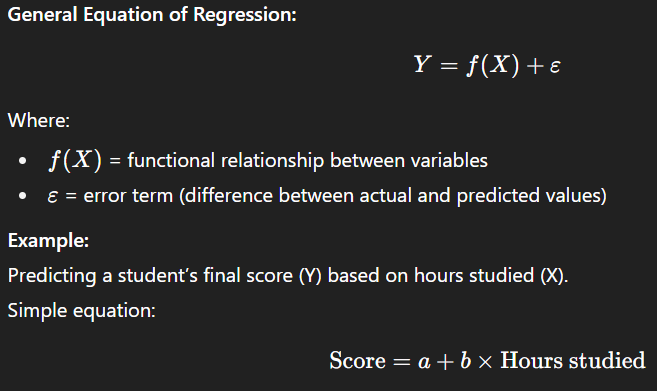
**ML\_U2**Regression is a **statistical and machine learning technique** used to model and analyze the relationship between a **dependent variable** (target) and one or more **independent variables** (predictors).  
Its goal is to predict or estimate the dependent variable based on known values of the independent variables.

* **Dependent Variable (Y):** The value we want to predict (e.g., house price, sales revenue).
* **Independent Variables (X):** The input features used for prediction (e.g., area, location, number of bedrooms).

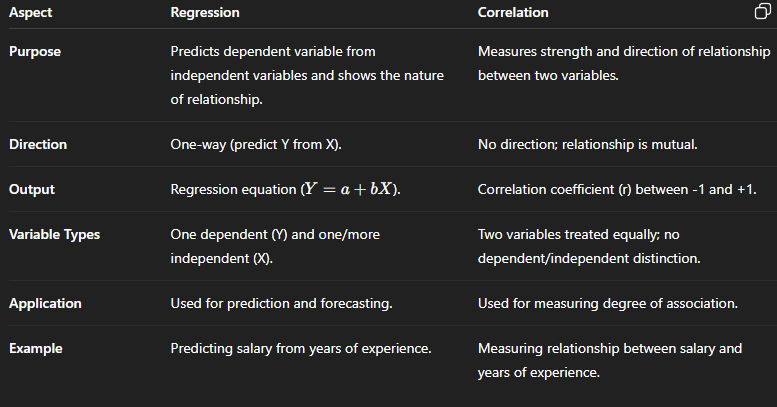


**Need for Regression**

Regression is important in many real-life scenarios because it provides both **prediction** and **relationship analysis**.

1. **Prediction of future outcomes**
   * Example: Predicting rainfall based on humidity and temperature.
2. **Relationship understanding**
   * Shows how the dependent variable changes when independent variables change.
3. **Forecasting trends**
   * Example: Estimating future sales from past sales data.
4. **Decision-making & planning**
   * Helps businesses choose strategies (e.g., which features increase product sales).
5. **Quantifying impact of factors**
   * Example: Finding which factor most influences customer satisfaction.

**Regression and Correlation**

**Types of Regression**

**1. Univariate vs. Multivariate Regression**

**Univariate Regression**

A regression model with only one independent variable (X) used to predict a single dependent variable (Y).  
**Purpose:** Helps to understand and quantify the relationship between two variables.  
**Equation:** Y = a + bX, where a = intercept, b = slope (change in Y per unit change in X).  
Example: Predicting student marks based on hours studied.  
**Graph:** Straight line in a 2D plot (X vs Y).

**Multivariate Regression**

A regression model with two or more independent variables (X1, X2, …) used to predict one dependent variable (Y).  
**Purpose:** Captures more complex relationships and accounts for multiple influencing factors.  
**Equation:** Y = a + b1X1 + b2X2 + … + bnXn, where b1, b2 … = coefficients for each predictor.  
Example: Predicting house price from size, location, and age.  
**Graph**: Can be visualized as a plane (for 2 predictors) or hyperplane (for more predictors).

**2. Linear vs. Nonlinear Regression**

**Linear Regression**

A regression model where the relationship between dependent and independent variables is linear in the parameters.  
**Key Point**: Variables can be transformed (e.g., polynomial), but coefficients appear in a linear way.  
**Equation**: Y = a + bX.  
Example: Predicting salary from years of experience.  
Advantages: Simple, fast to compute, easy to interpret.  
**Graph**: Straight line in 2D, flat plane in higher dimensions.

**Nonlinear Regression**

**Definition**: A regression model where the relationship is nonlinear in parameters. Parameters may appear in exponents, products, or other nonlinear forms.  
Equation Example: Y = a \* e^(bX).  
**Example**: Modeling population growth using an exponential curve.  
Advantages: Captures complex, curved patterns.  
Disadvantages: Harder to compute, may need iterative methods.  
**Graph**: Curved fit line (e.g., exponential, logarithmic, S-shape).

