* What is the difference between R² and adjusted R²?
* What is overfitting in decision tree regression, and how to control it?
* What happens when α = 0 in Ridge Regression?

# Regression in machine learning

Regression in machine learning refers to a [**supervised learning**](https://www.geeksforgeeks.org/machine-learning/supervised-machine-learning/)technique where the goal is to predict a continuous numerical value based on one or more independent features. It finds relationships between variables so that predictions can be made. we have two types of variables present in regression:

* **Dependent Variable (Target)**: The variable we are trying to predict e.g house price.
* **Independent Variables (Features)**: The input variables that influence the prediction e.g locality, number of rooms.

Regression analysis problem works with if output variable is a real or continuous value such as “salary” or “weight”. Many different regression models can be used but the simplest model in them is linear regression.

**Types of Regression**

Regression can be classified into different types based on the number of predictor variables and the nature of the relationship between variables:

**1. Simple Linear Regression**

[**Linear regression**](https://www.geeksforgeeks.org/machine-learning/ml-linear-regression/)is one of the simplest and most widely used statistical models. This assumes that there is a linear relationship between the independent and dependent variables. This means that the change in the dependent variable is proportional to the change in the independent variables.For example predicting the price of a house based on its size.

**2. Multiple Linear Regression**

[Multiple linear regression](https://www.geeksforgeeks.org/machine-learning/ml-multiple-linear-regression-using-python/) extends simple linear regression by using multiple independent variables to predict target variable.For example predicting the price of a house based on multiple features such as size, location, number of rooms, etc.

**3. Polynomial Regression**

[Polynomial regression](https://www.geeksforgeeks.org/machine-learning/python-implementation-of-polynomial-regression/)is used to model with non-linear relationships between the dependent variable and the independent variables. It adds polynomial terms to the linear regression model to capture more complex relationships.For example when we want to predict a non-linear trend like population growth over time we use polynomial regression.

**4. Ridge & Lasso Regression**

[Ridge & lasso regression](https://www.geeksforgeeks.org/machine-learning/ridge-regression-vs-lasso-regression/) are regularized versions of linear regression that help avoid overfitting by penalizing large coefficients.When there’s a risk of overfitting due to too many features we use these type of regression algorithms.

**5. Support Vector Regression (SVR)**

SVR is a type of regression algorithm that is based on the [Support Vector Machine (SVM)](https://www.geeksforgeeks.org/machine-learning/support-vector-machine-algorithm/) algorithm. SVM is a type of algorithm that is used for classification tasks but it can also be used for regression tasks. SVR works by finding a hyperplane that minimizes the sum of the squared residuals between the predicted and actual values.

**6. Decision Tree Regression**

[Decision tree](https://www.geeksforgeeks.org/machine-learning/decision-tree/) Uses a tree-like structure to make decisions where each branch of tree represents a decision and leaves represent outcomes. For example predicting customer behavior based on features like age, income, etc there we use decison tree regression.

**7. Random Forest Regression**

[Random Forest](https://www.geeksforgeeks.org/machine-learning/random-forest-algorithm-in-machine-learning/) is a ensemble method that builds multiple decision trees and each tree is trained on a different subset of the training data. The final prediction is made by averaging the predictions of all of the trees. For example customer churn or sales data using this.

**Regression Evaluation Metrics**

Evaluation in machine learning measures the performance of a model. Here are some popular evaluation metrics for regression:

* [**Mean Absolute Error (MAE):**](https://www.geeksforgeeks.org/python/how-to-calculate-mean-absolute-error-in-python/) The average absolute difference between the predicted and actual values of the target variable.
* [**Mean Squared Error (MSE):**](https://www.geeksforgeeks.org/python/python-mean-squared-error/)The average squared difference between the predicted and actual values of the target variable.
* [**Root Mean Squared Error (RMSE)**](https://www.geeksforgeeks.org/software-engineering/rmse-root-mean-square-error-in-matlab/)**:** Square root of the mean squared error.
* [**Huber Loss:**](https://www.geeksforgeeks.org/machine-learning/sklearn-different-loss-functions-in-sgd/) A hybrid loss function that transitions from MAE to MSE for larger errors, providing balance between robustness and MSE’s sensitivity to outliers.
* [R2 – Score](https://www.geeksforgeeks.org/machine-learning/python-coefficient-of-determination-r2-score/): Higher values indicate better fit ranging from 0 to 1.

**Regression Model Machine Learning**

Let's take an example of linear regression. We have a [**Housing data set**](https://vincentarelbundock.github.io/Rdatasets/csv/Ecdat/Housing.csv) and we want to predict the price of the house. Following is the python code for it.



import matplotlib

matplotlib.use('TkAgg') # General backend for plots

import matplotlib.pyplot as plt

import numpy as np

from sklearn import datasets, linear\_model

import pandas as pd

# Load dataset

df = pd.read\_csv("Housing.csv")

​

# Extract features and target variable

Y = df['price']

X = df['lotsize']

​

# Reshape for compatibility with scikit-learn

X = X.to\_numpy().reshape(len(X), 1)

Y = Y.to\_numpy().reshape(len(Y), 1)

​

# Split data into training and testing sets

X\_train = X[:-250]

X\_test = X[-250:]

Y\_train = Y[:-250]

Y\_test = Y[-250:]

​

# Plot the test data

plt.scatter(X\_test, Y\_test, color='black')

plt.title('Test Data')

plt.xlabel('Size')

plt.ylabel('Price')

plt.xticks(())

plt.yticks(())

​

# Train linear regression model

regr = linear\_model.LinearRegression()

regr.fit(X\_train, Y\_train)

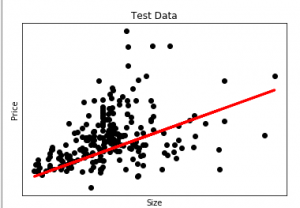
​

# Plot predictions

plt.plot(X\_test, regr.predict(X\_test), color='red', linewidth=3)

plt.show()

**Output:**



Here in this graph we plot the test data. The red line indicates the best fit line for predicting the price.

