B Tech – 6th Semester

MACHINE LEARNING

Hello!

I am Dr. Sumana De

Assistant Professor

Department of Computer Science and Engineering

KIIT University, Odisha

MODULE 1

Lecture	Topics
1	Introduction to Machine Learning, definition, and real-world applications.
2	Types of machine learning - Supervised, Unsupervised, Semi supervised learning, Definitions and examples.
3	Regression - Linear Regression, Intuition, Cost Function
4	Linear Regression - Gradient Descent
5	Multiple Linear regression
6	Closed-form Equation, Type of Gradient Descent (Batch, Stochastic, Mini-batch) - Definition, properties.
7	Normalization and Standardization (definition and why), Overfitting and Underfitting
8	Bias, Variance, Bias and Variance trade-off
9	Regularization - Lasso Regularization, Ridge Regularization

MODULE 2

Lecture	Topics
11	Classification, Logistic Regression - 1 (binary)
12	Logistic Regression - 2 (binary)
13	Nearest neighbour and K Nearest Neighbour
14	Error Analysis - Train/Test Split, validation set, Accuracy, Precision, Recall, F-measure, ROC curve, Confusion Matrix
15	Naive Bayes Classifier - 1
16	Naive Bayes Classifier - 2

17	Decision Tree: Introduction, Id3 Algorithm - 1
18	Decision Tree - Id3 Algorithm - 2
19	Decision Tree - Problem of Overfitting, Pre-pruning/post-pruning Decision Tree, Examples.
20	Support Vector Machine - Terminologies, Intuition, Learning, Derivation - 1
21	Support Vector Machine - Terminologies, Intuition, Learning, Derivation - 2
22	Support Vector Machine - KKT Condition - 3
23	Support Vector Machine - Kernel, Nonlinear Classification, and

25	Principal Component Analysis - Steps, merits, demerits, Intuition - 1
26	Principal Component Analysis - Steps, merits, demerits, Intuition - 2
27	Understanding and Implementing PCA using SVD for dimensionality reduction

MODULE 3

Lecture	Topics
28	Clustering: Introduction, K-means Clustering - 1
29	K-Median Clustering - 2
30	K-Means Clustering – 3 (Numerical)
31	DBSCAN Clustering - Why we use?, parameters, characterization of points, steps, determining parameters, time/space complexities
32	Mean Shift Clustering
33	Hierarchical Clustering - Agglomerative Clustering, Single/Complete/Average/Centroid Linkage
34	Hierarchical Clustering - Divisive hierarchical clustering

MODULE 4

Lecture	Topics
36	Introduction Neural networks, McCulloch-Pitts Neuron
37	Least Mean Square (LMS) Algorithm
38	Perceptron Model
39	Multilayer Perceptron (MLP) and Hidden layer representation
40	Non-linear problem solving, Activation Functions
41	Backpropagation Algorithm - 1
42	Backpropagation Algorithm - 2
43	Exploding Gradient Problem and Vanishing Gradient Problem, why and how to avoid
44	Introduction to Convolutional Neural Network (CNN)
45	Basic idea about their working and structure
46	Data Augmentation, Batch Normalization, Dropout

MODULE 5

Lecture	Topics
48	Introduce machine learning tools like Scikit Learn, PyTorch, TensorFlow, Kaggle competitions, etc.
	Case Study (Any Two)

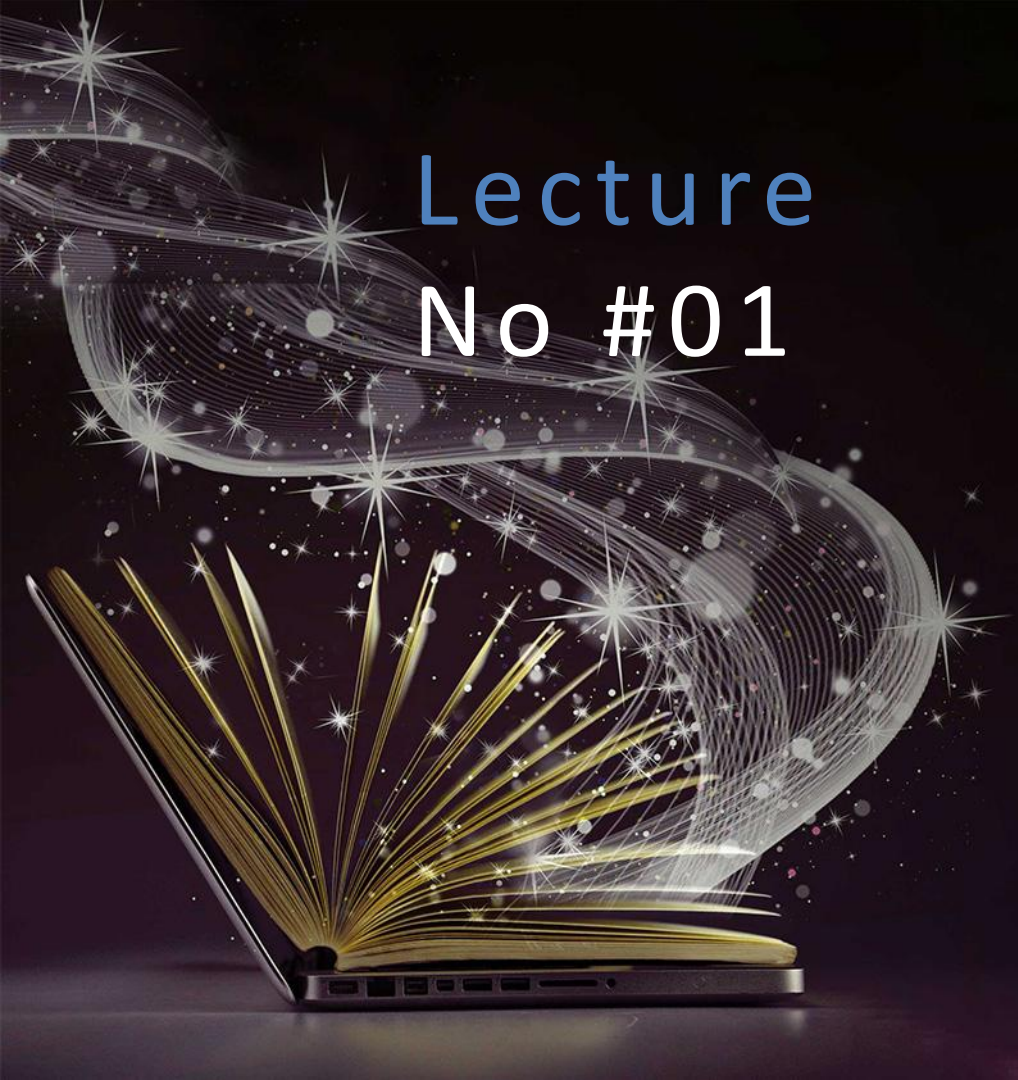
Case Study – 1: Implement linear regression to predict house prices based on features like size, location, and number of rooms.

Case Study – 2: Feature Extraction using PCA for Wine Dataset

Case Study – 3: Classification using Iris Dataset

Case Study – 4: Clustering using Iris Dataset

Case Study – 5: Classification of MNIST Dataset using CNN Model

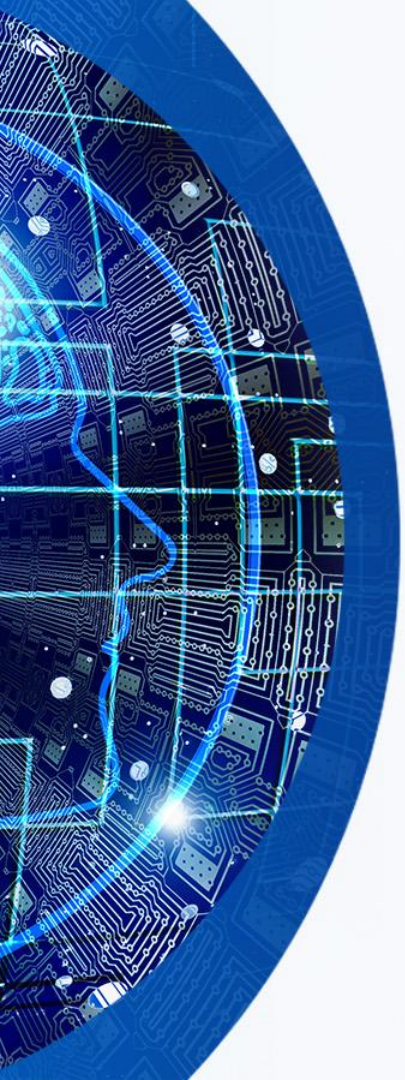


Lecture No #01

01 History of Machine Learning

02 How Machine Learning works

03 Real Life Examples of ML



What is Machine Learning

- Machine learning (ML) is a branch of artificial intelligence (AI) that enables computers to “self-learn” from training data and improve over time, without being explicitly programmed. Machine learning algorithms are able to detect patterns in data and learn from them, in order to make their own predictions. In short, machine learning algorithms and models learn through experience.

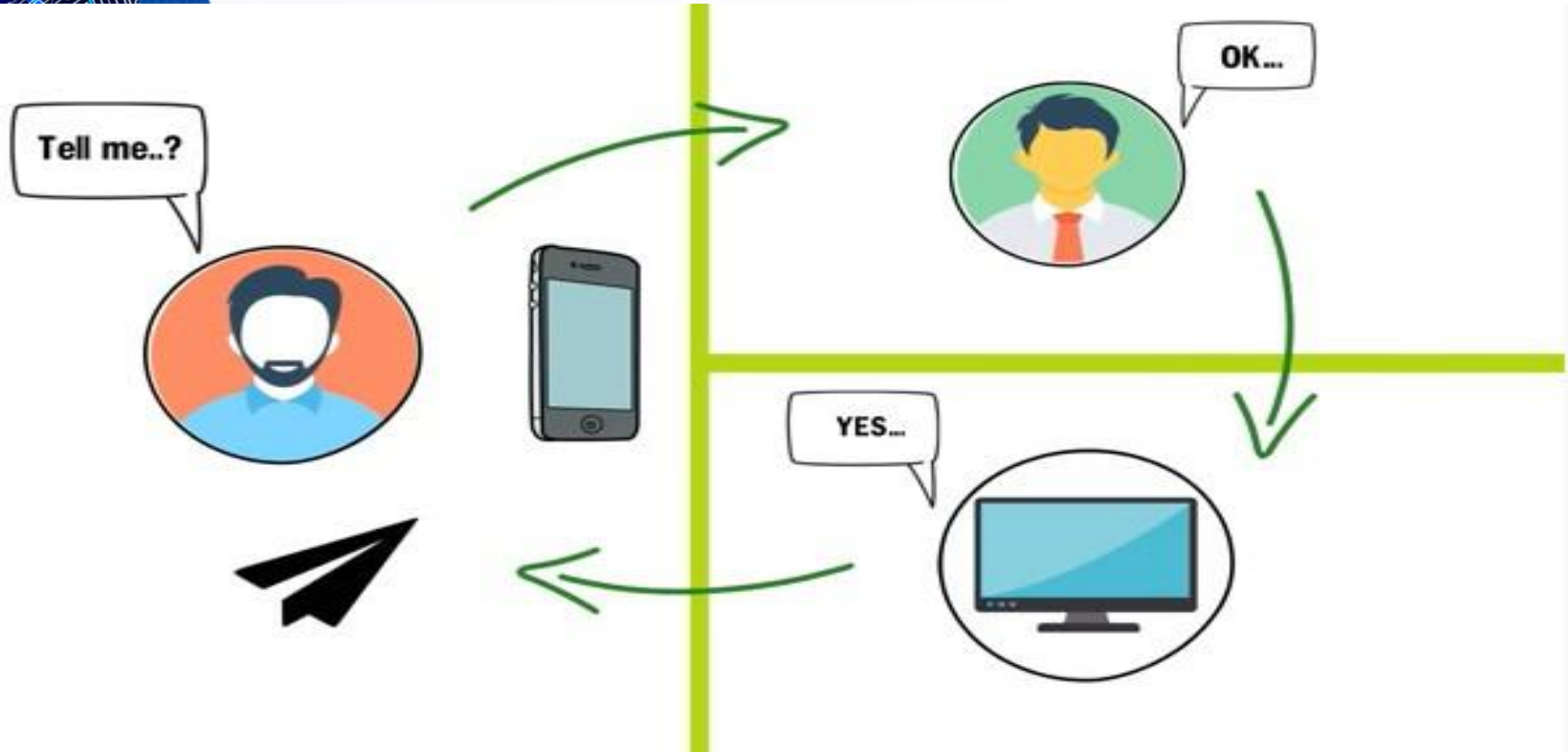
- Machine learning is an application of artificial intelligence that involves algorithms and data that automatically analyse and make decision by itself without human intervention.
- It describes how computer perform tasks on their own by previous experiences.
- Therefore we can say in machine language artificial intelligence is generated on the basis of experience

HISTORY

Alan Turing

1950

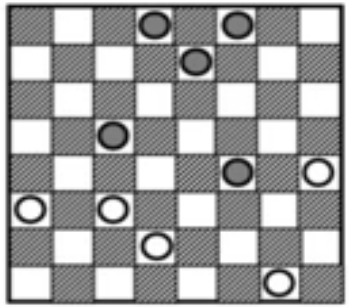
Can machines think like humans?

