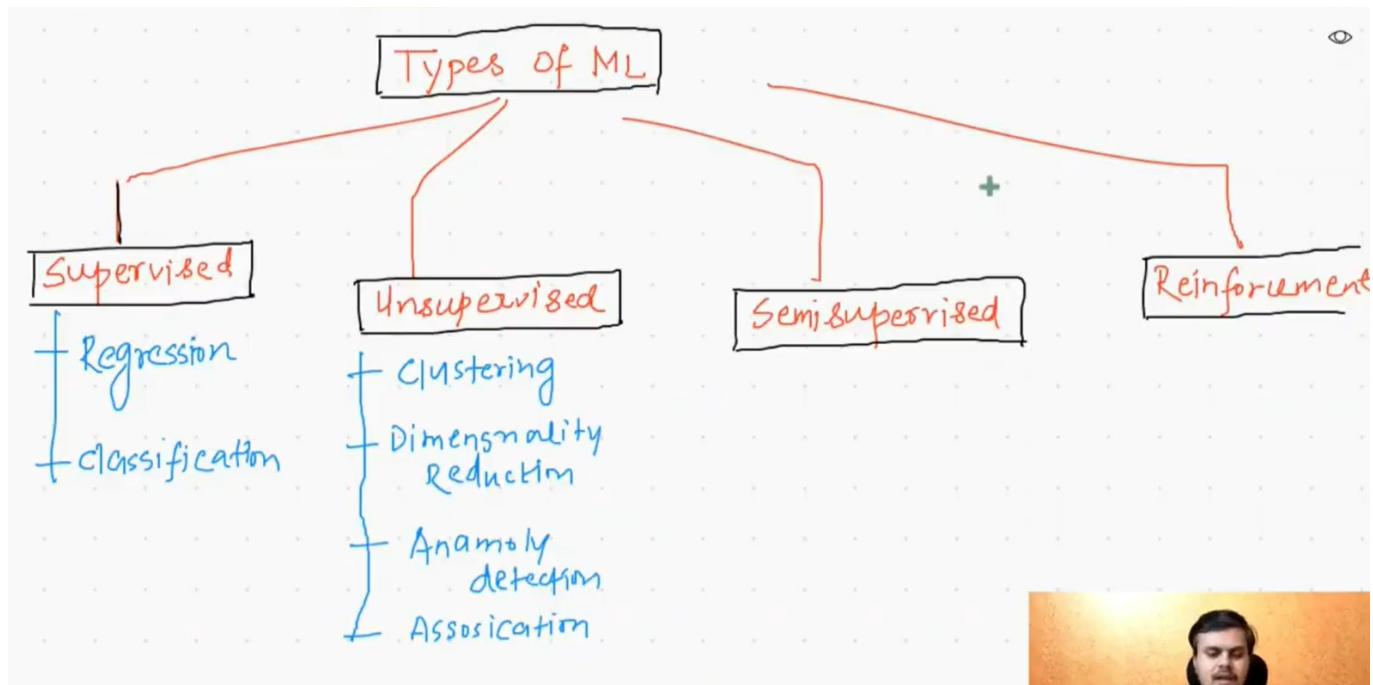


Machine learning is a field of computer science that uses statistical techniques to give computer systems the ability to "learn" with data, without being explicitly programmed.



## 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 data because it is high dimensional data so we can plot it. so we use dimensionality reduction to reduce dimension and then plot it  
E.g : MNIST Dataset

- **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

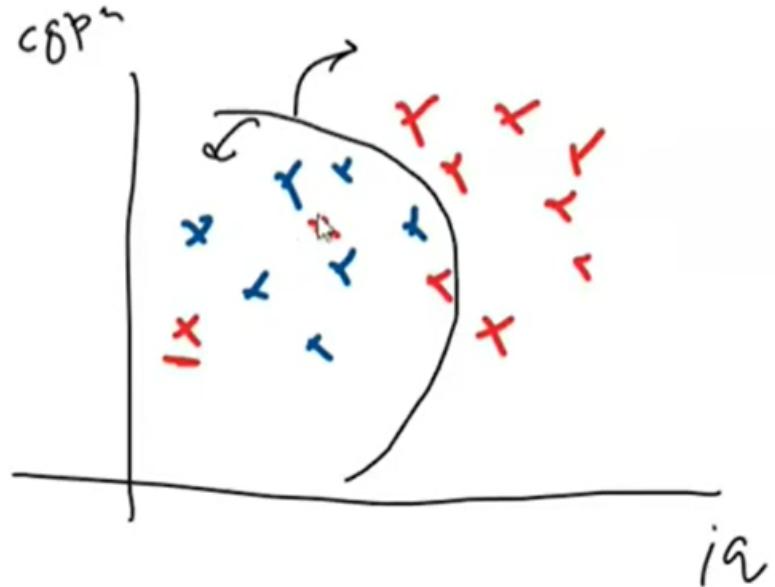
**Instance-based learning (memory-based / lazy learning)** : Learns by remembering instances. E.g. : K nearest neighbors