**3. Simple Linear vs. Multiple Linear Regression**

**Simple Linear Regression (SLR)**

Linear regression with one independent variable.  
**Equation**: Y = a + bX.  
Example: Predicting marks from hours studied.  
Interpretation: b tells how much Y changes for a one-unit increase in X.  
**Graph**: Straight line on 2D plot.

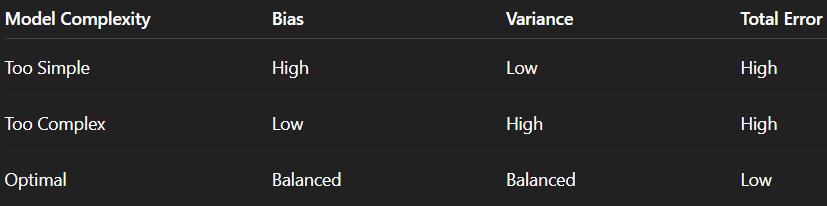
**Multiple Linear Regression (MLR)**

Linear regression with two or more independent variables.  
**Equation:** Y = a + b1X1 + b2X2 + … + bnXn.  
**Example:** Predicting salary based on experience, education level, and skill score.  
Interpretation: Each bi represents the effect of Xi on Y, keeping other variables constant.  
Graph: Plane for two predictors, hyperplane for more.

**Bias–Variance Trade-Off**

The **Bias–Variance Trade-Off** describes the balance between a model’s ability to **fit the training data** and its ability to **generalize to unseen data**.  
The goal is to minimize the **Total Error** (Generalization Error), which comes from three sources:

Total Error=Bias2+Variance+Irreducible Err



**High Bias** → Underfitting → Simple model.

H**igh Variance** → Overfitting → Complex model.

### **Bias**

* Error caused by **overly simplistic assumptions** in the model.
* High bias → **Underfitting** (model misses important patterns).
* Example: Using a straight line to fit non-linear data.

**Characteristics of High Bias:**

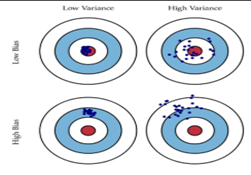
* Poor training accuracy.
* Poor test accuracy.
* Model is too simple.

### **Variance**

* Error caused by **too much sensitivity to training data fluctuations**.
* High variance → **Overfitting** (model memorizes training data but fails to generalize).
* Example: Using a high-degree polynomial to fit simple data.

**Characteristics of High Variance:**

* High training accuracy.
* Low test accuracy.
* Model is too complex.

****

## ****Overfitting****

When a model learns the training data too well, including noise, it performs well on training data but poorly on new (test) data.

* Low training error, high test error
* Predictions follow overly complex curves or patterns

**Causes:**

* Too many features or parameters
* Too little training data
* Long training without regularization
* Data leakage

**Detection:**

* Large gap between training and test performance
* High variation in cross-validation scores
* Learning curve shows low training error but high test error

**Remedies:**

* Add regularization
* Use simpler model
* Prune decision trees or limit depth
* Increase k in k-NN
* Early stopping or dropout (for neural networks)
* Data augmentation or cleaning
* Use cross-validation
* Apply ensembling (bagging)

## ****Underfitting****

When a model is too simple to capture the actual pattern in data, it performs poorly on both training and test sets.

* High training error and high test error
* Misses obvious patterns or trends

**Causes:**

* Model too simple
* Not enough features
* Too much regularization
* Training stopped too early

**Detection:**

* Learning curve shows both training and test errors are high and close
* Residual plots show clear patterns

**Remedies:**

* Use a more complex model
* Add more useful features
* Reduce regularization
* Train for longer
* Add interaction terms or nonlinear transformations

**Polynomial Regression**

Polynomial Regression is an **extension of Linear Regression** that allows us to model **non-linear relationships** between the independent variable(s) XXX and dependent variable YYY.

While **Linear Regression** fits a straight line to the data, Polynomial Regression fits a **curved line** by adding higher-degree terms of the predictor variable.

