Module II: Supervised Learning

Supervised Learning: Linear regression, Logistic regression, Decision trees, Naïve Bayes Classifier, K Nearest Neighbor, Support vector machines, Overfitting and underfitting, Regularization.

Two Ways to Learn from Data

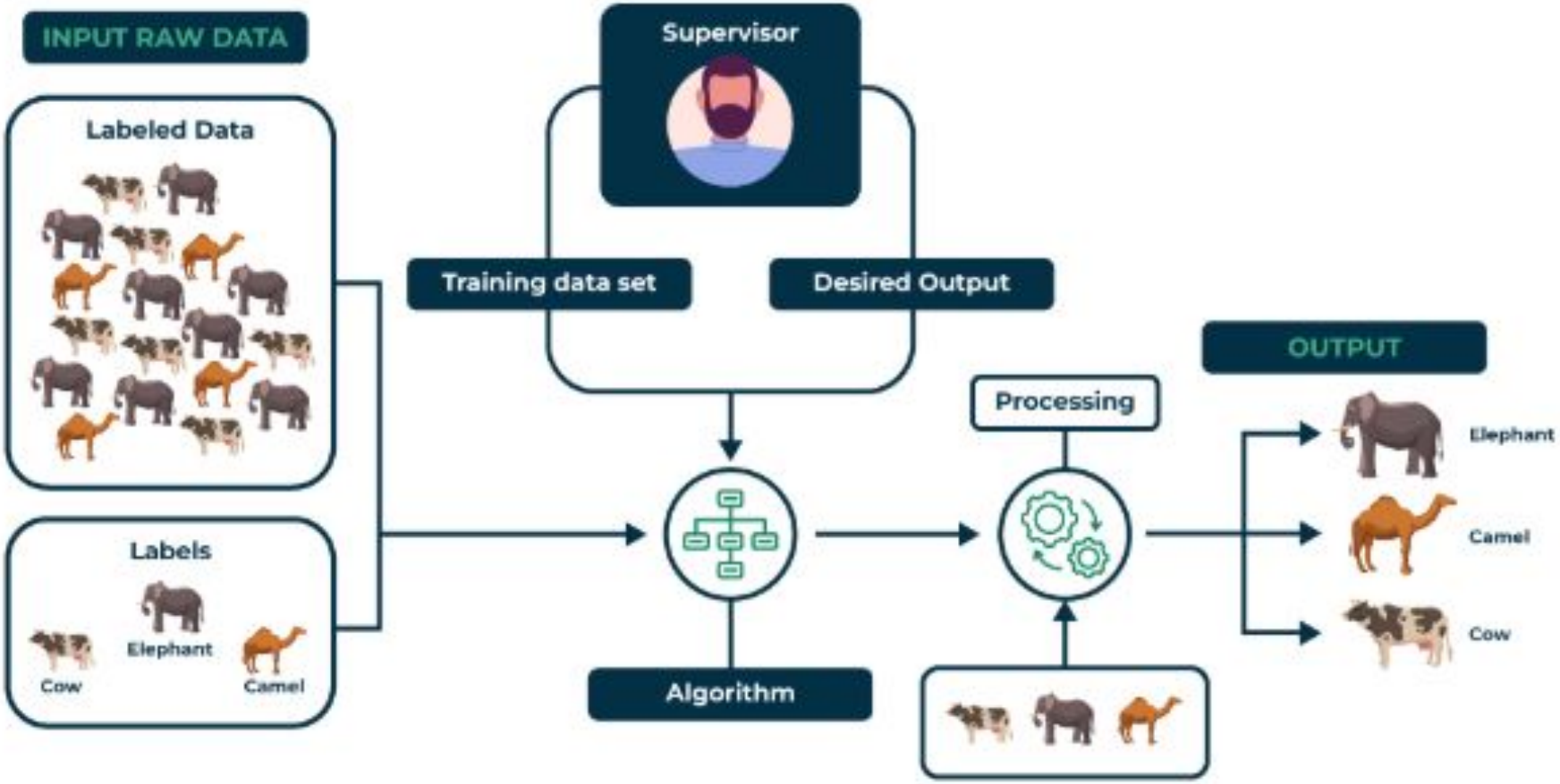
Supervised Learning

Learns from data with answers.

Unsupervised Learning

Finds hidden patterns in data without answers.

Supervised Learning



INPUT RAW DATA

- **Labeled Data**: Images of animals (cow, elephant, camel) where each image is associated with a known label.
- **Labels**: The categories — Cow, Elephant, and Camel — used to teach the algorithm.

Supervisor

- A "supervisor" provides the **training dataset** (input images + labels) and the **desired output(**model learns under guidance)

Algorithm

- The training dataset is fed into a machine learning **algorithm**.
- The algorithm uses the data to **learn the patterns** that differentiate the classes (e.g., physical features of cows vs elephants).

Processing

- The algorithm processes new/unseen data using what it learned during training.

OUTPUT

- The system is now able to **correctly classify new images** as Elephant, Camel, or Cow based on the learned features.

Types of Supervised Machine Learning

- Supervised Machine Learning
  - Regression
  - Classification

- A regression is used to predict **continuous** values such as house prices, stock prices or temperature. Regression algorithms learn how to connect input data to a specific number or value.
- A classification is used to predict **categorical** values such as whether a customer will buy or not, whether an email is spam or not or whether a medical image shows a tumor or not. Classification algorithms learn how to connect input data to the **probability** of belonging to different groups or categories.

User ID	Gender	Age	Salary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	1
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	1
15728773	Male	27	58000	1
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	1
15727311	Female	35	65000	0
15570769	Female	26	80000	1
15606274	Female	26	52000	0
15746139	Male	20	86000	1
15704987	Male	32	18000	0
15628972	Male	18	82000	0
15697686	Male	29	80000	0
15733883	Male	47	25000	1

Figure A: CLASSIFICATION

Temperature	Pressure	Relative Humidity	Wind Direction	Wind Speed
10.69261758	986.882019	54.19337313	195.7150879	3.278597116
13.59184184	987.8729248	48.0648859	189.2951202	2.909167767
17.70494885	988.1119385	39.11965597	192.9273834	2.973036289
20.95430404	987.8500366	30.66273218	202.0752869	2.965289593
22.9278274	987.2833862	26.06723423	210.6589203	2.798230886
24.04233986	986.2907104	23.46918024	221.1188507	2.627005816
24.41475295	985.2338867	22.25082295	233.7911987	2.448749781
23.93361956	984.8914795	22.35178837	244.3504333	2.454271793
22.68800023	984.8461304	23.7538641	253.0864716	2.418341875
20.56425726	984.8380737	27.07867944	264.5071106	2.318677425
17.76400389	985.4262085	33.54900114	280.7827454	2.343950987
11.25680746	988.9386597	53.74139903	68.15406036	1.650191426
14.37810685	989.6819458	40.70884681	72.62069702	1.553469896
18.45114201	990.2960205	30.85038484	71.70604706	1.005017161
22.54895853	989.9562988	22.81738811	44.66042709	0.264133632
24.23155922	988.796875	19.74790765	318.3214111	0.329656571

Figure B: REGRESSION

Types of Regression

- 1. Linear Regression
- 2. Logistic Regression
- 3. Decision Trees
- 4. Naive Bayes Classifier
- 5. K Nearest neighbor
- 6. Support Vector Machines

Linear regression

- Linear Regression is a supervised learning algorithm used for predicting continuous values.
- It models the relationship between input variables (independent) and a target variable (dependent) by fitting a linear equation.
- This relationship is represented by a straight line.

- **For example** we want to predict a student's exam score based on how many hours they studied. We observe that as students study more hours, their scores go up. In the example of predicting exam scores based on hours studied.

Here

- **Independent variable (input):** Hours studied because it's the factor we control or observe.
- **Dependent variable (output):** Exam score because it depends on how many hours were studied.
- We use the independent variable to predict the dependent variable.

Types of Linear Regression

- When there is only one independent feature it is known as Simple Linear Regression or Univariate Linear Regression .
- When there are more than one feature it is known as Multiple Linear Regression or Multivariate Regression.