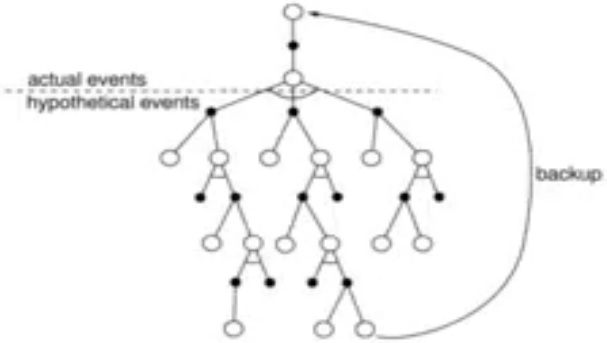

Tell me..?
OK...
YES...

IBM

Arthur Samual

1952

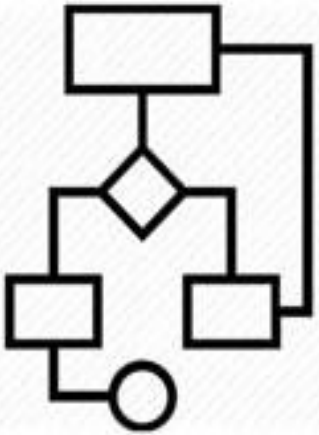




Self learning

Frank Rosenblatt

PERCEPTRON

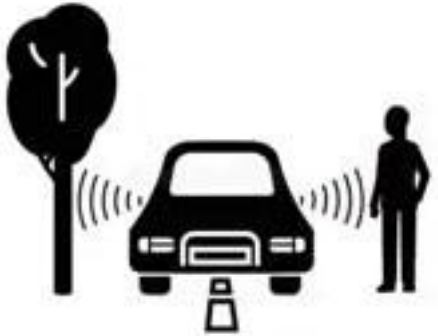


1958

1979

Stanford Cart

Stanford University



NETTALK

1985

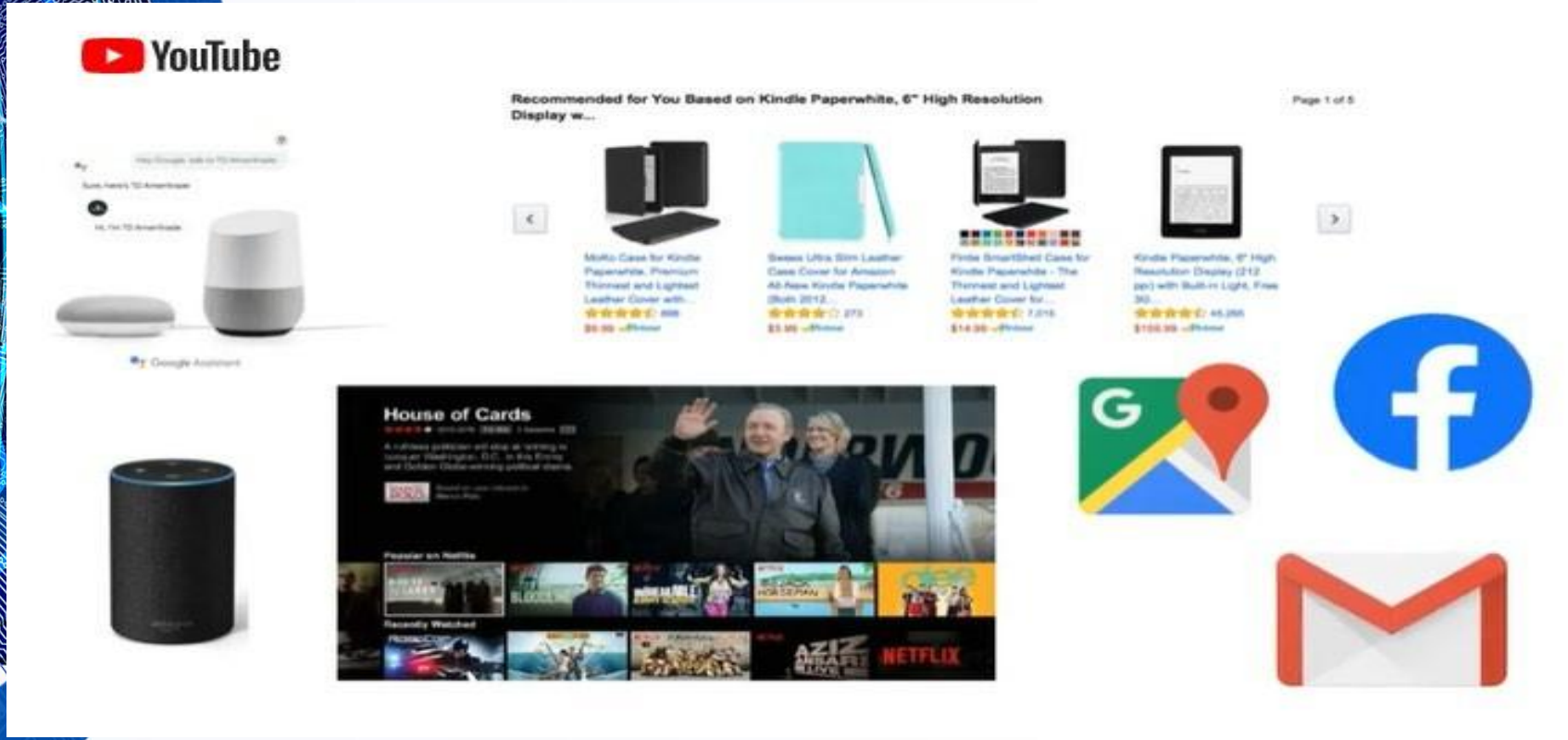
Terry Sejnowski

word pronounce learning


Hey Google, how is my Fidelity portfolio doing?
As of 9 a.m., your Fidelity portfolio is up 1.2%
Google Assistant


Hey!

Daily Life Application Examples for ML



Applications of Machine Learning

1. Image & Video Recognition

- Face recognition (e.g., mobile face unlock)
- Object detection in CCTV and autonomous cars
- Medical image analysis (X-ray, MRI)

2. Natural Language Processing (NLP)

- Chatbots and virtual assistants
- Language translation (Google Translate)
- Spam detection in emails
- Sentiment analysis (reviews, social media)

3. Healthcare

- Disease prediction (diabetes, cancer)
- Personalized treatment recommendations
- Drug discovery and genome analysis

4. Finance

- Fraud detection
- Credit scoring for loans
- Stock market prediction
- Algorithmic trading

5. E-commerce & Marketing

- Product recommendations (Amazon, Netflix)
- Dynamic pricing
- Customer segmentation
- Targeted advertising

6. Transportation

- Self-driving cars
- Traffic prediction and route optimization
- Predictive maintenance of vehicles

7. Manufacturing

- Quality inspection using cameras
- Supply-chain optimization
- Predictive maintenance of machine

8. Agriculture

- Crop disease detection
- Yield prediction
- Soil and weather analysis with sensors

9. Robotics

- Intelligent robots performing tasks
- Industrial automation
- Warehouse robots (Amazon)

10. Cybersecurity

- Detecting malware and intrusions
- Identifying unusual network activities

11. Education

- Personalized learning platforms
- Automatic grading
- Predicting student performance

12. Entertainment

- Music/movie recommendations
- Game AI
- Content personalization

Lecture No #02

01 What is Dataset and Features

02 How Machine Learning works

03 Why is machine learning important?

04 Real Life Examples of ML

Dataset

Outlook	Temperature	Humidity	Wind	Play Tennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Training and Testing Dataset

Outlook	Temperature	Humidity	Wind	Play Tennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Training Dataset