To make an individual prediction using the linear regression model:

**print("Predicted price for a lot size of 5000: " + str(round(regr.predict([[5000]])[0][0])))**

**Applications of Regression**

* **Predicting prices:** Used to predict the price of a house based on its size, location and other features.
* **Forecasting trends:** Model to forecast the sales of a product based on historical sales data.
* **Identifying risk factors:** Used to identify risk factors for heart patient based on patient medical data.
* **Making decisions:** It could be used to recommend which stock to buy based on market data.

**Advantages of Regression**

* Easy to understand and interpret.
* Robust to outliers.
* Can handle both linear relationships easily.

**Disadvantages of Regression**

* Assumes linearity.
* Sensitive to situation where two or more independent variables are highly correlated with each other i.e multicollinearity.
* May not be suitable for highly complex relationships.

**Conclusion**

Regression in machine learning is a fundamental technique for predicting continuous outcomes based on input features. It is used in many real-world applications like price prediction, trend analysis and risk assessment. With its simplicity and effectiveness regression is used to understand relationships in data.

[geekforgeeksregre](https://www.geeksforgeeks.org/machine-learning/regression-in-machine-learning/)

[builtinregre](https://builtin.com/data-science/regression-machine-learning)

**Linear Regression in Machine learning**

Linear regression is a type of [**supervised machine-learning algorithm**](https://www.geeksforgeeks.org/machine-learning/supervised-machine-learning/) that learns from the labelled datasets and maps the data points with most optimized linear functions which can be used for prediction on new datasets. It 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.

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**](https://www.geeksforgeeks.org/maths/least-square-method/). 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=yᵢ−y^ᵢResidual=yᵢ−y^​ᵢ*

**Where:**

* yᵢ*y*ᵢ is the actual observed value
* y^ᵢ*y*^​ᵢ is the predicted value from the line for that xᵢ*x*ᵢ

The least squares method minimizes the sum of the squared residuals:

*Sumofsquarederrors(SSE)=Σ(yᵢ−y^ᵢ)²Sumofsquarederrors(SSE)=Σ(yᵢ−y^​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.*

**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:

*h(x)=β₀+β₁xh(x)=β₀+β₁x*

**Where:**

* h(x)(ory^)*h*(*x*)(*ory*^​) is the predicted value of the dependent variable (y).
* x x*x*is the independent variable.
* β₀*β*₀ is the intercept, representing the value of y when x is 0.
* β₁*β*₁ is the slope, indicating how much y changes for each unit change in x.

For **multiple linear regression** (with more than one independent variable), the hypothesis function expands to:

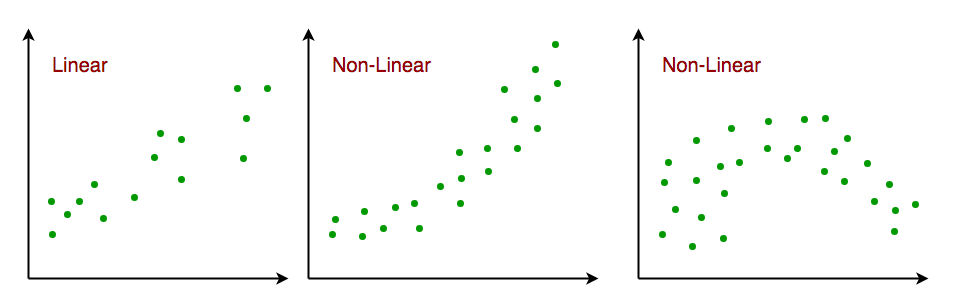
*h(x₁,x₂,...,xₖ)=β₀+β₁x₁+β₂x₂+...+βₖxₖh(x₁,x₂,...,xₖ)=β₀+β₁x₁+β₂x₂+...+βₖxₖ*

**Where:**

* x₁,x₂,...,xₖ*x*₁,*x*₂,...,*x*ₖ are the independent variables.
* β₀*β*₀ is the intercept.
* β₁,β₂,...,βₖ*β*₁,*β*₂,...,*β*ₖ are the coefficients, representing the influence of each respective independent variable on the predicted output.

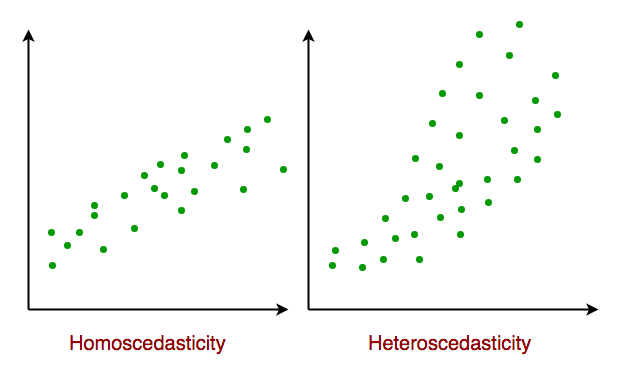
**Assumptions of the Linear Regression**

**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***](https://www.geeksforgeeks.org/machine-learning/multicollinearity-in-regression-analysis/)*.*

**Types of Linear Regression**

When there is only one independent feature it is known as Simple Linear Regression or [Univariate Linear Regression](https://www.geeksforgeeks.org/python/univariate-linear-regression-in-python/) and when there are more than one feature it is known as Multiple Linear Regression or [Multivariate Regression](https://www.geeksforgeeks.org/machine-learning/multivariate-regression/).