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

### 3. Model Based

Friday, March 19, 2021 4:06 PM

iq | cgpa | p bnm



## 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

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

- $\beta_0$ : The y-intercept (the value of  $Y$  when  $X = 0$ ).
- $\beta_1$ : The slope (the change in  $Y$  for a one-unit increase in  $X$ ).
- $\epsilon$ : 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 ( $\beta_0$ ) : Expected value of y when  $X = 0$
- $\beta_1$  : 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):

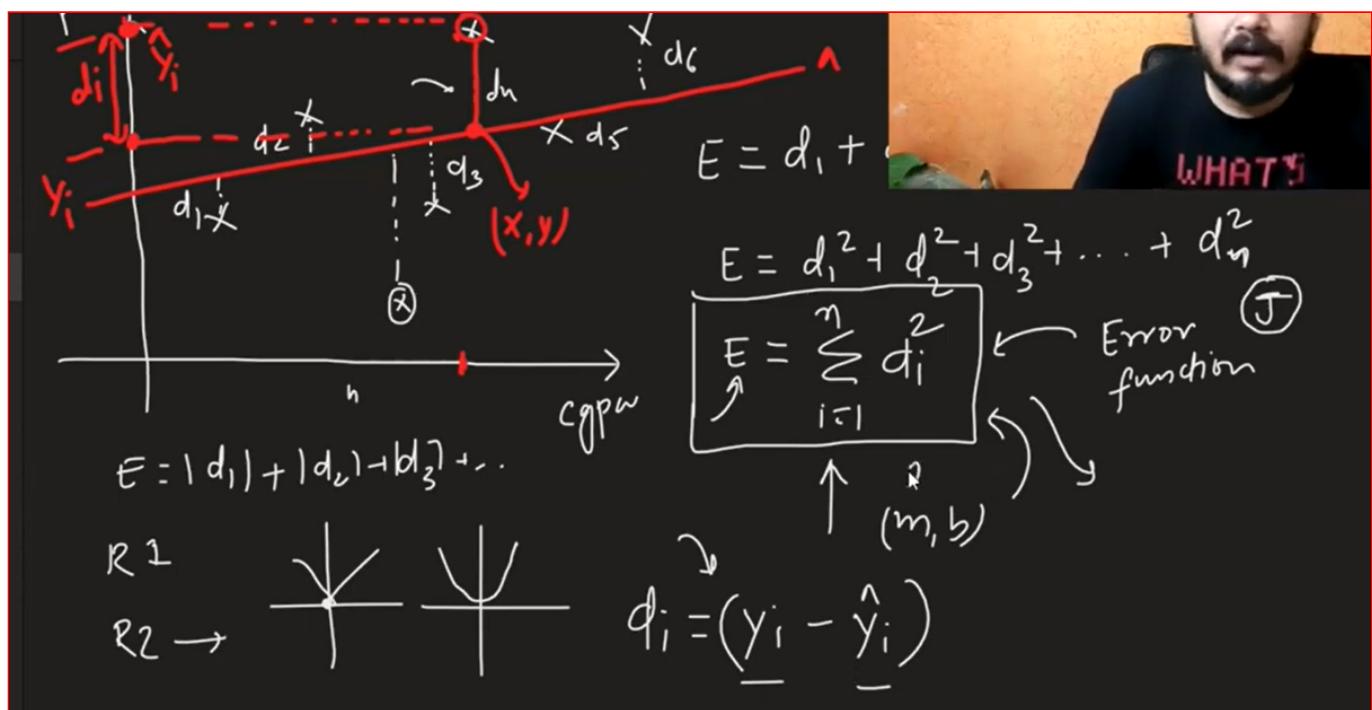
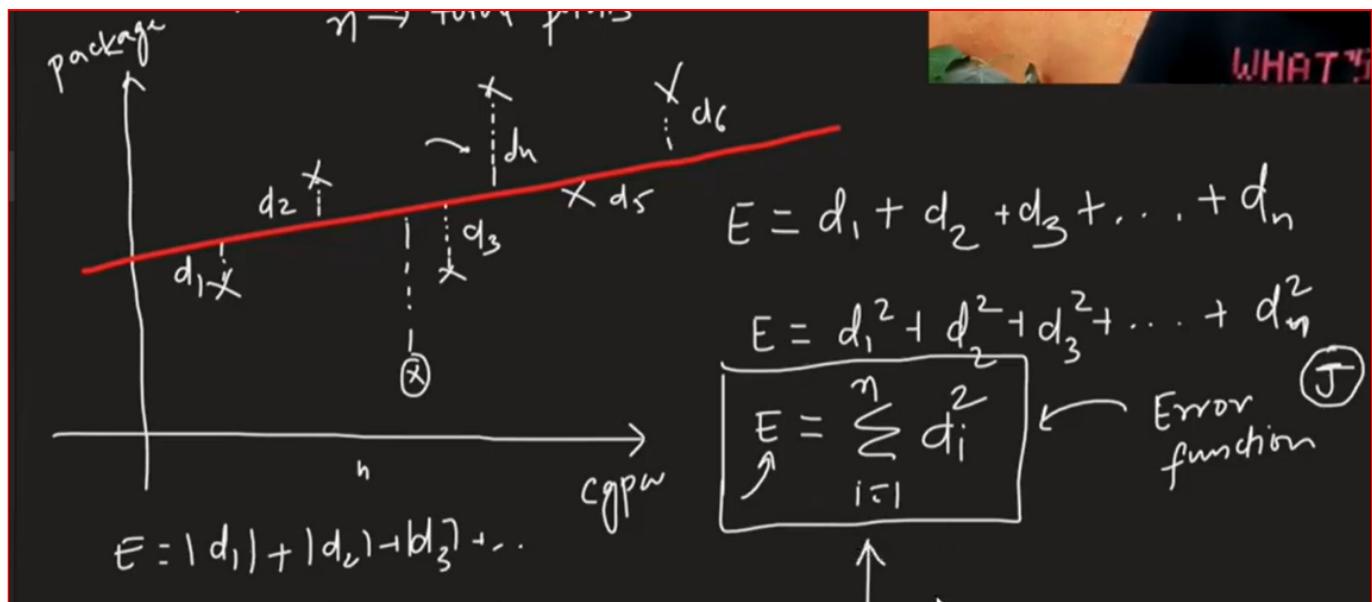
$$\hat{y} = \beta_0 + \beta_1 X$$