Simple Linear Regression

- Simple linear regression is used when we want to predict a target value (dependent variable) using only one input feature (independent variable).
- It assumes a straight-line relationship between the two.
- **One independent variable (X)** and **one dependent variable (Y)**
- The graph is a **2D straight line**:

$$Y=mX+b$$

Term	Meaning	Description
Y	**Dependent variable**	The value you're trying to predict (also called the target or output).
X	**Independent variable**	The input or feature — the value you use to predict Y.
m	**Slope of the line**	Tells how much Y changes for a one-unit increase in X. Represents the relationship strength between X and Y.
b	**Intercept**	The value of Y when X=0

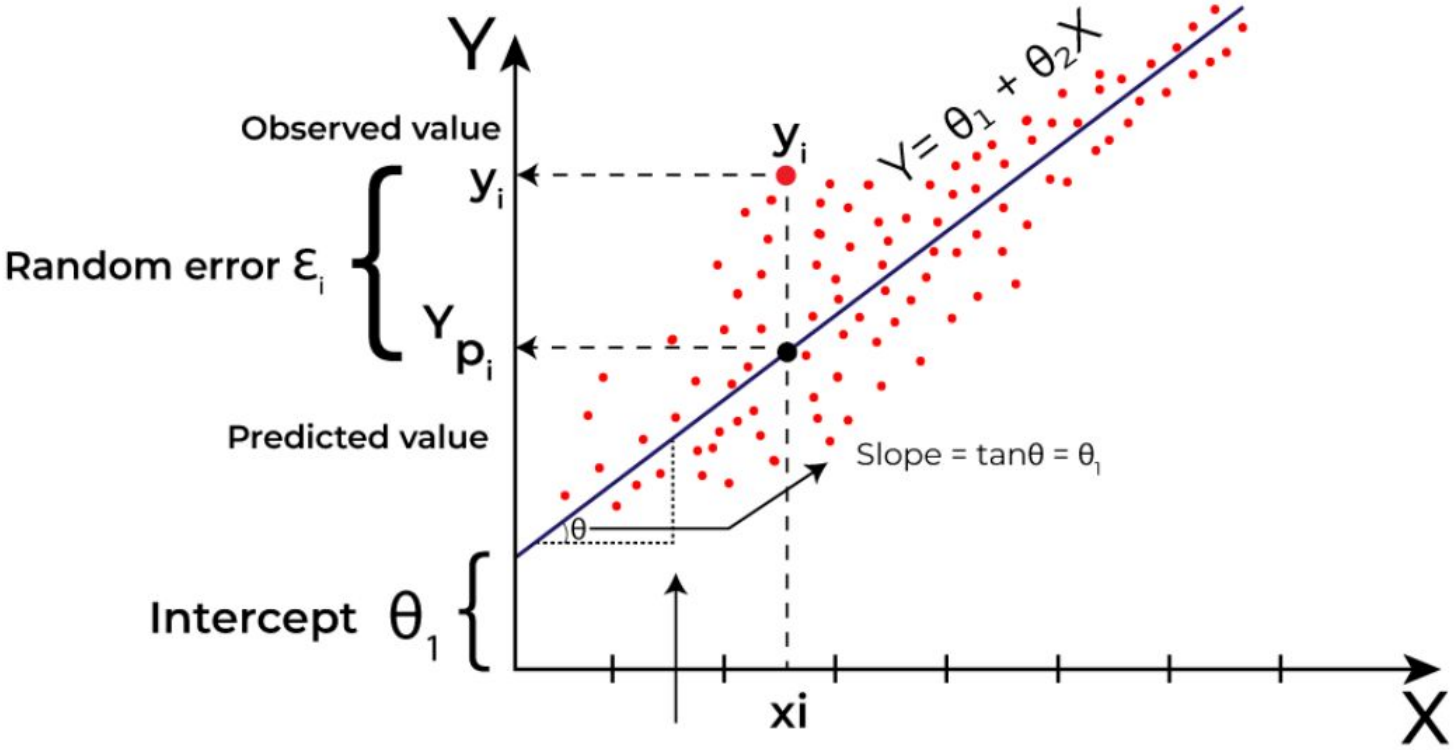
$$\hat{y} = \theta_0 + \theta_1 x$$

Where:

- $\hat{y}$ is the predicted value
- $x$is the input (independent variable)
- θ_0 is the intercept (value of $\hat{y}$ when x=0)
- θ_1 is the slope or coefficient (how much $\hat{y}$ changes with one unit of x)

$$Y = \theta_1 + \theta_2 X$$

- Y: Predicted output
- X: Input variable
- θ_1: Intercept (value of Y when $X = 0$)
- θ_2: Slope of the line (rate of change in Y with respect to X)



The Least Squares Method

- To find the best-fit line, we use a method called **Least Squares**.
- The idea behind this method is to minimize the sum of squared differences between the actual values (data points) and the predicted values from the line. These differences are called residuals.
- The formula for residuals is:

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

- **Where:**
- y_i is the actual observed value
- $\hat{y}_i$ is the predicted value from the line for that x_i

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

  $\text{Sumofsquarederrors(SSE)} = \Sigma(y_i - \hat{y}_i)^2$

- This method ensures that the line best represents the data where the sum of the squared differences between the predicted values and actual values is as small as possible.

🧮 How to Calculate the Line (Formulas)

Given data points (x_i, y_i), the slope θ_2 and intercept θ_1 are calculated as:

Slope (θ_2):

$$\theta_2 = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - (\sum x_i)^2}$$

Intercept (θ_1):

$$\theta_1 = \bar{y} - \theta_2 \bar{x}$$

Problem 1.

- A factory manager wants to predict the **production output** (Y) based on the **number of workers** (X).

Sl no.	Workers (X)	Output (Y)
1	2	40
2	4	50
3	6	65
4	8	80
5	10	95

Find the linear regression line

$$\hat{Y} = \theta_1 + \theta_2 X$$

Step 1: Calculate sums

We'll compute:

- $n = 5$
- $\sum X = 2 + 4 + 6 + 8 + 10 = 30$
- $\sum Y = 40 + 50 + 65 + 80 + 95 = 330$
- $\sum XY = (2)(40) + (4)(50) + (6)(65) + (8)(80) + (10)(95) = 80 + 200 + 390 + 640 + 950 = 2260$
- $\sum X^2 = 2^2 + 4^2 + 6^2 + 8^2 + 10^2 = 4 + 16 + 36 + 64 + 100 = 220$

Step 2: Compute slope θ_2

$$\theta_2 = \frac{n\sum XY - \sum X \sum Y}{n\sum X^2 - (\sum X)^2}$$

$$\theta_2 = \frac{5 \cdot 2260 - 30 \cdot 330}{5 \cdot 220 - 30^2} = \frac{11300 - 9900}{1100 - 900} = \frac{1400}{200} = 7$$

Step 3: Compute intercept θ_1

$$\theta_1 = \bar{Y} - \theta_2 \bar{X} = \frac{330}{5} - 7 \cdot \frac{30}{5} = 66 - 42 = 24$$

Regression Line Equation:

$$\hat{Y} = 24 + 7X$$

How much output is expected if there are **7 workers**?

$$\hat{Y} = 24 + 7 \cdot 7 = 24 + 49 = 73$$

Problem 2:

- A teacher wants to understand the relationship between **number of hours spent on revision (X)** and the **student's final grade (Y)**.

Revision Hours (X)	Final Grade (Y)
1	55
3	60
5	65
7	70
9	75

Find the linear regression line using the least squares method.