## ****Why Polynomial Regression?****

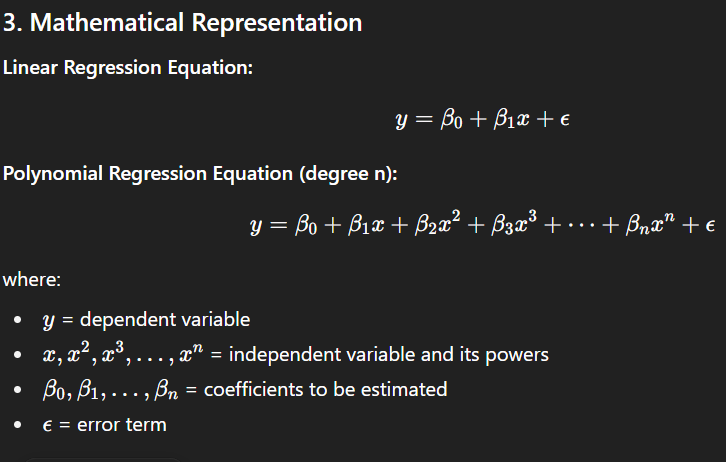
* In real-world problems, relationships are rarely perfectly linear.
* Example: Growth rate of bacteria, housing prices vs. area, learning curves, etc. often follow a curved trend.
* Linear regression would **underfit** such data because it assumes a constant slope, but polynomial regression can capture bends and curves.

## ****How It Works****

* We **transform** the original features by adding polynomial terms.
* Then we perform **Linear Regression** on these transformed features.
* Even though the curve looks non-linear in xxx, the regression is **linear in coefficients**, so it can be solved using the same least squares method.

Choosing the polynomial degree (n) is critical:

* **Too low degree** → Underfitting.
* **Too high degree** → Overfitting.
* Use **cross-validation** to choose optimal n.



**Advantages**

✅ Captures non-linear trends.  
✅ Easy to implement with existing linear regression algorithms.  
✅ Works well when the true relationship is polynomial-like.

**8. Limitations**

❌ High-degree polynomials are sensitive to noise → Overfitting.  
❌ Can oscillate wildly between points (Runge’s phenomenon).  
❌ Poor extrapolation — predictions outside the data range can be unrealistic.

**Example:** Predicting exam scores based on study hours where improvement rate slows after a certain point.

## ****Stepwise Regression****

Stepwise regression is a **variable selection method** for multiple linear regression that iteratively adds or removes predictors based on statistical criteria such as **p-values, AIC, BIC, or adjusted R²**.  
It aims to find a parsimonious model that explains the data without including unnecessary variables.

**Mathematical Expression:**  
Starts with:

y=β0+β1X1+⋯+βpXp+ϵ

The algorithm chooses the subset of XXX that minimizes an error metric.

**Working:**

1. **Forward Selection** – Start with no variables, add the one with the highest significance until no improvement.
2. **Backward Elimination** – Start with all variables, remove the least significant one at each step.
3. **Stepwise** – Combination of both methods.

**Advantages:**

* Reduces overfitting by eliminating irrelevant variables.
* Automates model building.
* Improves interpretability.

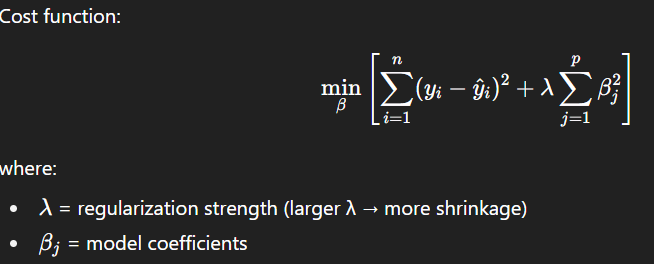
**Limitations:**

* Can ignore variable interactions.
* May overfit if selection is based solely on training data.
* Unstable when predictors are highly correlated.

**Example:** Selecting important economic indicators for predicting GDP growth.

## ****Ridge Regression****

**Concept:**  
Ridge Regression is a **regularized form of linear regression** that adds an **L2 penalty** to the loss function.  
This penalty shrinks large coefficient values, which helps reduce **overfitting** and handle **multicollinearity** (high correlation between predictors).  
It keeps all predictors in the model but with reduced magnitudes.



**Working Process:**

1. Start with the **Ordinary Least Squares (OLS)** regression formulation.
2. Add an **L2 penalty term** to the cost function.
3. Optimize to find coefficients that minimize both **prediction error** and **coefficient magnitude**.
4. Larger λ → more shrinkage; λ = 0 gives standard linear regression.

