Linear Regression Analysis

Linear regression is one of the simplest and most widely used statistical methods for analyzing relationships between two or more variables. It models the relationship between a **dependent variable (response)** and one or more **independent variables (predictors)** by fitting a linear equation to the observed data. The goal is to predict the dependent variable based on the values of the independent variables.

Key Concepts of Linear Regression

1. **Dependent Variable (Y)**: The outcome or target variable you want to predict or explain.
    - Example: Predicting house prices.
2. **Independent Variable(s) (X)**: The input variables used to predict the dependent variable.
    - Example: Factors like the size of the house, number of bedrooms, and location.
3. **Linear Equation**: The relationship is modeled as: [$ Y = \beta_0 + \beta_1X + \epsilon $] Where:
    - (Y): Dependent variable.
    - (X): Independent variable.
    - (β_0): Intercept (the value of (Y) when ($X = 0$)).
    - (β_1): Slope (the change in (Y) for a unit change in (X)).
    - (ϵ): Error term (captures variability not explained by (X)).
4. **Assumptions**:
    - Linearity: The relationship between (X) and (Y) is linear.
    - Independence: Observations are independent of each other.
    - Homoscedasticity: Constant variance of residuals (error terms).
    - Normality: Residuals are normally distributed.

Types of Linear Regression

1. **Simple Linear Regression**:
    - Involves one independent variable.
    - Example: Predicting a student's test score ((Y)) based on study hours ((X)).
2. **Multiple Linear Regression**:
    - Involves two or more independent variables.
    - Example: Predicting house prices ((Y)) based on size ((X_1)), number of bedrooms ((X_2)), and age of the house ((X_3)).

Steps in Linear Regression Analysis

1. **Data Collection**:
    - Collect data for both dependent and independent variables.
    - Example: A dataset with house prices, sizes, and number of bedrooms.
2. **Data Exploration and Preparation**:
    - Visualize data using scatterplots to check linearity.
    - Handle missing or outlier values.
3. **Fit the Model**:
    - Use statistical software (e.g., Python, R) to calculate (β_0) and (β_1).
4. **Evaluate the Model**:
    - **R-squared ((R^2))**: Explains the proportion of variance in (Y) explained by (X).
    - **Mean Squared Error (MSE)**: Measures the average squared error between predicted and actual (Y).
5. **Make Predictions**:
    - Use the fitted equation to predict outcomes for new (X) values.

Examples

Example 1: Simple Linear Regression

Scenario: A company wants to predict the salary ((Y)) of employees based on their years of experience ((X)).

Data:

Years of Experience ((X))	Salary ((Y))
1	40,000
2	50,000
3	60,000
4	70,000
5	80,000

Analysis:

- Fit the equation ($Y = \beta_0 + \beta_1X$).
- Using regression, we find: [$ Y = 30,000 + 10,000 \cdot X $] Here:

- Intercept ((β_0)) = 30,000
- Slope ((β_1)) = 10,000

Prediction:

- For ($X = 6$) (6 years of experience), predicted salary: [$ Y = 30,000 + 10,000 \cdot 6 = 90,000 $]

Example 2: Multiple Linear Regression

Scenario: Predicting house prices ((Y)) based on size in square feet ((X_1)) and number of bedrooms ((X_2)).

Data:

Size ((X_1))	Bedrooms ((X_2))	Price ((Y))
1200	2	300,000
1500	3	350,000
2000	4	450,000

Analysis:

- Fit the equation: [$ Y = \beta_0 + \beta_1X_1 + \beta_2X_2 $]
- After analysis, we find: [$ Y = 150,000 + 100 \cdot X_1 + 50,000 \cdot X_2 $]

Prediction:

- For a house with ($X_1 = 1800$) square feet and ($X_2 = 3$) bedrooms: [$ Y = 150,000 + 100 \cdot 1800 + 50,000 \cdot 3 = 380,000 $]

Applications

1. **Business**: Forecasting sales based on advertising budgets.
2. **Healthcare**: Predicting patient outcomes based on treatment and other factors.
3. **Economics**: Estimating GDP growth based on various economic indicators.