Step 1: Calculate values

$n = 5$

$\sum X = 1 + 3 + 5 + 7 + 9 = 25$

$\sum Y = 55 + 60 + 65 + 70 + 75 = 325$

$\sum XY = (1)(55) + (3)(60) + (5)(65) + (7)(70) + (9)(75) = 55 + 180 + 325 + 490 + 675 = 1725$

$\sum X^2 = 1^2 + 3^2 + 5^2 + 7^2 + 9^2 = 1 + 9 + 25 + 49 + 81 = 165$

Step 2: Compute slope θ_2

$$\theta_2 = \frac{n\sum XY - \sum X \sum Y}{n\sum X^2 - (\sum X)^2}$$

$$\theta_2 = \frac{5 \cdot 1725 - 25 \cdot 325}{5 \cdot 165 - 25^2} = \frac{8625 - 8125}{825 - 625} = \frac{500}{200} = 2.5$$

Step 3: Compute intercept θ_1

$$\theta_1 = \bar{Y} - \theta_2 \bar{X} = \frac{325}{5} - 2.5 \cdot \frac{25}{5} = 65 - 2.5 \cdot 5 = 65 - 12.5 = 52.5$$

Regression Equation:

$$\hat{Y} = 52.5 + 2.5X$$

If a student studies for **6 hours**, predicted grade is:

$$\hat{Y} = 52.5 + 2.5 \cdot 6 = 52.5 + 15 = 67.5$$

Multiple Linear Regression (multivariate regression)

- Multiple linear regression involves more than one independent variable and one dependent variable. The equation becomes:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

where:

- $\hat{y}$ is the predicted value
- $x_1, x_2, \ldots, x_n$ are the independent variables
- $\theta_1, \theta_2, \ldots, \theta_n$ are the coefficients (weights) corresponding to each predictor.
- θ_0 is the intercept.

Problem 3

A real estate agent wants to predict the **price of a house** (Y) based on:

- **Size in square feet (X_1)**
- **Number of bedrooms (X_2)**

Here's the data:

House	Size (sqft) X_1	Bedrooms X_2	Price (₹ in lakhs) Y
1	1000	2	50
2	1200	3	60
3	1500	3	70
4	1700	4	80

Find the **best-fitting line (or plane)** that models the relationship between:

- **Input variables (X_1 and X_2)**
- **Target variable (Y)**

This line will have the form:

$$\hat{Y} = \theta_0 + \theta_1 X_1 + \theta_2 X_2$$

Where:

- θ_0 is the intercept (value of Y when all Xs are 0),
- θ_1 is the coefficient for square footage,
- θ_2 is the coefficient for number of bedrooms.

Step 1: Represent the Data in Matrix Form

We write the data in the form of a **matrix** for calculation.

➤ Feature Matrix X:

We need to include a column of 1s for the **intercept**:

$$X = \begin{bmatrix} 1 & 1000 & 2 \\ 1 & 1200 & 3 \\ 1 & 1500 & 3 \\ 1 & 1700 & 4 \end{bmatrix}$$

- First column: 1 (for intercept θ_0)
- Second column: values of X_1 (size)
- Third column: values of X_2 (bedrooms)

Output Vector Y:

$$Y = \begin{bmatrix} 50 \\ 60 \\ 70 \\ 80 \end{bmatrix}$$

📜 Step 2: Use the Normal Equation

To find the **best-fitting parameters** $\theta_0, \theta_1, \theta_2$, we use this formula:

$$\theta = (X^T X)^{-1} X^T Y$$

Breakdown of formula:

- X^T is the transpose of matrix X (rows become columns).
- $X^T X$ is a square matrix resulting from multiplying X^T with X.
- $(X^T X)^{-1}$ is the inverse of that matrix.
- Multiplying that result by $X^T Y$ gives us the vector θ, which contains:

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}$$

Step 1: Write Matrices

X matrix (with intercept column):

$$X = \begin{bmatrix} 1 & 1000 & 2 \\ 1 & 1200 & 3 \\ 1 & 1500 & 3 \\ 1 & 1700 & 4 \end{bmatrix}$$

Y vector:

$$Y = \begin{bmatrix} 50 \\ 60 \\ 70 \\ 80 \end{bmatrix}$$

Step 2: Compute $X^T X$

$$X^T = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1000 & 1200 & 1500 & 1700 \\ 2 & 3 & 3 & 4 \end{bmatrix}$$

Now multiply $X^T \cdot X$:

$$X^T X = \begin{bmatrix} 4 & 5400 & 12 \\ 5400 & 7660000 & 16500 \\ 12 & 16500 & 38 \end{bmatrix}$$

Step 3: Compute $X^T Y$

$$X^T Y = \begin{bmatrix} 260 \\ 366000 \\ 795 \end{bmatrix}$$

📜 Step 4: Solve the Normal Equation

$$\theta = (X^T X)^{-1} X^T Y$$

We now solve the system of linear equations:

$$\begin{bmatrix} 4 & 5400 & 12 \\ 5400 & 7660000 & 16500 \\ 12 & 16500 & 38 \end{bmatrix} \cdot \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} 260 \\ 366000 \\ 795 \end{bmatrix}$$

This is a **3×3 system** of equations:

yaml

$$4\theta_0 + 5400\theta_1 + 12\theta_2 = 260$$
$$5400\theta_0 + 7660000\theta_1 + 16500\theta_2 = 366000$$
$$12\theta_0 + 16500\theta_1 + 38\theta_2 = 795$$

$$\theta_0 = 10, \quad \theta_1 = 0.03, \quad \theta_2 = 5$$

Final Regression Model:

$$\hat{Y} = 10 + 0.03X_1 + 5X_2$$

If house has:

- 1400 sqft
- 3 bedrooms

$$\hat{Y} = 10 + 0.03(1400) + 5(3) = 10 + 42 + 15 = ₹67 \text{ lakhs}$$

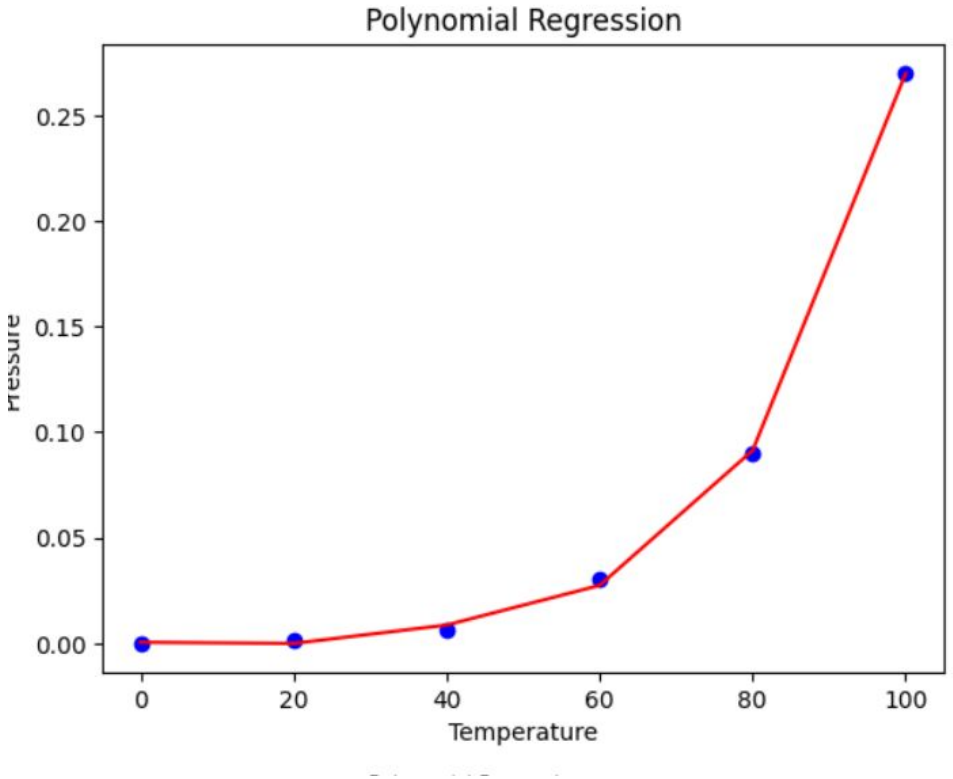
Polynomial Regression

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

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_n x^n + \epsilon$$

where:

- y is the dependent variable.
- x is the independent variable.
- $\beta_0, \beta_1, \ldots, \beta_n$ are the coefficients of the polynomial terms.
- $n$is the degree of the polynomial.
- ϵ represents the error term.