**Advantages:**

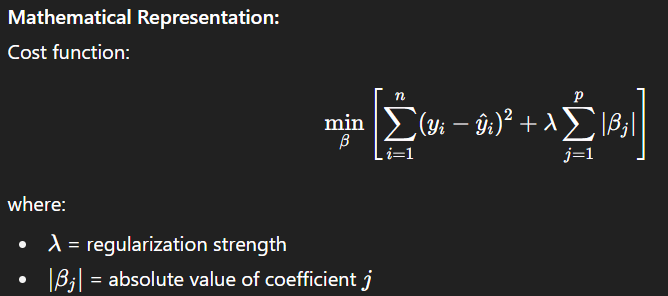
* Handles **multicollinearity** effectively.
* Reduces model complexity and **overfitting**.
* Works well when most predictors have small to moderate effects.

**Limitations:**

* Does **not** perform feature selection (keeps all predictors).
* Choice of λ requires **cross-validation**.
* Coefficients become biased due to shrinkage.

**Example:**  
Predicting **sales** using multiple highly correlated marketing variables (e.g., TV ads, online ads, print ads).

## ****Lasso Regression****

Lasso Regression (**Least Absolute Shrinkage and Selection Operator**) is a **regularized linear regression** method that uses an **L1 penalty** on the coefficients.  
Unlike Ridge Regression, Lasso can shrink some coefficients **exactly to zero**, effectively performing **feature selection** along with regularization.  
****

**Working Process:**

1. Start with **Ordinary Least Squares (OLS)** regression.
2. Add an **L1 penalty term** to the cost function.
3. Optimization via **coordinate descent** or **LARS algorithm** forces some coefficients to be exactly **zero**.
4. Larger λ → more coefficients shrink to zero.

**Advantages:**

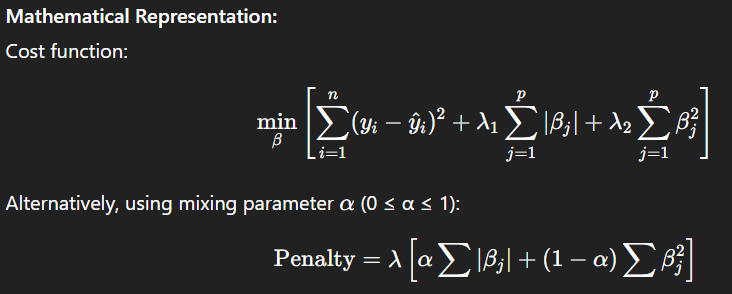
* Performs **automatic feature selection**.
* Produces simpler, more interpretable models.
* Useful when only a subset of predictors are important.

**Limitations:**

* In presence of highly correlated predictors, it tends to select only one and ignore others.
* Choice of λ is critical and requires cross-validation.
* May be unstable for small datasets.

**Example:**  
Selecting important **genes** from thousands of genomic variables to predict disease risk.

## ****ElasticNet Regression**** ElasticNet Regression is a **hybrid regularization method** that combines **L1 penalty** (from Lasso) and **L2 penalty** (from Ridge). It performs **feature selection** like Lasso while maintaining **stability** in the presence of highly correlated predictors like Ridge.

****

**Working Process:**

1. Transform problem into a mix of **L1 and L2 regularization**.
2. **α** controls the balance:
   * α = 1 → pure Lasso
   * α = 0 → pure Ridge
3. Solve using coordinate descent or similar optimization methods.
4. Select λ using cross-validation.

**Advantages:**

* Handles **multicollinearity** better than Lasso.
* Performs feature selection while keeping correlated variables.
* More flexible than Ridge or Lasso alone.

**Limitations:**

* Requires tuning of **two parameters** (λ and α).
* More computationally intensive than Ridge/Lasso.
* Interpretability slightly reduced compared to Lasso.

**Example:**  
Predicting **customer churn** using many correlated customer behavior features where both feature selection and stability are needed.

## ****Bayesian Linear Regression****

Bayesian Linear Regression is a **probabilistic approach** to linear regression where model parameters (coefficients) are treated as **random variables** with prior probability distributions.  
Using **Bayes’ theorem**, the model updates these priors into **posterior distributions** after seeing the data, providing both predictions and **uncertainty estimates**.

**Mathematical Representation:**  
Bayes’ theorem for regression:

p(β∣X,y)=(p(y∣X,β)⋅p(β))/p(y∣X) ​

where:

* p(β) = prior distribution of coefficients
* p(y∣X,β) = likelihood from data
* p(β∣X,y) = posterior distribution after observing data

Prediction distribution:

p(y∗∣x∗,X,y)=∫p(y∗∣x∗,β) p(β∣X,y) dβp

**Working Process:**

1. **Define prior** for coefficients (commonly Gaussian with mean 0).
2. **Calculate likelihood** from observed data using the linear model.
3. Apply **Bayes’ theorem** to obtain the posterior distribution.
4. Predict new values using the **posterior mean** (point estimate) or full distribution for uncertainty quantification.