Tools for Linear Regression

1. **Python**:

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
```

Geometric Intuition of Linear Regression

Linear regression, at its core, is about finding a line (or a plane in higher dimensions) that best fits a set of data points. The geometric intuition behind linear regression helps us understand this process in terms of distances, projections, and linear spaces.

Key Concepts in the Geometric Intuition

1. **Data as Points in Space**:
    - Each data point in the dataset can be thought of as a point in a multi-dimensional space.
    - For a dataset with one independent variable ((X)) and one dependent variable ((Y)), the points lie in a 2D plane.
    - For multiple independent variables (($X_1, X_2, \dots, X_p$)) and one dependent variable ((Y)), the points lie in a ($(p+1)$)-dimensional space.
2. **Linear Equation as a Line/Plane**:
    - The regression equation ($Y = \beta_0 + \beta_1X_1 + \beta_2X_2 + \dots + \beta_pX_p$) defines a hyperplane in the feature space.
    - For simple linear regression (($Y = \beta_0 + \beta_1X$)), this hyperplane is a straight line.
3. **Objective of Linear Regression**:
    - The goal is to minimize the vertical distances (residuals) between the observed points ((Y_i)) and the predicted points on the line/plane (($\hat{Y_i}$)).
    - Geometrically, this is equivalent to finding a line or plane that minimizes the sum of the squared vertical distances.

Detailed Geometric Breakdown

1. Simple Linear Regression (2D):

- Imagine a 2D graph with the independent variable (X) on the horizontal axis and the dependent variable (Y) on the vertical axis.
- Each data point ((x_i, y_i)) represents a coordinate on this plane.
- The regression line ($Y = \beta_0 + \beta_1X$):
    - Represents the best-fit line through these points.
    - Is found by minimizing the sum of the squared vertical distances (($y_i - \hat{y_i}$)) from each point to the line.

---

2. Multiple Linear Regression (Higher Dimensions):

- In multiple linear regression, the relationship is modeled in a ($p+1$)-dimensional space, where (p) is the number of independent variables.
- Geometrically:
    - Data points (($X_1, X_2, \dots, X_p, Y$)) are in a ($(p+1)$)-dimensional space.
    - The regression equation defines a hyperplane in this space.
    - The goal is to project the (Y)-vector (actual values) onto the subspace spanned by the (X)-vectors (independent variables).
    - The projected points (($\hat{Y}$)) lie on the hyperplane, minimizing the squared distance to the actual (Y).

Projection Intuition

1. **Vectors in a Linear Space**:
    - Think of the independent variables (($X_1, X_2, \dots, X_p$)) as basis vectors in a vector space.
    - The dependent variable (Y) is another vector in the same space.
2. **Projection of (Y) onto the Space of (X)**:
    - The regression line/plane represents the projection of (Y) onto the space spanned by the independent variables.
    - The projection minimizes the perpendicular (orthogonal) distance from (Y) to the hyperplane.
3. **Residuals as the Orthogonal Component**:
    - The residual vector ((ϵ)) is the difference between (Y) and its projection (($\hat{Y}$)).
    - Geometrically, (ϵ) is orthogonal to the plane spanned by the (X)-vectors.

Geometric Illustration with Examples

Example 1: Simple Linear Regression

- Dataset: [$ X = [1, 2, 3], , Y = [2, 4, 6] $]
- Geometric Interpretation:
    - Points ($(1, 2), (2, 4), (3, 6)$) lie on a straight line in the 2D plane.
    - The regression line ($Y = 2X$) perfectly fits the data, with no residuals.

Example 2: Multiple Linear Regression

- Dataset: [$ X_1 = [1, 2, 3], , X_2 = [4, 5, 6], , Y = [10, 20, 30] $]
- Geometric Interpretation:
    - Points ($(X_1, X_2, Y) = (1, 4, 10), (2, 5, 20), (3, 6, 30)$) lie in a 3D space.

- The regression plane minimizes the distance from (Y)-values to the predicted values ($\hat{Y}$) on the plane.

Why Geometric Intuition is Important

1. **Understanding the Regression Coefficients**:
   - ($\beta_1, \beta_2, \dots$) represent the direction and steepness of the hyperplane in the (X)-space.
   - They indicate how much (Y) changes for a unit change in (X), keeping other variables constant.
2. **Residual Analysis**:
   - Residuals ((ϵ)) represent deviations from the plane and provide insights into model accuracy.
   - If residuals are not random, the linear assumption may not hold.
3. **Interpreting Multicollinearity**:
   - If two independent variables ((X_1) and (X_2)) are highly correlated, their vectors in the space will nearly overlap, making it hard to find a unique regression plane.

Geometric Intuition of Multiple Linear Regression

In **multiple linear regression**, the geometric intuition extends from the 2D plane of simple linear regression into higher-dimensional spaces. The method models the relationship between a dependent variable ((Y)) and multiple independent variables (($X_1, X_2, \dots, X_p$)) by fitting a hyperplane that best approximates the data points in the space.

Understanding the Geometry of Multiple Linear Regression

1. **Data Points as Vectors in Higher Dimensions**:
   - Each data point in multiple linear regression is represented as a vector in a ($(p+1)$)-dimensional space, where (p) is the number of independent variables.
   - Example:
     - If there are 2 independent variables ((X_1, X_2)) and 1 dependent variable ((Y)), the data points exist in 3D space.
2. **Regression Plane (or Hyperplane)**:
   - The regression model fits a hyperplane in the ($(p+1)$)-dimensional space to minimize the error between the observed data points and the predictions.
   - The hyperplane is defined by the equation: [$ Y = \beta_0 + \beta_1X_1 + \beta_2X_2 + \dots + \beta_pX_p $]

- For 2 independent variables, this is a 2D plane in 3D space.
- For 3+ independent variables, it becomes a higher-dimensional hyperplane that is harder to visualize.

3. **Projection of (Y) onto the Space Spanned by (X)**:
    - The independent variables (($X_1, X_2, \dots, X_p$)) form a subspace in the ($(p+1)$)-dimensional space.
    - The goal of regression is to project the vector of observed (Y)-values (($\mathbf{Y}$)) onto this subspace.
    - The projection results in a vector of predicted (Y)-values (($\hat{\mathbf{Y}}$)) that lies in the span of the independent variables.
    - Residuals ((ϵ)) are the components of (Y) orthogonal to the (X)-space.

Key Geometric Components

1. **Dependent Variable Vector ((Y))**:
    - The vector (Y) represents the actual observed values of the dependent variable.
2. **Span of the Independent Variables ((X))**:
    - The independent variables form a subspace in the higher-dimensional space.
    - Each independent variable is a vector, and the span is the space formed by their linear combinations.
3. **Fitted Values (($\hat{Y}$))**:
    - ($\hat{Y}$) is the projection of (Y) onto the subspace of (X).
    - Geometrically, ($\hat{Y}$) is the closest point in the (X)-subspace to the original (Y).
4. **Residuals ((ϵ))**:
    - Residuals are the difference between (Y) and ($\hat{Y}$), i.e., the part of (Y) not explained by the independent variables.
    - Residuals are orthogonal to the (X)-subspace: [$ \epsilon = Y - \hat{Y} $]

Steps in Multiple Linear Regression (Geometrically)

1. **Define the Independent Variable Space**:
    - Each independent variable (($X_1, X_2, \dots, X_p$)) is treated as a vector in (n)-dimensional space, where (n) is the number of observations.
    - These vectors span a subspace in (n)-dimensional space.