Testing Dataset

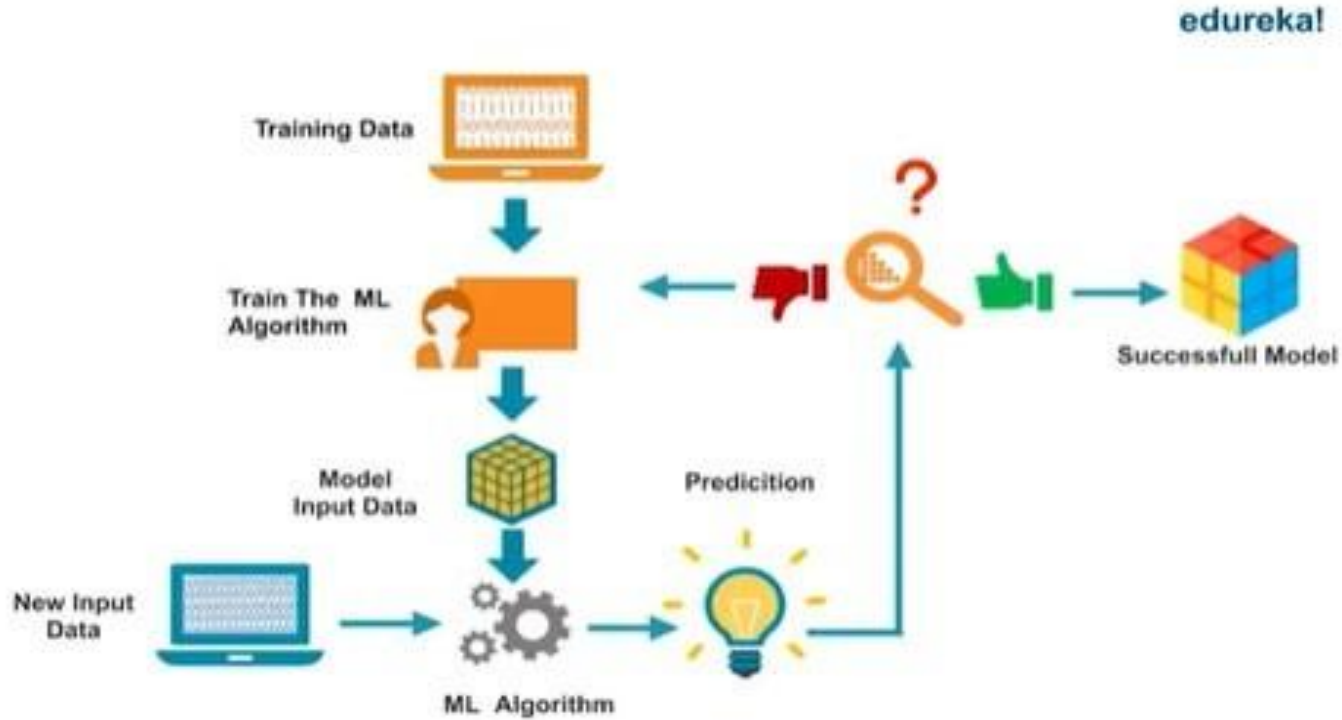
Features

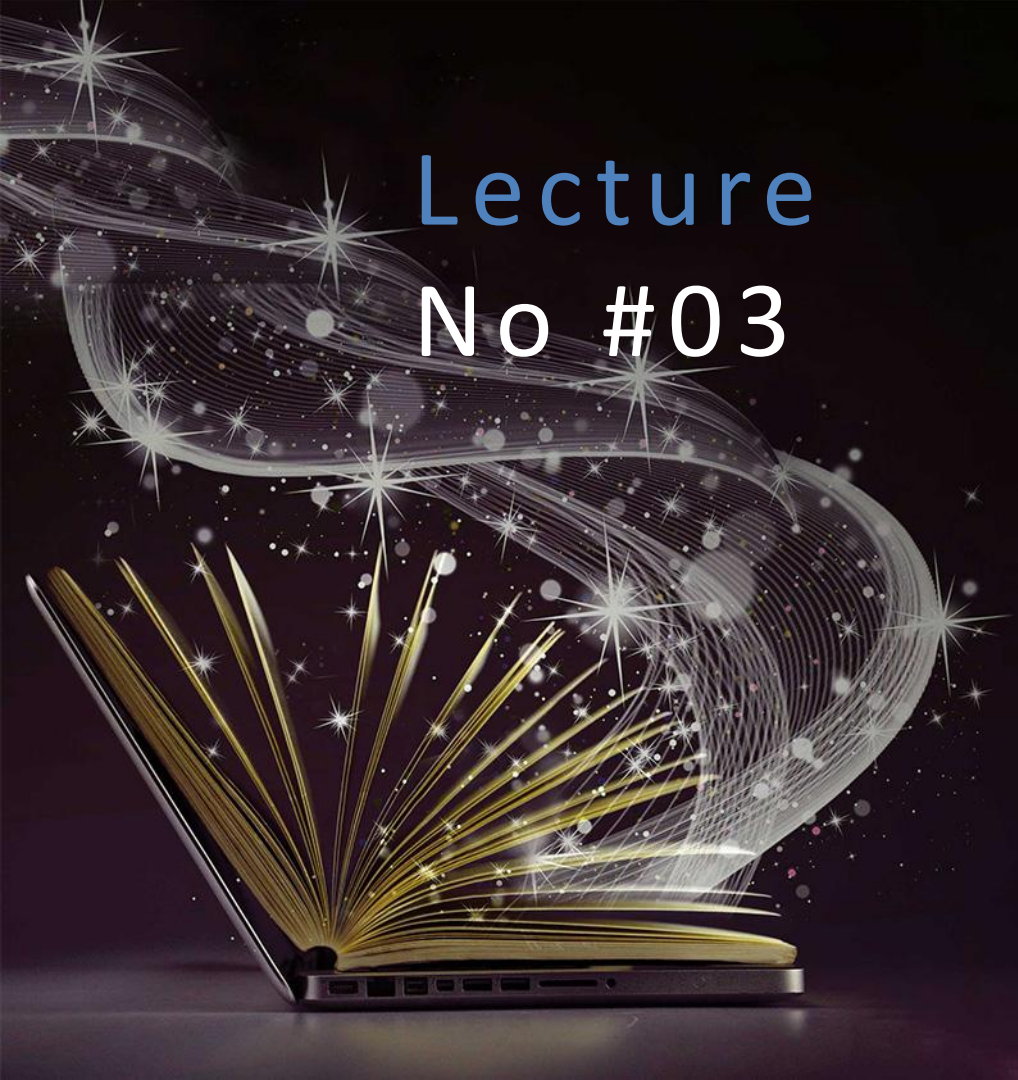
- A feature is a measurable property of the object you're trying to analyze. In datasets, features appear as columns:

Outlook	Temperature	Humidity	Wind	Play Tennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Input Features = [Outlook, Temperature, Humidity, Wind]
Output Feature = [Play Tennis]

How does Machine Learning Work?





Lecture No #03

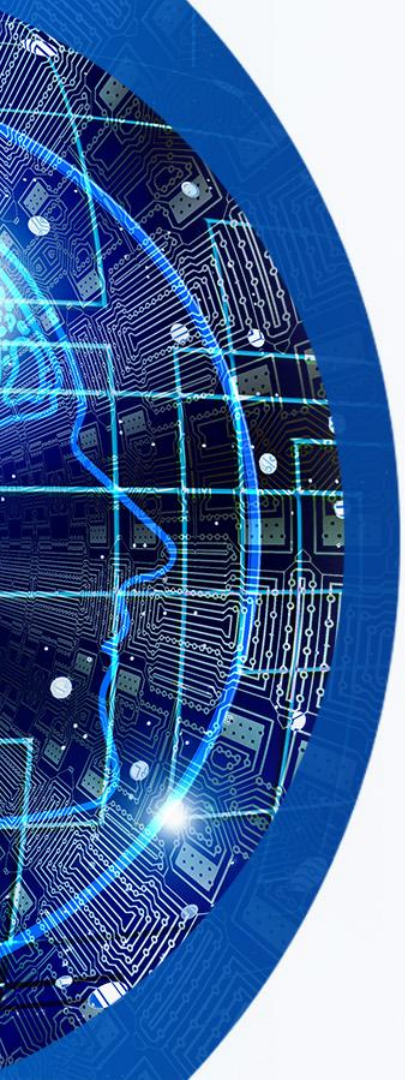
01 How Machine Learning works

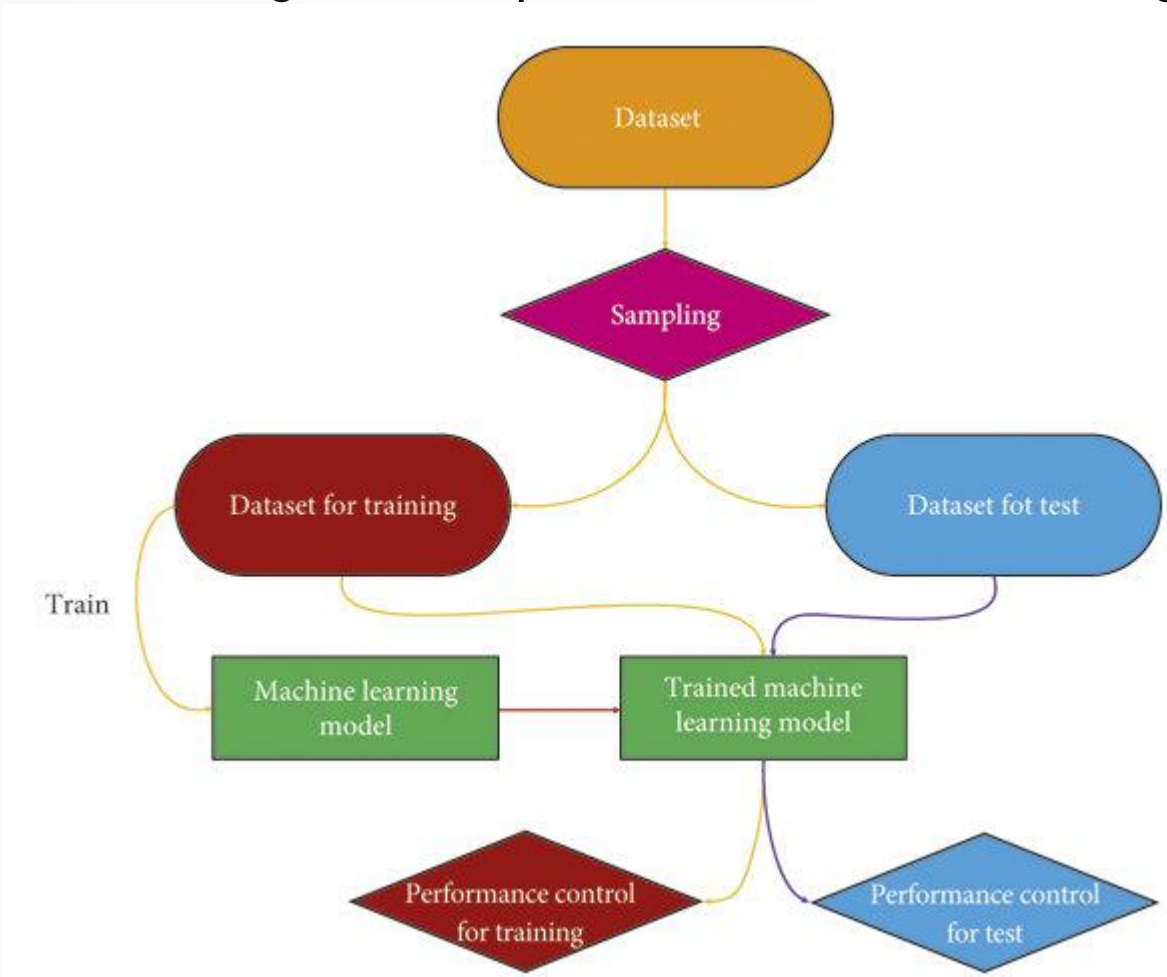
02 Why is machine learning important?

03 Real Life Examples of ML

04 Types of ML

Training and test process in machine learning.





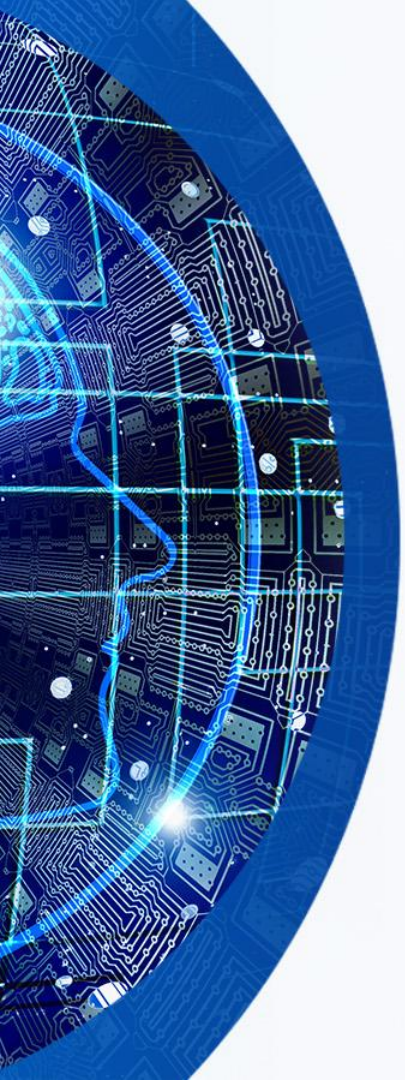
Train, Validation and Test

1. Training set is a set of examples used for learning a model (e.g., a classification model).
2. Validation set is a set of examples that cannot be used for learning the model but can help tune model parameters (e.g., selecting K in K-NN). Validation helps control overfitting.
3. Test set is used to assess the performance of the final model and provide an estimation of the test error.

Example: Split the data randomly into 60% for training, 20% for validation and 20% for testing.

Why is machine learning important?

- Machine learning is important because it gives enterprises a view of trends in customer behavior and business operational patterns, as well as supports the development of new products.
- Many of today's leading companies, such as Facebook, Google and Uber, make machine learning a central part of their operations. Machine learning has become a significant competitive differentiator for many companies.
- Machine Learning is a popular subfield of Artificial Intelligence used in various fields, including healthcare, finance, infrastructure, marketing, self-driving cars, recommendation systems, chatbots, social sites, gaming, cyber security, and others.
- Machine Learning can reduce costs, mitigate risks, and improve quality of life by recommending products/services, detecting cybersecurity breaches, and enabling self-driving cars. It is becoming more common and will soon integrate into many facets of life.

