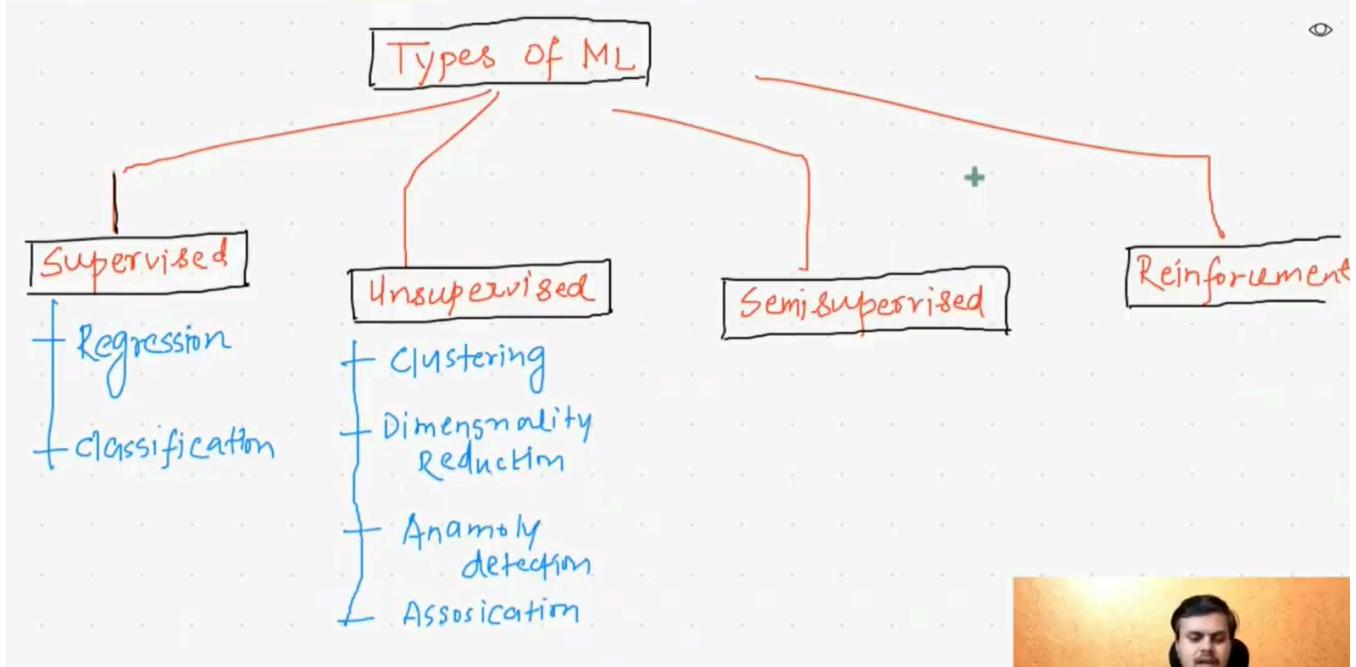


Machine learning is a field of computer science that uses statistical techniques to give computer systems the ability to "learn" with data, without being explicitly programmed.



Supervise Machine Learning

- Types
 - **Regression** : It is used when the target/output column is numerical
 - **Classification** : It is used when the output column is categorical

Unsupervised Machine:

- In Unsupervised ML you only have input
- Types :
 - **Clustering** : It detects that particulard data will fall in which group or category.
 - **Dimensionality Reduction** : When you are working with supervised ML you have too many input columns. it makes algo slow and it does not improve result because there are some columns that do not help in predicting. It is done using techniques like **PCA**

Also it is used in visualisation technique. Sometimes we cannot visualise a

- **Anomaly Detection** : It is used in detecting anomaly detection like detecting in manufacturing or credit card fraud detection so it basically detect outliers.
- **Association rule learning** : Association Rule Learning is an unsupervised machine learning technique used to discover hidden relationships (patterns) between items in large datasets. For example it can be used in super market to find relationship between products and using that we can create combo offers.

Semi Supervised

- It is partially unsupervised and partially **supervised**. It has small amount of labelled data and large amount of unlabeled data. Labelling data is expensive, slow and requires experts

Reinforcement Learning

- Reinforcement Learning (RL) is a type of machine learning where an agent learns to make decisions by interacting with an environment to maximize cumulative rewards. The goal is to learn a policy that maximizes long-term reward.

Extra Information

Instance-based learning (memory-based / lazy learning) : Learns by remembering instances. E.g. : K nearest neighbors

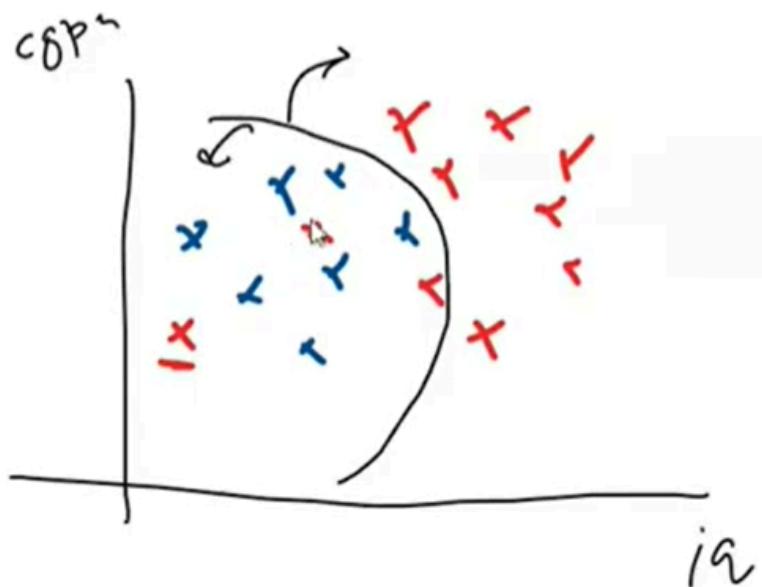
Model-based learning (eager learning) : The algorithm builds an explicit model that.

3. Model Based

Friday, March 19, 2021

4:06 PM

iq | cgpa | p6cm



Simple Linear Regression

Simple linear regression (SLR) is a fundamental supervised learning statistical method used to model the relationship between two variables: one independent variable and one dependent variable.

Types

- Simple Linear Regression
- Multiple Linear Regression
- Polynomial Linear Regression : It is used when our data is not linear

Simple Linear Regression

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

- β_0 : The y-intercept (the value of Y when $X = 0$).
- β_1 : The slope (the change in Y for a one-unit increase in X).
- ϵ : The error term (random noise, assumed to be normally distributed with mean 0 and constant variance).

- What is best fit line : It is the line that minimizes errors across all data points according to chosen rules. That rule will be mean squared error.
- Intercept (β_0) : Expected value of y when $X = 0$
- β_1 : On average, how much does y change when X increases by 1 unit?"

Important distinction:

- **True model:**

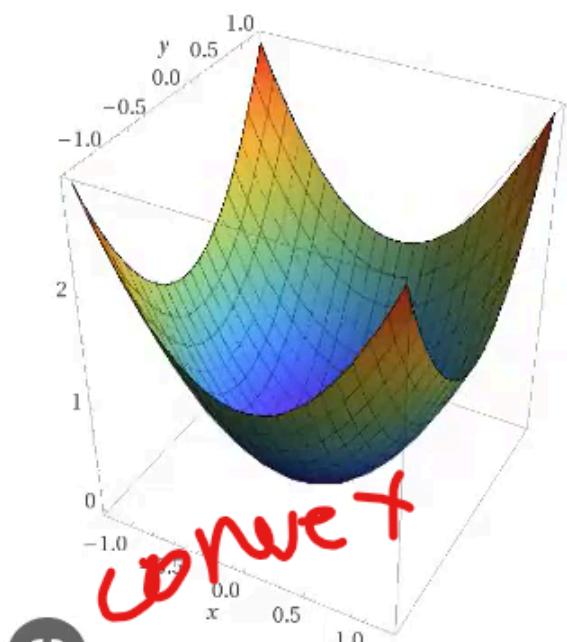
$$y = \beta_0 + \beta_1 X + \epsilon$$

- **Prediction (what the model outputs):**

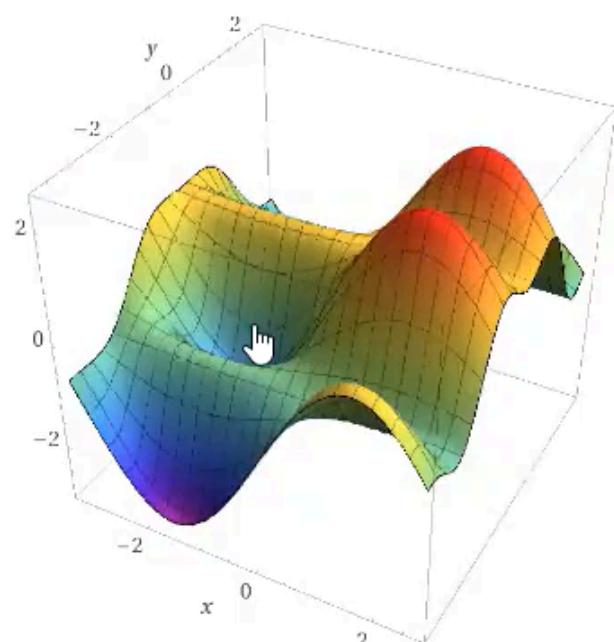
$$\hat{y} = \beta_0 + \beta_1 X$$

The model never predicts ϵ .

- There are 2 ways to find the value of m and b.
 - **Closed form solution** : (Direct mathematical formula using ordinary least square. Scikit learn is using this technique for linear regression algo).
 - Its only efficient for lower dimensional data. If a function is convex that means if the line between any two points on the function lies above the function having one global minima or maxima then we can use closed form solution.



Computed by Wolfram|Alpha



Computed by Wolfram|Alpha

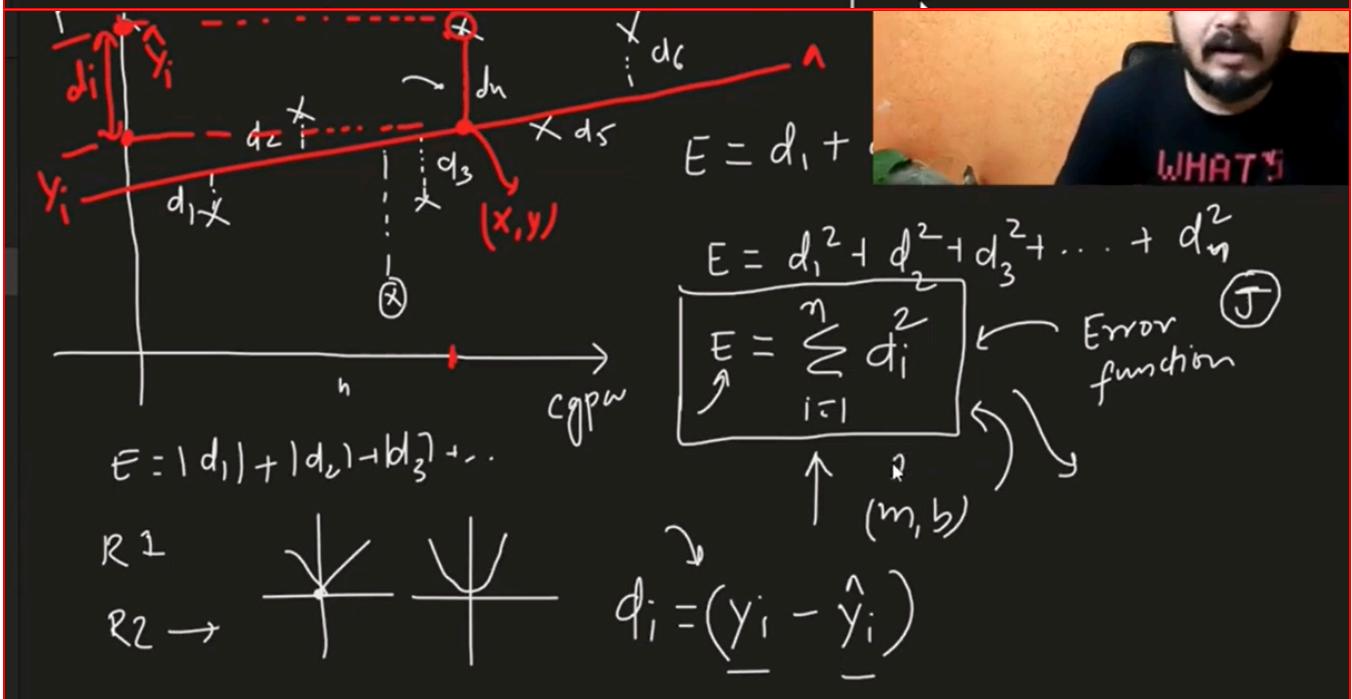
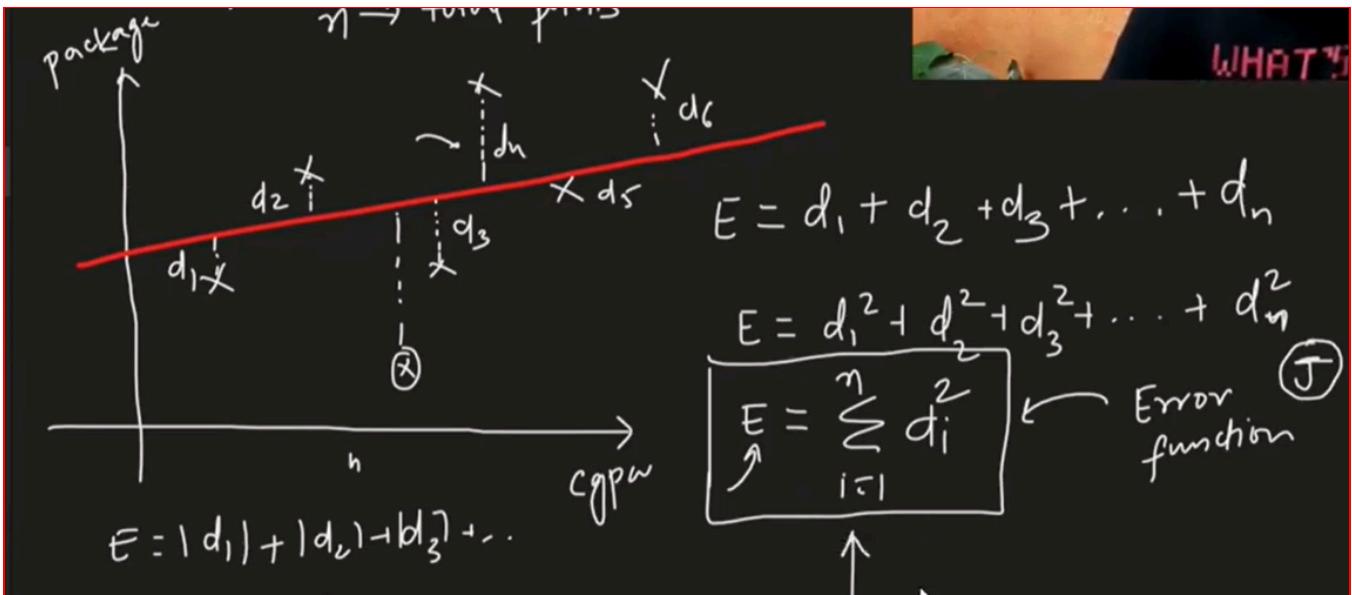
1,462 × 800

$$\frac{b}{\uparrow} = \bar{y} - m \bar{x}$$

↑

↗ mean

$$m = \frac{\sum_{i=1}^n (\underline{x_i} - \bar{x})(\underline{y_i} - \bar{y})}{\sum_{i=1}^n (\underline{x_i} - \bar{x})^2}$$



$$E = \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

$$\hat{y}_i = m x_i + b$$

$$E(m, b) = \sum_{i=1}^n (\hat{y}_i - y_i)^2$$

- **Non closed form solution** (Gradient Descent). Used for higher dimensional data. SGD Regressor in python uses this.

Regression evaluation metrics

- MAE (L1 Regression) : $\frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$

Advantages: Robust to outliers.

Disadvantages: It is not differentiable at 0.

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

Advantages: It is differentiable

Disadvantages: Not robust to outliers.

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

- R2 Score : $1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$

R² compares your model against a dumb model that always predicts mean(y).