2. **Project (Y) onto the Subspace**:

- The dependent variable (Y) is a vector in the same (n)-dimensional space.
- Regression finds the projection of (Y) onto the subspace spanned by the independent variables.
- The projection minimizes the squared distance between (Y) and the subspace.

3. **Compute Coefficients ((β))**:
    - The coefficients ($\beta_0, \beta_1, \dots, \beta_p$) define the orientation of the hyperplane that minimizes the sum of squared residuals.

Geometric Relationships

1. **Orthogonality**:
    - Residuals ((ϵ)) are orthogonal (perpendicular) to the subspace spanned by the independent variables (($X_1, X_2, \dots, X_p$)).
    - Mathematically: [$ X^T \epsilon = 0 $] Where ($X$) is the matrix of independent variables.
2. **Error Minimization**:
    - The regression hyperplane minimizes the squared Euclidean distance between (Y) and the subspace spanned by (X).
    - This ensures the best linear fit in the least-squares sense.
3. **Interpretation of Coefficients**:
    - Each coefficient (β_j) represents the contribution of the corresponding (X_j) to (Y), holding all other variables constant.

Example: Multiple Linear Regression in 3D

Problem:

- Predict house prices ((Y)) based on:
    - Size in square feet ((X_1)).
    - Number of bedrooms ((X_2)).

Dataset:

Size ((X_1))	Bedrooms ((X_2))	Price ((Y))
1200	3	300,000
1500	4	350,000
1800	4	400,000

Geometric Interpretation:

1. **Data Points**:
    - Each observation is a point in 3D space: ((X_1, X_2, Y)).
    - Example: ($(1200, 3, 300,000)$).
2. **Regression Plane**:
    - The regression model fits a plane in 3D space that best approximates these points.
    - Equation: [$ Y = \beta_0 + \beta_1X_1 + \beta_2X_2 $]
    - The plane minimizes the vertical distances from the observed points to the plane.
3. **Projection**:
    - For a given (X_1) and (X_2), the projection onto the plane gives ($\hat{Y}$) (predicted price).
4. **Residuals**:
    - The difference between the observed price ((Y)) and predicted price (($\hat{Y}$)) is the residual.

Generalization to Higher Dimensions

- In 4D or higher spaces, the regression plane becomes a hyperplane.
- The same principles apply:
    - Independent variables span a subspace.
    - Dependent variable (Y) is projected onto this subspace.
    - Residuals remain orthogonal to the subspace.

Lasso and Ridge Regression

Lasso and Ridge Regression are two advanced regression techniques used to handle **multicollinearity** (when independent variables are highly correlated) and prevent **overfitting**. These techniques introduce a penalty term to the linear regression equation to shrink the regression coefficients, making them particularly useful when dealing with high-dimensional datasets.

Core Concept of Regularized Regression

Both Lasso and Ridge Regression are forms of **regularized regression**, which modifies the cost function of Ordinary Least Squares (OLS) regression by adding a penalty term. This penalty discourages large regression coefficients, helping the model generalize better.

1. **Cost Function for Ordinary Linear Regression**: [$ \text{Cost Function} = \text{SSE} = \sum_{i=1}^n (y_i - \hat{y}_i)^2 $] Where:

    - (y_i): Actual value.
    - ($\hat{y}_i$): Predicted value.
    - (n): Number of observations.

2. **Adding a Regularization Term**:

    - Ridge Regression: Adds the sum of the squares of the coefficients (($L2$) norm).
    - Lasso Regression: Adds the sum of the absolute values of the coefficients (($L1$) norm).

Ridge Regression

Cost Function:

[$ \text{Ridge Cost Function} = \sum_{i=1}^n (y_i - \hat{y}*i)^2 + |lambda |sum*{j=1}^p \beta_j^2 $]

- (λ): Regularization parameter that controls the penalty term.
    - When ($\lambda = 0$): Ridge regression becomes ordinary linear regression.