The model never predicts  $\epsilon$ .

- 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.

$$\begin{aligned} b &= \bar{y} - m\bar{x} \\ m &= \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2} \end{aligned}$$

$\bar{x}$  → mean



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

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

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

- **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: It is differentiable Disadvantages: Not robust to outliers.

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

Advantages: Robust to outliers. Disadvantages: It is not differentiable at 0.

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

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

$R^2$  compares your model against a dumb model that always predicts  $\text{mean}(y)$ .

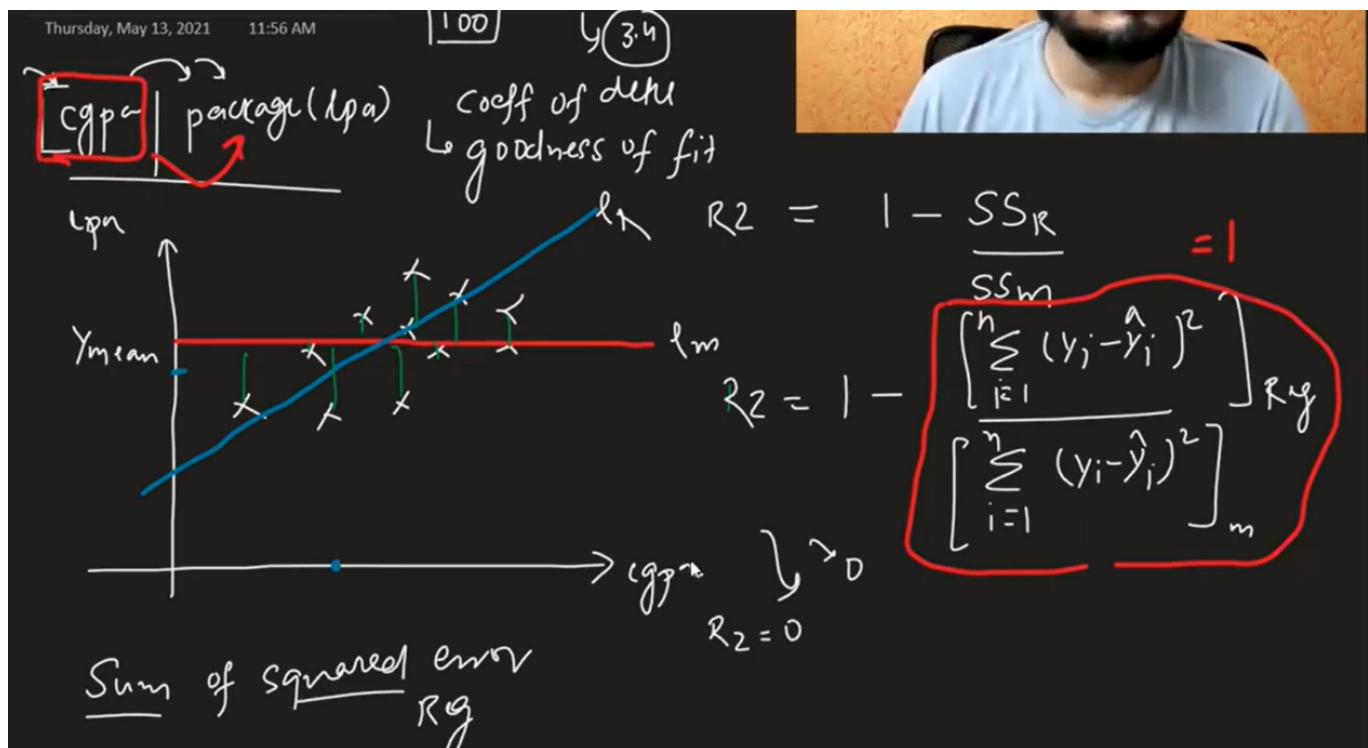
- $R^2 = 1 \rightarrow$  perfect fit (line explains everything)
- $R^2 = 0 \rightarrow$  no better than guessing the average
- Sometimes  $R^2 < 0 \rightarrow$  worse than guessing the average