**1. Simple Linear Regression**

[Simple linear regression](https://www.geeksforgeeks.org/machine-learning/simple-linear-regression-in-python/) 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.

**Formula:**

*y^=θ0+θ1xy^​=θ0​+θ1​x*

**Where:**

* y^*y*^​​ is the predicted value
* x*x*is the input (independent variable)
* θ0*θ*0​ is the intercept (value of y^​*y*^​​ when x=0)
* θ1*θ*1​​ is the slope or coefficient (how much y^*y*^​​ changes with one unit of x)

**Example:**

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

**2. Multiple Linear Regression**

[Multiple linear regression](https://www.geeksforgeeks.org/machine-learning/ml-multiple-linear-regression-using-python/) involves more than one independent variable and one dependent variable. The equation for multiple linear regression is:

*y^=θ0+θ1x1+θ2x2+⋯+θnxny^​=θ0​+θ1​x1​+θ2​x2​+⋯+θn​xn​*

**where:**

* y^*y*^​​ is the predicted value
* x1,x2,…,xn*x*1​,*x*2​,…,*xn*​ are the independent variables
* θ1,θ2,…,θn*θ*1​,*θ*2​,…,*θn*​ are the coefficients (weights) corresponding to each predictor.
* θ0*θ*0​ is the intercept.

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.

**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 Y^     *Y*^     and the true value Y and it is called [cost function](https://www.geeksforgeeks.org/microeconomics/what-is-cost-function/) or the[loss function](https://www.geeksforgeeks.org/machine-learning/ml-common-loss-functions/).

In Linear Regression, the Mean Squared Error (MSE) cost function is employed, which calculates the average of the squared errors between the predicted values y^i*y*^​*i*​ and the actual values yi*yi*​. The purpose is to determine the optimal values for the intercept θ1*θ*1​ and the coefficient of the input feature θ2*θ*2​ providing the best-fit line for the given data points. The linear equation expressing this relationship is y^i=θ1+θ2xi*y*^​*i*​=*θ*1​+*θ*2​*xi*​.

MSE function can be calculated as:

*Cost function(J)=1n∑ni(yi^−yi)2Cost function(J)=n1​∑ni​(yi​^​−yi​)2*

Utilizing the MSE function, the iterative process of gradient descent is applied to update the values of \θ1&θ2*θ*1​&*θ*2​. 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 \(\theta\_1\) and \(\theta\_2\) 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*](https://www.geeksforgeeks.org/machine-learning/gradient-descent-in-linear-regression/)

**Evaluation Metrics for Linear Regression**

A variety of [evaluation measures](https://www.geeksforgeeks.org/machine-learning/metrics-for-machine-learning-model/) 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)](https://www.geeksforgeeks.org/python/python-mean-squared-error/) is an evaluation metric that calculates the average of the squared differences between the actual and predicted values for all the data points. The difference is squared to ensure that negative and positive differences don't cancel each other out.

*MSE=1n∑i=1n(yi−yi^)2MSE=n1​∑i=1n​(yi​−yi​​)2*

Here,

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

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

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

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

Mathematically MAE is expressed as:

*MAE=1n∑i=1n∣Yi−Yi^∣MAE=n1​∑i=1n​∣Yi​−Yi​​∣*

Here,

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

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

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

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

*RMSE=RSSn=∑i=2n(yiactual−yipredicted)2nRMSE=nRSS​​=n∑i=2n​(yiactual​−yipredicted​)2​​*

Rather than dividing the entire number of data points in the model by the number of degrees of freedom, one must divide the sum of the squared residuals to obtain an unbiased estimate. Then, this figure is referred to as the Residual Standard Error (RSE).

In mathematical notation, it can be expressed as:

*RMSE=RSSn=∑i=2n(yiactual−yipredicted)2(n−2)RMSE=nRSS​​=(n−2)∑i=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](https://www.geeksforgeeks.org/maths/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**](https://www.geeksforgeeks.org/maths/residual-sum-of-squares/#:~:text=Residual%20sum%20of%20squares%20is%20used%20to%20calculate%20the%20variance,squares%2C%20the%20better%20the%20model.)(RSS): The sum of squares of the residual for each data point in the plot or data is known as the residual sum of squares or RSS. It is a measurement of the difference between the output that was observed and what was anticipated.

*RSS=∑i=1n(yi−b0−b1xi)2RSS=∑i=1n​(yi​−b0​−b1​xi​)2*

* **Total Sum of Squares (TSS):**The sum of the data points' errors from the answer variable's mean is known as the total sum of squares or TSS.

*TSS=∑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 R2*R*2measures the proportion of variance in the dependent variable that is explained by independent variables in a regression model. [Adjusted R-square](https://www.geeksforgeeks.org/machine-learning/ml-adjusted-r-square-in-regression-analysis/) accounts the number of predictors in the model and penalizes the model for including irrelevant predictors that don't contribute significantly to explain the variance in the dependent variables.