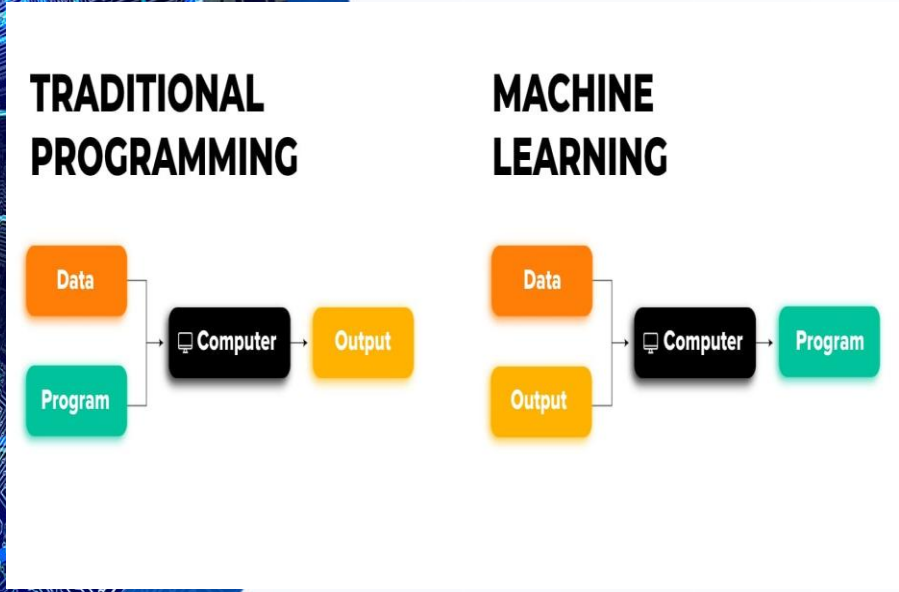


Use cases of Machine Learning Technology

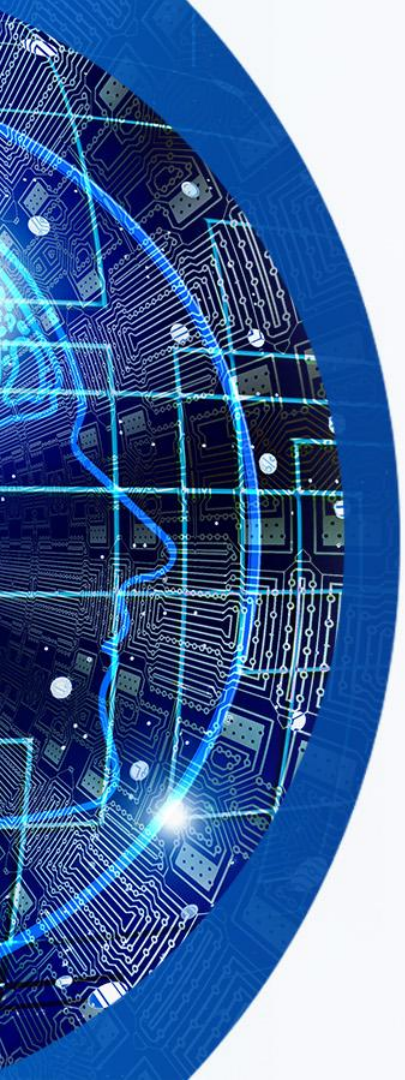
- Healthcare
- Automation
- Banking and Finance
- Transportation and Traffic Prediction
- Image Recognition
- Speech Recognition
- Product Recommendation
- Virtual Personal Assistance
- Email Spam and Malware detection & Filtering
- Self-driving cars
- Credit card fraud detection
- Stock Marketing and Trading
- Language Translation

Reference Link: https://www.javatpoint.com/importance-of-machine-learning

- Traditional programming (rule-based approach). In this case, we define the set of rules that will determine whether the claim is fraudulent or genuine, and then a developer translates them into code. This works great when such rules exist and we know about them.

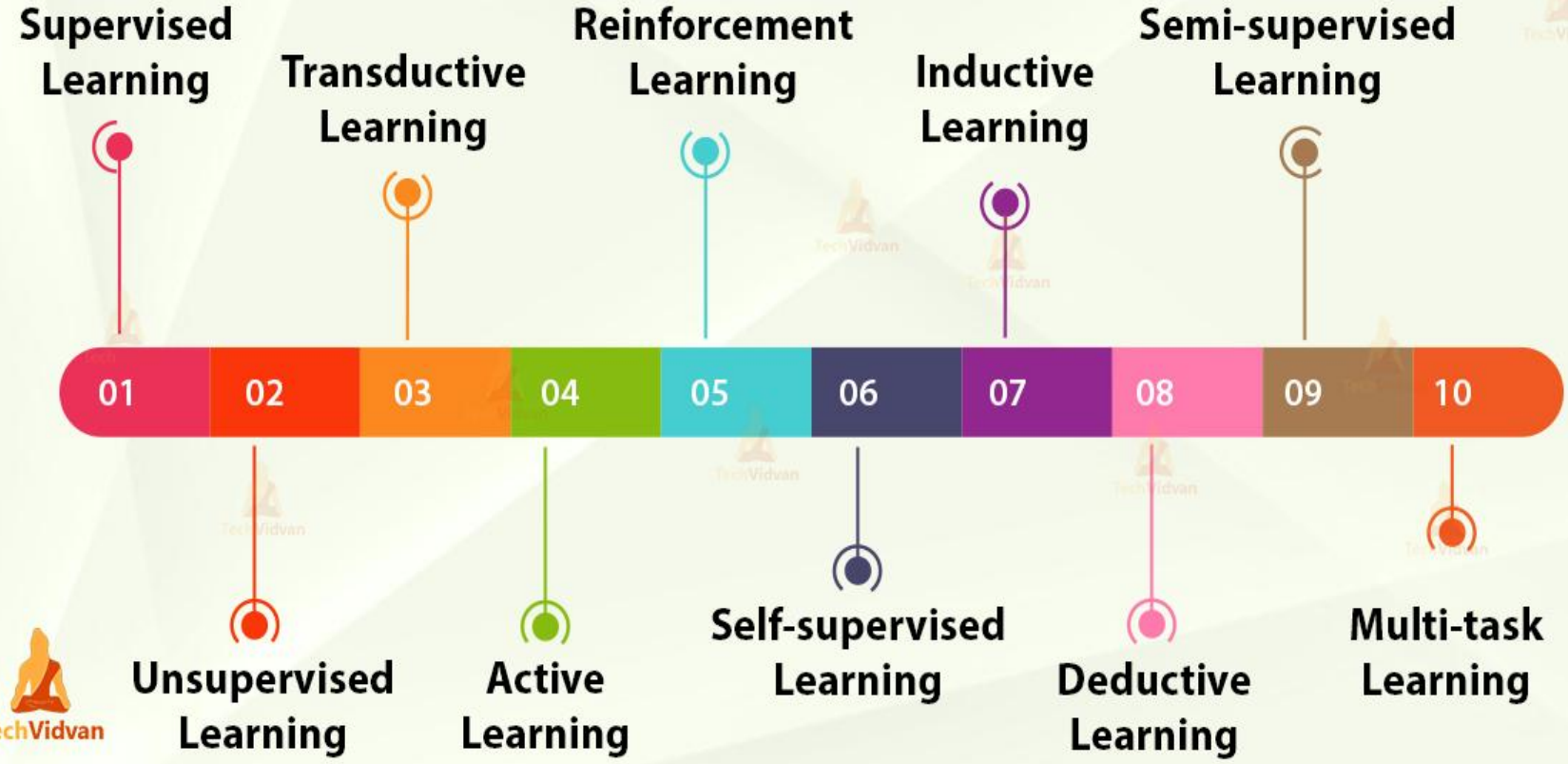


Whereas, Machine Learning algorithms is the ability to find rules using existing examples.