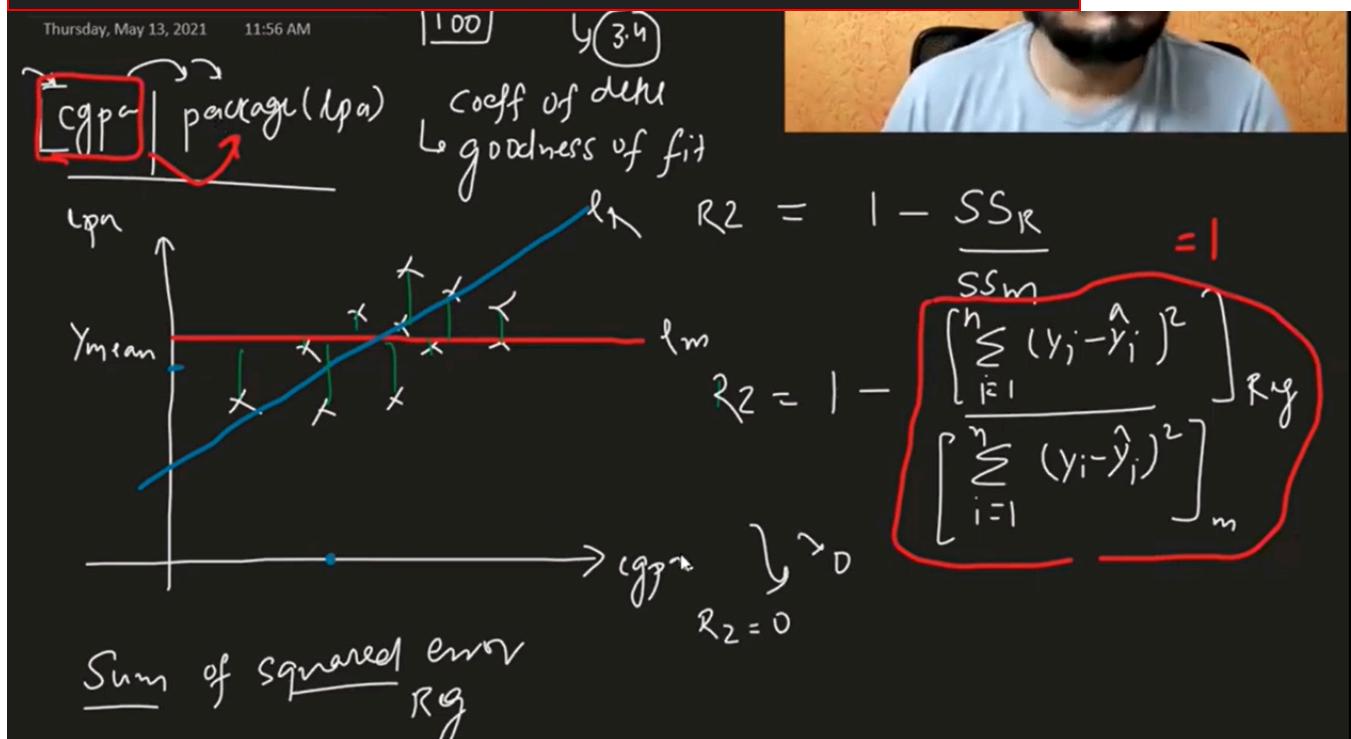
- $R^2 = 1 \rightarrow$ perfect fit (line explains everything)
- $R^2 = 0 \rightarrow$ no better than guessing the average
- Sometimes $R^2 < 0 \rightarrow$ worse than guessing the average

Step 1: Total Sum of Squares (TSS)

TSS measures how spread out the data is before using any model.

$$\text{TSS} = \sum (y_i - \bar{y})^2$$

- y_i : actual values
- \bar{y} : mean of the target variable



Problem: R^2 never decreases when you add more features (predictors), even if those features are useless.

- Adjusted R² score : It rewards you for adding useful features and punishes you for adding useless ones.

Adjusted R^2

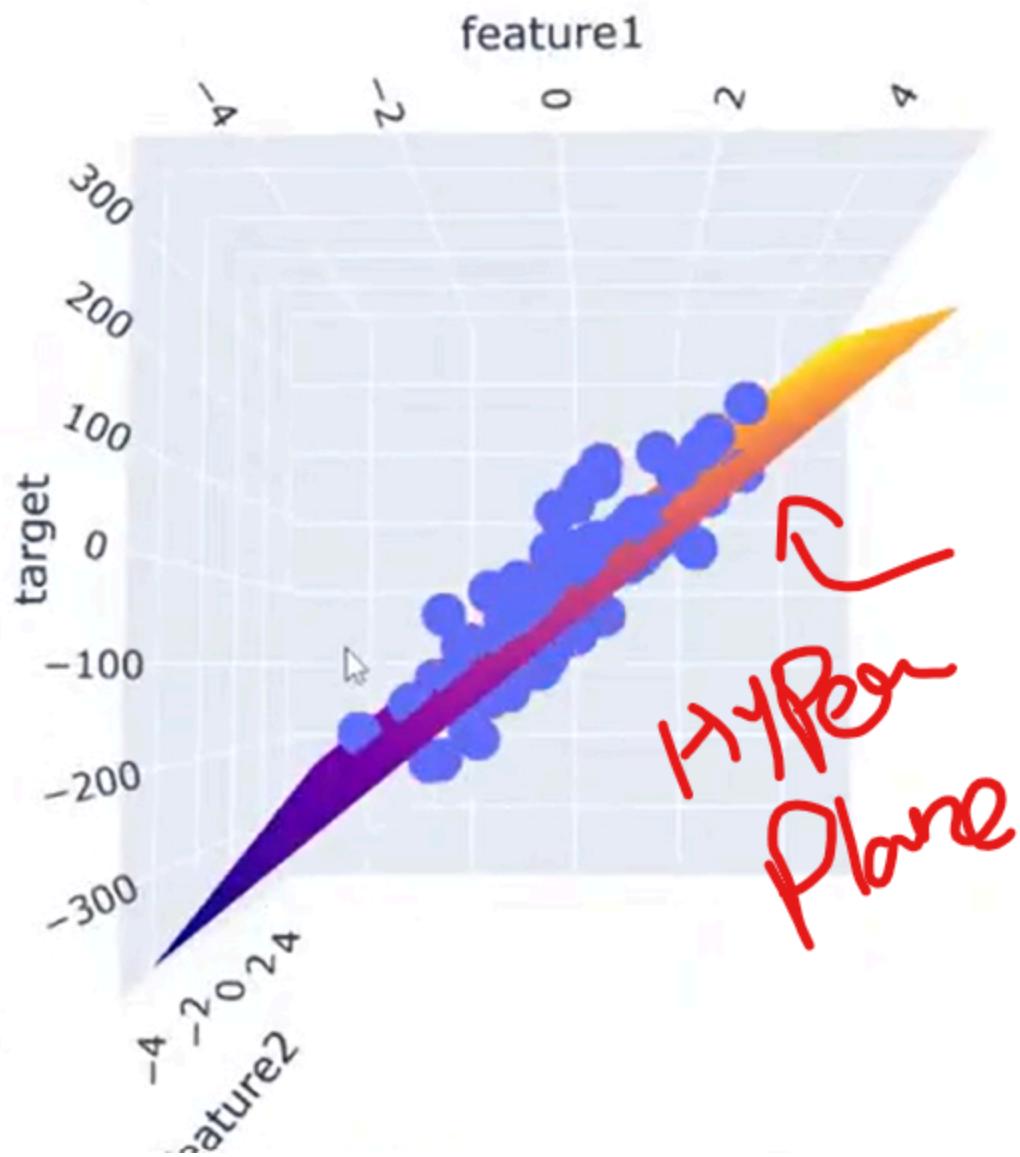
$$R^2_{adj} = 1 - \left[\frac{(1-R^2)(n-1)}{(n-1-K)} \right]$$

$R^2 \rightarrow$
 $n \rightarrow \text{no. of rows}$
 $K = \text{indepent}$
 $K=1, K=2, K=3$

Shape of the cost function creates a bowl shaped curve having a single global minimum.

Multi Linear Regression

- It is an extension of simple linear regression that uses multiple independent variables to predict the value of a dependent variable.
- The model is represented as: $y = \beta_0 + \beta_1x_1 + \beta_2x_2 + \dots + \beta_nx_n$
- In simple linear regression, we try to find the best-fit line because the relationship involves one independent variable and one dependent variable, which can be represented in a 2-dimensional plane. In multiple linear regression, we try to find the best-fit hyperplane because the relationship involves multiple independent variables, and the data exists in a higher-dimensional space (e.g., 3D or more).



- Formula for predicting value of y using matrix representation:

$$= \begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_n \end{bmatrix} = \begin{bmatrix} \beta_0 + \beta_1 x_{11} + \beta_2 x_{12} + \beta_3 x_{13} + \beta_4 x_{14} + \dots + \beta_m x_{1m} \\ \beta_0 + \beta_1 x_{21} + \beta_2 x_{22} + \dots + \dots + \beta_m x_{2m} \\ \beta_0 + \beta_1 x_{31} + \beta_2 x_{32} + \dots + \dots + \beta_m x_{3m} \\ \vdots \\ \beta_0 + \beta_1 x_{n1} + \beta_2 x_{n2} + \dots + \dots + \beta_m x_{nm} \end{bmatrix}$$

- This matrix can be decomposed to 2 matrices X and β where X is feature matrix and β is coefficient matrix. Dot product of these 2 matrices will give predicted value of y .

$$= \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1m} \\ 1 & x_{21} & x_{22} & \dots & x_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{nm} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_m \end{bmatrix} \quad (\text{n } \times 1) \quad \boxed{\hat{y} = X\beta}$$

Matrix representation

Multivariate regression uses a matrix-based setup to model multiple outcomes at the same time:

$$\mathbf{Y} = \mathbf{X}\mathbf{B} + \mathbf{E}$$

Where:

- Y is an $n \times p$ matrix of dependent variables (n observations, p response variables)
 - X is an $n \times (k+1)$ matrix of independent variables (including intercept)
 - B is a $(k+1) \times p$ matrix of regression coefficients
 - E is an $n \times p$ matrix of error terms
-
- The shape of X will be $(m, n+1)$ where m is number of training examples and n is number of features. And shape of β will be $(n+1, 1)$ because we have n features and each feature will have one coefficient. So the shape of predicted y will be $(m, 1)$ because we have m training examples. its $n+1$ because of intercept term.
 - cost function for multiple linear regression is as follows:

In Multiple Linear Regression (MLR), the key idea behind **Ordinary Least Squares (OLS)** is:

Choose coefficients β such that the **Sum of Squared Errors (SSE)** is minimized.

1) Core objects in Multiple Linear Regression

The MLR model is:

$$y = X\beta + \varepsilon$$

Where:

1.1 Output vector y ($n \times 1$)

y contains all actual target values:

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

- n = number of training examples
-

1.2 Feature matrix X ($n \times p$)

X stores all feature values.

- n = number of samples (rows)
- p = number of parameters (columns)
(including intercept column of ones)

Example: intercept + two features (x_1, x_2)

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} \\ 1 & x_{21} & x_{22} \\ \vdots & \vdots & \vdots \\ 1 & x_{n1} & x_{n2} \end{bmatrix}$$

1.3 Coefficient vector β ($p \times 1$)

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix}$$

- $\beta_0 = \text{intercept}$
 - $\beta_1, \beta_2, \dots = \text{weights of features}$
-

1.4 Predicted values \hat{y} ($n \times 1$)

$$\hat{y} = X\hat{\beta}$$

Which means:

$$\hat{y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix}$$

1.5 Residual (error) vector e ($n \times 1$)

Residuals are the difference between actual and predicted values:

$$e = y - \hat{y}$$

So:

$$e = \begin{bmatrix} y_1 - \hat{y}_1 \\ y_2 - \hat{y}_2 \\ \vdots \\ y_n - \hat{y}_n \end{bmatrix}$$

Each residual value:

$$e_i = y_i - \hat{y}_i$$

Residual transpose vector e^T is ($1 \times n$)

$$e^T = [y_1 - \hat{y}_1 \quad y_2 - \hat{y}_2 \quad \cdots \quad y_n - \hat{y}_n]$$

Residual vector e is ($n \times 1$)

$$e = \begin{bmatrix} y_1 - \hat{y}_1 \\ y_2 - \hat{y}_2 \\ \vdots \\ y_n - \hat{y}_n \end{bmatrix}$$

Now multiply them:

$$e^T e = [y_1 - \hat{y}_1 \quad y_2 - \hat{y}_2 \quad \cdots \quad y_n - \hat{y}_n] \begin{bmatrix} y_1 - \hat{y}_1 \\ y_2 - \hat{y}_2 \\ \vdots \\ y_n - \hat{y}_n \end{bmatrix}$$

Result is:

$$e^T e = (y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_2)^2 + \cdots + (y_n - \hat{y}_n)^2$$

$$\underline{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \quad \hat{\underline{y}} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix}_{n \times 1}$$

$$e = \underline{y} - \hat{\underline{y}} = \begin{bmatrix} y_1 - \hat{y}_1 \\ y_2 - \hat{y}_2 \\ \vdots \\ y_n - \hat{y}_n \end{bmatrix}$$

$$e^T e = \left[\begin{array}{c} y_1 - \hat{y}_1 \\ y_2 - \hat{y}_2 \\ \vdots \\ y_n - \hat{y}_n \end{array} \right]_{1 \times n} \cdot \left[\begin{array}{c} y_1 - \hat{y}_1 \\ y_2 - \hat{y}_2 \\ \vdots \\ y_n - \hat{y}_n \end{array} \right]_{n \times 1}^T =$$

$$e^T e = (y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_2)^2 + \dots + (y_n - \hat{y}_n)^2$$

$$= \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

✓ This is exactly the **Sum of Squared Errors (SSE)**.

3) Interpretation: What is SSE?

✓ SSE / RSS Meaning

$e^T e$ is known as:

- SSE = Sum of Squared Errors
- RSS = Residual Sum of Squares
- Sum of Squared Residuals

It measures the **total squared prediction error** of the model on the dataset.

If SSE is small → predictions are close to actual values.

If SSE is large → model predictions are far from actual values.

4) Why do we square the residuals?

For a sample:

$$e_i = y_i - \hat{y}_i$$

If we just summed residuals:

$$\sum_{i=1}^n e_i$$

Problems occur:

- positive and negative errors cancel out
- model may look “perfect” even when it isn’t

Example: errors $+10$ and -10

$$+10 + (-10) = 0$$

So we square them:

$$\sum_{i=1}^n e_i^2$$

Benefits of squaring:

- makes all errors positive
- penalizes large errors heavily
- gives a smooth differentiable objective (easy optimization)

5) The OLS objective in matrix form

Since:

$$e = y - X\beta$$

$$E = e^T e$$

$$E = (y - \hat{y})^T (y - \hat{y})$$

↑ minimize

$$E = e^T e$$

same

$$E = (y - \hat{y})^T (y - \hat{y}) = (y^T - \hat{y}^T) (y - \hat{y})$$

$$E = y^T y - \boxed{y^T \hat{y} - \hat{y}^T y} + \hat{y}^T \hat{y}$$

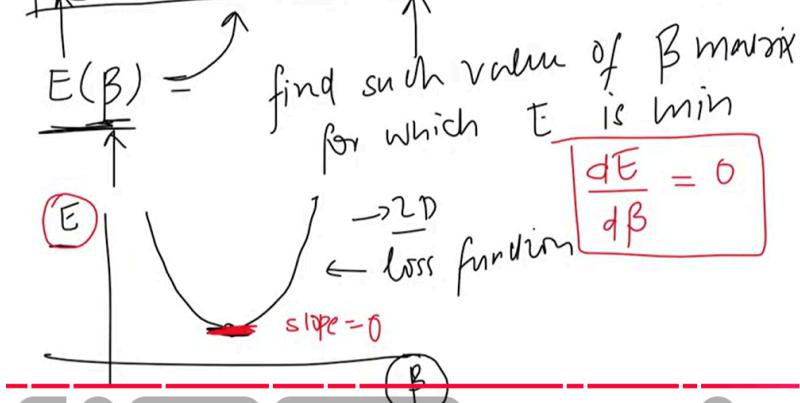
$$E = y^T y - 2 y^T \hat{y} + \hat{y}^T \hat{y}$$

→ eq ③

$$E = y^T y - 2y^T \hat{y} + \hat{y}^T \hat{y} \quad \rightarrow \text{eq } ③ \quad \hat{y} = x\beta$$

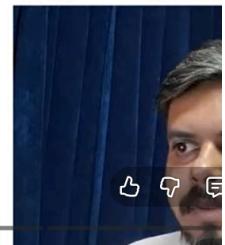
$$\begin{aligned} E &= y^T y - 2y^T x\beta + (x\beta)^T (x\beta) \\ E &= y^T y - 2y^T x\beta + \beta^T x^T x\beta \end{aligned} \quad \text{eq } ④ \quad y$$

$$\begin{aligned} E &= y^T y - 2y^T x\beta + (x\beta)^T (x\beta) \\ E &= y^T y - 2y^T x\beta + \beta^T x^T x\beta \end{aligned} \quad \text{eq } ④ \quad y$$



$y \rightarrow$ data output
 $x \rightarrow$ data input
 $f(x) = x^2 \rightarrow y$

Differentiate



We will differentiate the equation shown in the image to find the best value of beta which minimizes the error. The equation after performing differentiation is shown below:

$$\beta = (x^T x)^{-1} x^T y \quad \text{eq } ⑤$$

The shape of β will be $(m+1, 1)$ because we have m features and one intercept term.

$$\beta = \text{values}^{(m+1 \times 1)}$$

$$\beta = \text{values}^{(m+1 \times 1)}$$

$(X^T X)^{-1}$

$m+1 \times n \quad n \times (m+1)$

$(m+1) \times (m+1) \quad (m+1) \times n$

$(m+1) \times n \quad n \times 1$

y

(m+1) x 1

begin

The diagram illustrates the dimensions of the matrices involved in the formula. It shows a vertical stack of terms:

- The top term is $\beta = \text{values}^{(m+1 \times 1)}$.
- The second term is $(X^T X)^{-1}$.
- The third term is $(m+1 \times n)$, with a bracket under it labeled $n \times (m+1)$.
- The fourth term is $(m+1) \times (m+1)$, with a bracket under it labeled $(m+1) \times n$.
- The fifth term is $(m+1) \times n$, with a bracket under it labeled $n \times 1$.
- The bottom term is y .