**Advantages:**

* Produces **uncertainty intervals** for predictions.
* Can incorporate **prior knowledge** about parameters.
* Works well with small datasets.
* Naturally prevents overfitting through prior regularization.

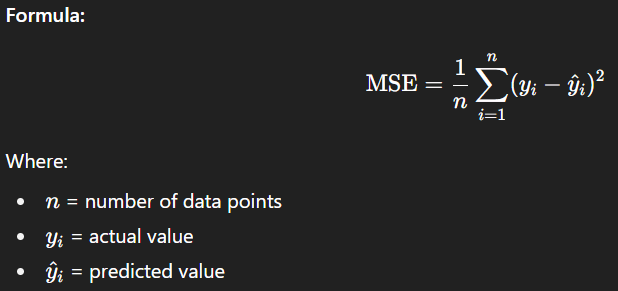
**Limitations:**

* Requires **specifying priors** carefully.
* Computationally expensive for large datasets.
* Analytical solutions may not exist for complex priors (requires approximation methods like MCMC).

**Example:**  
Forecasting **monthly sales** while accounting for uncertainty and prior knowledge about seasonal effects.

### **1. Mean Squared Error (MSE)**

MSE is the average of the squared differences between actual and predicted values.  
Squaring ensures all errors are positive and penalizes larger errors more heavily.

****

* **Interpretation**
  + Lower MSE → better fit.
  + Value is **always ≥ 0**; 0 means perfect predictions.
* **Advantages**
  + Easy to compute mathematically.
  + Penalizes large errors strongly → useful when big mistakes are costly.
* **Disadvantages**
  + Unit is **squared** (not same as target variable).
  + Highly sensitive to outliers.

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

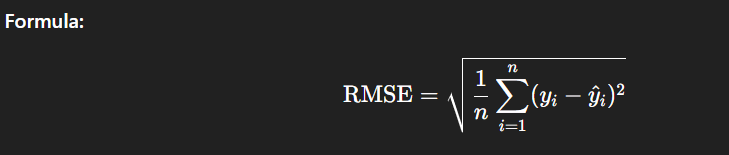
* **Definition**  
  MAE is the average of the absolute differences between actual and predicted values.  
  Unlike MSE, it does not square errors, so it treats all errors equally.

****

* **Interpretation**
  + Measures average magnitude of errors in the **same unit** as the target variable.
  + Gives an idea of how wrong predictions are, on average.
* **Advantages**
  + Easy to interpret.
  + Less sensitive to outliers than MSE.
* **Disadvantages**
  + Does not emphasize large errors → may be misleading if big mistakes are important.

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

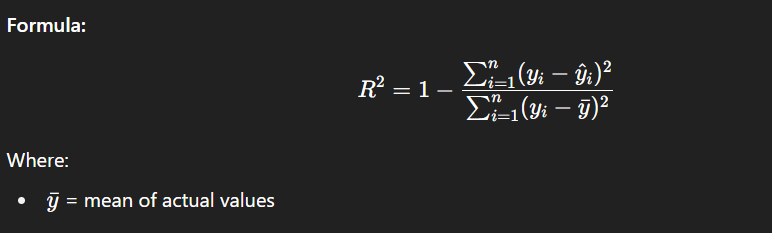
* **Definition**  
  RMSE is the square root of MSE.  
  It gives the error in the **same unit** as the target variable.



* **Interpretation**
  + Shows how far predictions are from actual values, on average.
  + Lower RMSE = better model.
* **Advantages**
  + Same units as target variable → easy to compare.
  + Still penalizes large errors more than MAE.
* **Disadvantages**
  + Sensitive to outliers.

### **4. R-squared (**R2R^2R2**)**

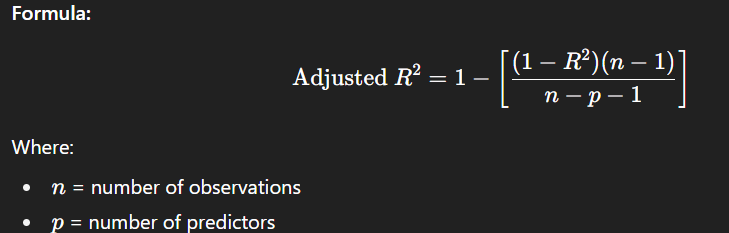
* **Definition**  
  Measures the proportion of variance in the dependent variable explained by the model.
* **Formula**