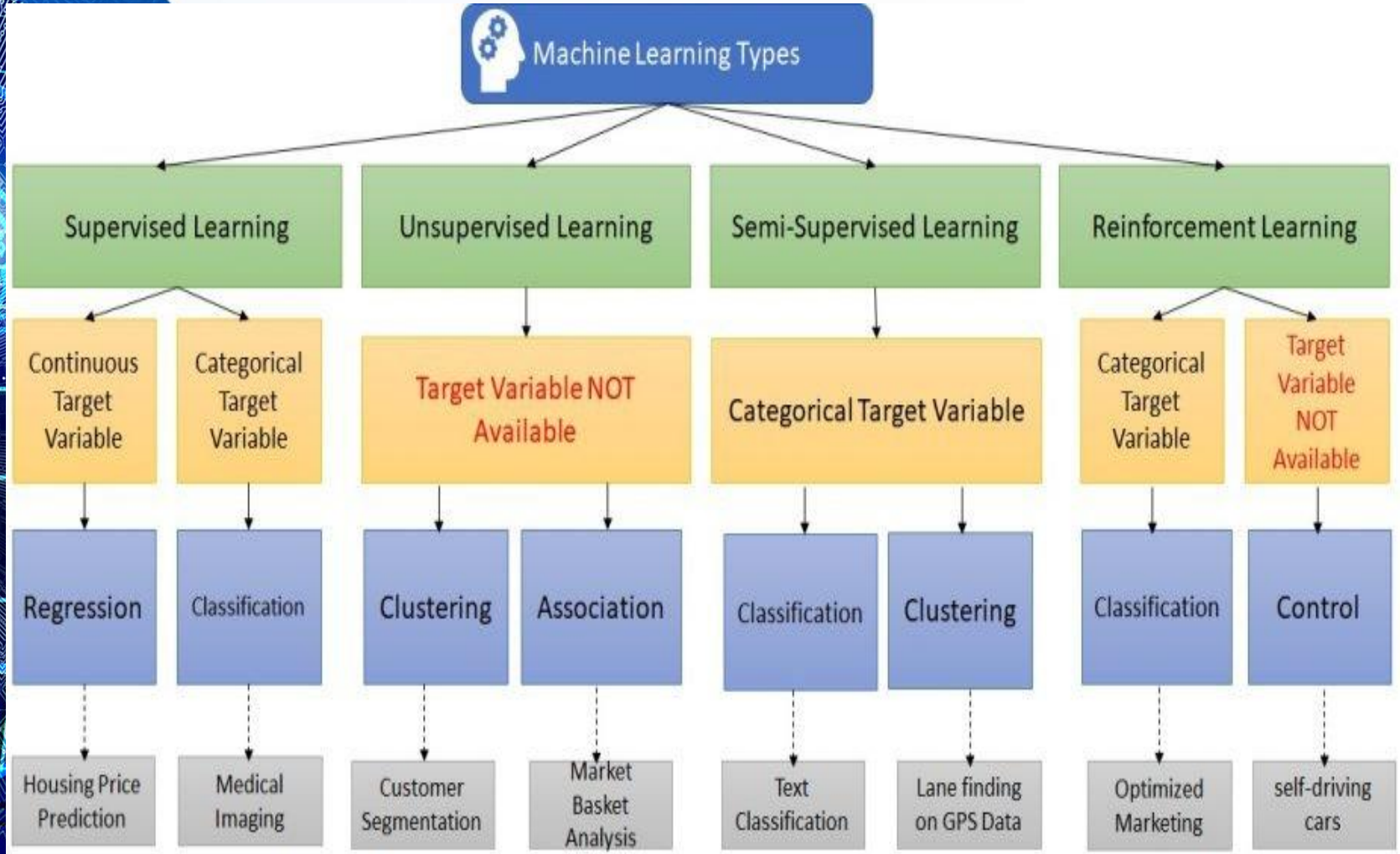


Classification of ML

Types of Machine Learning



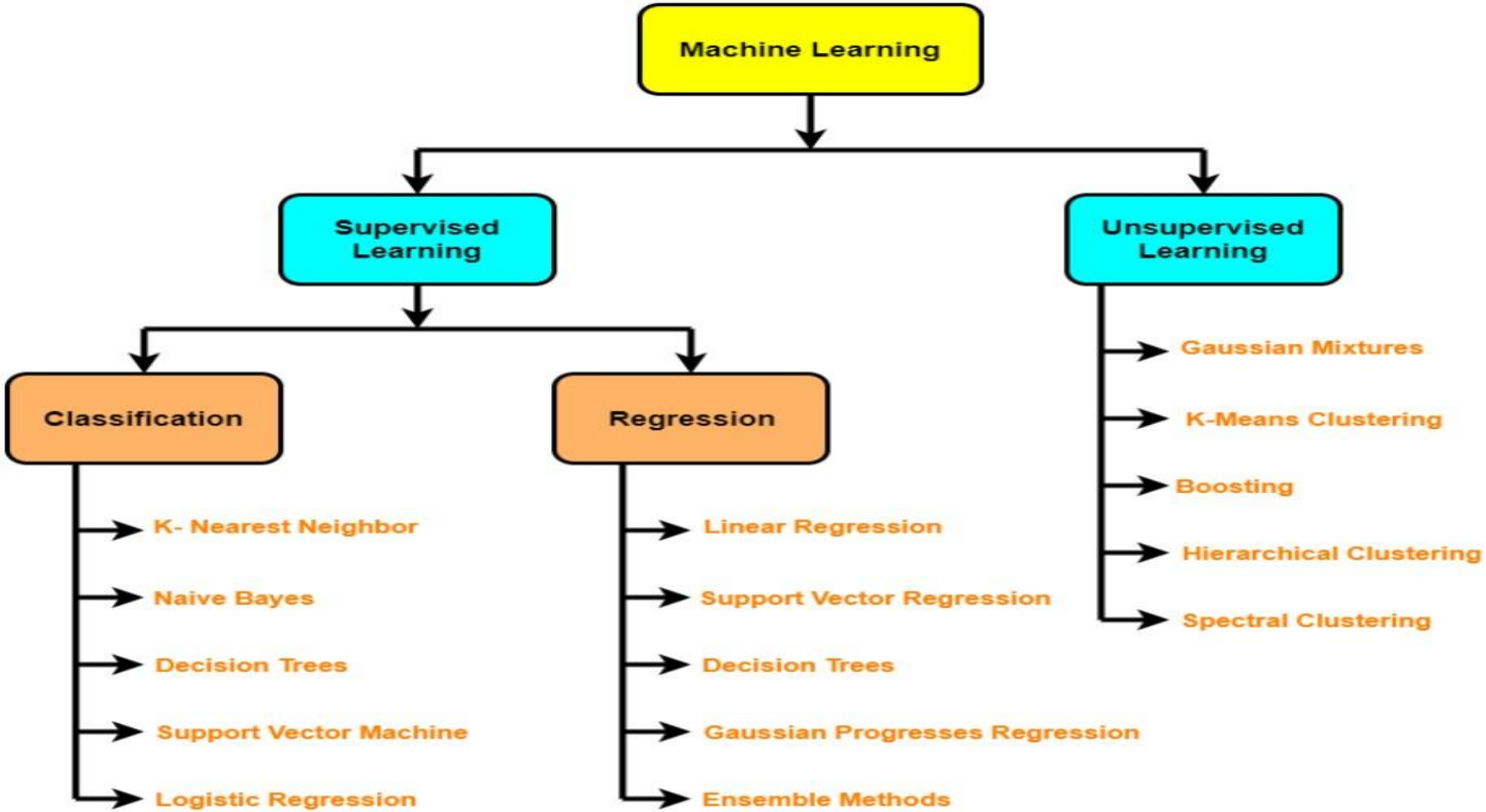
TechVidvan


Machine Learning Types
Supervised Learning
Unsupervised Learning
Semi-Supervised Learning
Reinforcement Learning
Continuous Target Variable
Categorical Target Variable
Target Variable NOT Available
Categorical Target Variable
Categorical Target Variable
Target Variable NOT Available
Regression
Classification
Clustering
Association
Classification
Clustering
Classification
Control
Housing Price Prediction
Medical Imaging
Customer Segmentation
Market Basket Analysis
Text Classification
Lane finding on GPS Data
Optimized Marketing
self-driving cars

Lecture No #04

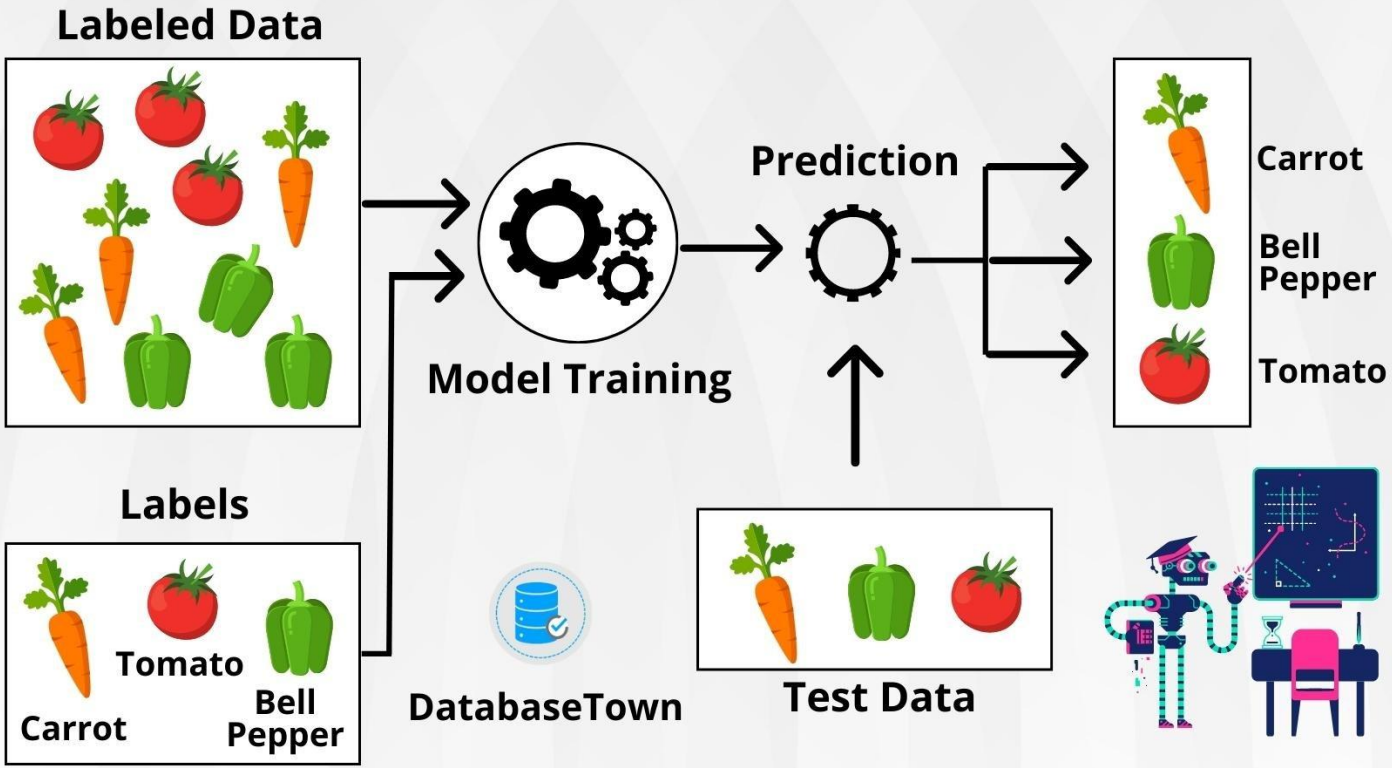
01 Classification of ML

02 Performances Matrices in ML


Machine Learning
Supervised Learning
Unsupervised Learning
Classification
Regression
K- Nearest Neighbor
Naive Bayes
Decision Trees
Support Vector Machine
Logistic Regression
Linear Regression
Support Vector Regression
Decision Trees
Gaussian Progresses Regression
Ensemble Methods
Gaussian Mixtures
K-Means Clustering
Boosting
Hierarchical Clustering
Spectral Clustering

SUPERVISED LEARNING

Supervised machine learning is a branch of artificial intelligence that focuses on training models to make predictions or decisions based on labeled training data.



1. SUPERVISED LEARNING

WEIGHT
HEIGHT
COLORS
NAMES

FEATURES & LABELS

FEATURES & LABELS

WEIGHT : 100G
HEIGHT : 5 INCH
COLOR : YELLOW

→ FEATURES

NAME : MANGO

→ LABEL

Labeled Dataset

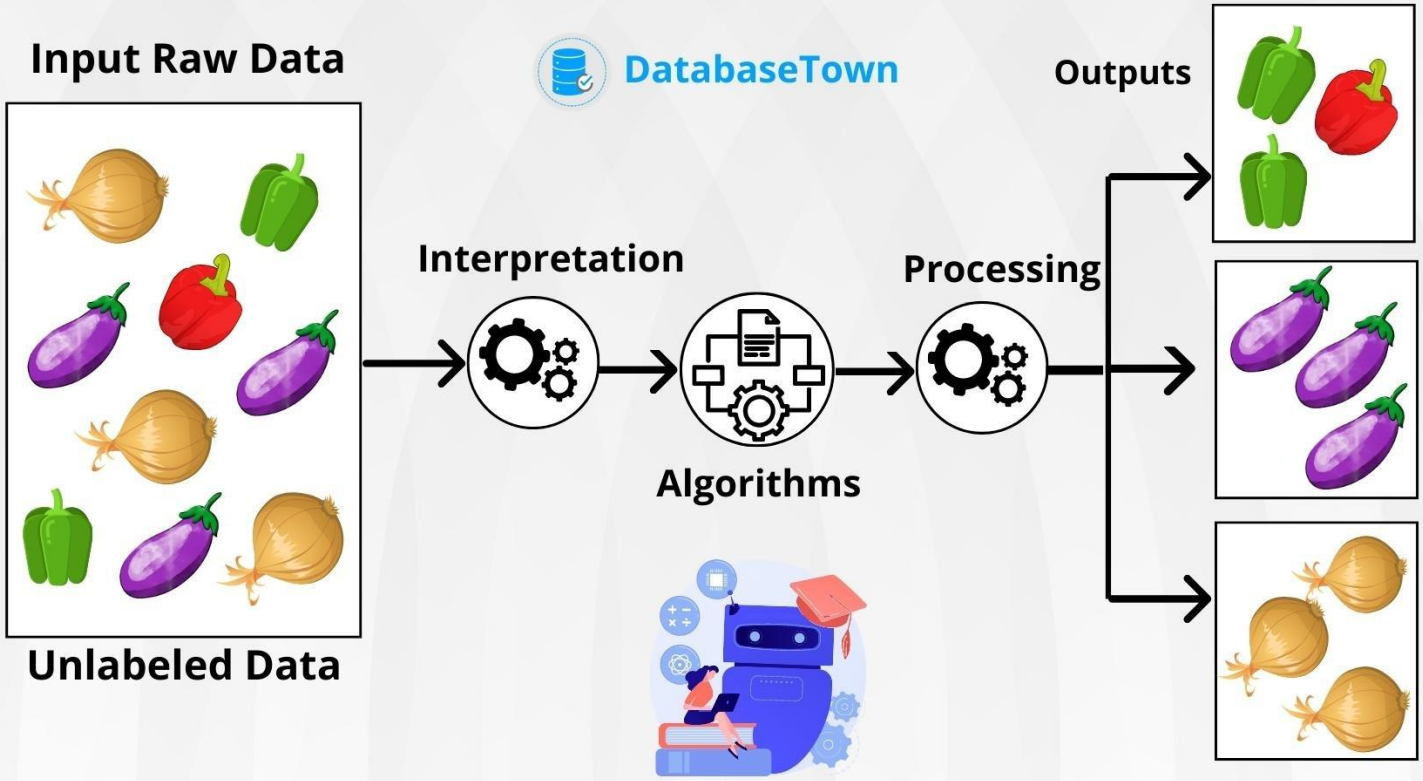
Features					Label
Position	Experience	Skill	Country	City	Salary ($)
Developer	0	1	USA	New York	103100
Developer	1	1	USA	New York	104900
Developer	2	1	USA	New York	106800
Developer	3	1	USA	New York	108700
Developer	4	1	USA	New York	110400
Developer	5	1	USA	New York	112300
Developer	6	1	USA	New York	114200
Developer	7	1	USA	New York	116100
Developer	8	1	USA	New York	117800
Developer	9	1	USA	New York	119700
Developer	10	1	USA	New York	121600

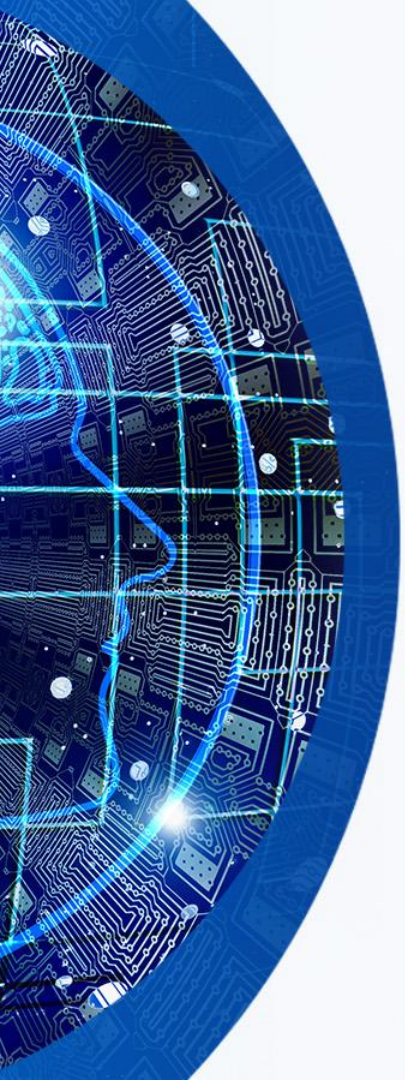
Applications of Supervised Learning

- **Image Segmentation:** Supervised Learning algorithms are used in image segmentation. In this process, image classification is performed on different image data with pre-defined labels.
- **Medical Diagnosis:** Supervised algorithms are also used in the medical field for diagnosis purposes. It is done by using medical images and past labelled data with labels for disease conditions. With such a process, the machine can identify a disease for the new patients.
- **Fraud Detection -** Supervised Learning classification algorithms are used for identifying fraud transactions, fraud customers, etc. It is done by using historic data to identify the patterns that can lead to possible fraud.
- **Spam detection -** In spam detection & filtering, classification algorithms are used. These algorithms classify an email as spam or not spam. The spam emails are sent to the spam folder.
- **Speech Recognition -** Supervised learning algorithms are also used in speech recognition. The algorithm is trained with voice data, and various identifications can be done using the same, such as voice-activated passwords, voice commands, etc.

