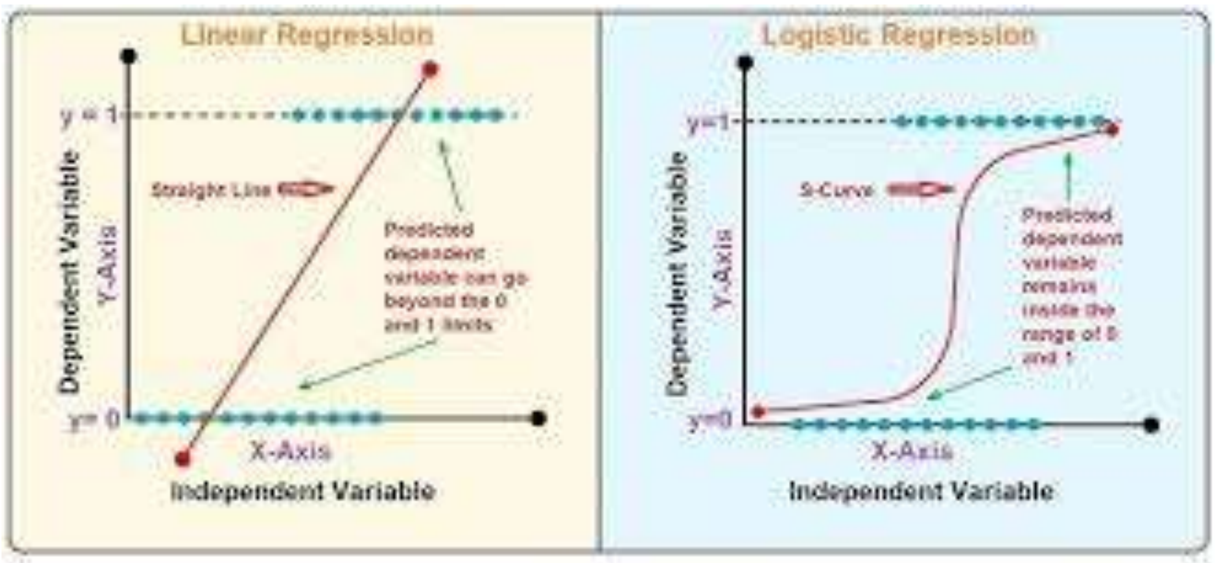
- **Accuracy**
- **Precision**
- **Recall**
- **F1 Score**
- **ROC Curve** (Receiver Operating Characteristic)
- **AUC (Area Under Curve)**
- **Confusion Matrix**