Mathematically, adjusted R2*R*2is expressed as:

*AdjustedR2=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 Models**

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

[Lasso Regression](https://www.geeksforgeeks.org/machine-learning/implementation-of-lasso-regression-from-scratch-using-python/) is a technique used for regularizing a linear regression model, it adds a penalty term to the linear regression objective function to prevent [overfitting](https://www.geeksforgeeks.org/machine-learning/underfitting-and-overfitting-in-machine-learning/).

The objective function after applying lasso regression is:

*J(θ)=12m∑i=1m(yi^−yi)2+λ∑j=1n∣θj∣J(θ)=2m1​∑i=1m​(yi​​−yi​)2+λ∑j=1n​∣θj​∣*

* the first term is the least squares loss, representing the squared difference between predicted and actual values.
* the second term is the L1 regularization term, it penalizes the sum of absolute values of the regression coefficient θj.

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

[Ridge regression](https://www.geeksforgeeks.org/machine-learning/implementation-of-ridge-regression-from-scratch-using-python/) 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:

*J(θ)=12m∑i=1m(yi^−yi)2+λ∑j=1nθj2J(θ)=2m1​∑i=1m​(yi​​−yi​)2+λ∑j=1n​θj2​*

* the first term is the least squares loss, representing the squared difference between predicted and actual values.
* the second term is the L1 regularization term, it penalizes the sum of square of values of the regression coefficient θj.

**3. Elastic Net Regression**

[Elastic Net Regression](https://www.geeksforgeeks.org/machine-learning/implementation-of-elastic-net-regression-from-scratch/) is a hybrid regularization technique that combines the power of both L1 and L2 regularization in linear regression objective.

*J(θ)=12m∑i=1m(yi^−yi)2+αλ∑j=1n∣θj∣+12(1−α)λ∑j=1nθj2J(θ)=2m1​∑i=1m​(yi​​−yi​)2+αλ∑j=1n​∣θj​∣+21​(1−α)λ∑j=1​nθj2​*

* the first term is least square loss.
* the second term is L1 regularization and third is ridge regression.
* λ*λ*is the overall regularization strength.
* α*α*controls the mix between L1 and L2 regularization.

Now that we have learned how to make a linear regression model, now we will implement it.

**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](https://media.geeksforgeeks.org/wp-content/uploads/20240320114716/data_for_lr.csv)

url = 'https://media.geeksforgeeks.org/wp-content/uploads/20240320114716/data\_for\_lr.csv'

data = pd.read\_csv(url)

data = data.dropna()

train\_input = np.array(data.x[0:500]).reshape(500, 1)

train\_output = np.array(data.y[0:500]).reshape(500, 1)

test\_input = np.array(data.x[500:700]).reshape(199, 1)

test\_output = np.array(data.y[500:700]).reshape(199, 1)

**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

**class** **LinearRegression**:

**def** \_\_init\_\_(self):

self.parameters = {}

**def** forward\_propagation(self, train\_input):

m = self.parameters['m']

c = self.parameters['c']

predictions = np.multiply(m, train\_input) + c

**return** predictions

**def** cost\_function(self, predictions, train\_output):

cost = np.mean((train\_output - predictions) \*\* 2)

**return** cost

**def** backward\_propagation(self, train\_input, train\_output, predictions):

derivatives = {}

df = (predictions-train\_output)

dm = 2 \* np.mean(np.multiply(train\_input, df))

dc = 2 \* np.mean(df)

derivatives['dm'] = dm

derivatives['dc'] = dc

**return** derivatives

**def** update\_parameters(self, derivatives, learning\_rate):

self.parameters['m'] = self.parameters['m'] - learning\_rate \* derivatives['dm']

self.parameters['c'] = self.parameters['c'] - learning\_rate \* derivatives['dc']

**def** train(self, train\_input, train\_output, learning\_rate, iters):

self.parameters['m'] = np.random.uniform(0, 1) \* -1

self.parameters['c'] = np.random.uniform(0, 1) \* -1

self.loss = []

fig, ax = plt.subplots()

x\_vals = np.linspace(min(train\_input), max(train\_input), 100)

line, = ax.plot(x\_vals, self.parameters['m'] \* x\_vals + self.parameters['c'], color='red', label='Regression Line')

ax.scatter(train\_input, train\_output, marker='o', color='green', label='Training Data')

ax.set\_ylim(0, max(train\_output) + 1)

**def** update(frame):

predictions = self.forward\_propagation(train\_input)

cost = self.cost\_function(predictions, train\_output)

derivatives = self.backward\_propagation(train\_input, train\_output, predictions)

self.update\_parameters(derivatives, learning\_rate)

line.set\_ydata(self.parameters['m'] \* x\_vals + self.parameters['c'])

self.loss.append(cost)

print("Iteration = **{}**, Loss = **{}**".format(frame + 1, cost))

**return** line,

ani = FuncAnimation(fig, update, frames=iters, interval=200, blit=**True**)

ani.save('linear\_regression\_A.gif', writer='ffmpeg')

plt.xlabel('Input')

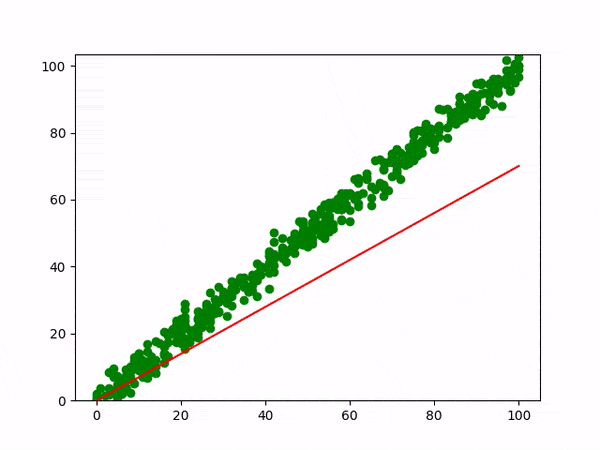
plt.ylabel('Output')

plt.title('Linear Regression')

plt.legend()

plt.show()

**return** self.parameters, self.loss

****

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).