 Red arrows point from the right side of the first four terms up towards the fifth term (y), indicating the flow of the calculation. A red curly brace groups the last three terms: $(m+1) \times n$, $n \times 1$, and y , with the label "begin" written above it.

$$e^T e = (y - X\beta)^T (y - X\beta)$$

6) Expanding the objective

Expanding:

$$(y - X\beta)^T (y - X\beta)$$

gives:

$$\begin{aligned} & \$\$ \\ & (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \end{aligned}$$

$$\begin{aligned} & \mathbf{y}^T \mathbf{y} - 2\boldsymbol{\beta}^T \mathbf{X}^T \mathbf{y} + \boldsymbol{\beta}^T \mathbf{X}^T \mathbf{X} \boldsymbol{\beta} \\ & \$\$ \end{aligned}$$

7) Minimization leads to Normal Equation

We minimize SSE by differentiating with respect to $\boldsymbol{\beta}$ and setting gradient to 0.

Result:

$$\mathbf{X}^T \mathbf{X} \boldsymbol{\beta} = \mathbf{X}^T \mathbf{y}$$

This is the **Normal Equation**.

Solving for $\boldsymbol{\beta}$:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

This is the **closed-form OLS solution** (works if $\mathbf{X}^T \mathbf{X}$ is invertible).

8) Geometric intuition (very important)

- \mathbf{y} is a vector in n -dimensional space
- $\hat{\mathbf{y}}$ lies in the column space (span) of \mathbf{X}

OLS chooses $\hat{\mathbf{y}}$ to be the **projection of \mathbf{y}** onto the space spanned by columns of \mathbf{X} .

Residual:

$$\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}}$$

OLS ensures residual is perpendicular to the feature space:

$$\mathbf{X}^T \mathbf{e} = 0$$

Substitute $e = y - X\beta$:

$$X^T(y - X\beta) = 0$$

Which becomes:

$$X^T X \beta = X^T y$$

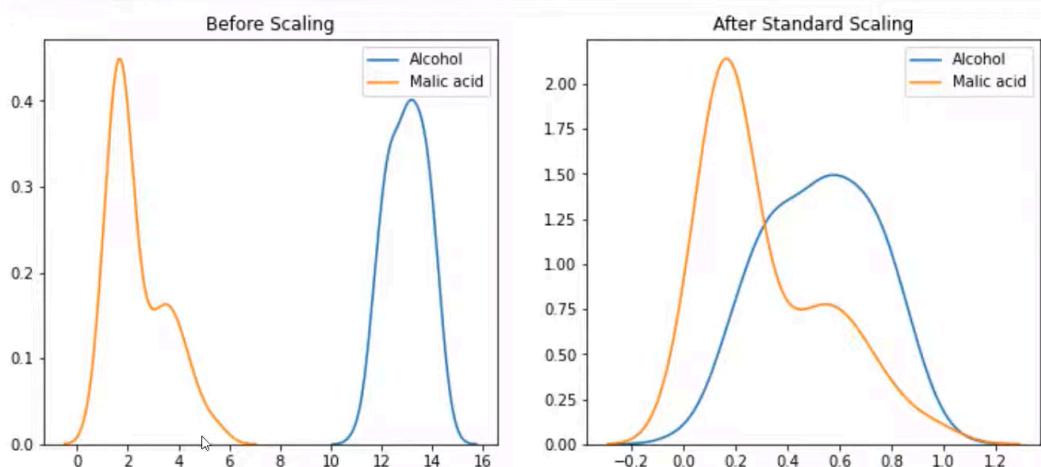
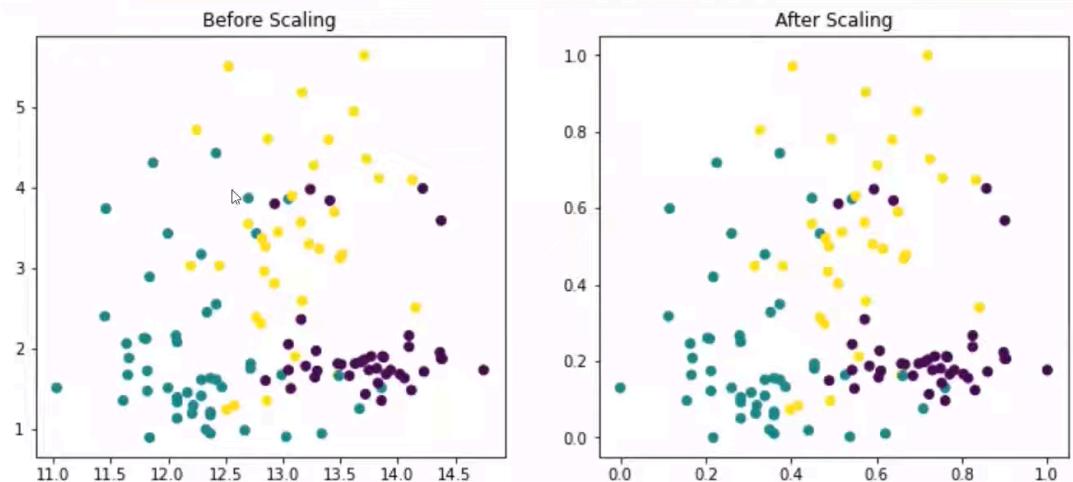
So the normal equation is actually a **perpendicularity condition**:

error must be orthogonal to the space of predictors.

Feature Scaling (Important for feature engineering):

- Feature scaling are mainly of 2 types which are as follows:
 - **Normalization**
 - Normalization is a data preprocessing technique that rescales numerical features to a common range, most commonly between 0 and 1, so that all features are on the same scale and differences in measurement units are eliminated.
 - It is not robust to outliers because if there is an outlier then min and max value will be affected and all other values will be compressed in small range.
 - It is not necessary that the distribution shape will remain same after normalization.
 - Types of normalization :
 - **Min Max Scaling:** It scales the data to a fixed range, usually 0 to 1. It is mostly used in image processing where pixel values are between 0 to 255 because we know the maximum and minimum values and we want to scale it between 0 and 1.

$$x_{\text{new}} = \frac{x_i - \min(X)}{\max(X) - \min(X)}$$



- As you can see that both these distribution first were not overlapping because of different scale but after min max scaling both are overlapping because both are in same scale now.
- **Max Abs Scaling** : It scales the data by dividing each value by the maximum absolute value of that feature. The resulting values will be in the range [-1, 1]. It is mostly used when data is sparse where significant portion of the values are zero.

$$x_{scaled} = \frac{x}{\max(x)}$$

- **Mean Normalization** : It is a technique used to scale features by subtracting the mean and dividing by the range (max - min) of the feature. This centers the data around zero and scales it to a range of -1 to 1.

Mean Normalization

Saturday, April 10, 2021 1:16 PM

Handwritten notes showing mean normalization. On the left, there is a list of values: wt [3.0], 2.8, 2.2, 1.6, 1.5, 1.4, 1.3, 1.2, 1.0, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0.0. An arrow points from this list to the formula $x'_i = \frac{x_i - \bar{x}_{\text{mean}}}{x_{\text{max}} - x_{\text{min}}}$. Above the formula, 'mean' and 'center' are written above \bar{x}_{mean} , and $x_{\text{max}} - x_{\text{min}}$ is written below the denominator.

$$x'_i = \frac{x_i - \bar{x}_{\text{mean}}}{x_{\text{max}} - x_{\text{min}}}$$

- Robust Scaling : It is used when data contains outliers. It uses median and interquartile range for scaling. It subtracts median from each value and then divides it by interquartile range.

Diagram illustrating the Robust Standardised Value formula:

$$x' = \frac{x - \text{median}(x)}{\text{Interquartile Range} = Q3 - Q1}$$

The formula is shown with arrows pointing to its components:

- An arrow from "Robust Standardised Value" points to the formula.
- An arrow from "Original Value" points to x .
- An arrow from "Sample Median" points to $\text{median}(x)$.
- An arrow from "Interquartile Range = Q3 - Q1" points to the denominator $(Q3 - Q1)$.

◦ Standardization

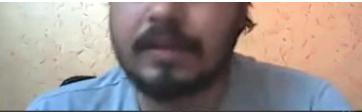
- It is used when we know the distribution of data. It scales data such that mean becomes 0 and standard deviation becomes 1. It is based on z score and that's why it is also called z score normalization.
- Z-score normalization (also called standardization) is the process of transforming every value in a feature into its z-score.
- Shape of distribution does not change after standardization.
- Standardization (z-score normalization) involves mean centering followed by scaling by the standard deviation. This transformation results in a feature with zero mean and unit variance, effectively rescaling the data without changing its distribution shape.

- If we standardize the data then we can easily find outliers because outliers will have z score greater than 3 or less than -3.
- Algorithms like decision tree, random forest do not require feature scaling because they are not based on distance.

$$z = \frac{X - \mu}{\sigma}$$

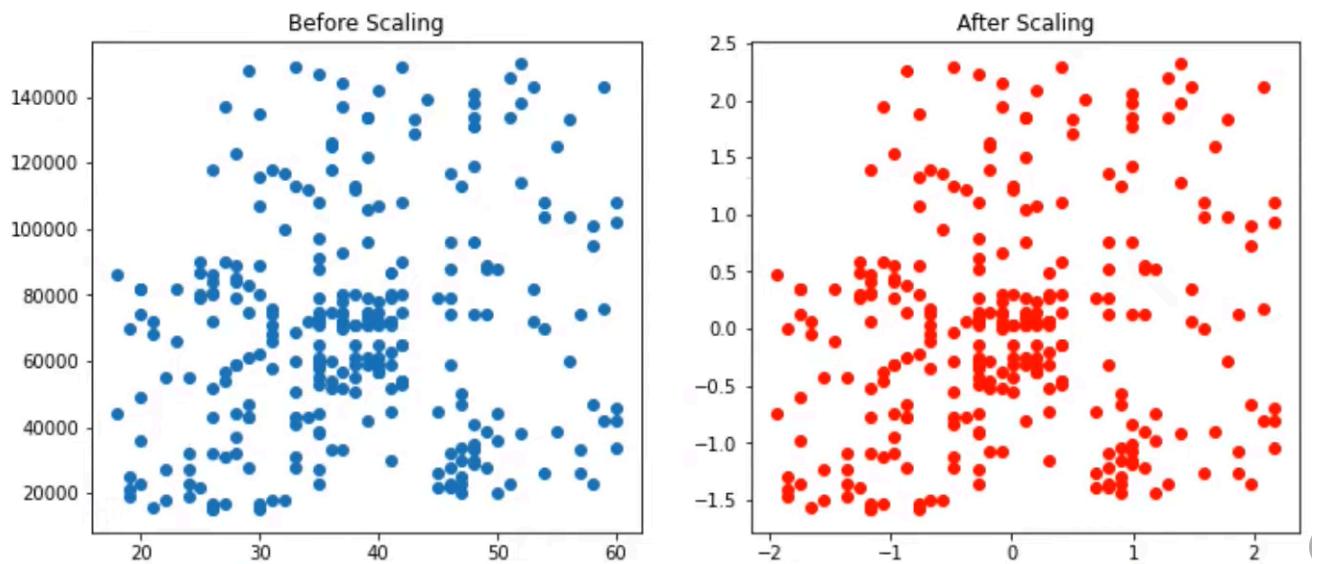
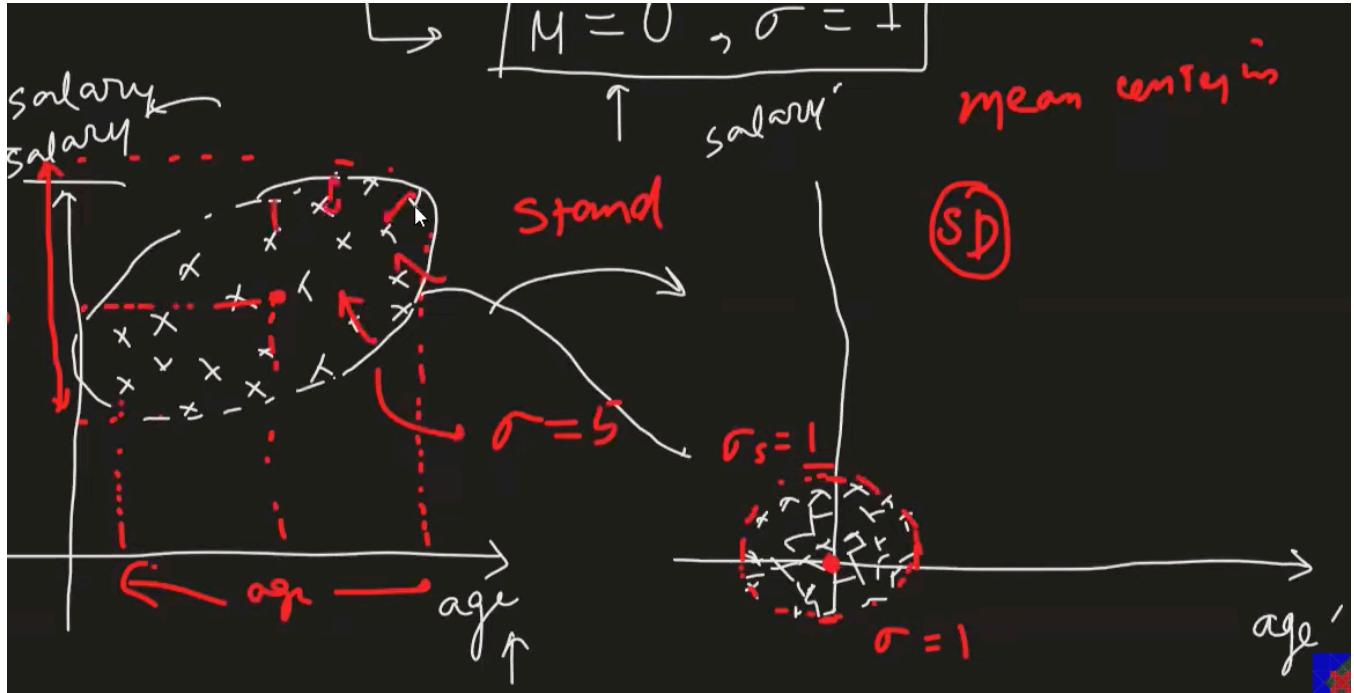
When to use Standardization?

Friday, April 9, 2021 4:28 PM



Algorithm(s)	Reason of applying feature scaling
1. K-Means	Use the Euclidean distance measure.
2. K-Nearest-Neighbours	Measure the distances between pairs of samples and these distances are influenced by the measurement units
3. Principal Component Analysis (PCA)	Try to get the feature with maximum variance
4. Artificial Neural Network	Apply Gradient Descent
5. Gradient Descent	Theta calculation becomes faster after feature scaling and the learning rate in the update equation of Stochastic Gradient Descent is the same for every parameter

- The main reason to do feature scaling is that some machine learning algorithms use distance between data points to make predictions. If one feature has a wide range of values, it can dominate the distance calculations and lead to biased results. By scaling features to a similar range, we ensure that all features contribute equally to the distance calculations.