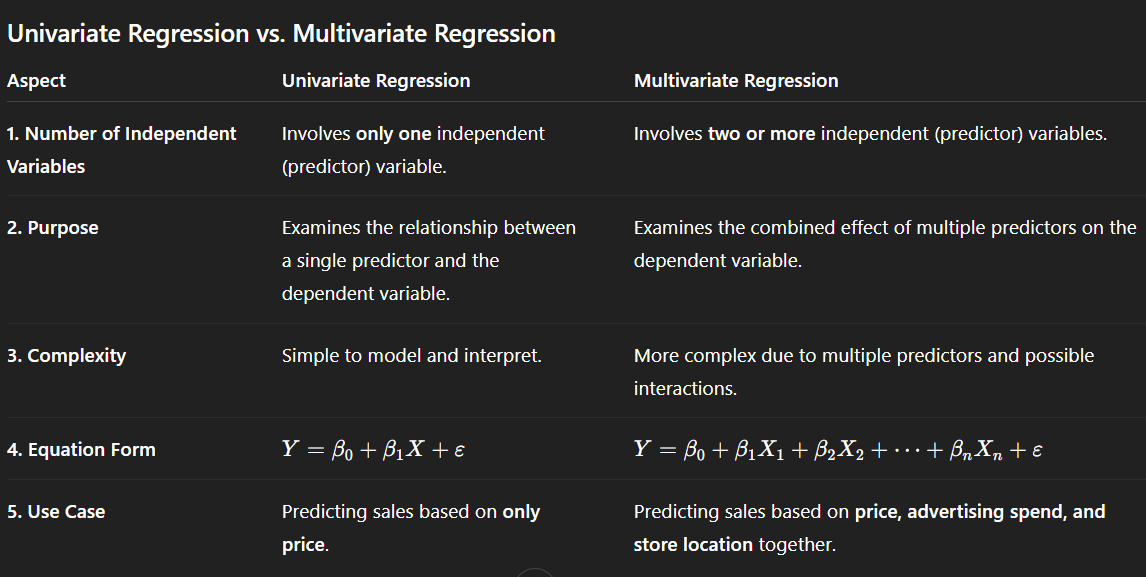
* **Interpretation**
  + R2R^2R2 = 1 → perfect fit
  + R2R^2R2 = 0 → model explains none of the variance
  + Can be **negative** if the model is worse than predicting the mean.
* **Advantages**
  + Intuitive measure of goodness-of-fit.
  + Easy to compare models on the same dataset.
* **Disadvantages**
  + Can increase artificially when adding more variables, even if they are irrelevant.
  + Does not indicate bias or overfitting.

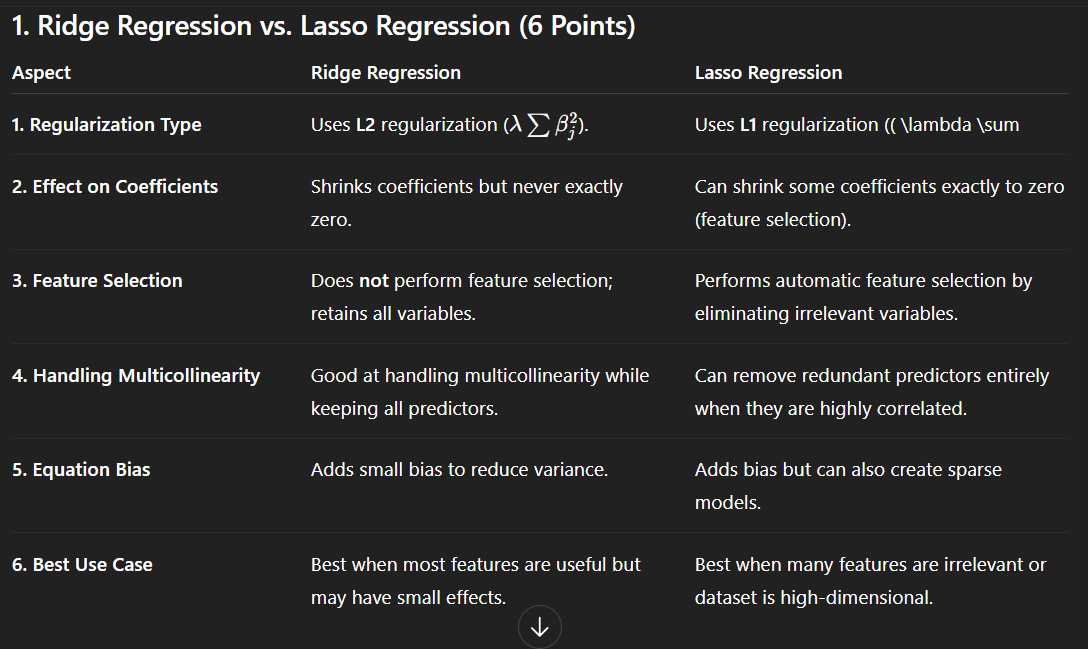
### **5. Adjusted R-squared**

* **Definition**  
  Modified version of R2R^2R2 that adjusts for the number of predictors in the model.  
  Penalizes adding predictors that don’t improve the model.