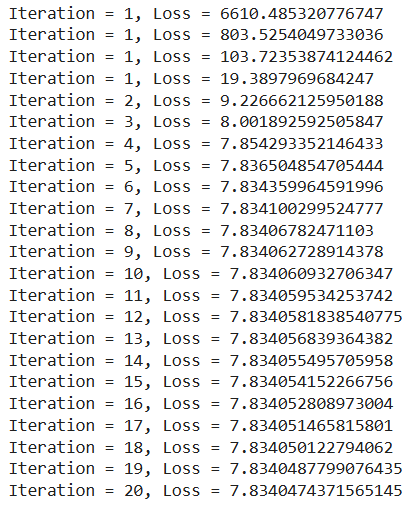
* **Positive Linear Regression Line**: A positive linear regression line indicates a direct relationship between the independent variable (X*X*) and the dependent variable (Y*Y*). This means that as the value of X increases, the value of Y also increases. The slope of a positive linear regression line 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 (X*X*) and the dependent variable (Y*Y*). 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**

linear\_reg = LinearRegression()

parameters, loss = linear\_reg.train(train\_input, train\_output, 0.0001, 20)

**Output**:

Model Training

**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*](https://www.geeksforgeeks.org/machine-learning/linear-regression-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.

# Implementation of Polynomial Regression

Polynomial Regression is a form of linear regression where the relationship between the independent variable (x) and the dependent variable (y) is modelled as an nth*nth* degree polynomial. It is useful when the data exhibits a non-linear relationship allowing the model to fit a curve to the data.

**Need for Polynomial Regression**

* **Non-linear Relationships:**Polynomial regression is used when the relationship between the independent variable (input) and dependent variable (output) is non-linear. Unlike linear regression which fits a straight line, it fits a polynomial equation to capture the curve in the data.
* **Better Fit for Curved Data:** When a researcher hypothesizes a curvilinear relationship, polynomial terms are added to the model. A linear model often results in residuals with noticeable patterns which shows a poor fit. It can capture these non-linear patterns effectively.
* **Flexibility and Complexity:**It does not assume all independent variables are independent. By introducing higher-degree terms, it allows for more flexibility and can model more complex, curvilinear relationships between variables.

**How does a Polynomial Regression work?**

Polynomial regression is an extension of [linear regression](https://www.geeksforgeeks.org/machine-learning/ml-linear-regression/) where higher-degree terms are added to model non-linear relationships. The general form of the equation for a polynomial regression of degree n*n*is:

y=β0+β1x+β2x2+…+βnxn+ϵ*y*=*β*0​+*β*1​*x*+*β*2​*x*2+…+*βn*​*xn*+*ϵ*

where:

* y*y* is the dependent variable.
* x*x* is the independent variable.
* β0,β1,…,βn*β*0​,*β*1​,…,*βn*​​ are the coefficients of the polynomial terms.
* n*n*is the degree of the polynomial.
* ϵ*ϵ* represents the error term.

The goal of regression analysis is to model the expected value of a dependent variable y*y* in terms of an independent variable x*x*. In simple linear regression, this relationship is modeled as:

y=a+bx+e*y*=*a*+*bx*+*e*

Here

* y*y* is a dependent variable
* a*a* is the y-intercept, b*b* is the slope
* e*e* is the error term

However in cases where the relationship between the variables is nonlinear such as modelling chemical synthesis based on temperature, a linear model may not be sufficient. Instead, we use polynomial regression which introduces higher-degree terms such as x2*x*2 to better capture the relationship.

For example, a quadratic model can be written as:

y=a+b1x+b2x2+e*y*=*a*+*b*1​*x*+*b*2​*x*2+*e*

Here:

* y*y* is the dependent variable on x*x*
* a*a*is the y-intercept and e*e*is the error rate.

In general, polynomial regression can be extended to the nth degree:

 y=a+b1x+b2x2+....+bnxn*y*=*a*+*b*1​*x*+*b*2​*x*2+....+*bn*​*xn*

While the regression function is linear in terms of the unknown coefficients 𝑏0,𝑏1,…,𝑏𝑛*b*0​,*b*1​,…,*bn*​, the model itself captures non-linear patterns in the data. The coefficients are estimated using techniques like Least Square technique to minimize the error between predicted and actual values.

Choosing the right polynomial degree n*n* is important: a higher degree may fit the data more closely but it can lead to overfitting. The degree should be selected based on the complexity of the data. Once the model is trained, it can be used to make predictions on new data, capturing non-linear relationships and providing a more accurate model for real-world applications.

**Real-Life Example for Polynomial Regression**

Let’s consider an example in the field of finance where we analyze the relationship between an employee's years of experience and their corresponding salary. If we check that the relationship might not be linear, polynomial regression can be used to model it more accurately.