Feature Encoding (Important for feature engineering):

Loss Functions

Formal definition

If you see:

$$\operatorname{argmin}_x f(x)$$

It means:

"The value(s) of x that make $f(x)$ as small as possible."

$$L(\beta_0, \beta_1, \beta_2) = \operatorname{argmin}_{\beta_0, \beta_1, \beta_2} \left[\sum_{i=1}^n (y_i - \hat{y}_i)^2 \right]$$

Gradient Descent

Problems faced in Gradient Descent:

Problems faced in Optimization

21 April 2023 16:39

→ G D

1. **Non-convexity:** For many machine learning models, such as artificial neural networks, the loss function is non-convex, which means it has a complex landscape with multiple local minima, maxima, and saddle points. This makes it difficult for optimization algorithms to find the global minimum and can result in suboptimal solutions.
2. **Ill-conditioning:** The loss function may be ill-conditioned, meaning the gradients in some dimensions are much larger than in others. This can cause gradient-based optimization algorithms, such as gradient descent, to oscillate and converge slowly.
3. **Vanishing and exploding gradients:** In deep neural networks, the gradients can become very small (vanish) or very large (explode) as they propagate through the layers. This can lead to slow convergence or unstable training dynamics, making it difficult to optimize the loss function.
4. **Overfitting:** When optimizing the loss function, the algorithm may overfit the training data, resulting in a model that performs poorly on unseen data. This occurs when the model is too complex and learns the noise in the training data instead of the underlying patterns.
5. **Scalability:** For large-scale problems with a high number of features, instances, or model parameters, optimizing the loss function can be computationally expensive and time-consuming. This can limit the applicability of certain optimization techniques or require significant computational resources.

Gradient Descent core rule

$$w = w - \alpha \frac{dL}{dw}$$

Where:

- α = learning rate (step size)
- $\frac{dL}{dw}$ = derivative (tells direction)

So:

Differentiation tells direction to move to reduce loss.

Gradient descent basically helps us to find the minima of a function. It is an optimization

algorithm used to minimize a function by iteratively moving towards the steepest descent, which is the direction of the negative gradient.

Step 1 : Initialize the parameters (weights) randomly or with some initial values.

Step 2 : Calculate the predicted output using current parameters.

Step 3 : Compute the loss (error) between predicted output and actual output using a loss function.

Step 4: Calculate the gradient of the loss function with respect to each parameter. The gradient represents the direction and rate of change of the loss function. It is computed using derivatives. If the gradient of a parameter is positive, increasing that parameter increases the loss, so the parameter value is decreased. If the gradient is negative, increasing the parameter decreases the loss, so the parameter value is increased. In this way, the parameters are adjusted in the direction opposite to the gradient to move toward the minimum of the loss function.

Step 5: Update the parameters using the calculated gradients and a learning rate, which determines the step size for each update. The learning rate is a hyperparameter that needs to be chosen carefully; too large a learning rate can cause overshooting of the minimum, while too small a learning rate can lead to slow convergence. So updated parameter = current parameter - learning rate * gradient(slope).

Step 6: Repeat steps 2 to 5 until convergence, which occurs when the change in loss is below a certain threshold or after a fixed number of iterations.

For $L(m,b)$ first we will calculate partial derivative with respect to m and b .

8 steps

1) init random vals for m and b
 $m = 1$ and $b = 0$

2) epochs = 100, $\eta = 0.01$

for i in epochs:

$$b = b - \eta \frac{\text{slope}}{\text{y}}$$

$$m = m - \eta \frac{\text{slope}}{\text{x}}$$

1 Partial derivative with respect to m

$$\frac{\partial L}{\partial m} = \sum_{i=1}^n 2(y_i - mx_i - b)(-x_i)$$

$$\boxed{\frac{\partial L}{\partial m} = -2 \sum_{i=1}^n x_i(y_i - mx_i - b)}$$

2 Partial derivative with respect to b

$$\frac{\partial L}{\partial b} = \sum_{i=1}^n 2(y_i - mx_i - b)(-1)$$

$$\boxed{\frac{\partial L}{\partial b} = -2 \sum_{i=1}^n (y_i - mx_i - b)}$$

Types of Gradient Descent:

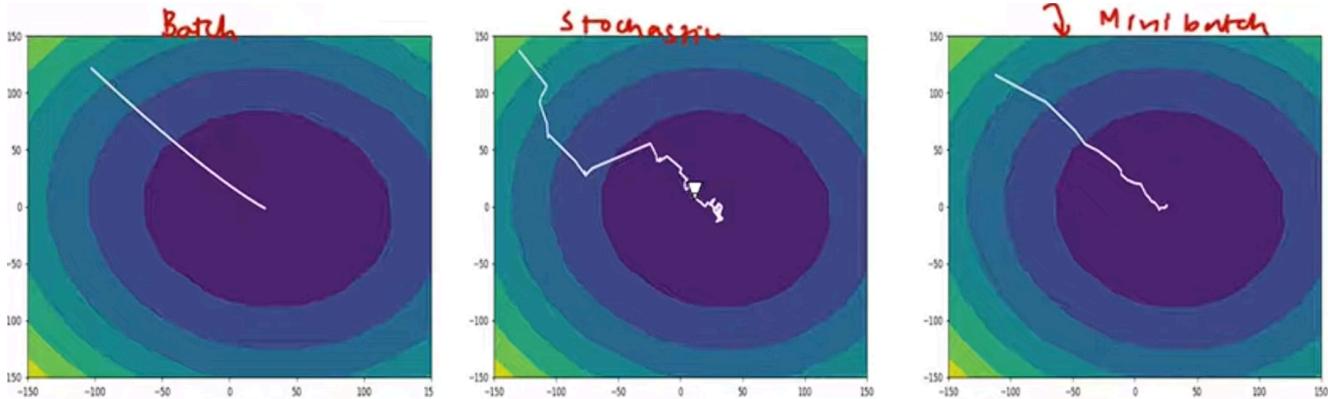
- **Batch Gradient Descent** : It uses the entire dataset to compute the gradient of the loss function for each iteration. It provides a stable and accurate estimate of the gradient but can be computationally expensive for large datasets. It is mainly used when the dataset is small enough to fit into memory and also converges smoothly towards the minimum of the loss function. Convex functions are best suited for batch gradient descent because BGD can get stuck in local minima or saddle points in non convex function.
- **Stochastic Gradient Descent** : It updates the parameters using the gradient computed from a single randomly selected data point. This makes it much faster and allows it to start improving the model right away. However, the updates can be noisy and may lead to a less stable convergence. Its name is stochastic because of the randomness involved in selecting data points for each update.
- The final solution may oscillate around the minimum rather than converging smoothly because of randomness in selecting data points for each update.
- In stochastic it is possible that **step n+1 is worse than step n because of randomness**. while in batch it is not possible because it uses entire data.
- In SGD we can face a problem wherein even near the solution the updates can be large and erratic because of high variance in gradient estimates from single data points. To mitigate this, techniques like learning rate scheduling (gradually decreasing the learning rate over time) and data shuffling (randomizing the order of data points before each epoch) are commonly used.
- Advantages of stochastic gradient descent include faster convergence, ability to escape local minima, and suitability for large datasets. It is mainly used when the dataset is too large to fit into memory or when we want to quickly iterate over the data. Non-convex functions are best suited for stochastic gradient descent.
- Gradient estimates in stochastic gradient descent are noisy, leading to oscillations around

the minimum; requires careful learning rate scheduling and data shuffling.

- SGD's noise can actually help escape saddle points and poor local minima. Deep learning is non-convex + large-scale and that's why SGD practical and effectively always used to train deep learning models.

- **Why SGD uses random rows (not sequential)**

- SGD updates model parameters using one random data row (or sequential rows after shuffling) because:
 - Prevents ordering bias: real datasets are often sorted/grouped (by class, time, category). Sequential updates can make SGD learn in a biased direction.
 - Reduces correlated gradients: consecutive rows are similar → gradients become similar → slow/unstable learning. Randomization breaks this correlation.
 - Unbiased gradient estimate: random sampling ensures the expected SGD gradient points toward the true/full gradient direction.
- **Mini-Batch Gradient Descent** : It is a compromise between batch and stochastic gradient descent. It divides the dataset into small batches and computes the gradient for each batch. This approach balances the computational efficiency of stochastic gradient descent with the stability of batch gradient descent.



Gradient Descent for n dimensional data

Mathematical Formulation
 Saturday, May 22, 2021 3:38 PM

n -dim-dataset 3-cols

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \quad (\text{eqn}) \quad (\text{cgp}) \quad (\text{ig})$$

cgpa	iq	lpa
x_1	x_2	y
8.1	93	3.2
7.5	95	3.5

$$\{ \beta_0, \beta_1, \beta_2 \} \quad L(\beta_0, \beta_1, \beta_2)$$

1) Random values

$$\beta_0 = 0, \beta_1, \beta_2 = 1$$

$$\frac{\partial L}{\partial \beta_0} = \frac{\partial L}{\partial \beta_1}, \frac{\partial L}{\partial \beta_2}$$

\Rightarrow epoch = 100, $\eta = 0.1$

$$\beta_0 = \beta_0 - \eta \underline{\text{slope}}$$

$$\beta_1 = \beta_1 - \eta \underline{\text{slope}}$$

$$\beta_2 = \beta_2 - \eta \underline{\text{slope}}$$

$$L = \frac{1}{2} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

$$= \frac{1}{2} [(y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_2)^2]$$

$$= \frac{1}{2} [(y_1 - \beta_0 - \beta_1 x_{11} - \beta_2 x_{12})^2 + (y_2 - \beta_0 - \beta_1 x_{21} - \beta_2 x_{22})^2]$$

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

2 Dots Only

xi	x2	y
8.1	93	3.2
7.5	95	3.5

$$\hat{y}_1 = \beta_0 + \beta_1 x_{11} + \beta_2 x_{12}$$

$$\hat{y}_2 = \beta_0 + \beta_1 x_{21} + \beta_2 x_{22}$$

Partial derivative with respect to β_0

$$\frac{\partial L}{\partial \beta_0} = \frac{1}{2} \left[2(y_1 - \hat{y}_1)(-1) + 2(y_2 - \hat{y}_2)(-1) \right]$$

$$\frac{\partial L}{\partial \beta_0} = -\frac{1}{2} \left[(y_1 - \hat{y}_1) + (y_2 - \hat{y}_2) \right]$$

Now generalising the method for n dimension

$$\begin{aligned} & \frac{\partial L}{\partial \beta_0} = \frac{1}{2} \left[(y_1 - \hat{y}_1) + (y_2 - \hat{y}_2) + (y_3 - \hat{y}_3) + \dots + (y_n - \hat{y}_n) \right] \\ &= \frac{-2}{n} \left[\underbrace{(y_1 - \hat{y}_1)}_{-} + \underbrace{(y_2 - \hat{y}_2)}_{-} + \underbrace{(y_3 - \hat{y}_3)}_{-} + \dots + \underbrace{(y_n - \hat{y}_n)}_{-} \right] \\ &= \boxed{\frac{-2}{n} \sum_{i=1}^n (y_i - \hat{y}_i)} = \frac{\partial L}{\partial \beta_0} \end{aligned}$$

凸 4

Partial derivative with respect to β_1

$$\begin{aligned} L &= \frac{1}{2} \sum_{i=1}^2 (y_i - \hat{y}_i)^2 \quad \frac{\partial}{\partial \beta_1} - \beta_1 x_{11} = -x_{11} \\ L &= \frac{1}{2} \left[(y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_2)^2 \right] \\ L &= \frac{1}{2} \left[(y_1 - \beta_0 - \beta_1 x_{11} - \beta_2 x_{12})^2 + (y_2 - \beta_0 - \beta_1 x_{21} - \beta_2 x_{22})^2 \right] \end{aligned}$$

x_1	x_2	y
8.1	9.3	3.2
7.5	9.5	3.5

$$\frac{\partial L}{\partial \beta_1} = \frac{1}{2} \left[2(y_1 - \hat{y}_1)(-x_{11}) + 2(y_2 - \hat{y}_2)(-x_{21}) \right]$$

Now generalising the method for n dimension

$$\frac{\partial L}{\partial \beta_1} = \frac{1}{2} \left[2(y_1 - \hat{y}_1)(-x_{11}) + 2(y_2 - \hat{y}_2)(-x_{21}) \right] \quad \boxed{x_{i1}} \quad \begin{matrix} x_{21} \\ x_{31} \\ x_{n1} \end{matrix}$$

$$\frac{\partial L}{\partial \beta_1} = \frac{-2}{n} \left[(y_1 - \hat{y}_1)x_{11} + (y_2 - \hat{y}_2)x_{21} + (y_3 - \hat{y}_3)x_{31} + \dots + (y_n - \hat{y}_n)x_{n1} \right]$$

$\frac{\partial L}{\partial \beta_1} = \frac{-2}{n} \sum_{i=1}^n (y_i - \hat{y}_i) \boxed{x_{i1}}$

$x_{i1} \rightarrow 1\text{ col data}$
 $\beta_1 \rightarrow \text{values of 1 col.}$



Partial derivative with respect to β_2

$$\frac{\partial L}{\partial \beta_2} = -\frac{2}{n} \sum_{i=1}^n (y_i - \hat{y}_i) x_{i2}$$

Generalised partial derivative for $\beta_1 \dots \beta_n$

$$\frac{\partial L}{\partial \beta_m} = -2 \sum_{i=1}^n (y_i - \hat{y}_i) x_{im}$$

What is statistical inference problems?

Statistical inference is the process of drawing conclusions about a population based on a sample of data taken from that population. It involves using statistical methods to analyze the sample data and make inferences about the characteristics, relationships, or patterns that exist in the larger population.

Reducible error vs Irreducible error:

- Reducible Error: This is the portion of the error that can be reduced or minimized by improving the model. It includes errors due to bias (systematic errors from incorrect assumptions in the model) and variance (errors due to sensitivity to fluctuations in the training data). By selecting a better model, tuning hyperparameters, or using more data, we can reduce this type of error.
- Irreducible Error: This is the portion of the error that cannot be reduced, regardless of the model used. It is inherent in the data due to noise, measurement errors, or other random factors that affect the outcome. No matter how well we optimize our model, this error will always be present.

For example, in a house price prediction model, reducible error could come from using a linear model when the relationship is non-linear (bias) or from overfitting to a small training set (variance). Irreducible error could come from unpredictable factors like sudden market changes or unique features of individual houses that are not captured in the data.

Difference between statistical learning and machine learning:

Statistical Learning:

- Focuses on understanding the underlying relationships between variables and making inferences about the population.
- Often uses traditional statistical methods like linear regression, logistic regression, and hypothesis testing.
- Typically deals with smaller datasets and focuses on parameter estimation.

Machine Learning:

- Focuses on making accurate predictions and improving performance on specific tasks.
- Utilizes a wide range of algorithms, including decision trees, neural networks, and ensemble methods
- Typically deals with large datasets and focuses on pattern recognition.

Difference between inference and prediction:

Inference:**

- Inference is the process of drawing conclusions about a population based on a sample of data.
- It focuses on understanding the relationships between variables and identifying significant factors that influence the outcome.
- It often involves hypothesis testing and estimating parameters of a statistical model.

Prediction:

- Prediction is the process of using a model to forecast or estimate the value of an outcome variable based on input features.
- The goal of prediction is to achieve high accuracy in estimating future outcomes.

Regression Analysis:

Regression analysis is a statistical method used to examine the relationship between a dependent variable (also known as the outcome or response variable) and one or more independent variables (also known as predictors or explanatory variables). The primary goal of regression analysis is to model the relationship between these variables in order to make predictions, understand the strength and nature of the relationships, and identify significant factors that influence the dependent variable.

Using libraries like statsmodels in Python, we can perform regression analysis and obtain detailed statistical summaries of the results, including coefficients, p-values, R-squared values,

and more.

OLS Regression Results						
Dep. Variable:	Sales	R-squared:	0.897			
Model:	OLS	Adj. R-squared:	0.896			
Method:	Least Squares	F-statistic:	570.3			
Date:	Sat, 29 Apr 2023	Prob (F-statistic):	1.58e-96			
Time:	07:32:56	Log-Likelihood:	-386.18			
No. Observations:	200	AIC:	780.4			
Df Residuals:	196	BIC:	793.6			
Df Model:	3					
Covariance Type:	nonrobust					
coef	std err	t	P> t	[0.025	0.975]	
const	2.9389	0.312	9.422	0.000	2.324	3.554
TV	0.0458	0.001	32.809	0.000	0.043	0.049
Radio	0.1885	0.009	21.893	0.000	0.172	0.206
Newspaper	-0.0010	0.006	-0.177	0.860	-0.013	0.011
Omnibus:	60.414	Durbin-Watson:	2.084			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	151.241			
Skew:	-1.327	Prob(JB):	1.44e-33			
Kurtosis:	6.332	Cond. No.	454.			