- Polynomial regression is **non-linear in terms of input variables (x)**,but it is still **linear in terms of the model parameters (coefficients)**
- **Polynomial Regression is Linear in Parameters**
- Even though the **relationship between x and y is non-linear**, the model is still **linear in parameters**:
- b0,b1,b2,b3 are not multiplied or raised to powers.

A company's HR department wants to model the relationship between an employee's **years of experience** and their **salary.** The data is as follows:

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

$$y = a_0 + a_1 x + a_2 x^2$$

So, we are solving for:

$$\vec{a} = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}$$

Using:

$$\vec{a} = (X^T X)^{-1} X^T Y$$

design matrix X and **vector** Y:

$$X = \begin{bmatrix} 1 & 1 & 1^2 \\ 1 & 2 & 2^2 \\ 1 & 3 & 3^2 \\ 1 & 4 & 4^2 \\ 1 & 5 & 5^2 \\ 1 & 6 & 6^2 \\ 1 & 7 & 7^2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \\ 1 & 5 & 25 \\ 1 & 6 & 36 \\ 1 & 7 & 49 \end{bmatrix}$$

$$Y = \begin{bmatrix} 50000 \\ 55000 \\ 65000 \\ 80000 \\ 110000 \\ 150000 \\ 200000 \end{bmatrix}$$

$X^T X$

$$X^T X = \begin{bmatrix} 7 & 28 & 140 \\ 28 & 140 & 784 \\ 140 & 784 & 4676 \end{bmatrix}$$

$X^T Y$

$$X^T Y = \begin{bmatrix} 710000 \\ 3470000 \\ 20024000 \end{bmatrix}$$

$$\vec{a} = (X^T X)^{-1} X^T Y$$

Let's calculate $(X^T X)^{-1} \cdot (X^T Y)$

I'll now compute this using matrix algebra:

Let:

$$A = \begin{bmatrix} 7 & 28 & 140 \\ 28 & 140 & 784 \\ 140 & 784 & 4676 \end{bmatrix}, \quad B = \begin{bmatrix} 710000 \\ 3470000 \\ 20024000 \end{bmatrix}$$

Now compute $A^{-1}B$

Using the least squares method, the coefficients are:

$$a_0 = 123428.57, \quad a_1 = -52166.67, \quad a_2 = 9333.33$$

So, the **polynomial regression model** is:

$$\boxed{y = 123428.57 - 52166.67x + 9333.33x^2}$$

How to make a linear regression model?

Cost function for Linear Regression

- Loss functions are integral to the training process of machine learning models
- They provide a measure of how well the model's predictions align with the actual data.
- By minimizing this loss, models learn to make more accurate predictions.
- The choice of a loss function can significantly affect the performance of a model

- In Linear Regression, the Mean Squared Error (MSE) cost function is employed, which calculates the average of the squared errors between the predicted values $y\hat{}i$ and the actual values yi.
- The purpose is to determine the optimal values for the intercept $\theta 1$ and the coefficient of the input feature $\theta 2$ providing the best-fit line for the given data points.

Mean Squared Error (MSE)

- The Mean Squared Error (MSE) is a common loss function in machine learning where the mean of the squared residuals is taken rather than just the sum.
- This ensures that the loss function is independent of the number of data points in the training set, making the metric more reliable across datasets of varying sizes.
- However, MSE is sensitive to outliers, as large errors have a disproportionately large impact on the final result.

- This squaring process is essential for most regression loss functions, ensuring that models can minimize error and improve performance.
- The formula is:

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

Where:

- y_i is the actual value for the i-th data point.
- $\hat{y}_i$ is the predicted value for the i-th data point.
- n is the total number of data points.

Gradient Descent for Linear Regression

- A linear regression model can be trained using the optimization algorithm gradient descent by iteratively modifying the model's parameters to reduce the mean squared error (MSE) of the model on a training dataset.
- To update θ_1 and θ_2 values in order to reduce the Cost function (minimizing RMSE value) and achieve the best-fit line the model uses Gradient Descent. The idea is to start with random θ_1 and θ_2 values and then iteratively update the values, reaching minimum cost.
- A gradient is nothing but a derivative that defines the effects on outputs of the function with a little bit of variation in inputs.

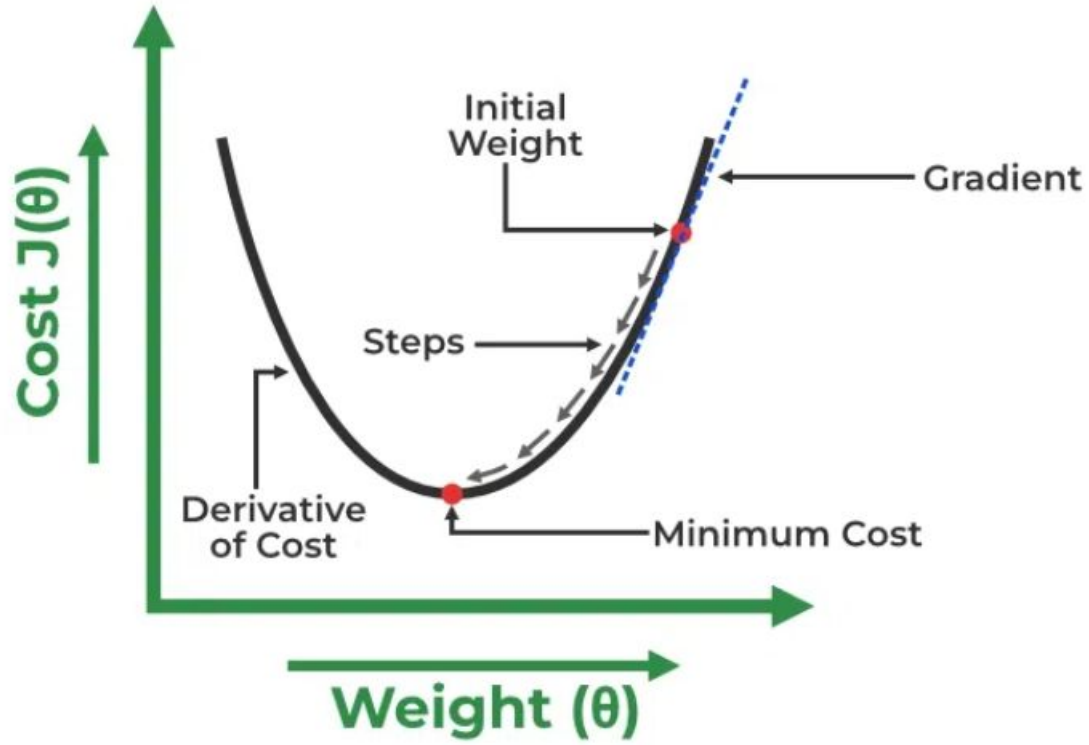
Let's differentiate the cost function(J) with respect to θ_1

$$
\begin{aligned}
J'_{\theta_1} &= \frac{\partial J(\theta_1, \theta_2)}{\partial \theta_1} \\
&= \frac{\partial}{\partial \theta_1} \left[\frac{1}{n} \left(\sum_{i=1}^{n} (\hat{y}_i - y_i)^2 \right) \right] \\
&= \frac{1}{n} \left[\sum_{i=1}^{n} 2(\hat{y}_i - y_i) \left(\frac{\partial}{\partial \theta_1} (\hat{y}_i - y_i) \right) \right] \\
&= \frac{1}{n} \left[\sum_{i=1}^{n} 2(\hat{y}_i - y_i) \left(\frac{\partial}{\partial \theta_1} (\theta_1 + \theta_2 x_i - y_i) \right) \right] \\
&= \frac{1}{n} \left[\sum_{i=1}^{n} 2(\hat{y}_i - y_i) \left(1 + 0 - 0 \right) \right] \\
&= \frac{1}{n} \left[\sum_{i=1}^{n} (\hat{y}_i - y_i) \left(2 \right) \right] \\
&= \frac{2}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)
\end{aligned}
$$