## 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  $R^2$

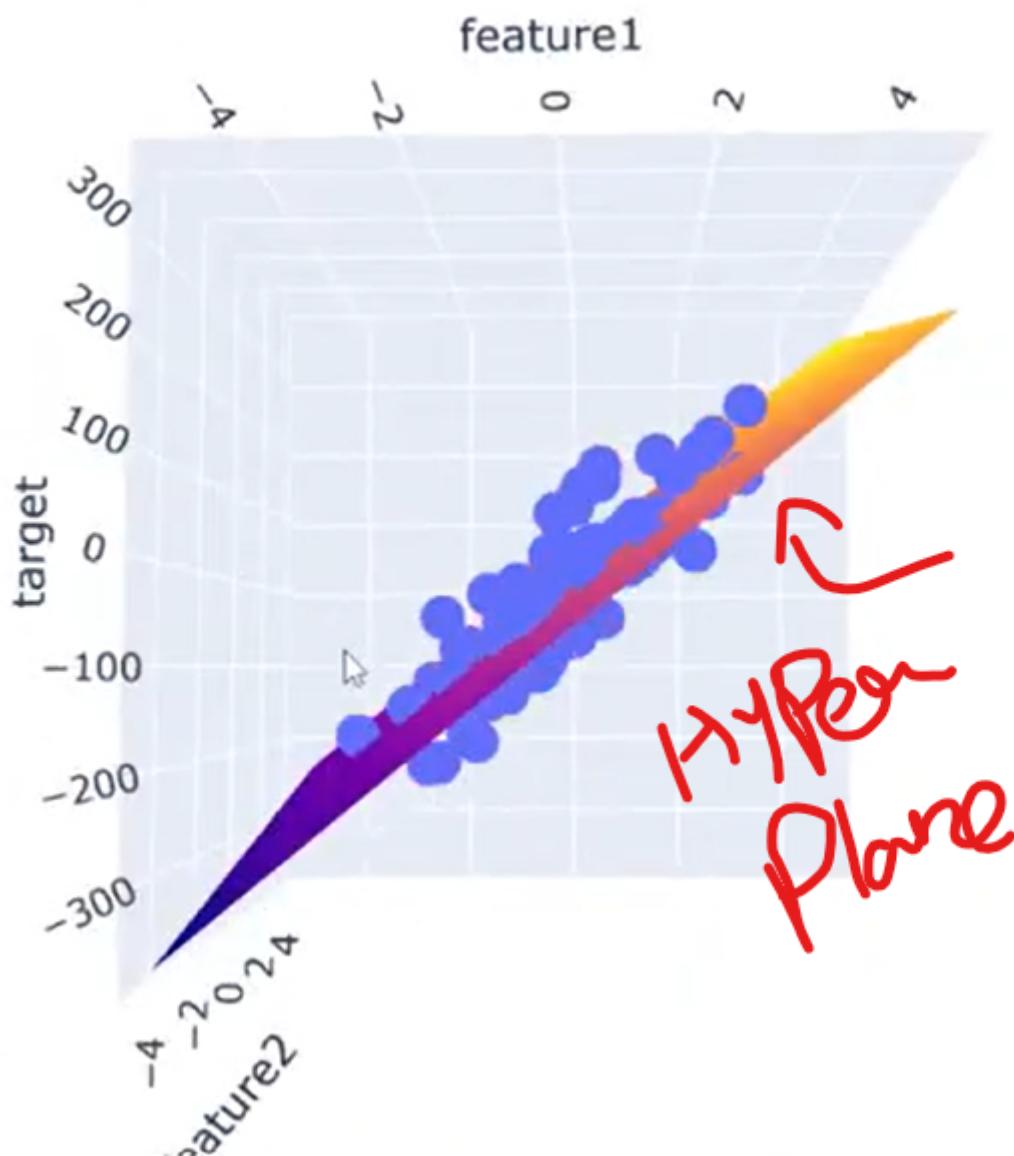
$$R^2_{adj} = 1 - \left[ \frac{(1-R^2)(n-1)}{(n-1-K)} \right]$$