TSS (Total Sum of Squares) : It measures the total variability in the dependent variable. It quantifies how much the observed values of the dependent variable deviate from their mean.

$$\text{TSS} = \sum (y_i - \bar{y})^2$$

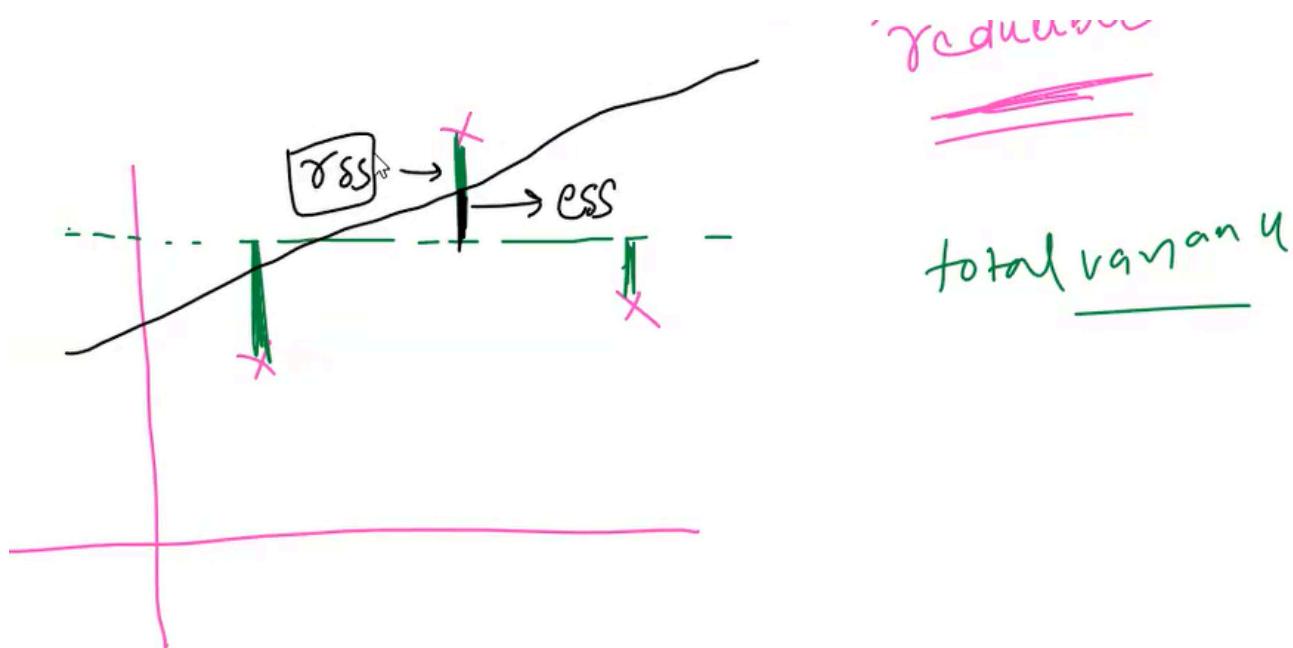
Definition of RSS

The Residual Sum of Squares (RSS) is a key concept in regression analysis that measures the total deviation of the predicted values from the actual values. It is calculated as the sum of the squares of the residuals, which are the differences between the observed values and the predicted values from the model.

$$\text{RSS} = \sum (y_i - \hat{y}_i)^2$$

ESS (Explained Sum of Squares) : It quantifies how much the predicted values generated by the regression model deviate from the mean of the dependent variable.

$$TSS - RSS$$



Degree of Freedom (DF) : It refers to the number of independent pieces of information that are available to estimate a parameter or calculate a statistic. In regression analysis, degrees of freedom are used to determine the number of independent observations that contribute to the estimation of the regression coefficients.

Formula for calculating degrees of freedom in regression analysis:

- For the regression model: $DF = k$ (where k is the number of independent variables)
- For the residuals (error term): $DF = n - k - 1$ (where n is the total number of observations and k is the number of independent variables)

Df Total : It represents the total number of observations minus one ($n - k - 1 + k = n - 1$), where n is the total number of observations in the dataset.

R-squared (R^2) : It is a statistical measure that represents the proportion of the variance in the dependent variable that is explained by the independent variables in the regression model. It ranges from 0 to 1, where a value of 1 indicates that the model perfectly explains the variability in the dependent variable, while a value of 0 indicates that the model does not explain any of the variability.

(155).

$$R^2 = \frac{ESS}{TSS}$$

Adjusted R-squared : It is a modified version of R-squared that takes into account the

f-statistic : It is used to determine whether there is a significant relationship between the dependent variable and the independent variables in the regression model. It tests the null hypothesis that all regression coefficients are equal to zero (i.e., no relationship) against the alternative hypothesis that at least one coefficient is not equal to zero.

Regression analysis can be used for various purposes, including:

1. Prediction: Estimating the value of the dependent variable based on the values of the independent variables.
2. Inference: Understanding the relationships between variables and determining which independent variables have a significant impact on the dependent variable.
3. Modeling: Creating mathematical models that describe the relationships between variables.
4. Hypothesis Testing: Testing hypotheses about the relationships between variables.

Regression analysis assumptions:

1. Linearity: The relationship between the independent and dependent variables is linear.
2. Independence: The observations are independent of each other.
3. Homoscedasticity: The variance of the residuals (errors) is constant across all levels of the independent variables.
4. Normality: The residuals are normally distributed.
5. No Multicollinearity: The independent variables are not highly correlated with each other.

6. No Autocorrelation: The residuals are not correlated with each other, especially in time series data.
7. Exogeneity: The independent variables are not correlated with the error term.
8. Measurement Level: The dependent variable is continuous, and the independent variables can be continuous or categorical.
9. No Outliers: There are no extreme values that can unduly influence the results of the regression analysis.
10. Correct Model Specification: The model includes all relevant variables and is correctly specified.
11. Stationarity (for time series data): The statistical properties of the time series data do not change over time.
12. No Perfect Multicollinearity: There should be no perfect linear relationship among the independent variables.
13. No Measurement Error: The independent variables are measured without error.
14. Additivity: The effects of the independent variables on the dependent variable are additive.
15. No Omitted Variable Bias: All relevant variables that influence the dependent variable are included in the model.
16. Sufficient Sample Size: The sample size should be large enough to provide reliable estimates of the regression coefficients.
17. Model Stability: The relationships modeled should be stable over time and across different samples.
18. Linearity in Parameters: The model should be linear in terms of the parameters (coefficients), even if the relationship between variables is non-linear.
19. No Influential Points: There should be no individual data points that have a disproportionate impact on the regression results.
20. Correct Functional Form: The chosen functional form of the regression model should accurately represent the underlying relationship between the variables.
21. No Endogeneity: The independent variables should not be correlated with the error term, which can lead to biased estimates.
22. No Heteroscedasticity: The variance of the residuals should be constant across all levels of the independent variables.

if all these assumptions are met, the results of the regression analysis can be considered valid and reliable. Violations of these assumptions can lead to biased estimates, incorrect inferences, and unreliable predictions.

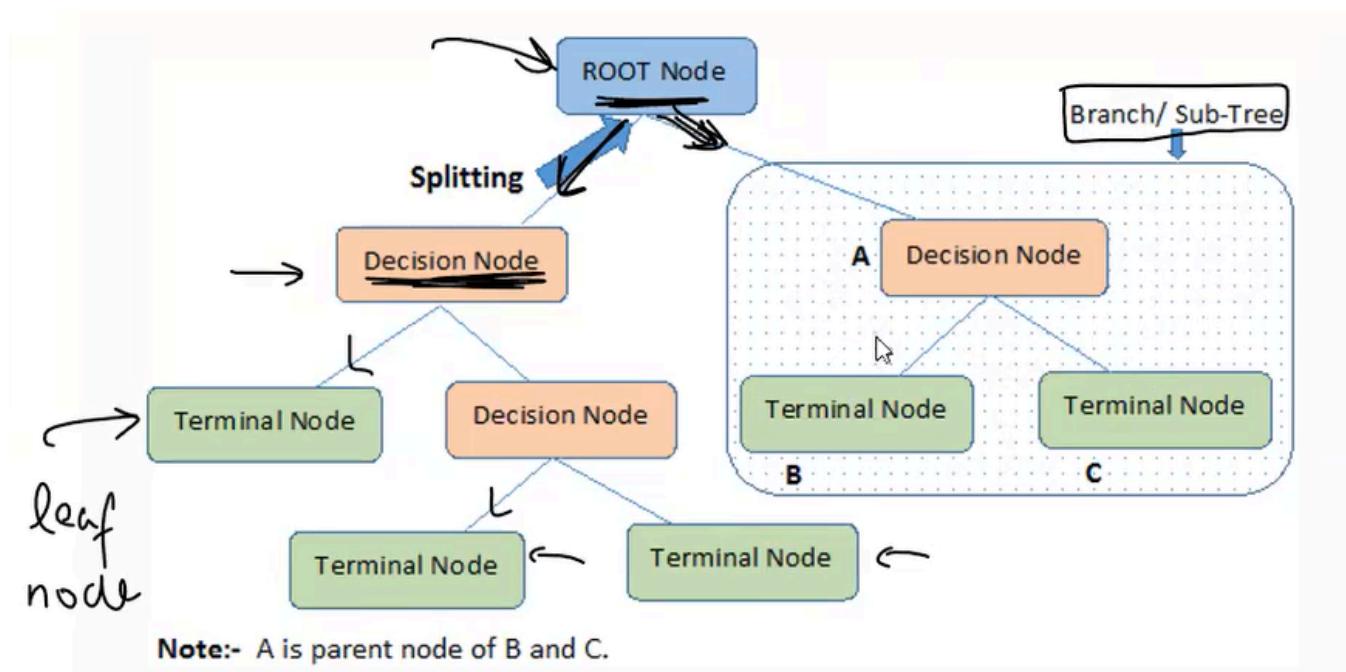
TREE BASED MODELS

Tree based models are a type of supervised machine learning algorithms that use decision trees to make predictions. Decision trees are hierarchical structures that recursively split the data into subsets based on the values of input features. Each internal node represents a feature, each branch represents a decision rule, and each leaf node represents a predicted outcome.

Tree based models can be used for both classification and regression tasks.

Types of Tree based models:

Decision Tree



It is a simple tree based model that splits the data into subsets based on the values of input features. It can be used for both classification and regression tasks.

It is white box model because we can easily interpret the model by visualizing the decision tree. We can see how the model is making decisions and which features are important for making predictions.

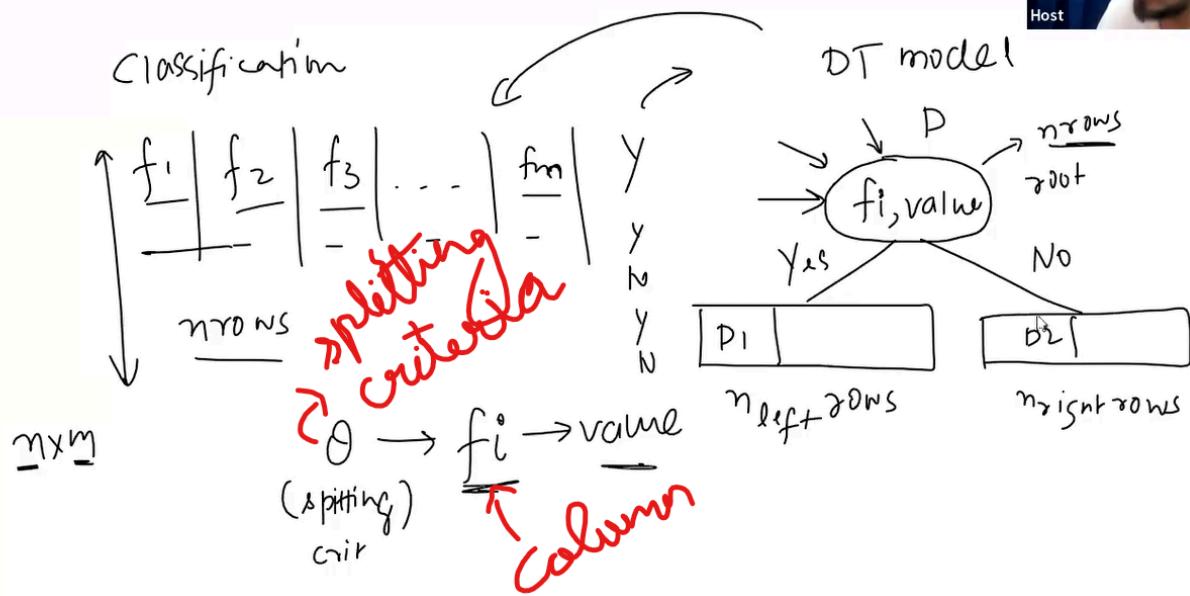
It is non parametric model because it does not make any assumptions about the underlying data distribution. It can handle both numerical and categorical data.

It is mother of all tree based models because other tree based models like random forest, gradient boosting etc are based on decision tree.

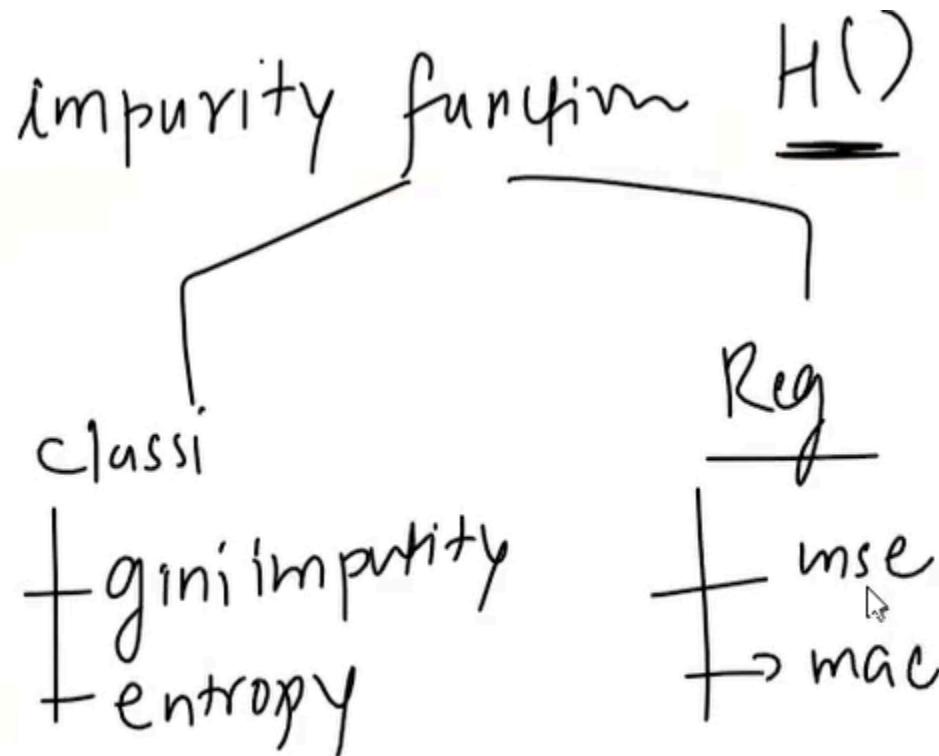
Different algorithms to build decision tree:

- **ID3 (Iterative Dichotomiser 3)** : It uses information gain as the splitting criterion. It selects the feature that maximizes the information gain at each node. It can only handle categorical features.
- **C4.5** : It is an extension of ID3 that uses gain ratio as the splitting criterion. It selects the feature that maximizes the gain ratio at each node. It can handle both continuous and categorical features.
- **CART (Classification and Regression Trees)** : It uses Gini impurity as the splitting criterion for classification tasks and mean squared error for regression tasks. It selects the feature that minimizes the Gini impurity or mean squared error at each node. It creates binary trees, where each node has at most two children. It can handle both continuous and categorical features. sklearn's DecisionTreeClassifier and DecisionTreeRegressor are based on CART algorithm.

Impurity : Impurity is a measure of how mixed the classes are in a dataset. It quantifies the degree of uncertainty or disorder in the data. The goal of a decision tree is to create splits that result in subsets with low impurity, meaning that the instances within each subset belong to the same class as much as possible.