    - As (λ) increases: Coefficients shrink closer to zero but do not become zero.

Key Characteristics:

1. Penalizes large coefficients by adding their squared values.
2. Helps reduce multicollinearity by shrinking correlated coefficients together.
3. Does not perform variable selection (all coefficients are shrunk but retained).

Geometric Intuition:

- Ridge regression constrains the coefficients to lie within a **circular region** (Euclidean distance constraint). The solution is where the error function intersects this region.

Lasso Regression

Cost Function:

[$ \text{Lasso Cost Function} = \sum_{i=1}^n (y_i - \hat{y}*i)^2 + |lambda |sum*{j=1}^p |\beta_j| $]

- (λ): Regularization parameter that controls the penalty term.
    - When ($\lambda = 0$): Lasso regression becomes ordinary linear regression.
    - As (λ) increases: Some coefficients are shrunk to exactly zero.

Key Characteristics:

1. Performs both **shrinkage** and **variable selection** (some coefficients are reduced to zero, effectively removing irrelevant features).
2. Particularly useful in high-dimensional datasets where many variables may be irrelevant.

Geometric Intuition:

- Lasso regression constrains the coefficients to lie within a **diamond-shaped region** (Manhattan distance constraint). The sharp edges of the diamond encourage some coefficients to shrink exactly to zero.

Comparison of Lasso and Ridge Regression

Feature	Ridge Regression	Lasso Regression
Penalty Term	($\lambda \sum_{j=1}^p \beta_j^2$)	($\lambda \sum_{j=1}^p
Effect on Coefficients	Shrinks coefficients toward zero but retains all variables.	Shrinks coefficients and sets some to zero, performing feature selection.
Dimensionality	Useful when all features contribute to the output.	Useful when some features are irrelevant or redundant.
Geometric Constraint	Circular constraint (L2 norm).	Diamond-shaped constraint (L1 norm).

Elastic Net Regression

Elastic Net combines Lasso and Ridge regression: [$ \text{Elastic Net Cost Function} = \sum_{i=1}^n (y_i - \hat{y}*i)^2 + |lambda_1 |sum*{j=1}^p |\beta_j| + \lambda_2 \sum_{j=1}^p \beta_j^2 $]

- Balances between Lasso and Ridge by tuning the parameters (λ_1) and (λ_2).
- Useful when features are highly correlated and variable selection is required.

Example: Comparing Ridge and Lasso Regression

Problem:

You want to predict house prices ((Y)) based on:

- (X_1): Size of the house.
- (X_2): Number of bedrooms.
- (X_3): Distance to the city center.

Dataset:

(X_1) (sq. ft)	(X_2) (# bedrooms)	(X_3) (miles)	(Y) (price)
1200	3	10	300,000

1500	4	8	350,000
1800	4	5	400,000

Observations:

- (X_1) and (X_2) are highly correlated (multicollinearity issue).
- Ridge regression will shrink coefficients but retain both (X_1) and (X_2).
- Lasso regression will likely set the coefficient of one of these to zero if it finds it redundant.

Results:

1. **Ridge Regression**:
    - ($\beta_1 = 150, \beta_2 = 0.5, \beta_3 = -10$).
    - Both (X_1) and (X_2) are retained but with reduced coefficients.
2. **Lasso Regression**:
    - ($\beta_1 = 200, \beta_2 = 0, \beta_3 = -12$).
    - (X_2) is removed (coefficient is zero), suggesting it is not essential.

Applications

1. **Ridge Regression**:
    - Used when all features are potentially useful.
    - Examples:
        - Predicting stock prices using multiple correlated financial indicators.
        - Medical studies with many correlated health metrics.
2. **Lasso Regression**:
    - Used when some features are irrelevant and should be excluded.
    - Examples:
        - Feature selection in text classification problems (e.g., spam detection).
        - Genomics, where only a few genes influence a trait.

Implementation in Python

Ridge Regression:

```
from sklearn.linear_model import Ridge
ridge_model = Ridge(alpha=1.0)  # Alpha is the regularization parameter
(λ)
ridge_model.fit(X_train, y_train)
```

Lasso Regression:

```
from sklearn.linear_model import Lasso
lasso_model = Lasso(alpha=0.1)  # Alpha is the regularization parameter
(λ)
lasso_model.fit(X_train, y_train)
```

Polynomial Regression

Polynomial Regression is an extension of linear regression where the relationship between the independent variable ((X)) and the dependent variable ((Y)) is modeled as a polynomial function of degree (n). It is useful when the data shows a non-linear relationship that a straight line cannot capture.

Mathematical Representation

The model equation for polynomial regression is: [$ Y = \beta_0 + \beta_1X + \beta_2X^2 + \beta_3X^3 + \dots + \beta_nX^n + \epsilon $] Where:

- (Y): Dependent variable.
- (X): Independent variable.
- ($\beta_0, \beta_1, \beta_2, \dots, \beta_n$): Coefficients to be estimated.
- (n): Degree of the polynomial.
- (ϵ): Error term.

For example:

- If ($n = 2$), it's a quadratic regression (parabolic curve).
- If ($n = 3$), it's a cubic regression.

How Polynomial Regression Works

1. **Transforming the Input Data**:
    - The original feature (X) is transformed into polynomial features (($X^2, X^3, \dots, X^n$)).
    - For example, if ($X = [1, 2, 3]$) and degree = 2, the transformed data becomes: [$ [1, 1^2], [2, 2^2], [3, 3^2] = [1, 1], [2, 4], [3, 9] $]
2. **Applying Linear Regression**:
    - Once the input data is transformed, linear regression is applied to fit a polynomial curve

to the data.

3. **Curve Fitting**:
    - The model tries to minimize the sum of squared residuals (the difference between observed and predicted values) to fit the curve.

Key Concepts in Polynomial Regression

1. **Degree of the Polynomial**:
    - Determines the flexibility of the curve.
    - A higher degree results in more complex curves, but too high a degree can lead to overfitting.
2. **Overfitting and Underfitting**:
    - **Underfitting**: A polynomial of too low degree (e.g., a straight line) cannot capture the data's complexity.
    - **Overfitting**: A polynomial of too high degree fits the noise in the data rather than the actual trend.
3. **Regularization**:
    - Techniques like Ridge and Lasso regression can be used with polynomial regression to prevent overfitting by penalizing large coefficients.

Steps in Polynomial Regression

1. **Visualize the Data**:
    - Plot the data to observe if the relationship between (X) and (Y) is non-linear.
2. **Transform Features**:
    - Convert (X) into polynomial features (($X, X^2, X^3, \dots, X^n$)).
3. **Fit the Model**:
    - Use linear regression on the transformed features to estimate the coefficients.
4. **Evaluate the Model**:
    - Use metrics like Mean Squared Error (MSE), R-squared ((R^2)), and visualization of the fit to assess performance.

Example

Scenario:

A company wants to predict sales ((Y)) based on advertising budget ((X)).

Dataset:

Advertising Budget ((X))	Sales ((Y))
1	5
2	9
3	15
4	25
5	39

Observations:

- A linear regression model would fail to capture the increasing trend.
- Polynomial regression can fit a curve to better model the relationship.

Model:

Let's fit a degree-2 polynomial (($n = 2$)): [$ Y = \beta_0 + \beta_1X + \beta_2X^2 $]

After fitting the model: [$ Y = 2 + 3X + 0.5X^2 $]

Predictions:

- For ($X = 6$): [$ Y = 2 + 3(6) + 0.5(6^2) = 2 + 18 + 18 = 38 $]

Applications of Polynomial Regression

1. **Economics**:
    - Modeling GDP growth, market trends, and economic indicators.
2. **Healthcare**:
    - Predicting disease progression or treatment effectiveness.
3. **Engineering**:
    - Estimating physical phenomena like material strength or pressure relationships.
4. **Marketing**:
    - Predicting sales based on advertising or pricing.

Advantages and Disadvantages

Advantages	Disadvantages

Can model non-linear relationships.	Prone to overfitting with high-degree polynomials.
Easy to implement and interpret.	Sensitive to outliers.
Works well with small datasets.	High-degree polynomials may lack generalizability.

Implementation in Python

Code Example:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Sample data
X = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)
Y = np.array([5, 9, 15, 25, 39])

# Transform features to polynomial (degree 2)
poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X)

# Fit the model
model = LinearRegression()
model.fit(X_poly, Y)

# Predictions
Y_pred = model.predict(X_poly)

# Evaluate
print("Coefficients:", model.coef_)
print("Intercept:", model.intercept_)
print("MSE:", mean_squared_error(Y, Y_pred))

# Plot
plt.scatter(X, Y, color='blue', label='Actual')
plt.plot(X, Y_pred, color='red', label='Predicted')
plt.legend()
plt.show()
```