$R^2 \rightarrow$   
 $n \rightarrow$  no. of rows  
 $K =$  independent  
 $K=1, K=2, K=3$

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:

$$\begin{aligned}
 &= \begin{bmatrix} \hat{Y}_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix} = \begin{bmatrix} \beta_0 + \beta_1 x_{11} + \beta_2 x_{12} + \beta_3 x_{13} + \beta_4 x_{14} + \dots + \beta_m x_{1m} \\ \beta_0 + \beta_1 x_{21} + \beta_2 x_{22} + \dots + \dots + \beta_m x_{2m} \\ \beta_0 + \beta_1 x_{31} + \beta_2 x_{32} + \dots + \dots + \beta_m x_{3m} \\ \vdots \\ \beta_0 + \beta_1 x_{n1} + \beta_2 x_{n2} + \dots + \dots + \beta_m x_{nm} \end{bmatrix}
 \end{aligned}$$

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

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

## 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  $\beta$  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 ( $\beta$ ) 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 (( $x_1, x_2$ ))

$$\$ \$ X = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1n} \\ 1 & x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{nn} \end{bmatrix} \$ \$$$


---

### 1.3 Coefficient vector ( $\beta$ ) ( $p \times 1$ )

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

- ( $\beta_0$ ) = intercept
  - ( $\beta_1, \beta_2, \dots, \beta_p$ ) = 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 ( $1 \times n$ )

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

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:

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

Result is:

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

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

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

$$e^T e = \begin{bmatrix} y_1 - \hat{y}_1 & y_2 - \hat{y}_2 & \dots & y_n - \hat{y}_n \end{bmatrix}_{1 \times n} \begin{bmatrix} y_1 - \hat{y}_1 \\ y_2 - \hat{y}_2 \\ \vdots \\ y_n - \hat{y}_n \end{bmatrix}_{n \times 1} =$$

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

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

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:

$$\$ \$ e = y - X\beta \$ \$$$

$$E = e^T e$$

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

↑  
miniz

$$E = e^T e$$

I-1  
II  
↓

same

$$E = (y - \hat{y})^T (y - \hat{y}) = (y^1 - \hat{y}^1)^T (y - \hat{y})$$

↑  
miniz

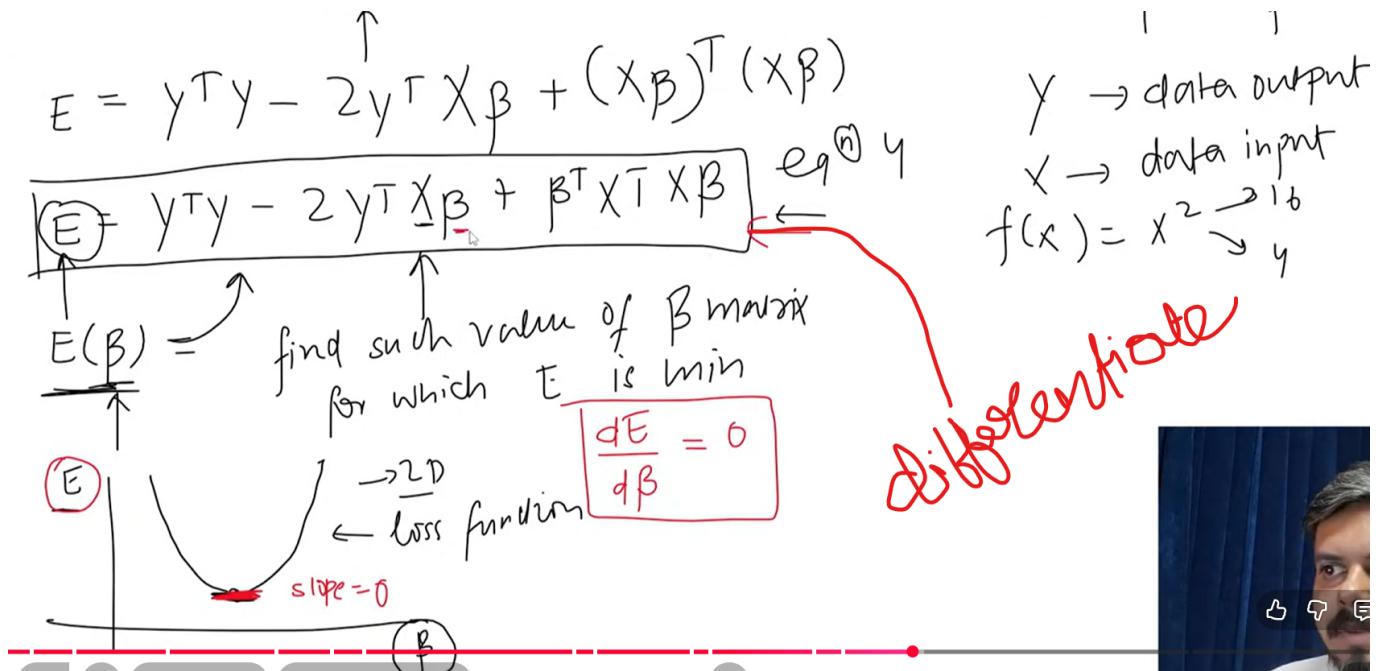
$$E = y^T y - [y^T \hat{y} - \hat{y}^T y] + \hat{y}^T \hat{y}$$