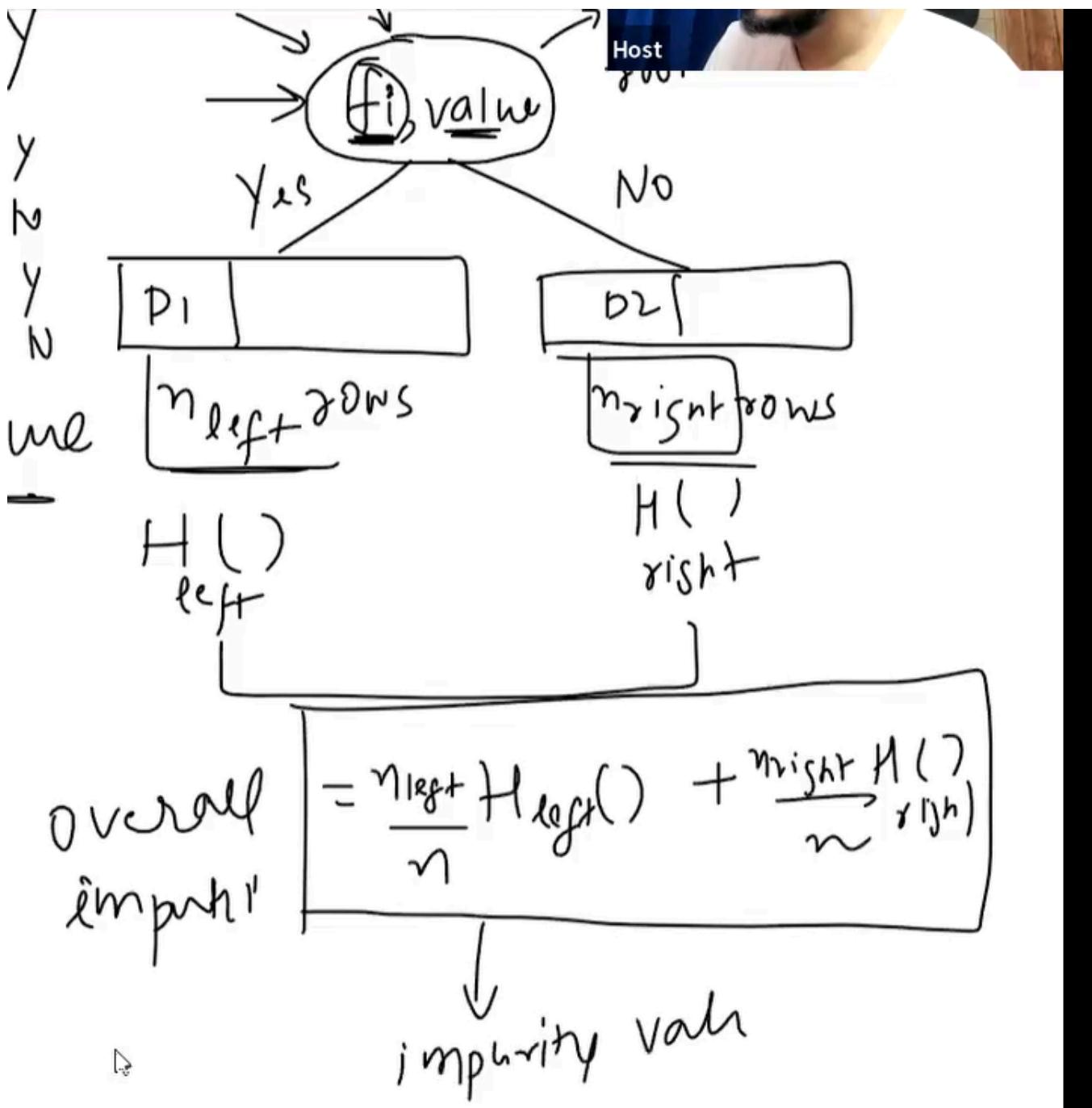
Impurity functions are used to measure the quality of a split in decision trees. They quantify how well a feature separates the classes in the dataset. The goal is to find the feature that results in the most homogeneous subsets after the split. It is the deciding factor for selecting the best feature to split the data at each node of the decision tree.



Calculate $h()$ left and $h()$ right for each split and then calculate the weighted impurity of the split

called overall impurity and is represented using (G).

so calculate G for each column and select the column which gives minimum G value because minimum G value means maximum reduction in impurity.



Impurity Measures:

- **Gini Impurity** : It measures the probability of misclassifying a randomly chosen element from the dataset. It is calculated as: $\text{Gini} = 1 - \sum(p_i)^2$ where p_i is the proportion of

instances belonging to class i. It ranges from $1 < \text{gini} \leq 0$. It is computationally efficient and works well in practice. It is used in CART algorithm. It is less sensitive to class imbalance compared to entropy. It is faster to compute than entropy because it does not involve logarithmic calculations. It tends to create balanced trees, which can lead to better generalization. It is less affected by outliers compared to entropy.

So for multi-class, the maximum impurity is **not always 0.5**; it increases with K and approaches 1 as $K \rightarrow \infty$.

Medium +1

1. Gini Impurity

Gini Impurity checks how often a randomly selected sample would be mislabeled if assigned by class probability. It is computationally simple and used in tree-based classifiers.

Formula:

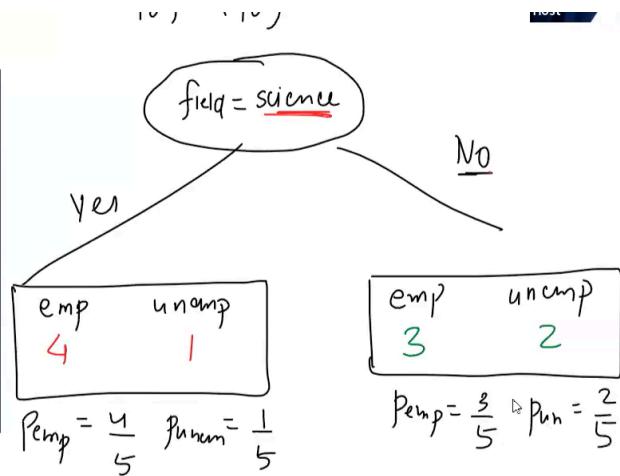
$$\text{Gini} = 1 - \sum_{i=1}^n p_i^2$$

Where p_i is the probability of class i .

Properties:

- Lower values indicate cleaner and more homogeneous nodes.
- Nodes become pure when all samples belong to one class.
- Slightly biased toward dominant classes during split selection.

	Degree_Type	Field	Average_Grade	Job_Outcome
0	Undergraduate	Science	89	Employed 1
1	Undergraduate	Arts	92	Unemployed
2	Postgraduate	Science	95	Employed 2
3	PhD	Science	85	Employed 3
4	Postgraduate	Arts	98	Unemployed
5	PhD	Arts	90	Employed
6	Undergraduate	Science	88	Unemployed 1
7	Postgraduate	Arts	93	Employed
8	Undergraduate	Arts	94	Unemployed
9	PhD	Science	86	Employed 4



$$\begin{array}{ccc}
 & \downarrow & \\
 & 5 & \\
 \downarrow & & \\
 \text{gini} & & \\
 1 - \left(\frac{4}{5}\right)^2 - \left(\frac{1}{5}\right)^2 & & \\
 1 - \frac{16}{25} - \frac{1}{25} = \frac{8}{25} & & \\
 & & \\
 & \downarrow & \\
 & \text{gini} & \\
 1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2 & & \\
 = 1 - \frac{9}{25} - \frac{4}{25} = \frac{12}{25} & & \\
 & & \rightarrow
 \end{array}$$

$$\begin{array}{c}
 \boxed{\frac{5}{10} \times \frac{8}{25} + \frac{5}{10} \times \frac{12}{25}} \\
 0.16 + 0.24 \longrightarrow \boxed{0.40} \quad g() \text{ value} \\
 g \rightarrow (\text{field, sciency})
 \end{array}$$

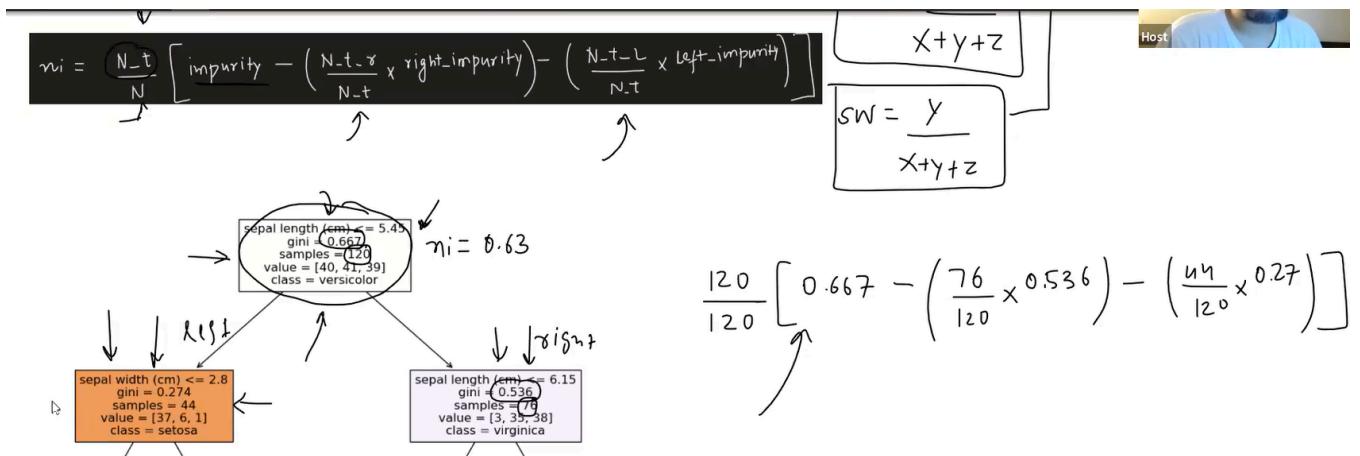
For numerical features, we need to find the best threshold to split the data. We can do this by sorting the unique values of the feature and then calculating the impurity for each possible split point (midpoint between two consecutive unique values). The split point that results in the lowest impurity is selected as the best threshold.

Feature importance using gini:

steps:

1. Calculate the Gini impurity of the parent node (before the split).
2. For each feature, iterate through all possible split points:
 - a. Split the data into left and right subsets based on the split point.
 - b. Calculate the Gini impurity for the left and right subsets.
 - c. Calculate the weighted Gini impurity of the split using the formula: Weighted Gini = $(n_{\text{left}} / n_{\text{total}}) * \text{Gini}_{\text{left}} + (n_{\text{right}} / n_{\text{total}}) * \text{Gini}_{\text{right}}$ where n_{left} and n_{right} are the number of instances in the left and right subsets, respectively, and n_{total} is the total number of instances.
 - d. Calculate the Gini gain for the split using the formula: Gini Gain = $\text{Gini}_{\text{parent}} - \text{Weighted Gini}$

3. Select the split point that results in the highest Gini gain as the best threshold for that feature.
4. Repeat steps 2-3 for all features and select the feature with the highest Gini gain as the best feature to split the data at the current node.
5. The Gini gain for the selected feature can be used as a measure of feature importance. Features with higher Gini gain are considered more important for making predictions.
6. Repeat the process recursively for each child node until a stopping criterion is met (e.g., maximum depth, minimum samples per leaf, or pure leaf nodes).
7. The final decision tree can be used for making predictions on new data by traversing the tree based on the feature values of the input instance.
8. The feature importance can be calculated by summing the Gini gains for each feature across all nodes in the tree where that feature is used for splitting. The importance scores can then be normalized to obtain relative importance values.
9. The resulting feature importance scores can be used to identify the most relevant features for the prediction task and can also be used for feature selection or dimensionality reduction.



$$f_{ik} = \frac{\sum_{j \in \text{node split on feature } k} n_i}{\sum_{j \in \text{all nodes}} n_i}$$

CART FOR REGRESSION:

In regression tasks, CART uses mean squared error (MSE) as the splitting criterion. The goal is to minimize the MSE in the resulting subsets after the split. The MSE is calculated as the average of the squared differences between the actual values and the predicted values (mean of the subset).

Instead of fitting one global equation like Linear Regression, CART does this: "Let me ask a sequence of simple questions that split houses into groups where prices are similar."

While in case of classification majority is considered as the predicted value for a leaf node, in regression mean of all the datapoints in region is considered as the predicted value for a leaf node.

Steps to calculate MSE for regression:

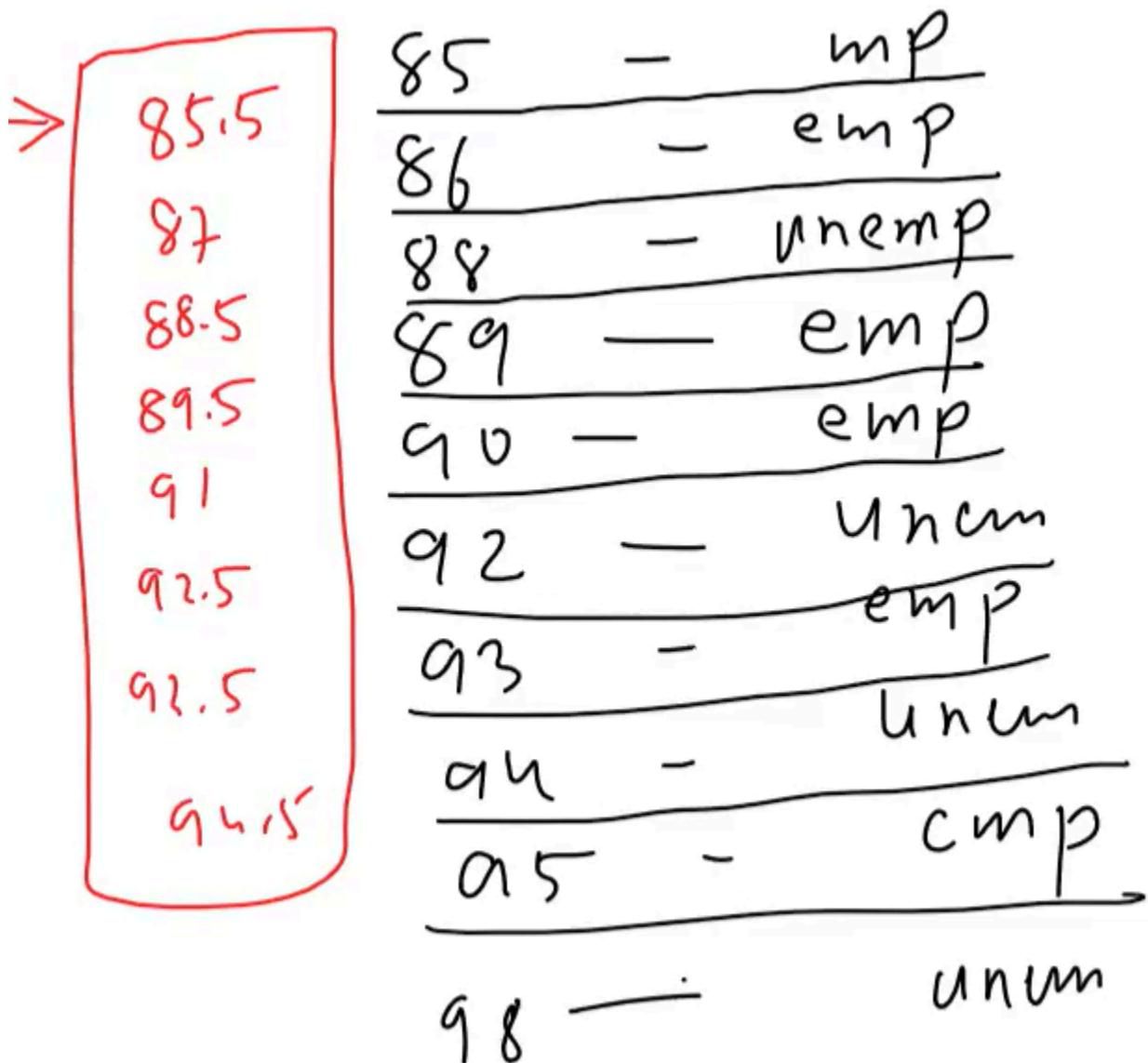
1. For each possible split point in the feature, divide the data into two subsets: left and right.
2. Calculate the mean of the target variable for each subset.
3. Calculate the MSE for each subset using the formula: $MSE = (1/n) * \sum(y_i - \text{mean})^2$
where n is the number of instances in the subset, y_i is the actual value, and mean is the mean of the target variable for that subset.
4. Calculate the weighted MSE for the split using the formula: $\text{Weighted MSE} = (n_{\text{left}} / n_{\text{total}}) MSE_{\text{left}} + (n_{\text{right}} / n_{\text{total}}) MSE_{\text{right}}$ where n_{left} and n_{right} are the number of instances in the left and right subsets, respectively, and n_{total} is the total number of instances.
5. Select the split point that results in the lowest weighted MSE as the best threshold.

CART for Regression

CART is used for regression tasks where the output variable is numerical.

How it Works

- Splits the data to minimize residual error between actual and predicted values.
- Uses Mean Squared Error (MSE) or Residual Sum of Squares (RSS) as the splitting criterion.
- Each leaf node stores the mean of the target values within that node, which becomes the prediction for new samples.