Let's differentiate the cost function(J) with respect to θ_2

$$
\begin{aligned}
J'_{\theta_2} &= \frac{\partial J(\theta_1, \theta_2)}{\partial \theta_2} \\
&= \frac{\partial}{\partial \theta_2} \left[\frac{1}{n} \left(\sum_{i=1}^{n} (\hat{y}_i - y_i)^2 \right) \right] \\
&= \frac{1}{n} \left[\sum_{i=1}^{n} 2(\hat{y}_i - y_i) \left(\frac{\partial}{\partial \theta_2} (\hat{y}_i - y_i) \right) \right] \\
&= \frac{1}{n} \left[\sum_{i=1}^{n} 2(\hat{y}_i - y_i) \left(\frac{\partial}{\partial \theta_2} (\theta_1 + \theta_2 x_i - y_i) \right) \right] \\
&= \frac{1}{n} \left[\sum_{i=1}^{n} 2(\hat{y}_i - y_i) \left(0 + x_i - 0 \right) \right] \\
&= \frac{1}{n} \left[\sum_{i=1}^{n} (\hat{y}_i - y_i) \left(2x_i \right) \right] \\
&= \frac{2}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i) \cdot x_i
\end{aligned}
$$

FINDING THE COEFFICIENTS OF A LINEAR EQUATION THAT BEST FITS THE TRAINING DATA IS THE OBJECTIVE OF LINEAR REGRESSION. BY MOVING IN THE DIRECTION OF THE MEAN SQUARED ERROR NEGATIVE GRADIENT WITH RESPECT TO THE COEFFICIENTS, THE COEFFICIENTS CAN BE CHANGED. AND THE RESPECTIVE INTERCEPT AND COEFFICIENT OF X WILL BE IF A IS THE LEARNING RATE.



$$\theta_1 = \theta_1 - \alpha \left(J'_{\theta_1} \right)$$

$$= \theta_1 - \alpha \left(\frac{2}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i) \right)$$

$$\theta_2 = \theta_2 - \alpha \left(J'_{\theta_2} \right)$$

$$= \theta_2 - \alpha \left(\frac{2}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i) \cdot x_i \right)$$

Categorical data

- Categorical data represents **qualities or characteristics** that can be divided into groups. There are **two main types** of categorical data:

Nominal Data (Unordered Categories)

- Categories **without any natural order**.
- **Examples**:
  - Gender: Male, Female, Other
  - Color: Red, Blue, Green
  - Nationality: Indian, American, French
- You **cannot say** one is greater or smaller than the other.
- Usually encoded using **One Hot Encoding**.

Ordinal Data (Ordered Categories)

Categories **with a clear, meaningful order**, but the intervals between them are not necessarily equal.

- **Examples**:
  - Education level: High School < Bachelor < Master < PhD
  - Satisfaction: Poor < Average < Good < Excellent
  - Rank: 1st, 2nd, 3rd
- You **can compare** the categories
- Usually encoded using **Label Encoding** or custom mapping (e.g., Poor=1, Average=2, Good=3).

One-Hot Encoding

- **One-Hot Encoding** is a technique to convert **categorical (nominal)** data into a format that can be provided to **machine learning algorithms**, which typically require numerical input.
- For each category, One-Hot Encoding creates a **new binary (0 or 1) column**.

Example

```
import pandas as pd
df = pd.DataFrame({'Color': ['Red', 'Blue', 'Green', 'Red']})
encoded_df = pd.get_dummies(df, columns=['Color'])
print(encoded_df)
```

- **Drop One Column to Avoid Multicollinearity (Dummy Variable Trap)**
- encoded_df = pd.get_dummies(df, columns=['Color'], drop_first=True)
- This will keep only Color_Blue and Color_Green, dropping Color_Red (the reference category).

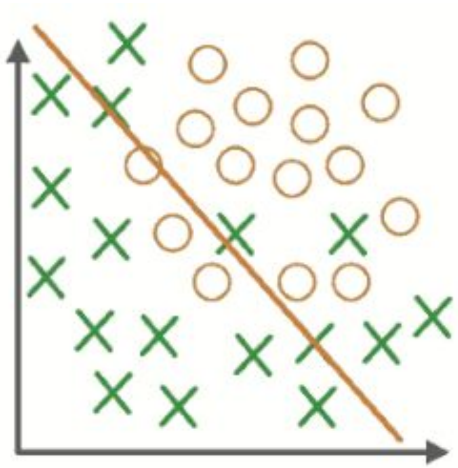
Multicollinearity

- **Multicollinearity** occurs when **two or more independent (predictor) variables in a regression model are highly correlated** with each other.
- This means that one predictor can be linearly predicted from another with a high degree of accuracy.

Overfitting and Underfitting

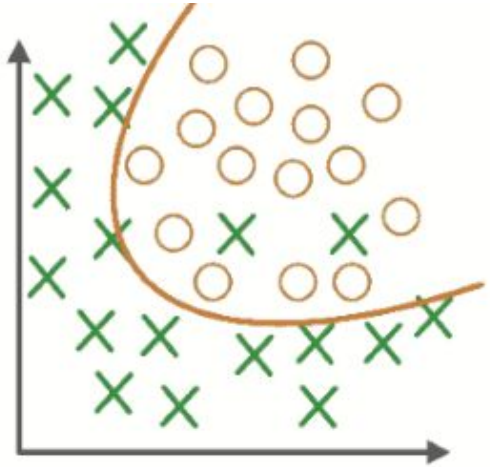
- **Overfitting** happens when a machine learning model learns the training data too well including the noise and random details. This makes the model to perform poorly on new, unseen data because it memorizes the training data instead of understanding the general patterns.
- For example, if we only study last week's weather to predict tomorrow's i.e our model might focus on one-time events like a sudden rainstorm which won't help for future predictions.

- **Underfitting** is the opposite problem which happens when the model is too simple to learn even the basic patterns in the data.
- An underfitted model performs poorly on both training and new data.
- To fix this we need to make the model more complex or add more features.
- For example if we use only the average temperature of the year to predict tomorrow's weather hence the model misses important details like seasonal changes which results in bad predictions.

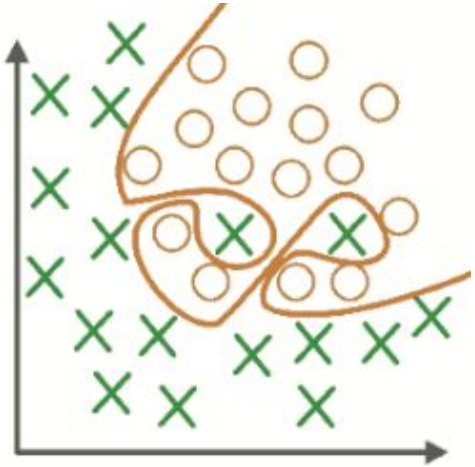


Under-fitting

(too simple to explain the variance)



Appropirate-fitting



Over-fitting

(forcefitting--too good to be true)