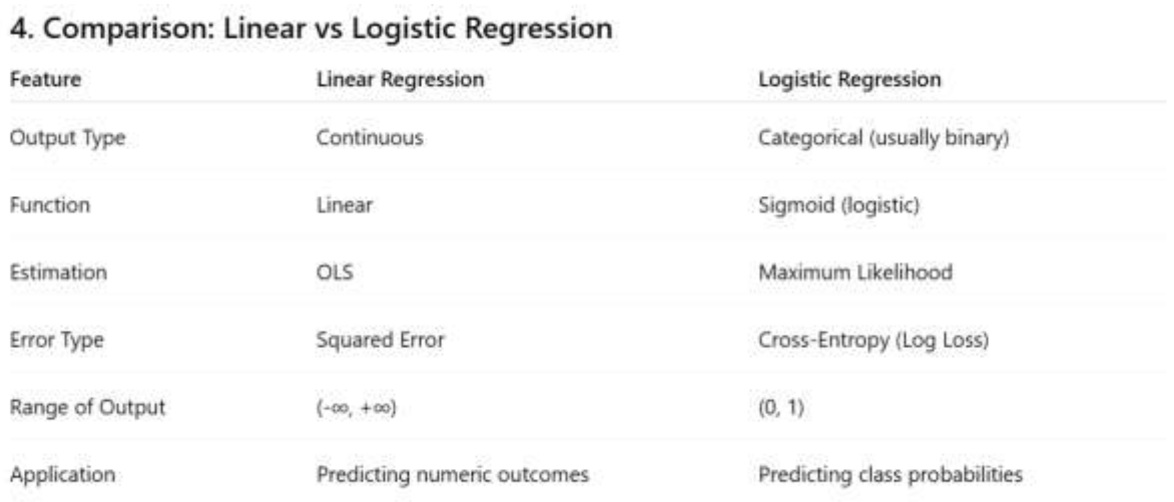
3.8 Extensions

- **Multinomial Logistic Regression:** For multi-class problems.
- **Ordinal Logistic Regression:** For ordered categorical outcomes.
- **Regularized Logistic Regression:** Includes penalties like L1 (Lasso) and L2 (Ridge).

Difference Between Linear Regression and Logistic Regression

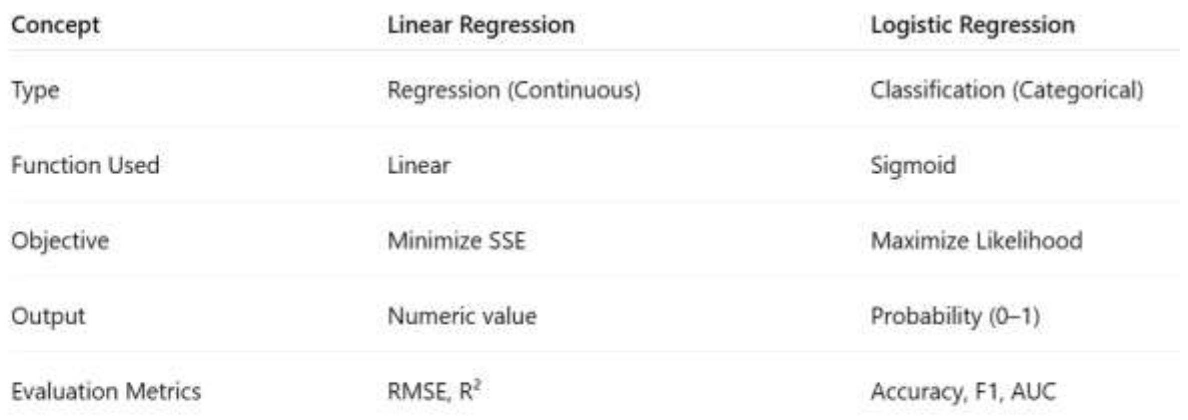
Linear Regression and Logistic Regression are Supervised Machine Learning models that use labelled datasets to make predictions. However, there's a fundamental difference in their usage – Linear Regression is used for Regression problems, whereas Logistic Regression is mainly used for solving classification problems.





4. Comparison: Linear vs Logistic Regression

Feature	Linear Regression	Logistic Regression
Output Type	Continuous	Categorical (usually binary)
Function	Linear	Sigmoid (logistic)
Estimation	OLS	Maximum Likelihood
Error Type	Squared Error	Cross-Entropy (Log Loss)
Range of Output	(-∞, +∞)	(0, 1)
Application	Predicting numeric outcomes	Predicting class probabilities



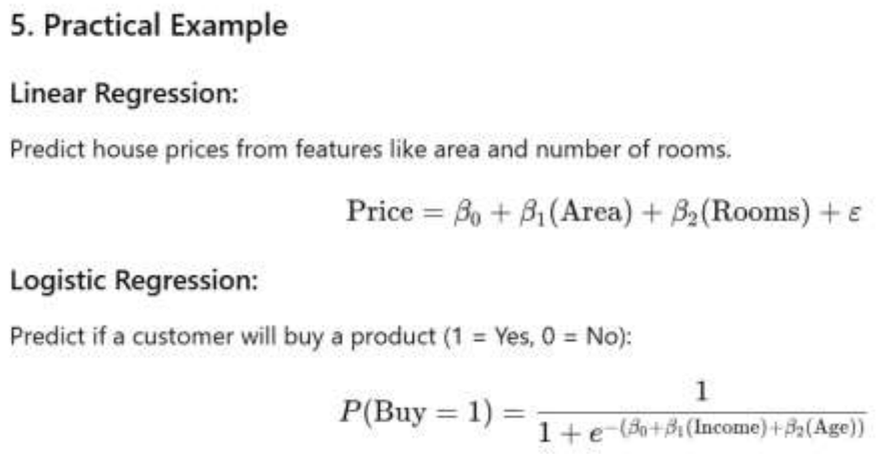
Concept	Linear Regression	Logistic Regression
Type	Regression (Continuous)	Classification (Categorical)
Function Used	Linear	Sigmoid
Objective	Minimize SSE	Maximize Likelihood
Output	Numeric value	Probability (0–1)
Evaluation Metrics	RMSE, R^2	Accuracy, F1, AUC