- **Entropy** : It measures the amount of uncertainty or disorder in the dataset. It is calculated as: Entropy = $-\sum(p_i \cdot \log_2(p_i))$ where p_i is the proportion of instances belonging to

class i.

2. Entropy

Entropy measures uncertainty in a node's class distribution and originates from information theory. Higher entropy indicates greater disorder among class labels.

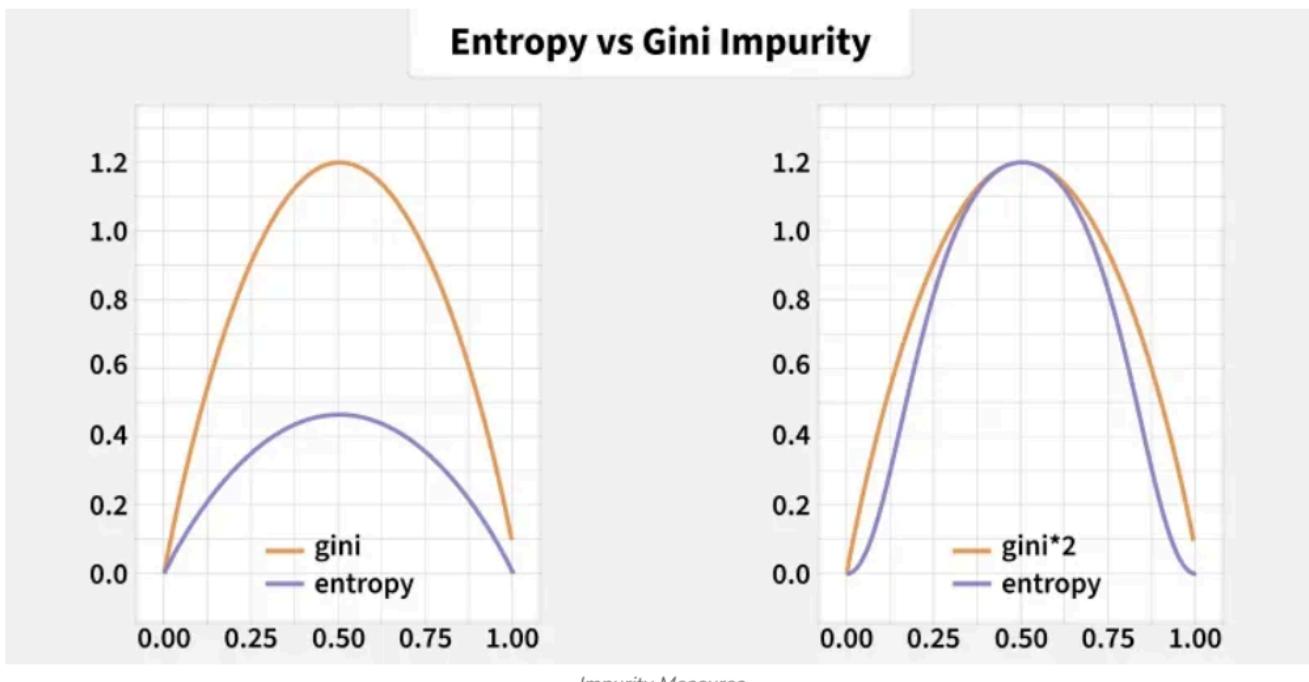
Formula:

$$\text{Entropy} = - \sum_{i=1}^n p_i \log_2(p_i)$$

Where p_i represents the proportion of class i in the node.

Properties:

- Zero entropy corresponds to perfectly pure splits.
- Sensitive to small fluctuations across class ratios.
- Often yields balanced splits with meaningful boundaries.



Splitting Criteria:

- **Information Gain** : It measures the reduction in entropy or impurity after splitting the dataset based on a feature. It is calculated as: $\text{Information Gain} = \text{Entropy}(\text{parent}) - \sum(|\text{child}| / |\text{parent}|) * \text{Entropy}(\text{child})$ where $|\text{child}|$ is the number of instances in the child node and $|\text{parent}|$ is the number of instances in the parent node.
- **Gini Gain** : It measures the reduction in Gini impurity after splitting the dataset based on a feature. It is calculated as: $\text{Gini Gain} = \text{Gini}(\text{parent}) - \sum(|\text{child}| / |\text{parent}|) * \text{Gini}(\text{child})$

where $|\text{child}|$ is the number of instances in the child node and $|\text{parent}|$ is the number of instances in the parent node.

- **Gain Ratio :** It is a modified version of information gain that takes into account the intrinsic information of a feature. It is calculated as: Gain Ratio = Information Gain / Split Information where Split Information = $-\sum\left(\frac{|\text{child}|}{|\text{parent}|} * \log_2\left(\frac{|\text{child}|}{|\text{parent}|}\right)\right)$

Advantages

- Simple to understand and to interpret. Trees can be visualized.
- Requires little data preparation. Other techniques often require data normalization, dummy variables need to be created and blank values to be removed. Note however that this module does not support missing values.
- The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree.
- Able to handle both numerical and categorical data.
- Can work on non-linear datasets
- |
- Can give you feature importance.

Disadvantages

- Decision-tree learners can create over-complex trees that do not generalize the data well. This is called overfitting. Mechanisms such as pruning, setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid this problem.
- Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This problem is mitigated by using decision trees within an ensemble.
- Predictions of decision trees are neither smooth nor continuous, but piecewise constant approximations as seen in the above figure. Therefore, they are not good at extrapolation.
This limitation is inherent to the structure of decision tree models. They are very useful for interpretability and for handling non-linear relationships within the range of the training data, but they aren't designed for extrapolation. If extrapolation is important for your task, you might need to consider other types of models.

One of the major problems with decision trees is overfitting. Overfitting occurs when the model

learns the training data too well, including noise and outliers, resulting in poor generalization to new data. To prevent overfitting, we can use techniques such as:

- Pruning : It involves removing branches from the decision tree that do not provide significant predictive power. This can be done by setting a maximum depth for the tree, minimum samples required to split a node, or minimum samples required at a leaf node.
 - There are two types of pruning:
 - Pre-pruning : It involves stopping the growth of the tree before it reaches its maximum depth. This can be done by setting a maximum depth for the tree, minimum samples required to split a node, or minimum samples required at a leaf node.
 - Post-pruning : It involves growing the full tree and then removing branches that do not provide significant predictive power. This can be done by evaluating the performance of the tree on a validation set and removing branches that do not improve the performance.

Pruning is a technique used in machine learning to reduce the size of decision trees and to avoid overfitting. Overfitting happens when a model learns the training data too well, including its noise and outliers, which results in poor performance on unseen or test data.

Decision trees are susceptible to overfitting because they can potentially create very complex trees that perfectly classify the training data but fail to generalize to new data. Pruning helps to solve this issue by reducing the complexity of the decision tree, thereby improving its predictive power on unseen data.

There are two main types of pruning: pre-pruning and post-pruning.

1. Pre-pruning (Early stopping): This method halts the tree construction early. It can be done in various ways: by setting a limit on the maximum depth of the tree, setting a limit on the minimum number of instances that must be in a node to allow a split, or stopping when a split results in the improvement of the model's accuracy below a certain threshold.
2. Post-pruning (Cost Complexity Pruning): This method allows the tree to grow to its full size, then prunes it. Nodes are removed from the tree based on the error complexity trade-off. The basic idea is to replace a whole subtree by a leaf node, and assign the most common class in that subtree to the leaf node.

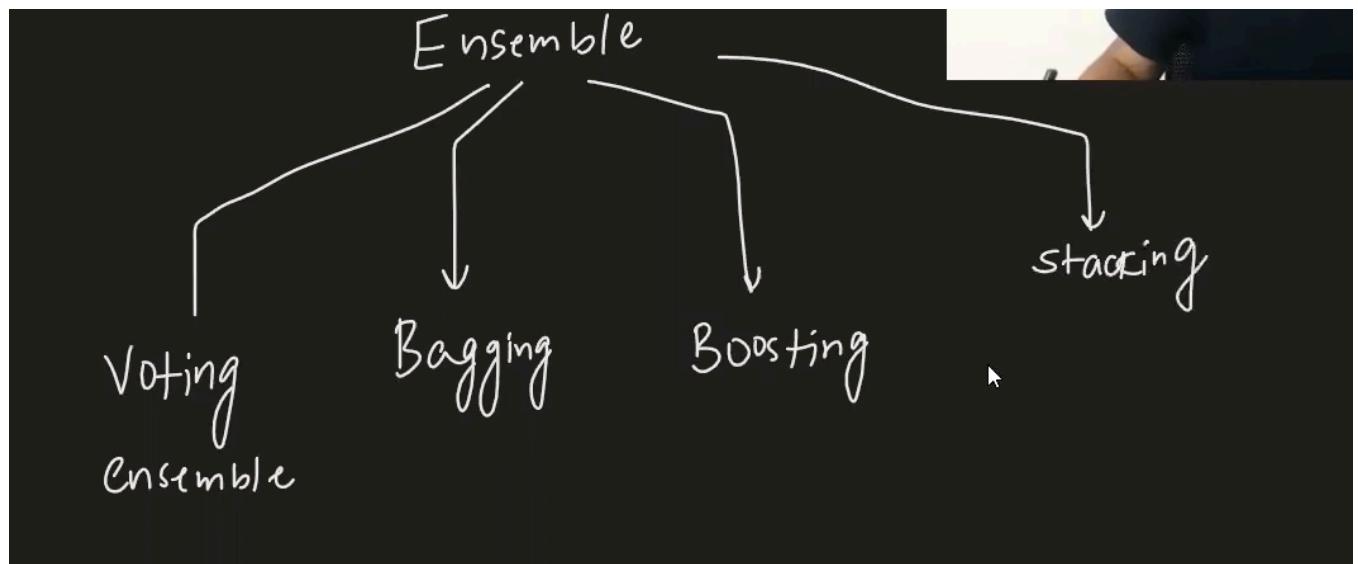
- Setting minimum samples per leaf : It ensures that each leaf node has a minimum number of samples, preventing the tree from creating overly specific rules that may not generalize well.

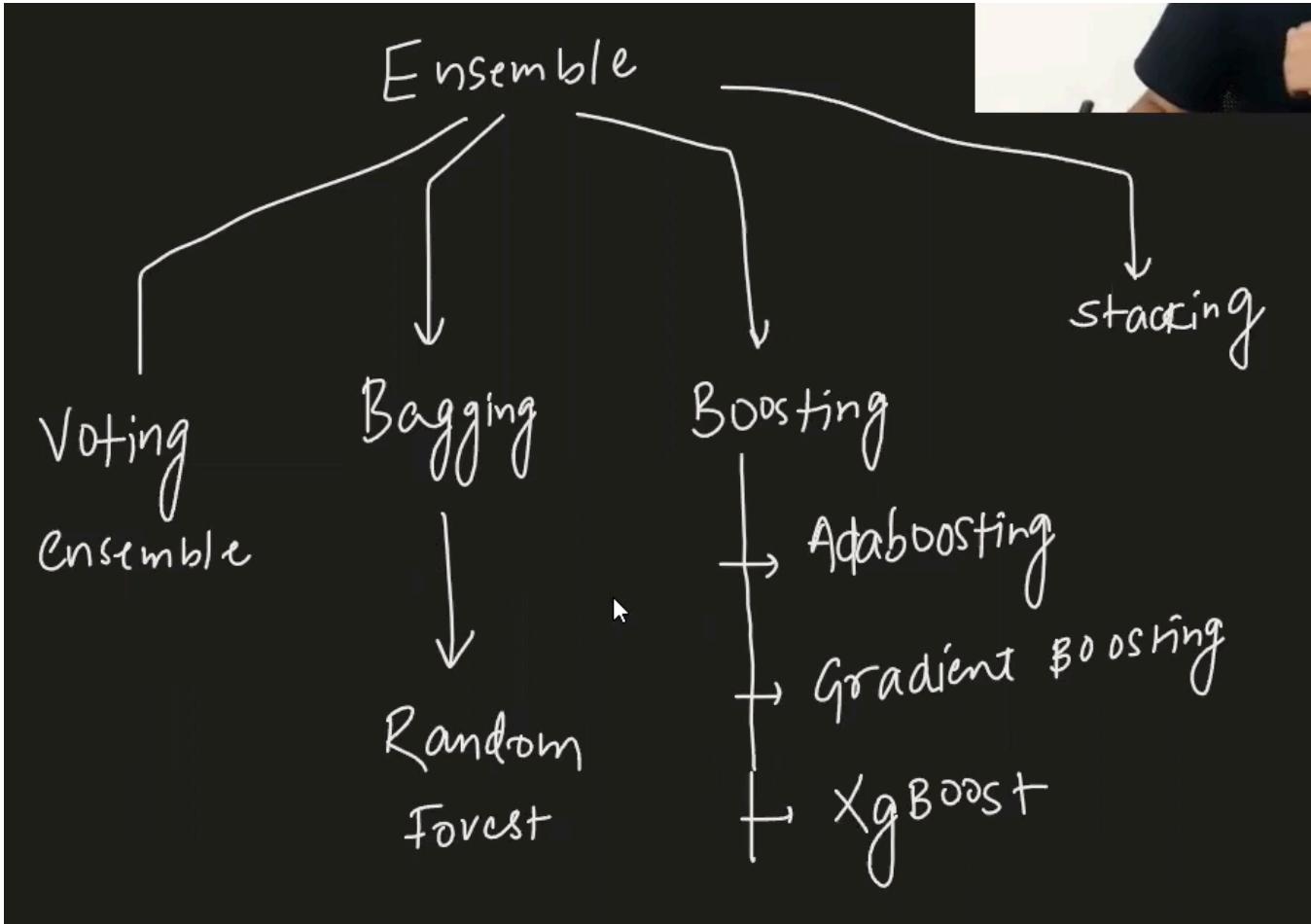
- Setting maximum depth : It limits the depth of the tree, preventing it from growing too deep and capturing noise in the data.
- Setting minimum samples per split : It ensures that a node must have a minimum number of samples before it can be split, preventing the tree from creating splits based on very few samples.

And also other biggest disadvantage of decision tree is extrapolation. Decision trees cannot extrapolate beyond the range of the training data. For example, if a decision tree is trained on house prices ranging from 100,000 to 500,000, it cannot accurately predict the price of a house worth \$600,000. This is because decision trees make predictions based on the splits created during training, and they do not have a mechanism or equation like linear regression to infer values outside the observed range. So when the input range is restricted then only we can go with decision tree

Ensemble Methods

Ensemble methods combine multiple decision trees to improve the accuracy and robustness of predictions. Some popular ensemble methods include:

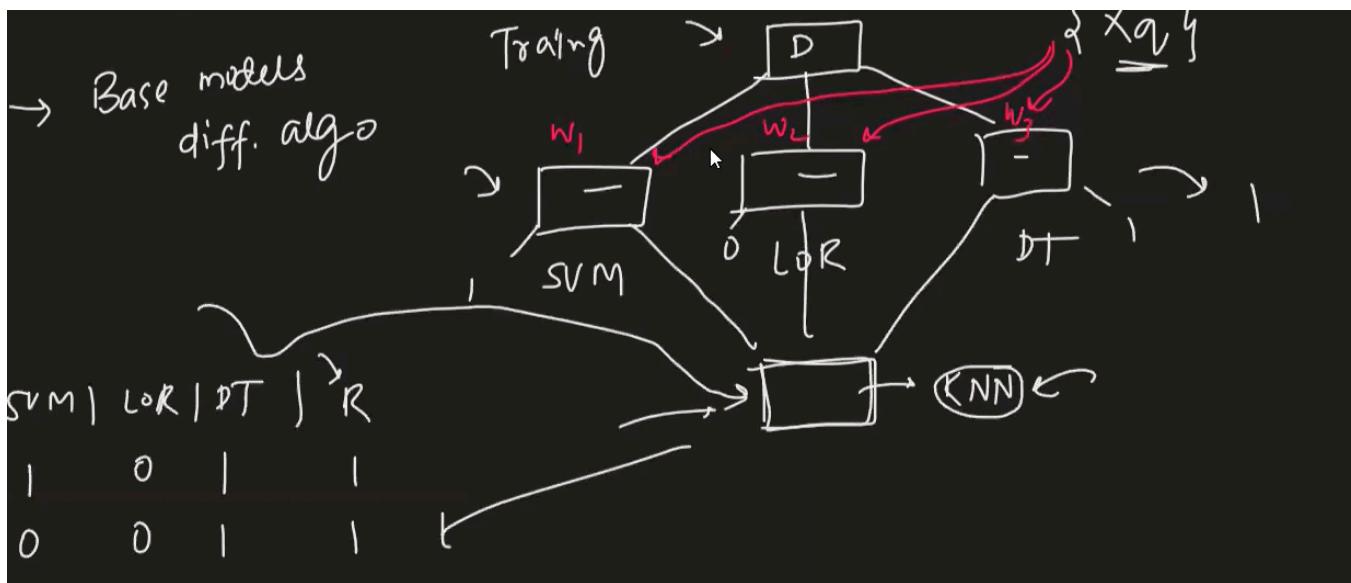




- Random Forest : It is an ensemble learning method that combines multiple decision trees to improve the accuracy and robustness of predictions. Each tree is trained on a random subset of the data and a random subset of features. The final prediction is made by aggregating the predictions of all the trees (majority voting for classification and averaging for regression).
- Gradient Boosting Machines (GBM) : It is another ensemble learning method that builds decision trees sequentially, where each tree tries to correct the errors of the previous tree. The final prediction is made by combining the predictions of all the trees, weighted by their performance.
- Extreme Gradient Boosting (XGBoost) : It is an optimized implementation of gradient boosting that is designed to be highly efficient and scalable. It includes several advanced features, such as regularization, parallel processing, and handling missing values, which make it a popular choice for machine learning competitions and real-world applications.
- LightGBM : It is a gradient boosting framework that uses tree-based learning algorithms. It is designed to be efficient and scalable, with a focus on speed and memory usage. LightGBM uses a histogram-based approach to split the data, which allows it to handle