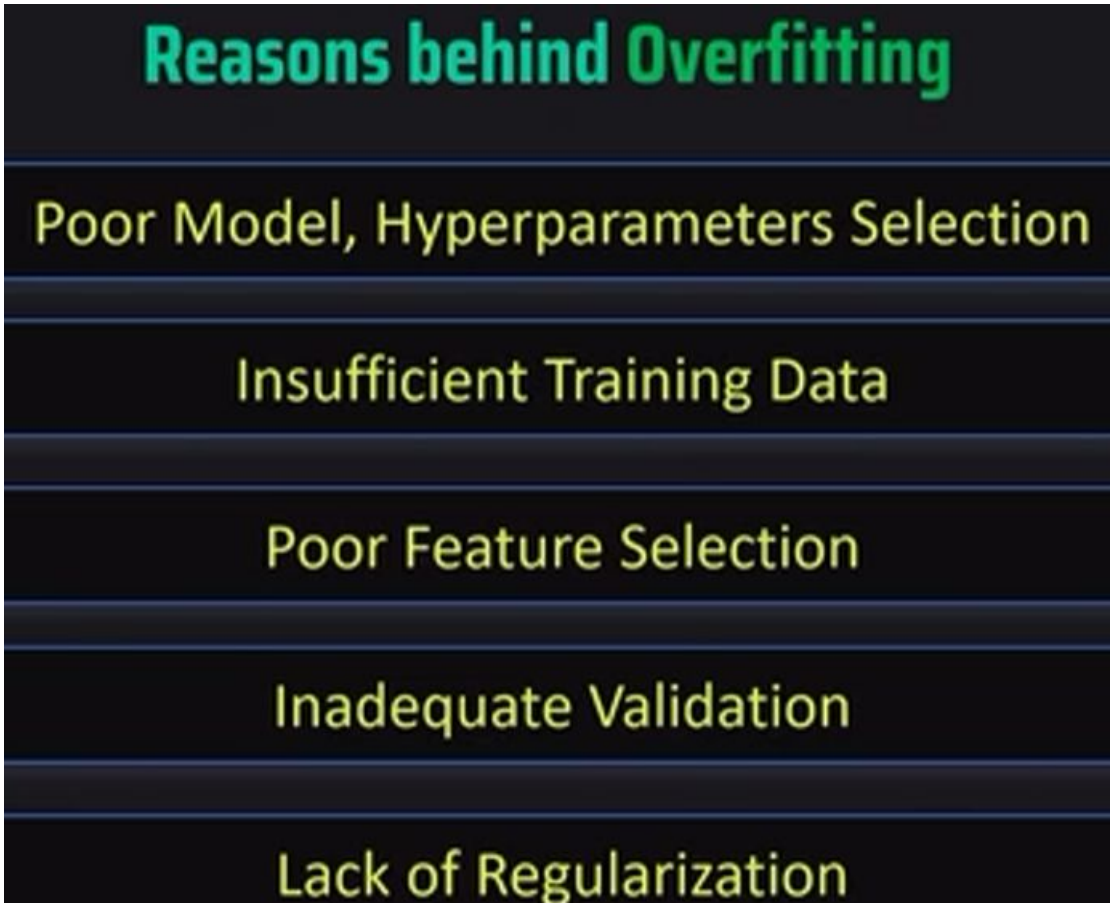
Examples

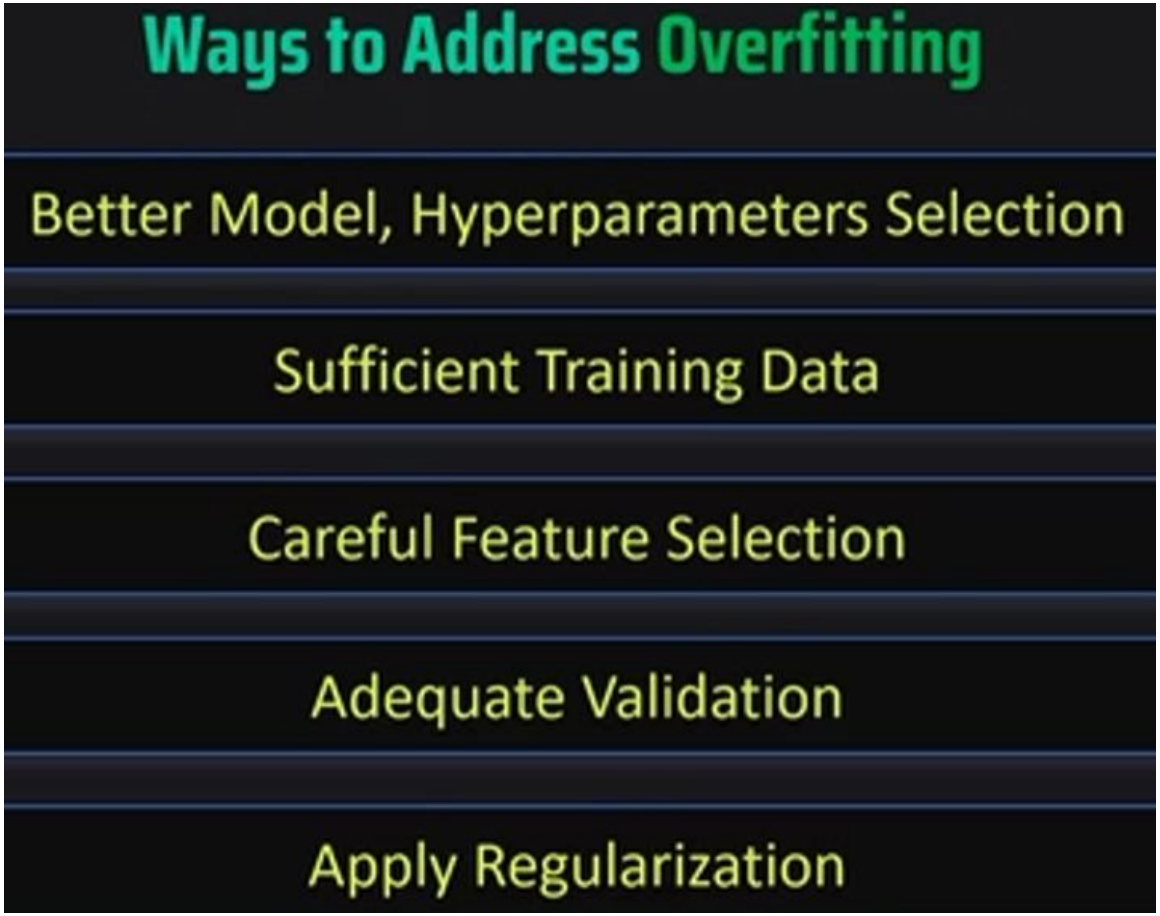
Education

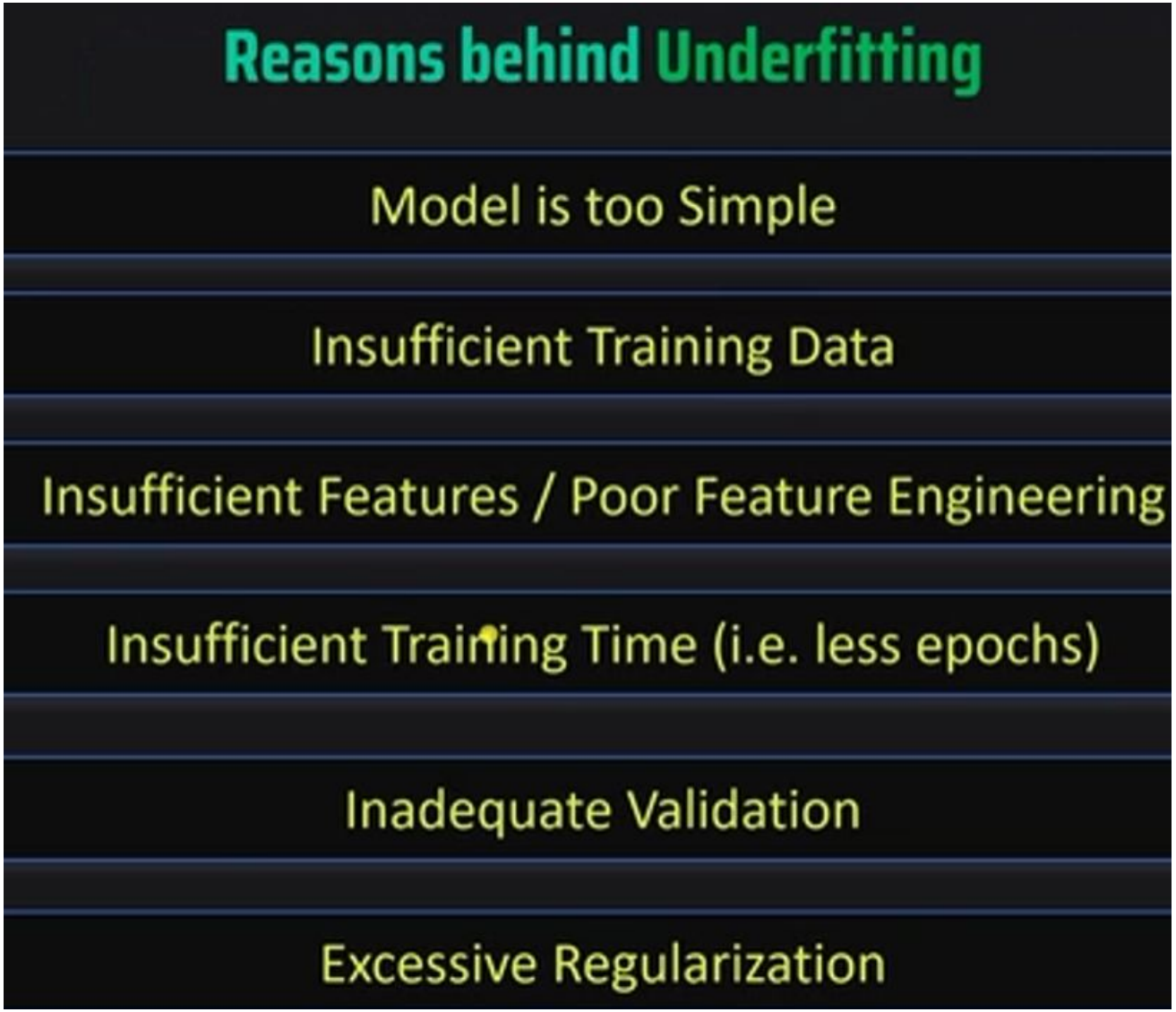
- **Underfitting**: Predicting student success using only age
- **Balanced**: Using attendance, assignment scores, and exam marks
- **Overfitting**: Using all past question patterns, even rare ones, so the model memorizes instead of generalizing