**Example Data:**

| **Years of Experience** | **Salary (in dollars)** |
| --- | --- |
| **1** | 50,000 |
| **2** | 55,000 |
| **3** | 65,000 |
| **4** | 80,000 |
| **5** | 110,000 |
| **6** | 150,000 |
| **7** | 200,000 |

Now, let's apply polynomial regression to model the relationship between years of experience and salary. We'll use a quadratic polynomial (degree 2) which includes both linear and quadratic terms for better fit. The quadratic polynomial regression equation is:

Salary=β0+β1×Experience+β2​×Experience2+ϵ*Salary*=*β*0​+*β*1​×*Experience*+*β*2​​×*Experience*2+*ϵ*

To find the coefficients β0,β1,β2*β*0​,*β*1​,*β*2​ that minimize the difference between the predicted and actual salaries, we can use the[**Least Squares method**](https://www.geeksforgeeks.org/maths/least-square-method/)**.** The objective is to minimize the sum of squared differences between the predicted salaries and the actual data points which allows us to fit a model that captures the non-linear progression of salary with respect to experience.

**Implementation of Polynomial Regression**

Here we will see how to implement polynomial regression using Python.

**Step 1: Importing Required Libraries**

We'll using [Pandas](https://www.geeksforgeeks.org/pandas/pandas-tutorial/), [NumPy](https://www.geeksforgeeks.org/python/introduction-to-numpy/), [Matplotlib](https://www.geeksforgeeks.org/data-visualization/data-visualization-using-matplotlib/)and [Sckit-Learn](https://www.geeksforgeeks.org/machine-learning/learning-model-building-scikit-learn-python-machine-learning-library/)libraries and a random dataset for the analysis of Polynomial Regression which you can download from [here](https://media.geeksforgeeks.org/wp-content/uploads/data.csv).

**import** **numpy** **as** **np**

**import** **matplotlib.pyplot** **as** **plt**

**import** **pandas** **as** **pd**

**from** **sklearn.linear\_model** **import** LinearRegression

**from** **sklearn.preprocessing** **import** PolynomialFeatures

**Step 2: Loading and Preparing the Data**

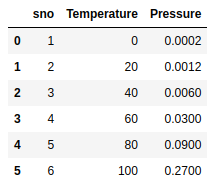
Here we will load the dataset and print it for our understanding.



datas = pd.read\_csv('/content/data.csv')

datas

**Output:**

First Five rows of the dataset

**Step 3: Defining Feature and Target Variables**

Our feature variable that is X will contain the Column between 1st and the target variable that is y will contain the 2nd column.

X = datas.iloc[:, 1:2].values

y = datas.iloc[:, 2].values

**Step 4: Fitting the Linear Regression Model**

We will first fit a simple linear regression model to the data.

lin = LinearRegression()

lin.fit(X, y)

**Output:**

Fitting the model

**Step 5: Fitting the Polynomial Regression Model**

Now we will apply polynomial regression by adding polynomial terms to the feature space. In this example, we use a polynomial of degree 4.

poly = PolynomialFeatures(degree=4)

X\_poly = poly.fit\_transform(X)

poly.fit(X\_poly, y)

lin2 = LinearRegression()

lin2.fit(X\_poly, y)

**Step 6: Visualizing the Linear Regression Results**

Visualize the results of the linear regression model by plotting the data points and the regression line.

plt.scatter(X, y, color='blue')

plt.plot(X, lin.predict(X), color='red')

plt.title('Linear Regression')

plt.xlabel('Temperature')

plt.ylabel('Pressure')

plt.show()

**Output:**

Linear Regression

A scatter plot of the feature and target variable with the linear regression line fitted to the data.

**Step 7: Visualize the Polynomial Regression Results**

Now visualize the polynomial regression results by plotting the data points and the polynomial curve.

plt.scatter(X, y, color='blue')

plt.plot(X, lin2.predict(poly.fit\_transform(X)),

color='red')

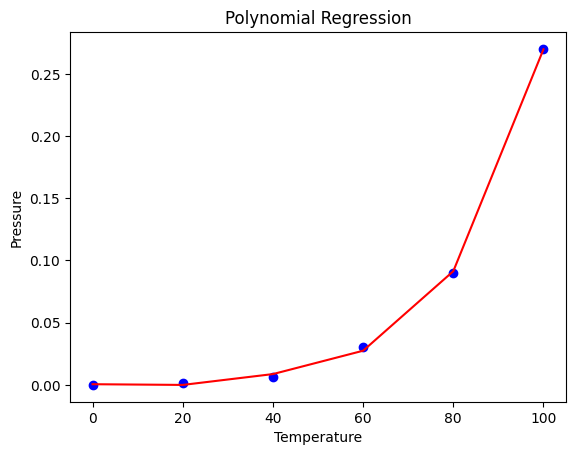
plt.title('Polynomial Regression')

plt.xlabel('Temperature')

plt.ylabel('Pressure')

plt.show()

**Output:**

Polynomial Regression

A scatter plot of the feature and target variable with the polynomial regression curve fitted to the data.

**Step 8: Predict New Results**

To predict new values using both linear and polynomial regression we need to ensure the input variable is in a 2D array format.

pred = 110.0

predarray = np.array([[pred]])

lin.predict(predarray)

**Output:**

*array([0.20675333])*

pred2 = 110.0

pred2array = np.array([[pred2]])

lin2.predict(poly.fit\_transform(pred2array))

**Output:**

*array([0.43295877])*

**Balancing Overfitting and Underfitting in Polynomial Regression**

In polynomial regression, overfitting happens when the model is too complex and fits the training data too closely helps in making it perform poorly on new data. To avoid this, we use techniques like [Lasso](https://www.geeksforgeeks.org/machine-learning/what-is-lasso-regression/) and [Ridge regression](https://www.geeksforgeeks.org/machine-learning/what-is-ridge-regression/) which helps to simplify the model by limiting the size of the coefficients.