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

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

$$E = y^T y - 2y^T X\beta + (X\beta)^T (X\beta)$$

$$\boxed{E = y^T y - 2y^T X\beta + \beta^T X^T X\beta} \quad \text{eq } ④ \quad y$$



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:

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

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

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

$$\beta = \text{values}^T (X^{T+1})^{-1}$$

logistic

Dimensions:

- $\text{values}^T$ :  $(m+1 \times 1)$
- $(X^T X)^{-1}$ :  $(m+1 \times m+1)$
- $y$ :  $n \times 1$
- $X$ :  $(m+1 \times n)$

$$e^T \text{values} = (y - X\beta)^T (y - X\beta)$$


---

## 6) Expanding the objective

Expanding:

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

gives:

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


---

$$y^T y - 2\beta^T X^T y + \beta^T X^T X \beta$$

## 7) Minimization leads to Normal Equation

We minimize SSE by differentiating with respect to ( $\beta$ ) and setting gradient to 0.

Result:

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

This is the **Normal Equation**.

Solving for ( $\beta$ ):

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

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

---

## 8) Geometric intuition (very important)

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

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

Residual:

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

OLS ensures residual is perpendicular to the feature space:

$$\$ \$ X^T e = 0 \$ \$$$

Substitute ( $e=y-X\beta$ ):

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

Which becomes:

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

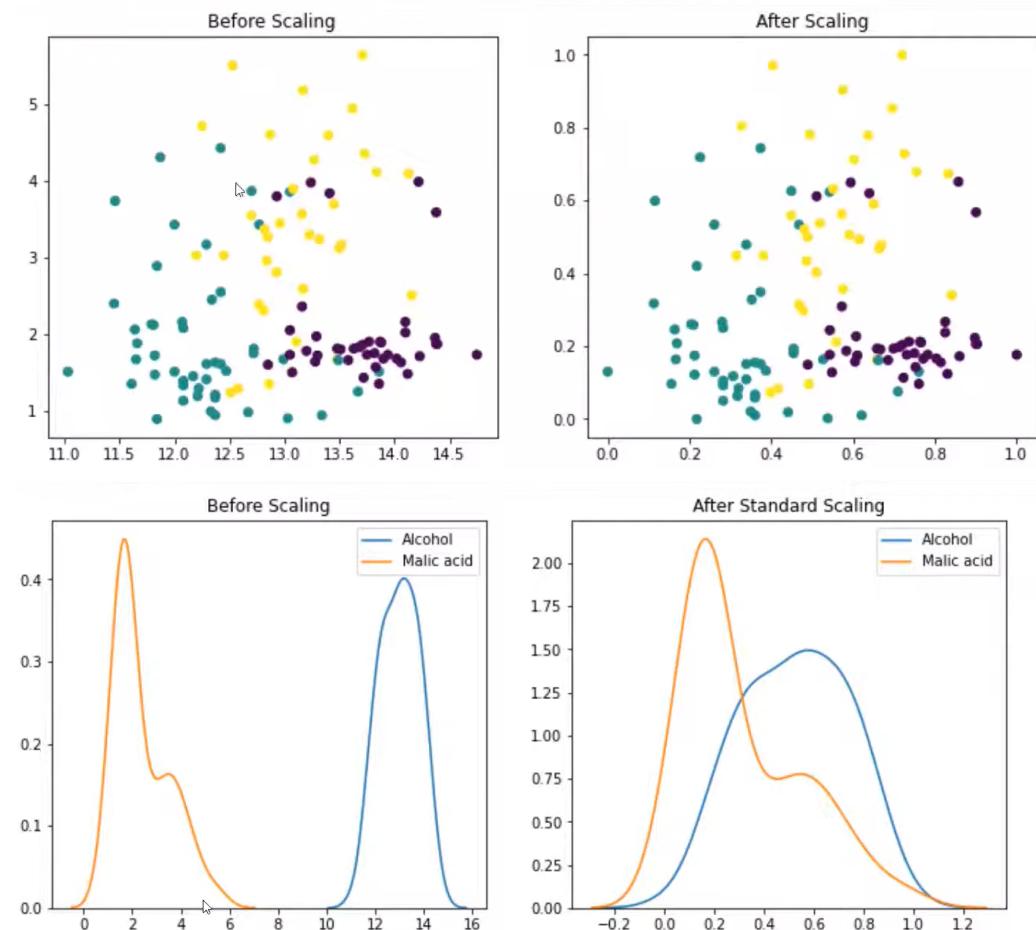
So the normal equation is actually a **perpendicularity condition**:

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.

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

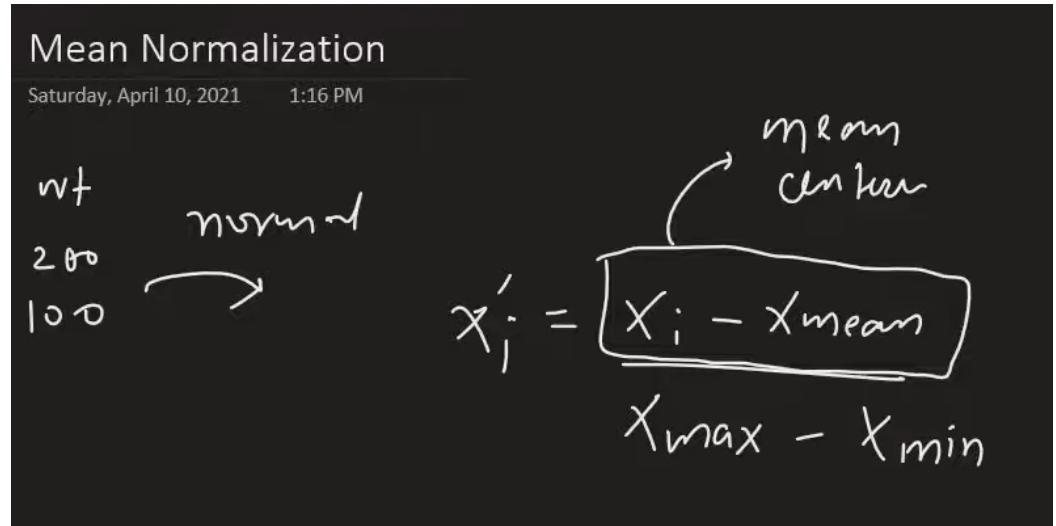


- As you can see that both these distribution first were not overlapping because of different scale but after min max scaling both are overlapping because both are in same scale now.
- Max Abs Scaling** : It scales the data by dividing each value by the maximum absolute value of that feature. The resulting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

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

- 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.



- 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.

Robust Standardised Value      Original Value      Sample Median

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

$\text{Interquartile Range} = Q3 - Q1$

### o 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

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

are not based on distance.