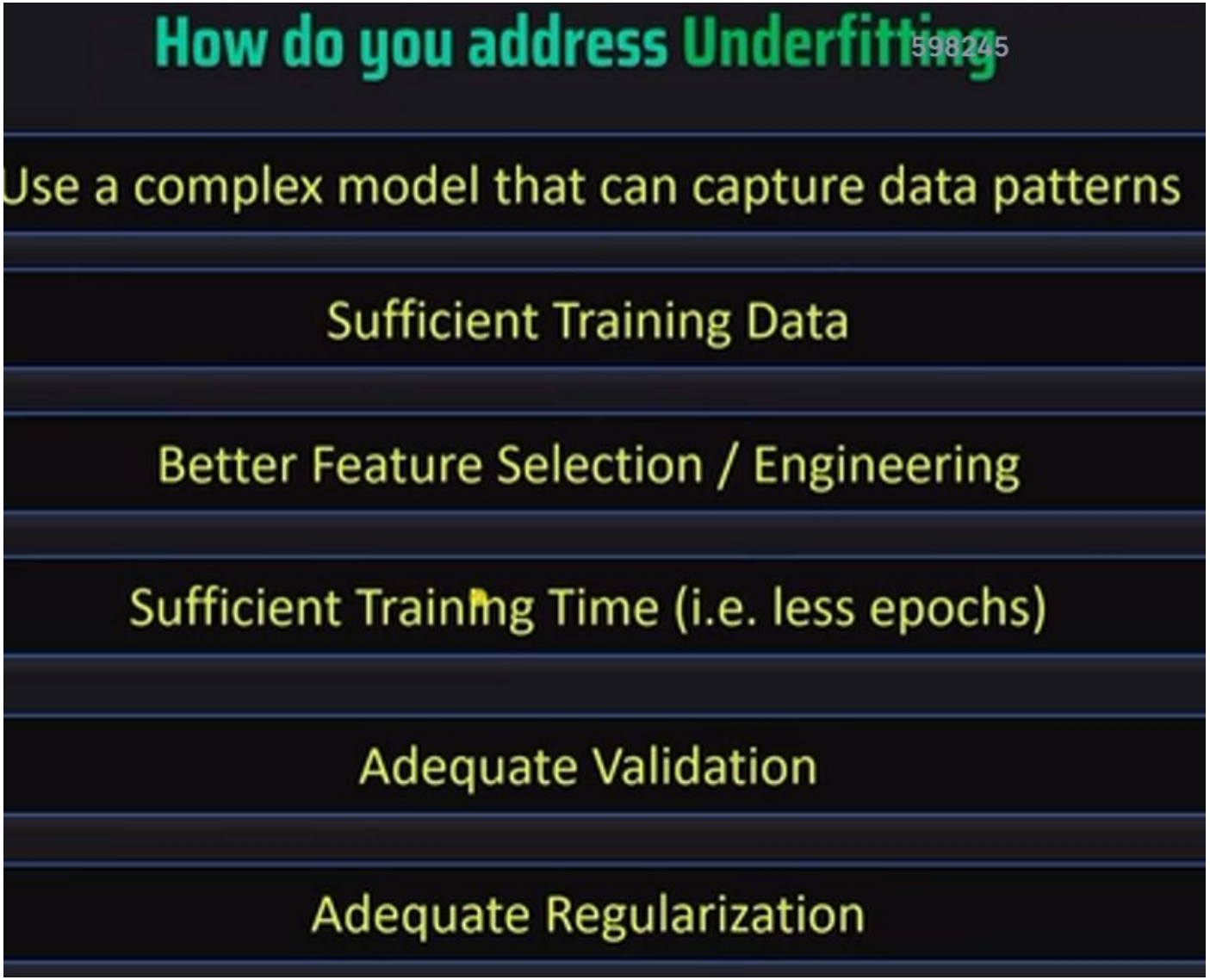
Medical Diagnosis

- **Underfitting**: Diagnosing a disease using only age and gender
- **Balanced**: U
- sing symptoms, test results, history, and risk factors
- **Overfitting**: Using too many specific variables, including rarely occurring biomarkers


Reasons behind Overfitting
Poor Model, Hyperparameters Selection
Insufficient Training Data
Poor Feature Selection
Inadequate Validation
Lack of Regularization


Ways to Address Overfitting
Better Model, Hyperparameters Selection
Sufficient Training Data
Careful Feature Selection
Adequate Validation
Apply Regularization


Reasons behind Underfitting
Model is too Simple
Insufficient Training Data
Insufficient Features / Poor Feature Engineering
Insufficient Training Time (i.e. less epochs)
Inadequate Validation
Excessive Regularization


How do you address Underfitting
598245
Use a complex model that can capture data patterns
Sufficient Training Data
Better Feature Selection / Engineering
Sufficient Training Time (i.e. less epochs)
Adequate Validation
Adequate Regularization

Regularization in Machine Learning

- **Regularization** is an important technique in machine learning that helps to improve model accuracy by preventing overfitting which happens when a model learns the training data too well including noise and outliers and perform poor on new data

Regularization Helps By:

- **Adding a penalty term** to the model's loss function.
- **Discouraging large coefficients** (weights) in the model.
- **Simplifying the model** → better generalization to new data.

Type	Technique	Penalty Term	Effect
L1	Lasso Regression	Adds sum of absolute values of coefficients	Can make some coefficients exactly **zero** (feature selection)
L2	Ridge Regression	Adds sum of squared coefficients	Shrinks coefficients but doesn't make them zero

Types of Regularization
1. Lasso Regression

- A regression model which uses the **L1 Regularization** technique is called **LASSO (Least Absolute Shrinkage and Selection Operator)** regression.
- It adds the **absolute value of magnitude** of the coefficient as a penalty term to the loss function(L).
- This penalty can shrink some coefficients to zero which helps in selecting only the important features and ignoring the less important ones.

$$\text{Cost} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{i=1}^{m} |w_i|$$

where

- m - Number of Features
- n- Number of Examples
- yi- Actual Target Value
- $\hat{y}_i$ - Predicted Target Value

n = Number of samples

- This is the number of **observations**, **rows**, or **data points** in your dataset.
- Example: If you have data for **100 houses**, then n = 100.

m = Number of features

- This is the number of **independent variables** or **predictors** (i.e., columns used for prediction).
- Example: If you use Area, Bedrooms, and Age to predict Price, then p = 3.

Hyperparameter – λ (Lambda)

- Controls the strength of regularization.
- Larger λ → higher penalty → simpler model.
- λ = 0 → becomes normal linear regression (no regularization).

When to Use Lasso Regression (L1 Regularization)

Use Lasso When...	Explanation
If some features are irrelevant	Lasso can shrink some coefficients **exactly to zero**, effectively removing them. This makes it useful for **feature selection**.
You want a **simpler model**	It automatically removes less important variables, reducing overfitting.
You have **high-dimensional data**	Especially useful when the number of features is **greater than** the number of samples.
To **understand which variables matter most**	Lasso helps identify key predictors.