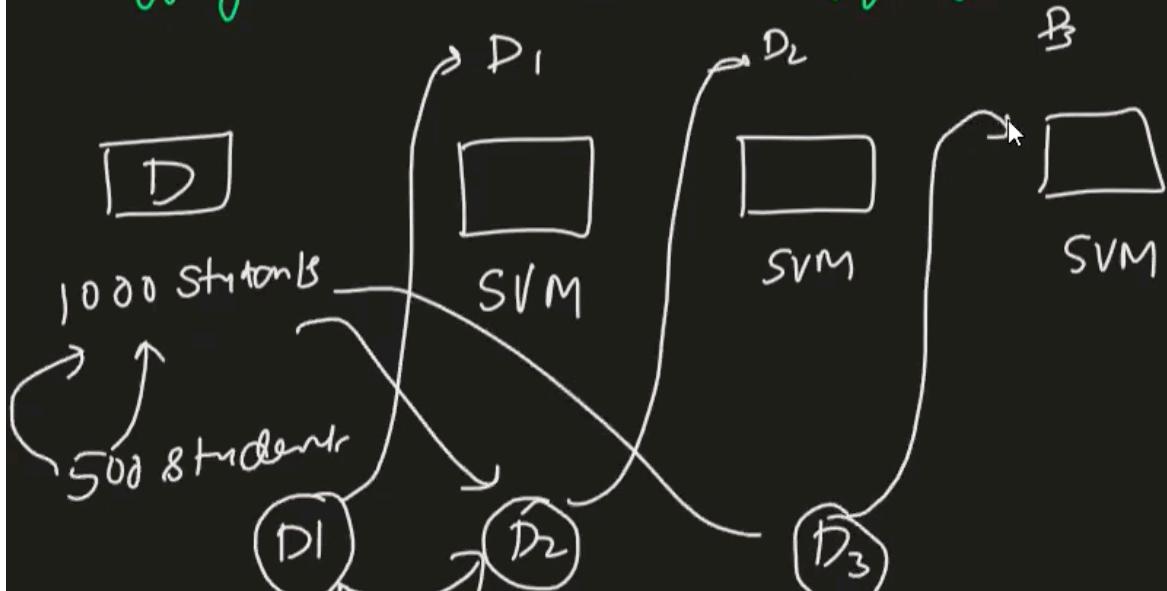
large datasets with high dimensionality.

- CatBoost : It is a gradient boosting library that is designed to handle categorical features automatically. It uses a combination of decision trees and gradient boosting to make predictions, and includes several advanced features, such as handling missing values and reducing overfitting.
- AdaBoost : It is an ensemble learning method that combines multiple weak classifiers to create a strong classifier. It works by iteratively training weak classifiers on the data, where each classifier focuses on the misclassified instances from the previous classifier. The final prediction is made by combining the predictions of all the classifiers, weighted by their performance.
- Stacking : It is an ensemble learning method that combines multiple base models to create a meta-model. The base models are trained on the data, and their predictions are used as input features for the meta-model. The meta-model is then trained to make the final prediction based on the predictions of the base models.

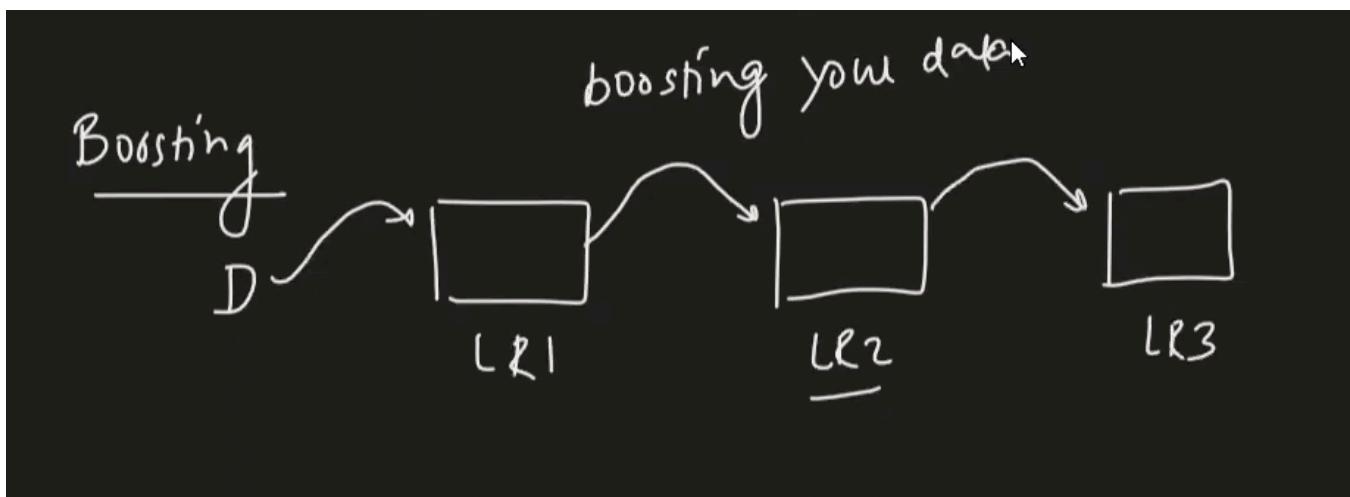


- Voting Classifier : It is an ensemble learning method that combines multiple classifiers to make a final prediction. Each classifier makes a prediction, and the final prediction is made by majority voting (for classification) or averaging (for regression).
- Bagging (Bootstrap Aggregating) : It is an ensemble learning method that combines multiple models using same algorithm but it is trained on different subsets of the data. Each model is trained on a random subset of the data (with replacement), and the final prediction is made by aggregating the predictions of all the models (majority voting for classification and averaging for regression).

Bagging → Bootstrapped Aggregation



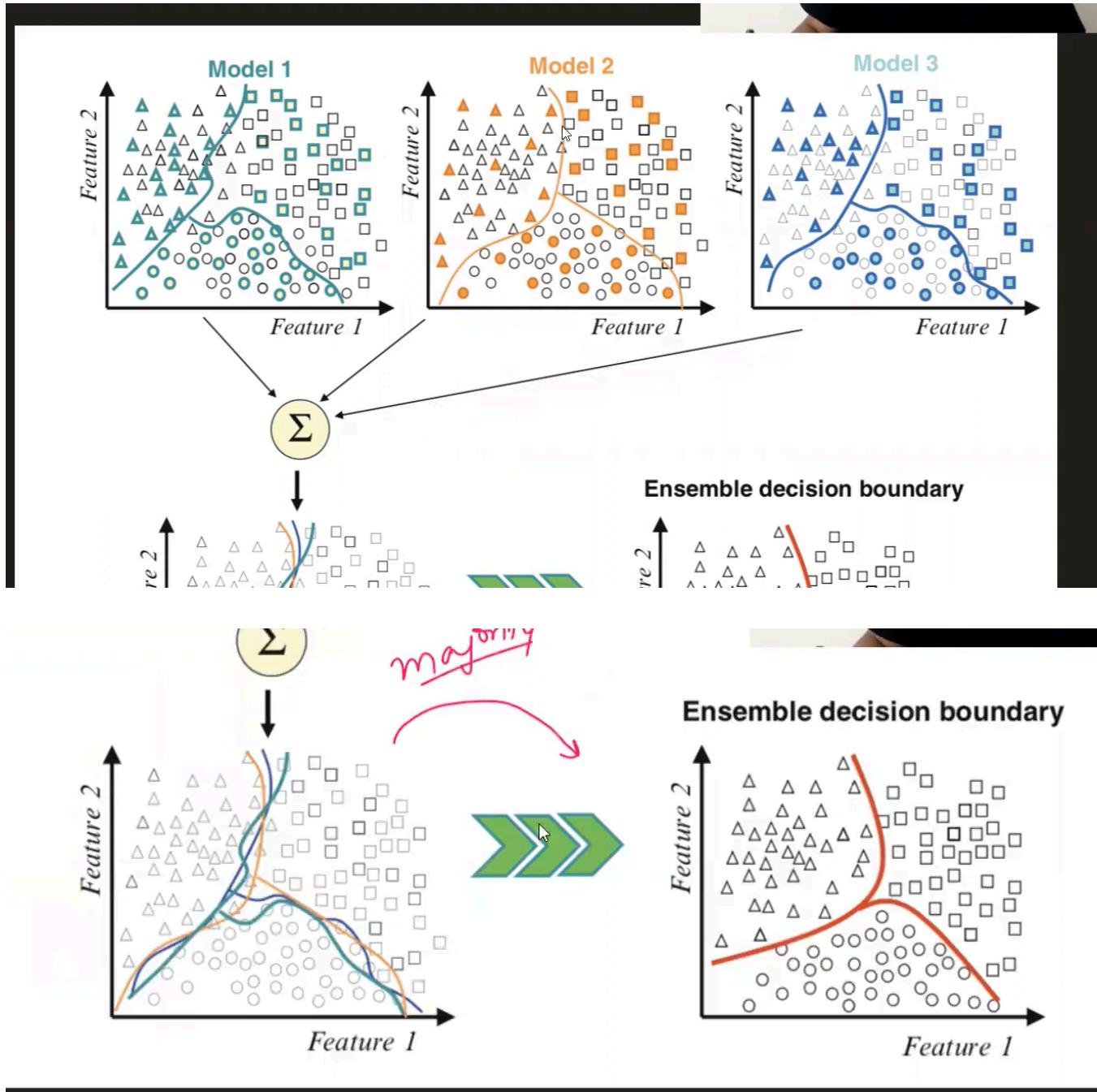
- Boosting : It is an ensemble learning method that combines multiple weak models to create a strong model. Each model is trained sequentially, where each model tries to correct the errors of the previous model. The final prediction is made by combining the predictions of all the models, weighted by their performance.



Why ensemble methods work?

Ensemble methods work because they combine the strengths of multiple models to create a more robust and accurate prediction. By aggregating the predictions of multiple models, ensemble methods can reduce the impact of individual model errors and improve overall

performance. This is particularly effective when the individual models are diverse and make different types of errors.



In machine learning, **bias** and **variance** are two key sources of prediction error that determine how well a model generalizes to unseen data. The **bias-variance tradeoff** is central to building models that avoid both underfitting and overfitting.

Bias is the error from overly simplistic assumptions in the model.

- **High bias:** Model underfits, missing important patterns (e.g., linear regression on non-linear data).
- **Low bias:** Model captures patterns well but may risk overfitting.
- **Reduction methods:** Use more complex models, add relevant features, or reduce regularization strength .

Variance is the error from excessive sensitivity to training data.

- **High variance:** Model overfits, capturing noise (e.g., deep decision trees).
- **Low variance:** Model is stable but may miss patterns.
- **Reduction methods:** Simplify the model, add more training data, apply regularization, or use ensemble methods .

For e.g in decision tree we mostly face the problem of low bias and high variance. By using ensemble methods like random forest and boosting we can reduce the variance and improve the accuracy of the model.

Similarly in case of linear regression we mostly face the problem of high bias and low variance. By using ensemble methods like bagging we can reduce the bias and improve the accuracy of the model.

New Topics :

- Out of core learning : Out-of-core learning is a machine learning approach designed to handle datasets that are too large to fit into a computer's main memory (RAM). Instead of loading the entire dataset at once, the algorithm processes the data in small chunks. E.g Mini batch processing, Stochastic gradient descent.

There are mainly 2 types of ML Models

Parametric Models : In parametric models, the model structure is defined by a fixed number of parameters. Once these parameters are learned from the training data, the model can make

predictions without needing to refer back to the entire dataset. Examples of parametric models include linear regression, logistic regression, and neural networks.

- Parametric models make strong assumptions about the data, such as assuming a specific functional form (e.g., linearity) or data distribution (e.g., Gaussian, linear). This can lead to underfitting if the assumptions do not hold true for the data.
- Fixed number of parameters regardless of the size of the training data. This can lead to faster training and prediction times, especially for large datasets.

Non-Parametric Models : Non-parametric models do not assume a fixed number of parameters. Instead, they can adapt their complexity based on the amount of training data available. These models often require storing the entire dataset or a significant portion of it to make predictions. Examples of non-parametric models include k-nearest neighbors (KNN), decision trees, and kernel density estimation.

What is residual error?

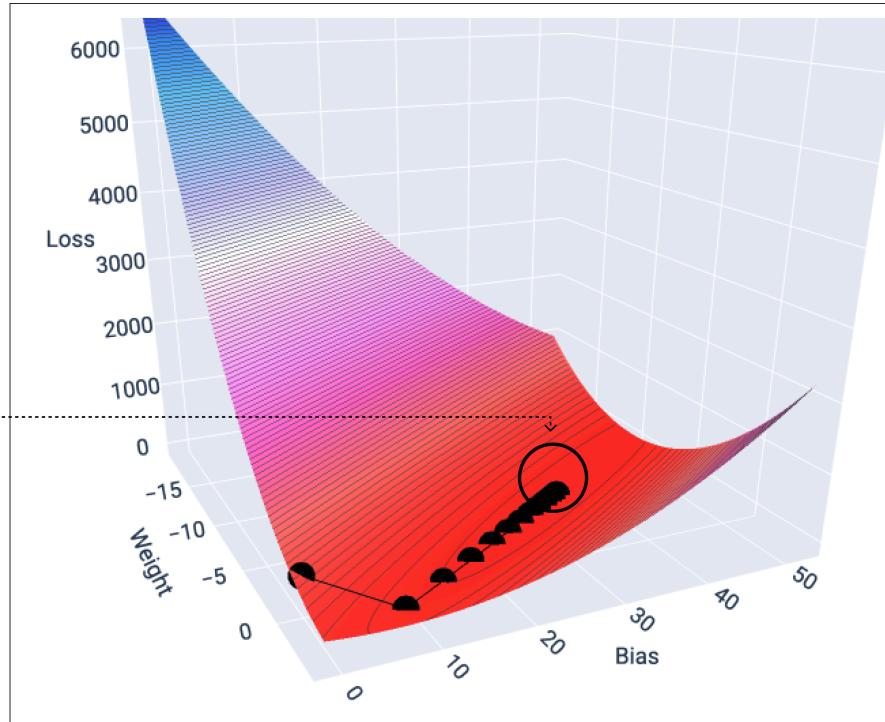
Residual error, also known as residuals or prediction errors, refers to the difference between the observed (actual) values and the predicted values generated by a regression model. In other words, it quantifies how much the model's predictions deviate from the true values in the dataset. It is combination of both reducible and irreducible error.

- Dividing by n → average error (scale-independent)
- Dividing by 2 → cancels a constant during differentiation

So sometimes you see:

$$J = \frac{1}{2n} \sum (y_i - \hat{y}_i)^2$$

Linear regression models converge because the loss surface is convex and contains a point where the weight and bias have a slope that's almost zero.



How to select a good Loss Function

21 April 2023 16:37

1. **Problem type:** The choice of a loss function depends on the type of problem you are solving. For example, in regression tasks, mean squared error (MSE) or mean absolute error (MAE) are commonly used. For binary classification, cross-entropy loss or hinge loss can be employed. For multi-class classification, categorical cross-entropy or multi-class hinge loss can be used. Choose a loss function that aligns with the objectives of the specific problem you are addressing.
2. **Robustness to outliers:** Some loss functions, like mean squared error, are more sensitive to outliers, which can lead to a model that is overly influenced by extreme values. If your dataset contains outliers or is prone to noise, consider using a loss function that is more robust to outliers, such as mean absolute error (MAE) or Huber loss.