On the other hand, underfitting occurs when the model is too simple to capture the real patterns in the data. This usually happens with a low-degree polynomial. The key is to choose the right polynomial degree to ensure the model is neither too complex nor too simple which helps it work well on both the training data and new data.

**Bias Vs Variance Tradeoff**

[Bias Vs Variance Tradeoff](https://www.geeksforgeeks.org/machine-learning/ml-bias-variance-trade-off/) helps us avoid both overfitting and underfitting by selecting the appropriate polynomial degree. As we increase the polynomial degree, the model fits the training data better but after a certain point, it starts to overfit. This is visible when the gap between training and validation errors begins to widen. The goal is to choose a polynomial degree where the model captures the data patterns without becoming too complex which ensures a good generalization.

**Application of Polynomial Regression**

1. **Modeling Growth Ra**tes: Polynomial regression is used to model non-linear growth rates such as the growth of tissues over time.
2. **Disease Epidemic Progression:**It helps track and predict the progression of disease outbreaks, capturing the non-linear nature of epidemic curves.
3. **Environmental Studies:** It is applied in studies like the distribution of carbon isotopes in lake sediments where relationships are non-linear.
4. **Economics and Finance:** It is used to analyze non-linear relationships in financial markets, predicting trends and fluctuations over time.

**Advantages**

1. **Fits a Wide Range of Curves:**Polynomial regression can model a broad range of non-linear relationships helps in making it versatile for complex data.
2. **Captures Non-linear Patterns:** It provides a more accurate approximation of relationships when data follows a curvilinear pattern, unlike linear regression.
3. **Flexible Modeling:** It can capture and represent the nuances in the data helps in making it ideal for situations where linear models fail.

**Disadvantages**

1. **Sensitivity to Outliers:** It is highly sensitive to outliers and a few extreme values can skew the results significantly.
2. **Overfitting:** With higher-degree polynomials, there’s a risk of overfitting where the model becomes too complex and fails to generalize well to new data.
3. **Limited Outlier Detection:**Unlike linear regression, it has fewer built-in methods for detecting or handling outliers helps in making it challenging to identify when the model is affected by them.

By mastering polynomial regression, we can better model complex data patterns which leads to more accurate predictions and valuable insights across various fields.

# 🔹 Key Regression Concepts

## 1️⃣ **Homoscedasticity**

* **Definition**: A situation where the **variance of the residuals (errors)** is **constant** across all levels of the independent variable(s).
* In other words, the spread of errors does **not depend** on the value of predictors.
* ✅ Assumption in OLS regression.
* **Why it matters**: If violated, the model may underestimate or overestimate prediction uncertainty.

**Example**:  
Predicting house prices → residuals have equal spread for small and large houses.

## 2️⃣ **Heteroscedasticity**

* **Definition**: Opposite of homoscedasticity → residual variance **changes** with the level of predictors.
* ❌ Violates OLS assumption, leading to **biased standard errors** and unreliable hypothesis tests.
* **Fixes**:
  + Transform target (e.g., log).
  + Use robust standard errors.
  + Use models that handle non-constant variance (GLMs, weighted least squares).

**Example**:  
Predicting income → variance of errors larger for high-income people than low-income.

## 3️⃣ **Loss Function**

* **Definition**: A mathematical function that measures the **error** between predicted values \(\hat{y}\) and actual values \(y\).
* In regression, the goal is to **minimize the loss**.

Common regression loss functions:

* **MSE (Mean Squared Error)** → penalizes large errors more.
* **MAE (Mean Absolute Error)** → more robust to outliers.

## 4️⃣ **Huber Loss**

* Hybrid loss function that combines **MSE** (quadratic) and **MAE** (linear).
* Defined as:

Lδ(y,y^)={12(y−y^)2if ∣y−y^∣≤δδ∣y−y^∣−12δ2otherwiseL\_\delta(y, \hat{y}) = \begin{cases} \frac{1}{2}(y-\hat{y})^2 & \text{if } |y-\hat{y}| \leq \delta \\ \delta |y-\hat{y}| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases}

* **Why use it**:
  + Quadratic for small errors → smooth optimization.
  + Linear for large errors → robust to outliers.

## 5️⃣ **Log-Cosh Loss**

* Defined as:

L(y,y^)=∑log⁡(cosh⁡(y−y^))L(y, \hat{y}) = \sum \log(\cosh(y - \hat{y}))

* Similar to MAE but **smooth and differentiable everywhere**.
* **Advantages**:
  + Like MSE for small differences.
  + Like MAE for large differences.
* Often used in deep learning regression.

## 6️⃣ **Importance of Intercept in Regression**

* **Intercept (β₀)**: baseline prediction when all features = 0.
* Without intercept, the regression line is **forced through the origin**, which may bias results unless you know the true relationship passes through zero.
* **Why it matters**:
  + Shifts regression line up/down.
  + Ensures residuals have mean = 0 (important for OLS assumptions).