In a housing price dataset with 100 features, only 10 might be significant. Lasso will retain only those 10 and remove the rest.

2. Ridge Regression

- A regression model that uses the **L2 regularization** technique is called **Ridge regression**.
- It adds the **squared magnitude** of the coefficient as a penalty term to the loss function(L).

$$\text{Cost} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda\sum_{i=1}^{m} w_i^2$$

where,

- n= Number of examples or data points
- m = Number of features i.e predictor variables
- y_i = Actual target value for the ith example
- $\hat{y}_i$ = Predicted target value for the ith example
- w_i = Coefficients of the features
- λ= Regularization parameter that controls the strength of regularization

When to Use Ridge Regression (L2 Regularization)

Use Ridge When...	Explanation
All features are **somewhat relevant**	Ridge shrinks all coefficients but **doesn't make them zero**, so all variables stay in the model.
Multicollinearity among features	Ridge handles highly correlated features better than Lasso by distributing weights more evenly.
To **keep all features** in the model	Useful when removing features is not acceptable (e.g., for interpretability or regulation).
If more focused on **prediction accuracy** than interpretability	Ridge keeps more information, which may help in prediction.

In a dataset where many features influence the outcome slightly but none should be dropped, Ridge is preferred.

3. Elastic Net Regression

- **Elastic Net Regression** is a combination of both **L1 as well as L2 regularization.**
- That shows that we add the **absolute norm of the weights** as well as the **squared measure of the weights**.
- With the help of an extra hyperparameter that controls the ratio of the L1 and L2 regularization.

$$\text{Cost} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda\left((1-\alpha)\sum_{i=1}^{m}|w_i| + \alpha\sum_{i=1}^{m}w_i^2\right)$$

where

- n = Number of examples (data points)
- m = Number of features (predictor variables)
- y_i = Actual target value for the ith example
- $\hat{y}_i$ = Predicted target value for the ith example
- wi= Coefficients of the features
- λ= Regularization parameter that controls the strength of regularization
- α = Mixing parameter where 0 ≤ α≤ 1 and α= 1 corresponds to Lasso (L_1) regularization, α= 0 corresponds to Ridge (L_2) regularization and Values between 0 and 1 provide a balance of both L1 and L2 regularization

Variance

- If our model is allowed to view the data too many times, it will learn very well for only that data.
- It will capture most patterns in the data, but it will also learn from the unnecessary data present, or from the noise.
- Variance can be defined as the model's sensitivity to fluctuations in the data.
- Our model may learn from noise. This will cause our model to consider trivial features as important.

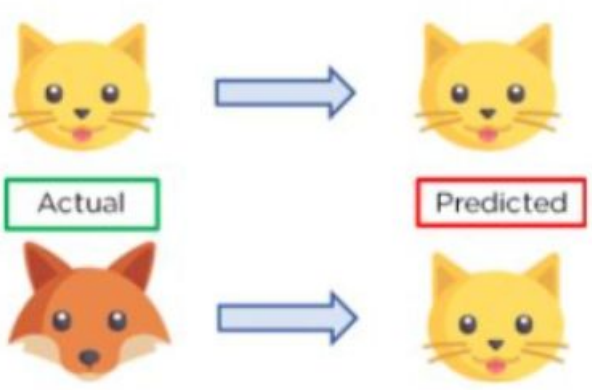


Figure 4: Example of Variance

- In the figure, we can see that our model has learned extremely well for our training data, which has taught it to identify cats.
- But when given new data, such as the picture of a fox, our model predicts it as a cat, as that is what it has learned.
- This happens when the Variance is high, our model will capture all the features of the data given to it, including the noise, will tune itself to the data, and predict it very well but when given new data, it cannot predict on it as it is too specific to training data.

- Hence, our model will perform really well on testing data and get high accuracy but will fail to perform on new, unseen data.
- New data may not have the exact same features and the model won't be able to predict it very well. This is called Overfitting.

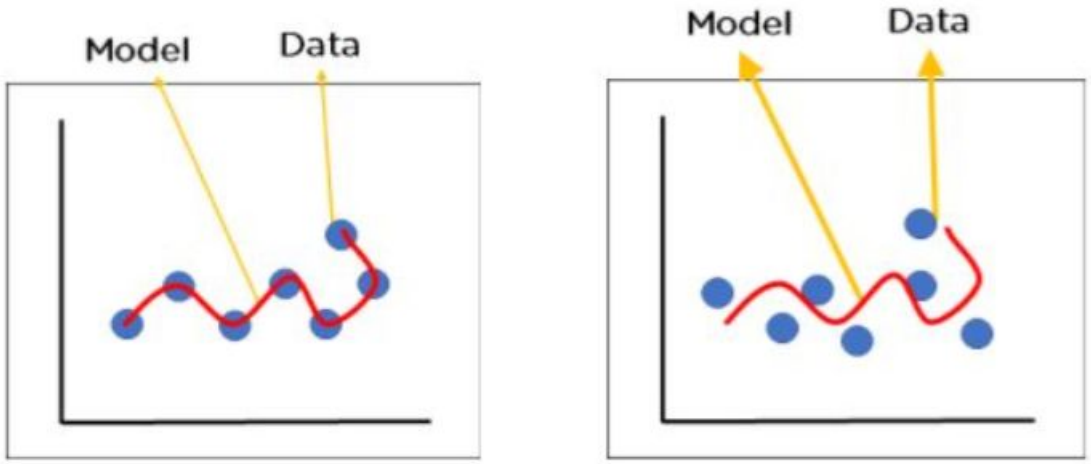


Figure 5: Over-fitted model where we see model performance on, a) training data b) new data

Bias

- Bias is the difference between our actual and predicted values.
- Bias is the simple assumptions that our model makes about our data to be able to predict new data.

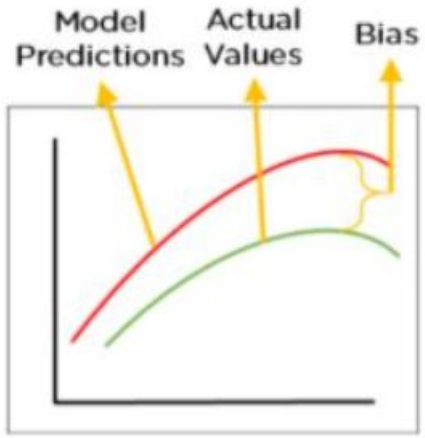


Figure 2: Bias

- When the Bias is high, assumptions made by our model are too basic, the model can't capture the important features of our data.
- This means that our model hasn't captured patterns in the training data and hence cannot perform well on the testing data too.
- If this is the case, our model cannot perform on new data and cannot be sent into production.
- This instance, where the model cannot find patterns in our training set and hence fails for both seen and unseen data, is called Underfitting.

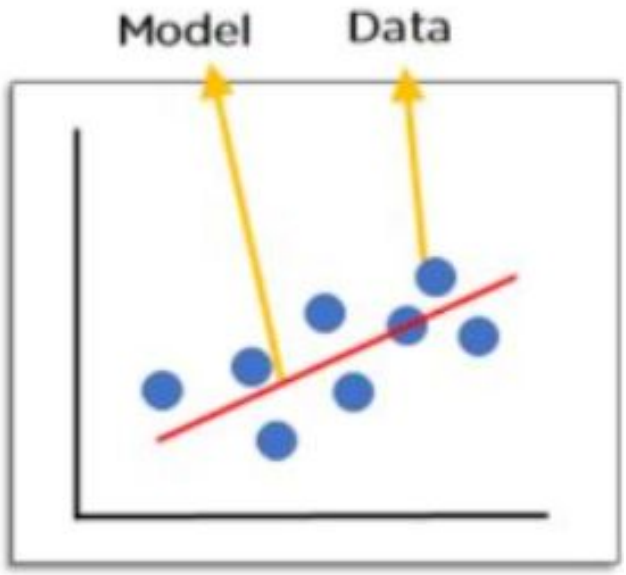


Figure 3: Underfitting