UNSUPERVISED LEARNING

Unsupervised learning is a type of machine learning where the algorithm learns from unlabeled data without any predefined outputs or target variables.



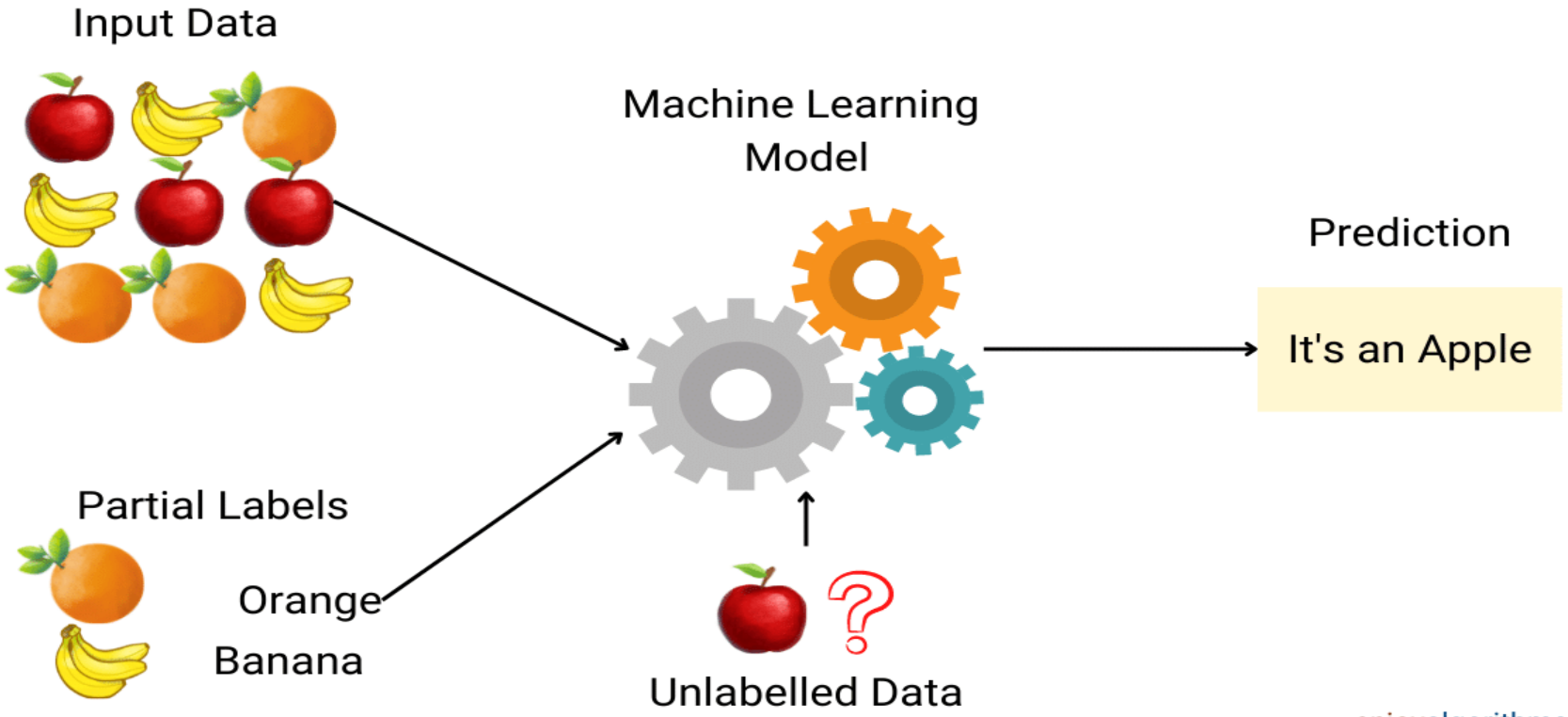


cyl	hp	wt	gear	mpg
6	110	2.620	4	MISSING
6	110	2.875	4	MISSING
4	93	2.320	4	MISSING
6	110	3.215	3	MISSING
8	175	3.440	3	MISSING
6	105	3.460	3	MISSING
8	245	3.570	3	MISSING
4	62	3.190	4	MISSING
4	95	3.150	4	MISSING
6	123	3.440	4	MISSING

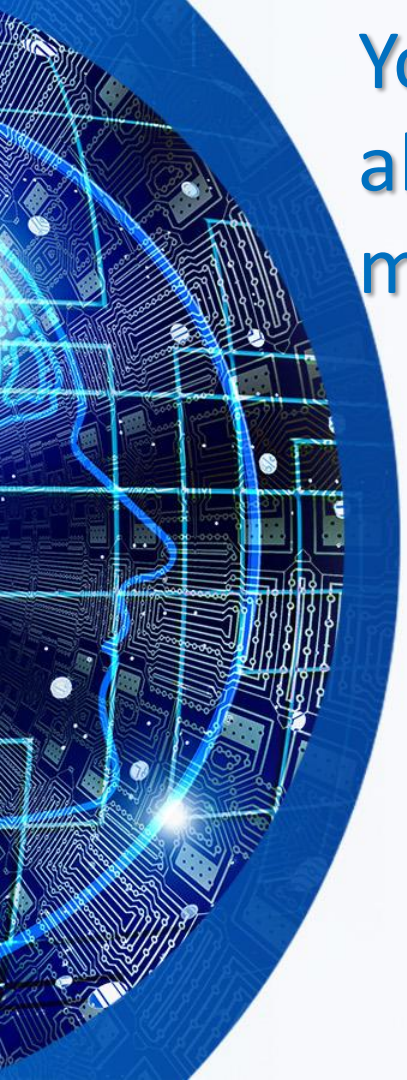
Unsupervised Learning Application

- Applications of Unsupervised Learning
- **Network Analysis:** Unsupervised learning is used for identifying plagiarism and copyright in document network analysis of text data for scholarly articles.
- **Recommendation Systems:** Recommendation systems widely use unsupervised learning techniques for building recommendation applications for different web applications and e-commerce websites.
- **Anomaly Detection:** Anomaly detection is a popular application of unsupervised learning, which can identify unusual data points within the dataset. It is used to discover fraudulent transactions.
- **Singular Value Decomposition:** Singular Value Decomposition or SVD is used to extract particular information from the database. For example, extracting information of each user located at a particular location.

Semisupervised Learning



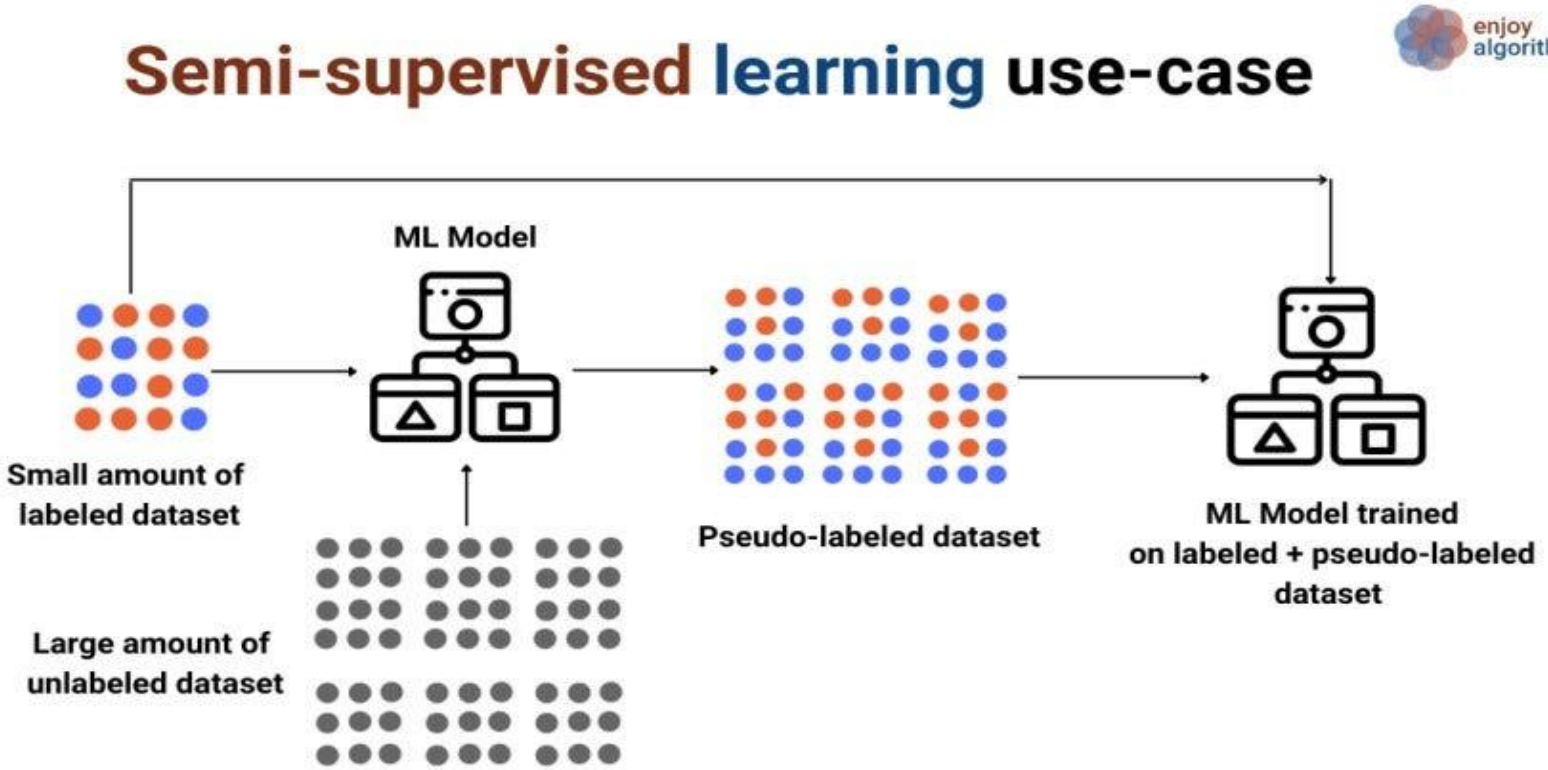
Name	Loan Amount	Loan Repaid	Fraud
Ashley	100000	1	1
Chuck	25000	0	0
Tim	4000	1	1
Mike	150000	1	1
Colin	200000000	0	
Libby	400400	1	0
Sheila	3200	1	1
Mandi	34850	1	
Gareth	6570	0	0