****

* **Interpretation**
  + Increases only if the new variable improves the model significantly.
  + Useful for comparing models with different numbers of predictors.
* **Advantages**
  + Prevents misleading improvement from adding irrelevant predictors.
  + Better measure for model selection.
* **Disadvantages**
  + More complex formula.
  + Still doesn’t guarantee the model is correct.





**Techniques to Reduce Overfitting in Regression**

**1. Cross-Validation**

* Instead of splitting data into just training and testing, use **k-fold cross-validation**.
* The dataset is divided into *k* subsets. The model trains on (k−1) subsets and validates on the remaining one.
* Helps ensure the model generalizes well and not just fits a single train-test split.

**2. Simpler Model**

* Reduce model complexity.
* In regression:
  + Use **lower-degree polynomials** instead of very high-degree ones.
  + Avoid adding unnecessary interaction terms.
* This helps to reduce variance and prevents memorizing training data.

**3. Regularization**

Regularization introduces a penalty term in the cost function to prevent coefficients from becoming too large.

* **Ridge Regression (L2 Regularization):**  
  Adds penalty = λ Σ (βᵢ²).  
  Shrinks coefficients but never makes them exactly zero. Good for multicollinearity.
* **Lasso Regression (L1 Regularization):**  
  Adds penalty = λ Σ |βᵢ|.  
  Can shrink some coefficients to **exactly zero**, performing **feature selection**.
* **Elastic Net (Combination of L1 + L2):**  
  Balances Ridge and Lasso. Works well when there are many correlated features.

**4. Feature Selection / Dimensionality Reduction**

* Too many irrelevant or correlated features increase model variance.
* Solutions:
  + Use **feature selection** techniques (Forward selection, Backward elimination).
  + Use **dimensionality reduction** methods like **PCA (Principal Component Analysis)**.
* This reduces noise and improves generalization.

**5. Increase Training Data**

* With more data, the model learns the true pattern better and noise effect reduces.
* Especially useful when the dataset is small and the model tends to memorize.
* Data augmentation (in images, text, etc.) can also be applied when collecting more data is hard

**6. Early Stopping (for iterative optimization models)**

* In gradient descent–based models (like polynomial regression with iterative solvers), monitor error on **validation set**.
* If validation error starts increasing while training error decreases → stop training.
* Prevents the model from over-learning noise.

**Decision Tree Regression**

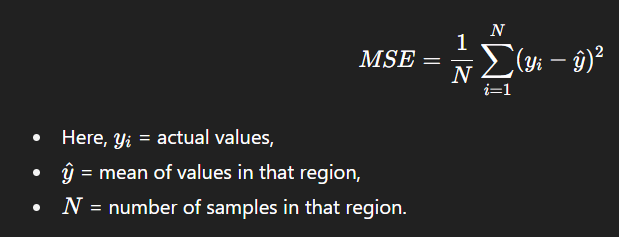
* Decision Tree Regression is a **supervised learning technique** used for **predicting continuous numerical values**.
* Unlike classification trees (which predict categories), regression trees predict a **real number output** (e.g., predicting house prices).

**2. Working**

1. **Splitting the Data:**
   * The input space is divided into smaller regions by asking a sequence of “if–else” questions (based on feature values).
   * Example: *“Is area < 1000 sq.ft?” → split into two groups.*
2. **Recursive Partitioning:**
   * The tree keeps splitting data into subsets until a stopping criterion is met (e.g., maximum depth, minimum samples per leaf).
3. **Prediction at Leaves:**
   * For regression, the value predicted at each **leaf node** is usually the **mean (or median)** of the target values in that region.

**3. Objective Function**

* At each split, the algorithm chooses the feature and threshold that minimizes the **prediction error**.
* Common error measure: **Mean Squared Error (MSE)**



**4. Example**

Suppose we want to predict **house prices** based on “Size of House”:

* Root Node: *Is size < 1000 sq.ft?*
  + If **Yes:** Average price = ₹25 lakh.
  + If **No:** Next split → *Is size < 2000 sq.ft?*
    - If **Yes:** Average price = ₹45 lakh.
    - If **No:** Average price = ₹70 lakh.