**Example**:  
Predicting exam score from study hours. Intercept ≈ expected score if a student studies 0 hours.

## 7️⃣ **Nonlinear Regression**

* **Definition**: Regression where the relationship between predictors and target is **not a straight line**.
* Examples:
  + Polynomial regression: \(y = β\_0 + β\_1x + β\_2x^2 + ε\)
  + Exponential/logarithmic models.
  + Tree-based models (piecewise non-linear).
* **Why important**: Many real-world processes are nonlinear (e.g., growth curves, diminishing returns).

## 8️⃣ **Multicollinearity**

* **Definition**: Strong correlation between independent variables (predictors).
* **Problems**:
  + Makes it hard to estimate coefficients (unstable, high variance).
  + Coefficients may flip signs unexpectedly.
* **Detection**:
  + Correlation matrix.
  + VIF (Variance Inflation Factor).
* **Fixes**:
  + Remove/reduce correlated variables.
  + Use Ridge regression (L2 penalty).
  + Use PCA or dimensionality reduction.

## 9️⃣ **Endogeneity**

* **Definition**: When an independent variable is **correlated with the error term**.
* Violates OLS assumption → biased and inconsistent estimates.
* Causes:
  + **Omitted variable bias** (important variable missing).
  + **Simultaneity** (mutual causality).
  + **Measurement error** in predictors.
* **Fixes**:
  + Instrumental Variables (IV) regression.
  + Panel data models.
  + Causal inference methods (Difference-in-Differences, etc.).

## 🔟 **Variance Inflation Factor (VIF)**

* **Definition**: A measure of how much the variance of a regression coefficient is inflated due to multicollinearity.

Formula:

VIFj=11−Rj2VIF\_j = \frac{1}{1 - R\_j^2}

where \(R\_j^2\) is the \(R^2\) from regressing feature \(j\) on all other features.

* **Interpretation**:
  + VIF = 1 → no correlation.
  + VIF > 5 → moderate multicollinearity.
  + VIF > 10 → severe multicollinearity.

**Fix**: drop/reduce correlated features or use regularization.

# ✅ Quick Recap Table

| **Concept** | **Key Idea** | **Why It Matters** |
| --- | --- | --- |
| **Homoscedasticity** | Constant error variance | Needed for OLS assumptions |
| **Heteroscedasticity** | Non-constant error variance | Leads to biased standard errors |
| **Loss Function** | Measures prediction error | Guides model training |
| **Huber Loss** | Mix of MSE + MAE | Robust to outliers |
| **Log-Cosh Loss** | Smooth MAE-like loss | Useful in deep learning |
| **Intercept** | Baseline prediction | Centers regression line |
| **Nonlinear Regression** | Captures nonlinear relationships | More realistic models |
| **Multicollinearity** | Correlated predictors | Inflated coefficient variance |
| **Endogeneity** | Predictor correlated with error | Biased estimates |
| **VIF** | Measure of multicollinearity | Detects instability in features |

Ordinary Least Squares (OLS) regression is one of the most widely used methods in statistics and machine learning. But for the results to be valid and unbiased, some **assumptions** must hold true. These are called the **OLS assumptions** (also known as Gauss–Markov assumptions).

### **OLS Assumptions**

1. **Linearity in Parameters**
   * The relationship between the dependent variable (**Y**) and the independent variables (**X**) is linear in terms of parameters.
   * Example:  
     Y=β0+β1X1+β2X2+ϵY = \beta\_0 + \beta\_1X\_1 + \beta\_2X\_2 + \epsilon  
     The parameters (β\beta) enter linearly, even if X is transformed (e.g., log⁡(X)\log(X)).
2. **Random Sampling**
   * The data is collected through a random sampling process.
   * This ensures observations are representative and not biased.
3. **No Perfect Multicollinearity**
   * Independent variables should not be perfectly correlated.
   * If X1=2X2X\_1 = 2X\_2, then coefficients cannot be estimated uniquely.
   * Detectable using **Variance Inflation Factor (VIF)**.
4. **Zero Conditional Mean (Exogeneity)**
   * The error term (ϵ\epsilon) has an expected value of zero given any values of X:  
     E[ϵ∣X]=0E[\epsilon | X] = 0
   * Meaning: No omitted variable bias, no reverse causality, and no measurement errors.
5. **Homoscedasticity** (Constant Variance of Errors)
   * The error terms have constant variance across all levels of X:  
     Var(ϵ∣X)=σ2Var(\epsilon | X) = \sigma^2
   * If violated → **heteroscedasticity** (can lead to inefficient estimates, though OLS remains unbiased).
6. **No Autocorrelation (Independence of Errors)**
   * Error terms are not correlated with each other:  
     Cov(ϵi,ϵj)=0for i≠jCov(\epsilon\_i, \epsilon\_j) = 0 \quad \text{for } i \neq j
   * Important in **time series regression**.
   * If violated → inefficient estimates and wrong standard errors.
7. **Normality of Errors** (for inference)
   * The error terms should follow a normal distribution:  
     ϵ∼N(0,σ2)\epsilon \sim N(0, \sigma^2)
   * Not necessary for OLS estimates to be **BLUE** (Best Linear Unbiased Estimator) but required for **valid hypothesis testing** (t-tests, F-tests, confidence intervals).

### **Summary**

OLS assumptions ensure that:

* Coefficients (β\beta) are **unbiased** and **efficient**.
* Statistical inference (p-values, confidence intervals) is valid.
* Predictions are reliable.

⚡ Mnemonic for remembering:  
**LINE**

* **L**inearity
* **I**ndependence (no autocorrelation)
* **N**ormality (of residuals for inference)
* **E**qual variance (homoscedasticity)