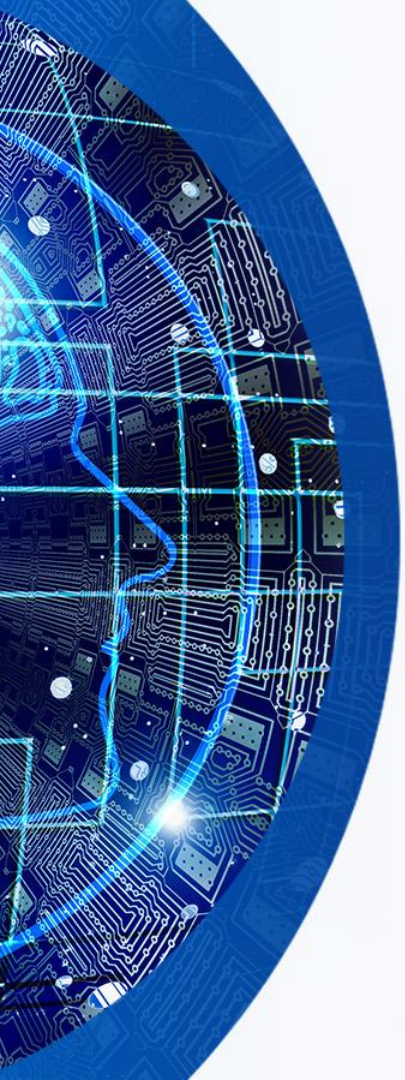


You can use a semi-supervised learning algorithm to label the data, and retrain the model with the newly labeled dataset:

Name	Loan Amount	Loan Repaid	Fraud
Ashley	100000	1	1
Chuck	25000	0	0
Tim	4000	1	1
Mike	150000	1	1
Colin	200000000	0	0
Libby	400400	1	0
Sheila	3200	1	1
Mandi	34850	1	1
Gareth	6570	0	0

Semisupervised Learning

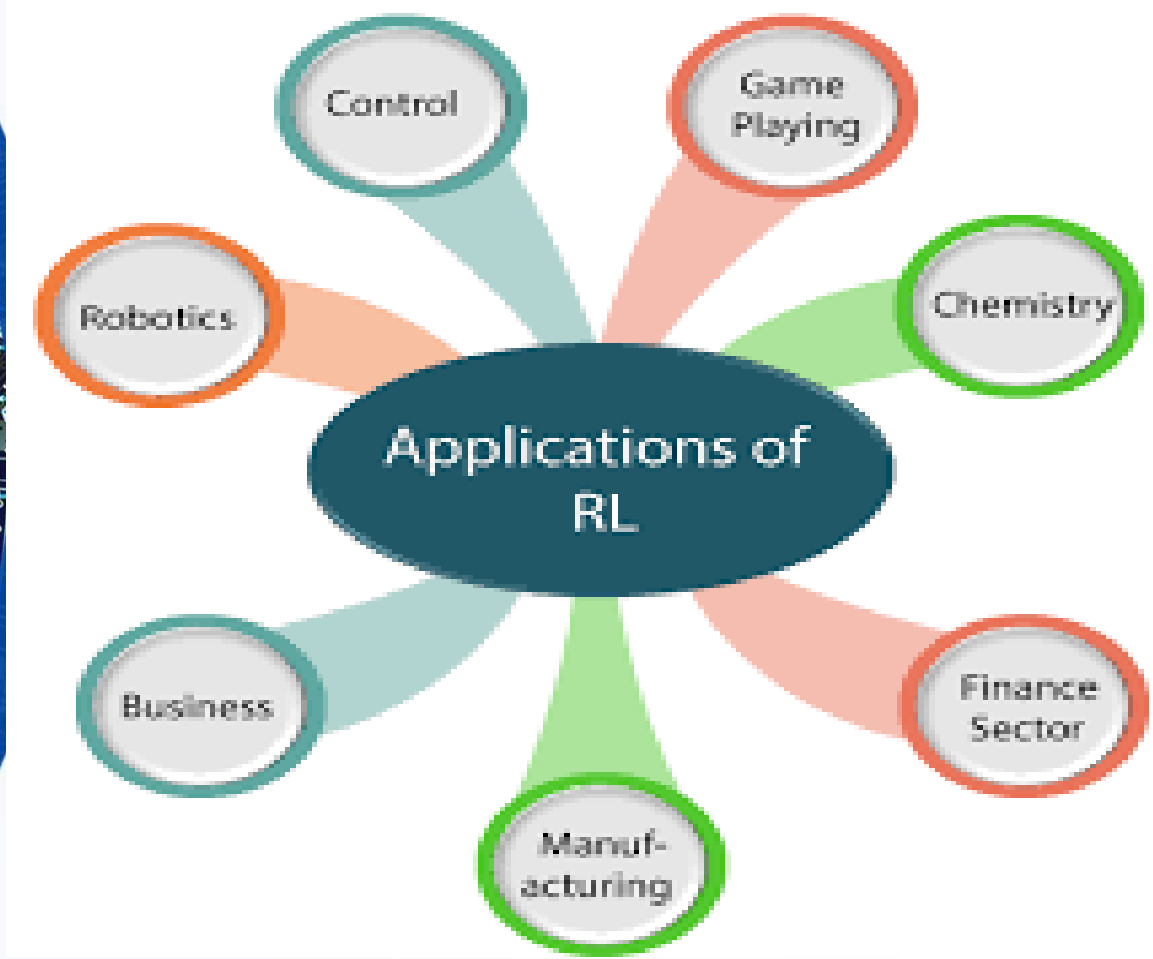


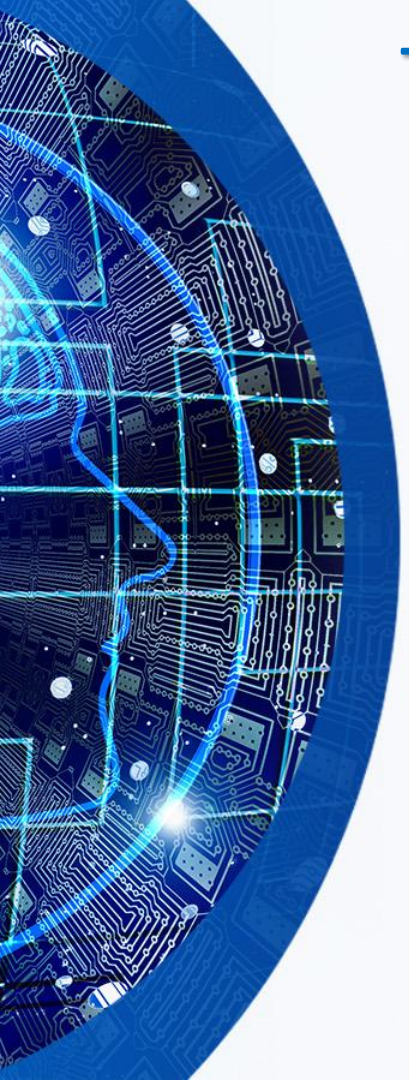


Semi-Supervised Application

- Image and Speech Analysis: This is the most popular example of semi-supervised learning models. Images and audio files are usually not labeled. To label them is an arduous task that is expensive as well. With the help of human expertise, you can label a small data set. Once the data is trained, we can then implement SSL to label the rest of the audio and Image files and thus improve Image and speech analytic models.
- Web Content Classification: There are billions of websites on the internet with different classified content. To make this information available to web users requires a vast team of human resources who can organize and classify the content on the web pages. SSL can help by labeling the content and classifying it, thus improving the user experience. Many search engines, including Google, use a semi-supervised learning model to label and rank web pages in their search result.

Reinforcement Learning in ML

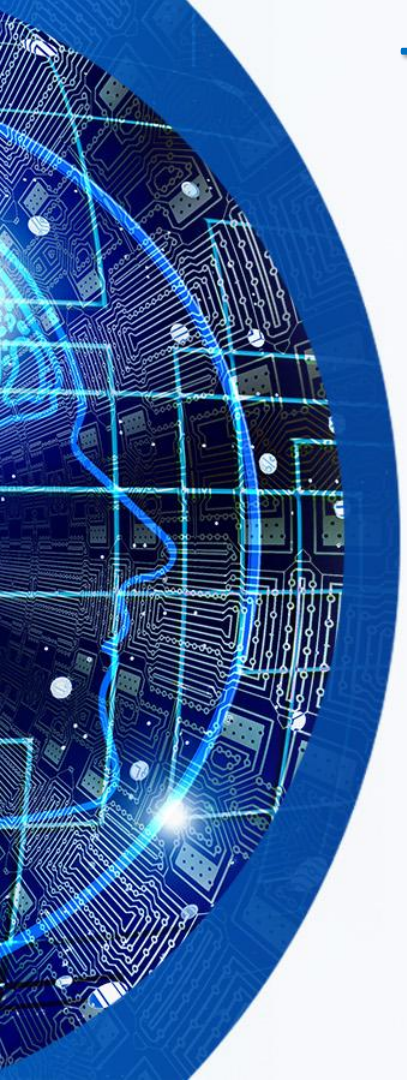

Control
Game Playing
Robotics
Chemistry
Applications of RL
Business
Finance Sector
Manuf-acturing



Types of Reinforcement Learning

Positive Reinforcement Learning

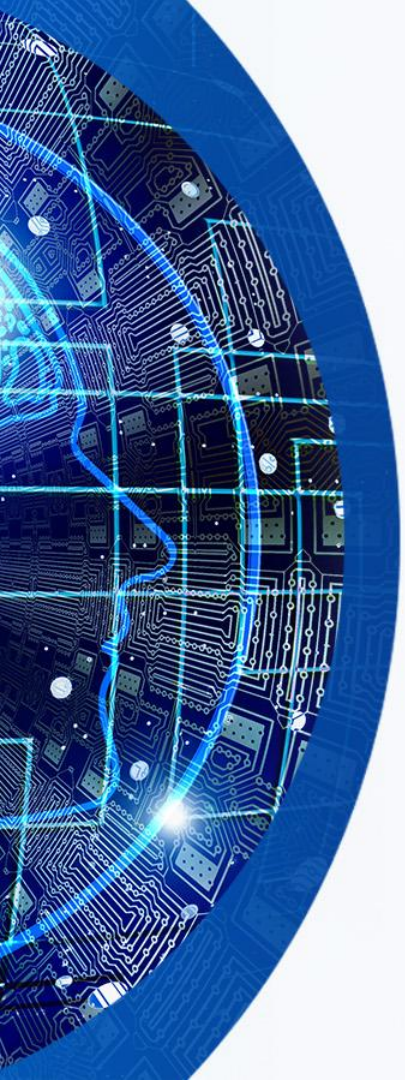
- In this type of RL, the algorithm receives a type of reward for a certain result.
- In other words, here we try to add a reward for every good result in order to increase the likelihood of a good result.
- Like, the parents promise to give the child something that he or she loves like chocolate. This rather has a good impact as it automatically makes the child work as they think of the reward. In this learning, we are adding a good reward to increase the likelihood of task completion.



Types of Reinforcement Learning

Negative Reinforcement Learning

- This RL Type is a bit different from positive RL. Here, we try to remove something negative in order to improve performance.
- We can take the same child-parent example here as well. Some parents punish kids for not cleaning their rooms.
- The punishment can be no video games for one week or sometimes a month. To avoid the punishment the kids often work harder or complete the job assigned to them.