Thus, predictions are made by following the path in the tree.

**5. Advantages**

* Easy to understand and interpret.
* Handles both numerical and categorical features.
* Non-linear relationships can be captured.
* No need for feature scaling (works directly on raw data).

**6. Limitations**

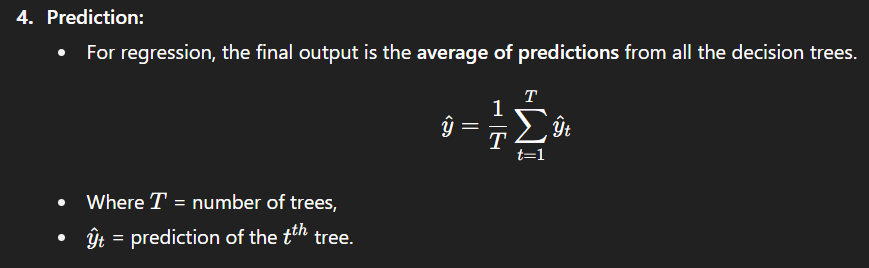
* Can easily **overfit** if tree depth is not controlled.
* Predictions are **piecewise constant** (not smooth).
* Small changes in data can lead to very different trees.

**Random Forest Regression**

* Random Forest Regression is a **supervised ensemble learning method** used for predicting continuous values.
* It combines the predictions of **multiple decision trees** to improve accuracy and reduce overfitting.
* Instead of relying on a single decision tree, it builds many trees and averages their outputs.

**2. Working Principle**

1. **Bootstrap Sampling (Bagging):**
   * From the training dataset, multiple random samples are taken **with replacement**.
   * Each sample is used to build a separate decision tree.
2. **Random Feature Selection:**
   * At each split in the tree, a **random subset of features** is chosen (not all features).
   * This ensures trees are diverse and not highly correlated.
3. **Tree Building:**
   * Each decision tree is grown (often without pruning).
   * Trees may overfit individually, but randomness + averaging reduces this effect.



**3. Example**

Suppose we want to predict **house prices**:

* Tree 1 predicts ₹42 lakh,
* Tree 2 predicts ₹45 lakh,
* Tree 3 predicts ₹44 lakh.
* **Final prediction = (42 + 45 + 44)/3 = ₹43.67 lakh.**

**4. Advantages**

* Reduces **overfitting** compared to a single decision tree.
* Works well for both linear and non-linear relationships.
* Handles large datasets and high-dimensional features effectively.
* Robust to missing values and noisy data.

**5. Limitations**

* More **complex and computationally expensive** than a single tree.
* Less interpretable (harder to visualize than one decision tree).
* Requires tuning (number of trees, max depth, etc.).

**Support Vector Regression (SVR)**

* Support Vector Regression (SVR) is a **supervised machine learning algorithm** used for predicting continuous values.
* It is based on the principles of **Support Vector Machines (SVM)** but adapted for regression tasks.
* SVR tries to fit the best line (or hyperplane) within a predefined margin of tolerance, rather than minimizing error for every data point.

**2. Working Principle**

1. **Margin of Tolerance (ε-insensitive loss):**
   * SVR introduces a margin (epsilon, ε) where errors within this boundary are ignored.
2. **Support Vectors:**
   * Only data points lying outside the ε margin (errors beyond tolerance) influence the model.
   * These critical points are called **support vectors**, as they define the regression function.
3. **Kernel Trick:**
   * SVR can use different kernels (linear, polynomial, RBF, etc.) to handle both **linear and non-linear relationships**.
   * Kernels map the input data into higher dimensions where a linear regression hyperplane can be fitted.

**3. Example**

Suppose we want to predict house prices:

* The true prices vary slightly around ₹44 lakh.
* SVR defines a margin (say ε = ₹1 lakh).
* Predictions within **₹43–45 lakh** are considered acceptable (no penalty).
* Only points outside this band affect the model fitting.

**4. Advantages**

* Effective in **high-dimensional spaces**.
* Works well for both **linear and non-linear regression** using kernels.
* **Robust to outliers** (small effect, since only support vectors outside the margin matter).
* Provides good **generalization** due to margin maximization.

**5. Limitations**

* **Computationally expensive**, especially for large datasets.
* Requires careful **parameter tuning** (C = penalty parameter, ε = margin, kernel choice).
* Performance is sensitive to **scaling of data** (feature normalization often required).