Linear regression and logistic regression are among the most commonly used models in supervised machine learning. Each model has their application, characteristics, advantages, and limitations. This article will provide a comprehensive understanding of both (linear and logistic regression) models, highlighting their differences, limitations and real-life applications.

Difference Between Linear Regression and Logistic Regression

Parameter	**Linear Regression**	**Logistic Regression**
Outcome Variable Type	Continuous variable (e.g., price, temperature)	Categorical variable, typically binary (e.g., yes/no, 0/1)
Model Purpose	Regression (predicting numerical values)	Classification (categorizing into discrete classes)
Equation/Function	Linear equation: Y = β0 + β1X + ε	Logistic (Sigmoid) function: p(X) = 1 / (1 + e^-(β0 + β1X))
Output Interpretation	Predicted value of the dependent variable	Probability of a particular class or event
Relationship Between Variables	Assumes a linear relationship between variables	Does not assume a linear relationship; models probability
Error Distribution	Assumes normally distributed errors	Does not assume a normal distribution of errors
Estimation Method	Ordinary Least Squares (OLS)	Maximum Likelihood Estimation (MLE)
Sensitivity to Outliers	More sensitive to outliers	Less sensitive to outliers
Homoscedasticity Assumption	Assumes homoscedasticity (constant variance of errors)	No assumption of homoscedasticity
Application Scope	Suitable for forecasting, effect analysis of	Ideal for binary classification in various fields

Parameter	Linear Regression	Logistic Regression
	independent variables	



5. Practical Example

Linear Regression:

Predict house prices from features like area and number of rooms.

$$\text{Price} = \beta_0 + \beta_1(\text{Area}) + \beta_2(\text{Rooms}) + \varepsilon$$

Logistic Regression:

Predict if a customer will buy a product (1 = Yes, 0 = No):

$$P(\text{Buy} = 1) = \frac{1}{1 + e^{-(\beta_0+\beta_1(\text{Income})+\beta_2(\text{Age}))}}$$

Linear Regression Example

Goal:

Predict a student’s exam score based on the number of study hours.

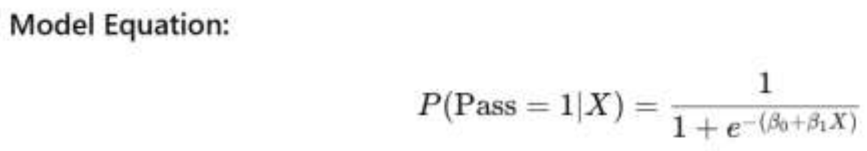
Model Equation:

$$Y = \beta_0 + \beta_1 X$$

Logistic Regression Example

Goal:

Predict whether a student passes (1) or fails (0) an exam based on study hours.



Model Equation:

$$P(\text{Pass} = 1|X) = \frac{1}{1 + e^{-(\beta_0+\beta_1 X)}}$$

Conclusion

Linear Regression and Logistic Regression are two of the most popularly used algorithms in Supervised Machine Learning. I hope this blog on **Linear Regression vs Logistic Regression** was helpful in understanding the difference between the two and it aroused your interest in Machine Learning.