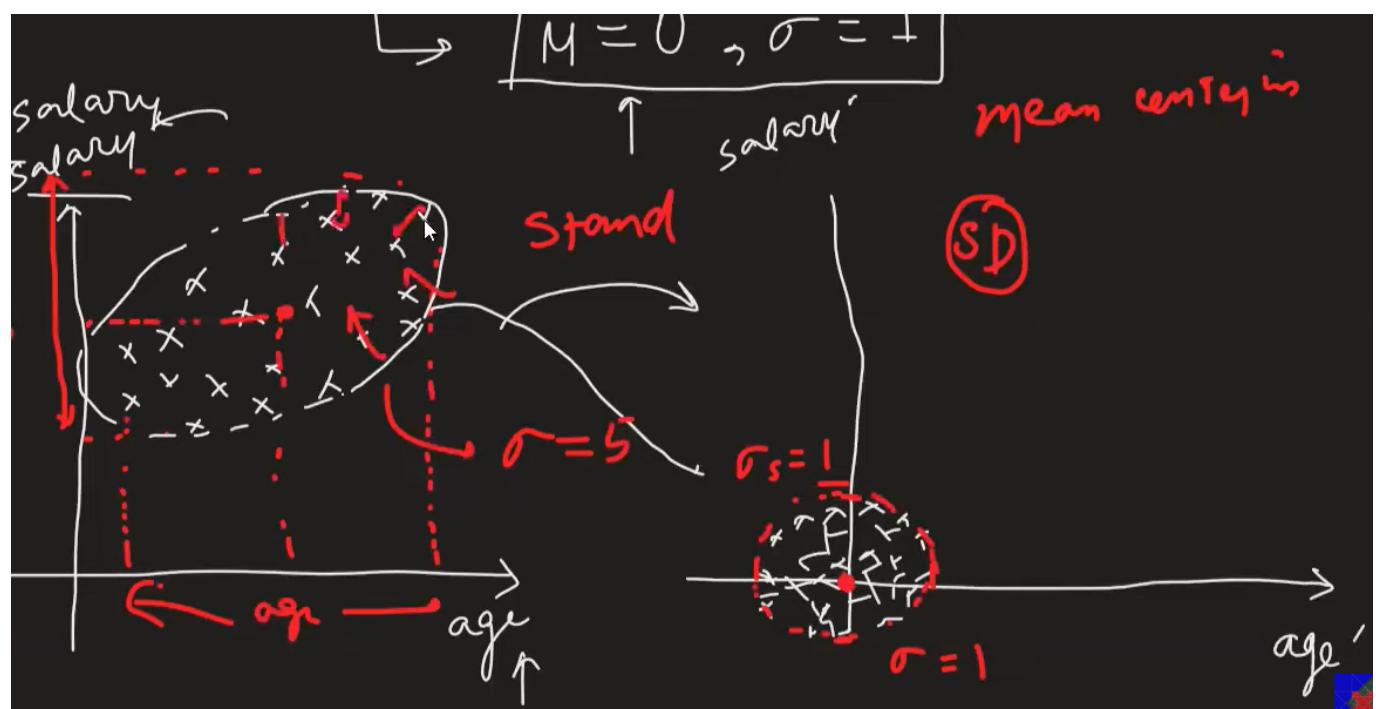
### When to use Standardization?

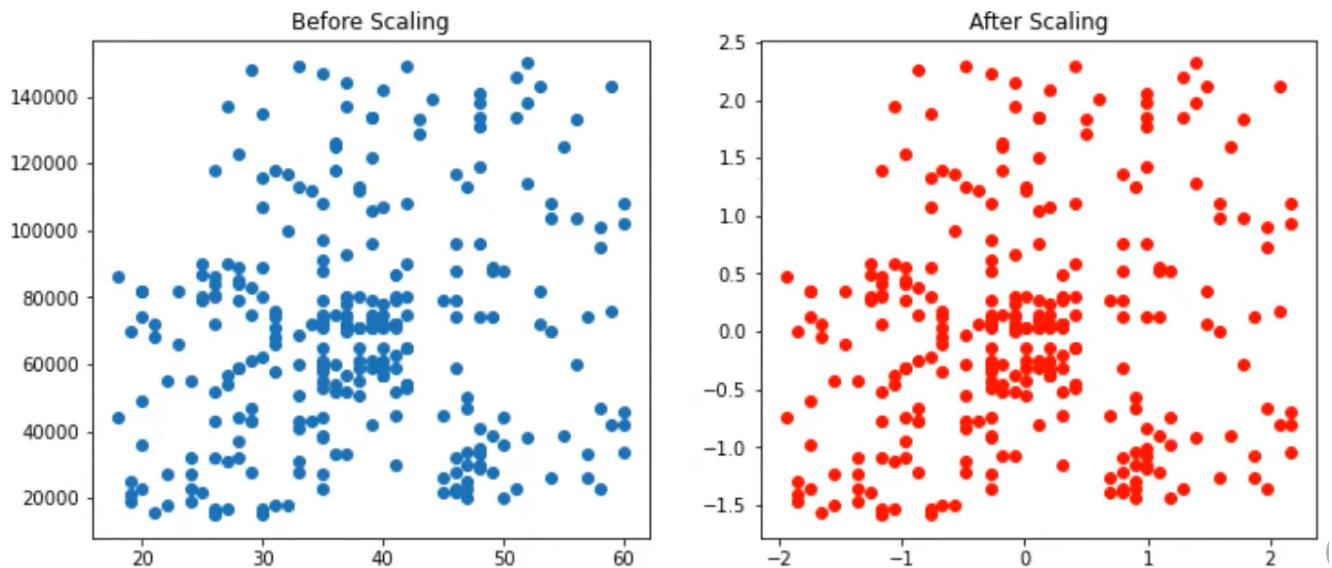
Friday, April 9, 2021 4:28 PM



Algorithm(s)	Reason of applying feature scaling
1. K-Means	Use the Euclidean distance measure.
2. K-Nearest-Neighbors	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): \*

#### 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.

- Dividing by  $n \rightarrow$  average error (scale-independent)
- Dividing by 2  $\rightarrow$  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.