Output:

- Coefficients: [0, 3, 0.5] (corresponding to (X) and (X^2)).
- Intercept: 2.
- MSE: A numerical value indicating model fit.

Diagnostics of Regression

Regression diagnostics involve evaluating the performance and validity of a regression model. Metrics like **R-squared (R^2)**, **Mean Squared Error (MSE)**, and **Root Mean Squared Error (RMSE)** help assess the model's goodness-of-fit, predictive power, and accuracy. Let’s dive into each metric.

---

1. R-squared ((R^2))

Definition:

R-squared is a statistical measure that indicates the proportion of the variance in the dependent variable ((Y)) that is explained by the independent variable(s) ((X)) in the regression model.

[$ R^2 = 1 - \frac{\text{SS}*{|text{Residual}}}{|text{SS}*}{\text{Total}} $]

Where:

- ($\text{SS}_{\text{Residual}} = \sum (y_i - \hat{y}_i)^2$): Sum of squared residuals (unexplained variance).
- ($\text{SS}_{\text{Total}} = \sum (y_i - \bar{y})^2$): Total variance in ($Y$).

Interpretation:

- ($R^2 = 1$): Perfect fit (model explains 100% of the variance in (Y)).
- ($R^2 = 0$): Model explains none of the variance in (Y) (essentially no predictive power).
- Higher (R^2): Better fit, but very high values can indicate overfitting.

Limitations:

- Adding more variables always increases (R^2), even if they are irrelevant.
- Does not indicate if the model is appropriate (e.g., it may not reflect non-linearity or multicollinearity).

---

2. Mean Squared Error (MSE)

Definition:

MSE measures the average squared difference between the observed actual values ((y_i)) and the predicted values (($\hat{y}_i$)).

[$ \text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 $]

Where:

- (n): Number of observations.

Interpretation:

- MSE quantifies the overall error of the model. Smaller MSE values indicate better model performance.
- Squaring amplifies the impact of larger errors, making MSE sensitive to outliers.

---

3. Root Mean Squared Error (RMSE)

Definition:

RMSE is the square root of MSE. It provides a measure of error in the same units as the dependent variable ((Y)).

[$ \text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} $]

Interpretation:

- Like MSE, RMSE penalizes large errors more than smaller ones.
- RMSE is often preferred because it is easier to interpret in the context of the data.

Comparison of (R^2), MSE, and RMSE

Metric	Description	Scale	Use Cases
(R^2)	Proportion of variance explained by the model.	Unitless (0 to 1).	Model fit and explanatory power.
MSE	Average squared prediction error.	Squared units of (Y).	Error magnitude for model training.
RMSE	Square root of MSE.	Same units as (Y).	Direct interpretability of errors.

Examples

Dataset:

(X)	(Y)	Predicted ($Y (\hat{Y})$)
1	2	1.8
2	4	4.2
3	6	6.1
4	8	7.9
5	10	9.8

1. **Calculate (R^2):**
    - ($\text{SS}_{\text{Residual}} = (2 - 1.8)^2 + (4 - 4.2)^2 + (6 - 6.1)^2 + (8 - 7.9)^2 + (10 - 9.8)^2 = 0.06$)
    - ($\text{SS}_{\text{Total}} = (2 - 6)^2 + (4 - 6)^2 + (6 - 6)^2 + (8 - 6)^2 + (10 - 6)^2 = 40$)
    - ($R^2 = 1 - \frac{\text{SS}_{\text{Residual}}}{\text{SS}_{\text{Total}}} = 1 - \frac{0.06}{40} = 0.9985$).
2. **Calculate MSE:** [$ \text{MSE} = \frac{1}{5} \left[(2 - 1.8)^2 + (4 - 4.2)^2 + (6 - 6.1)^2 + (8 - 7.9)^2 + (10 - 9.8)^2 \right] = 0.012 $]
3. **Calculate RMSE:** [$ \text{RMSE} = \sqrt{\text{MSE}} = \sqrt{0.012} = 0.109 $]

Applications in Model Diagnostics

1. **Model Fit**:
    - Use (R^2) to evaluate how well the independent variables explain the variance in (Y).
2. **Prediction Error**:
    - Use MSE and RMSE to understand how far off predictions are from actual values.
3. **Model Comparison**:
    - Compare RMSE values across models to identify the best-performing one.

Python Implementation

```
import numpy as np
from sklearn.metrics import mean_squared_error, r2_score

# Data
y_actual = np.array([2, 4, 6, 8, 10])
y_pred = np.array([1.8, 4.2, 6.1, 7.9, 9.8])

# R-squared
r2 = r2_score(y_actual, y_pred)
print("R-squared:", r2)

# MSE
mse = mean_squared_error(y_actual, y_pred)
print("MSE:", mse)
```

```
# RMSE
rmse = np.sqrt(mse)
print("RMSE:", rmse)
```