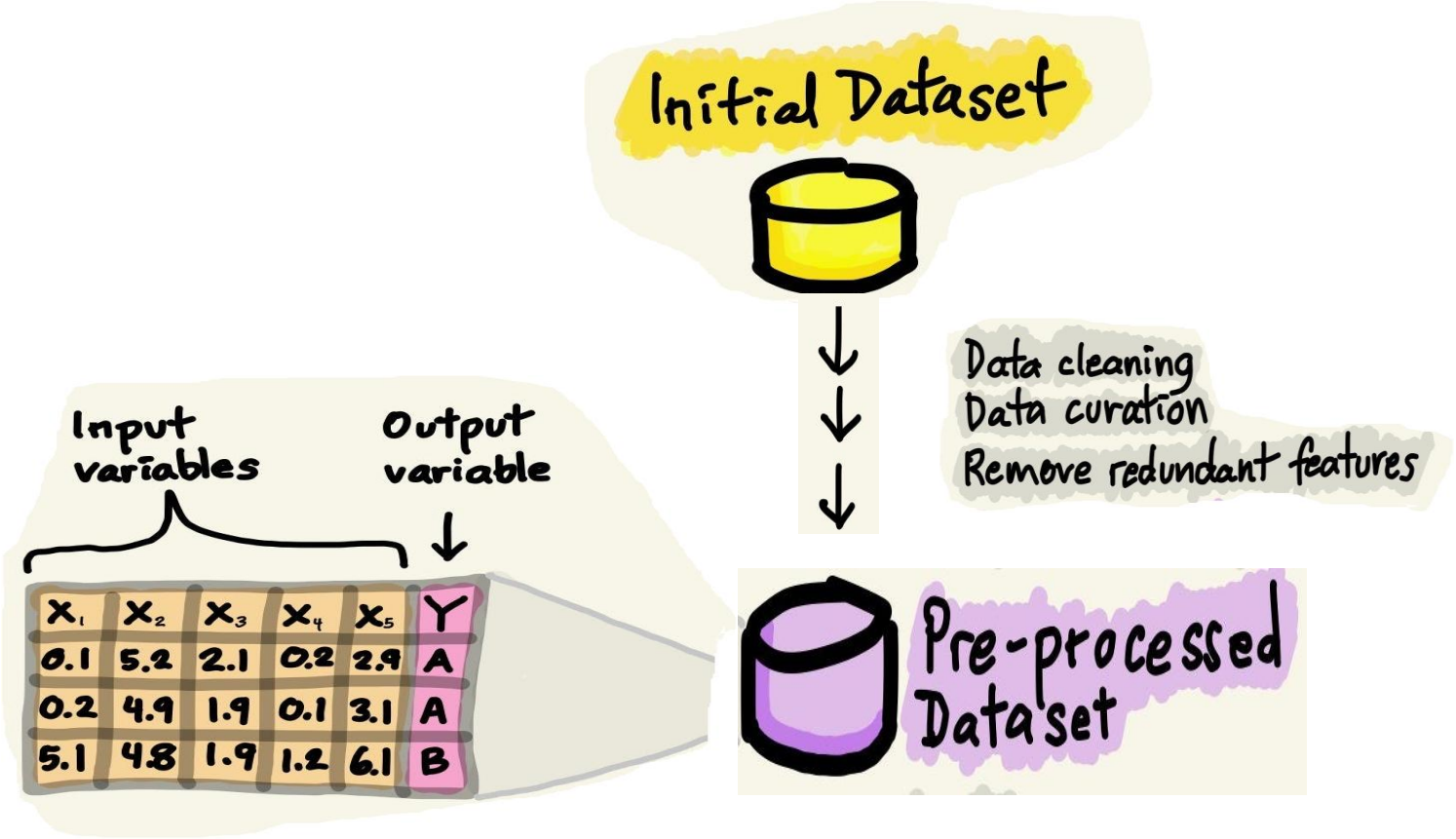


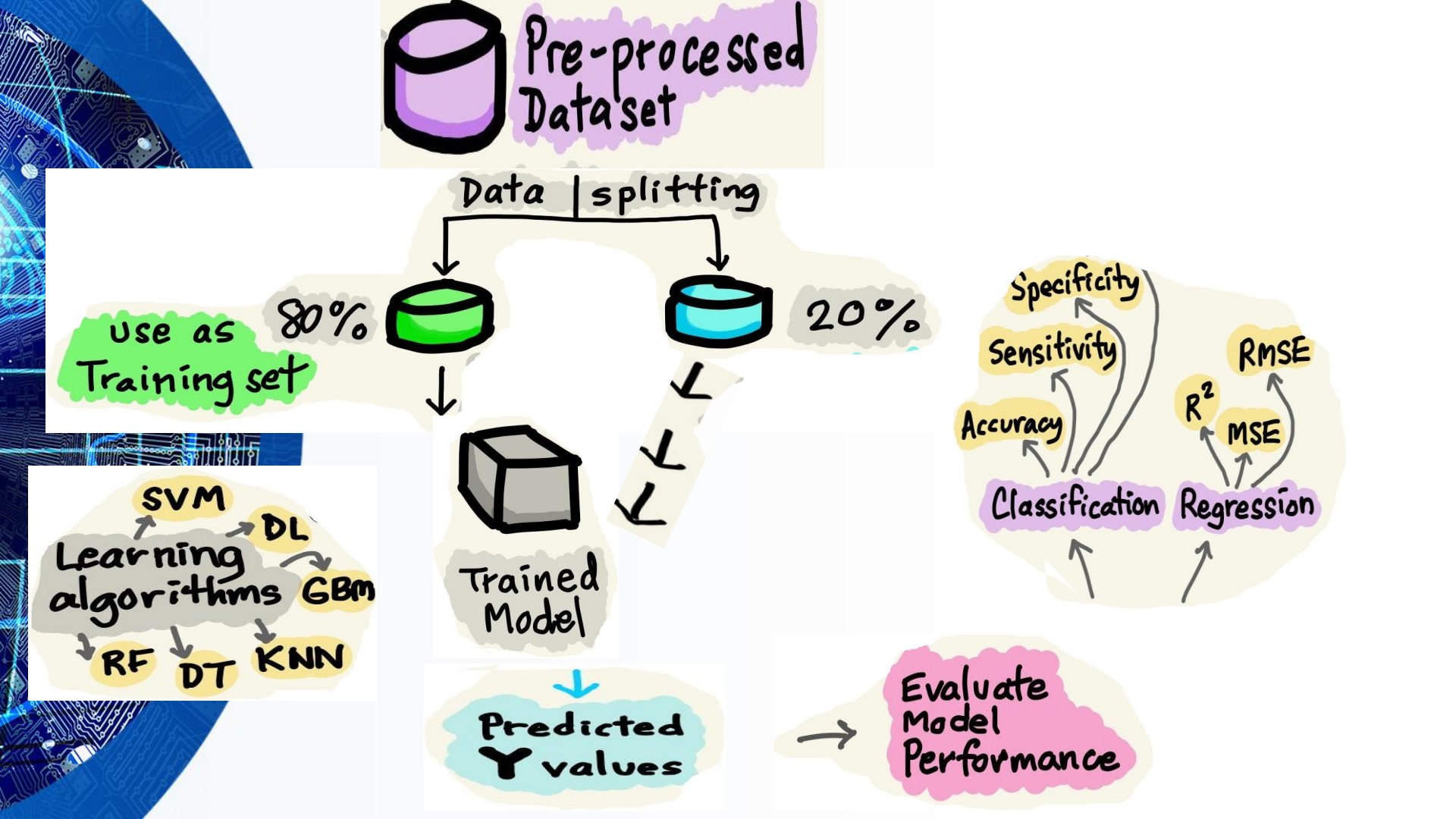
ID	Color	Weight	Broken	Class
1	Black	80	Yes	1
2	Yellow	100	No	2
3	Yellow	120	Yes	2
4	Blue	90	No	2
5	Blue	85	No	2
6	**?**	60	No	1
7	Yellow	100	**?**	2
8	**?**	40	**?**	1

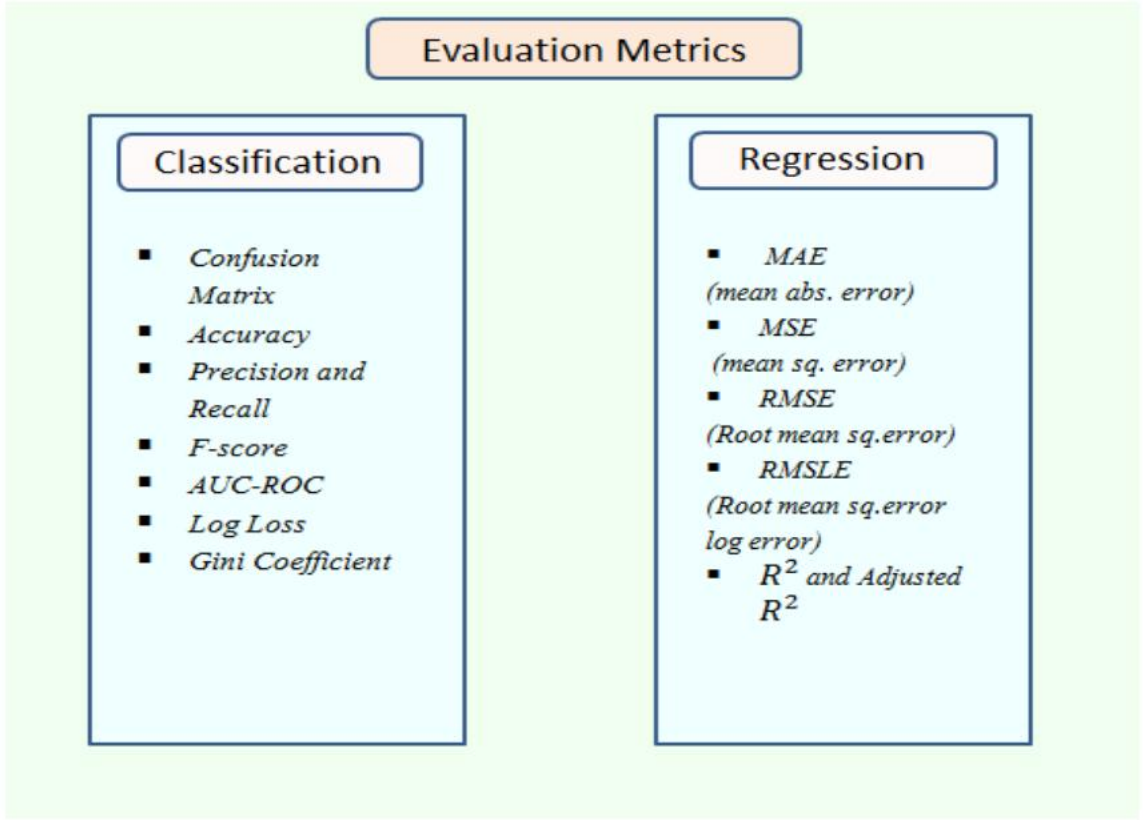
So, You want to build one ML Model

Name	Loan Amount	Loan Repaid	Fraud
Ashley	100000	1	1
Chuck	25000	0	0
Tim	4000	1	1
Mike	150000	1	1
Colin	200000000	0	
Libby	400400	1	0
Sheila	3200	1	1
Mandi	34850	1	
Gareth	6570	0	0

So, You want to build one ML Model




Pre-processed Dataset
Data splitting
80%
20%
use as Training set
Trained Model
Predicted Y values
Evaluate Model Performance
Learning algorithms
SVM
DL
GBm
RF
DT
KNN
Specificity
Sensitivity
Accuracy
Classification
RMSE
R^2
MSE
Regression


Evaluation Metrics
Classification
Confusion Matrix
Accuracy
Precision and Recall
F-score
AUC-ROC
Log Loss
Gini Coefficient
Regression
MAE
(mean abs. error)
MSE
(mean sq. error)
RMSE
(Root mean sq.error)
RMSLE
(Root mean sq.error log error)
R^2 and Adjusted R^2

Performance Metrics in Machine Learning for classification

- To check the performance of our machine learning or deep learning model we will discuss terms like:

1. Accuracy
2. Precision
3. Recall
4. F1 Score
5. Confusion matrix

-

Accuracy

True Positive = 4

False Positive = 3

Truth									
Prediction	Dog	Dog	Dog	No Dog	Dog	No Dog	Dog	No Dog	Dog	Dog
			✗		✓		✗		

True Negative = 1

False Negative = 2

How many we got right? → 5

$$\frac{TP + TN}{TP + FP + TN + FN}$$

Accuracy → 5/10 → **0.5**

Accuracy

- Accuracy is a metric used in classification problems used to tell the percentage of accurate predictions. We calculate it by dividing the number of correct predictions by the total number of predictions.

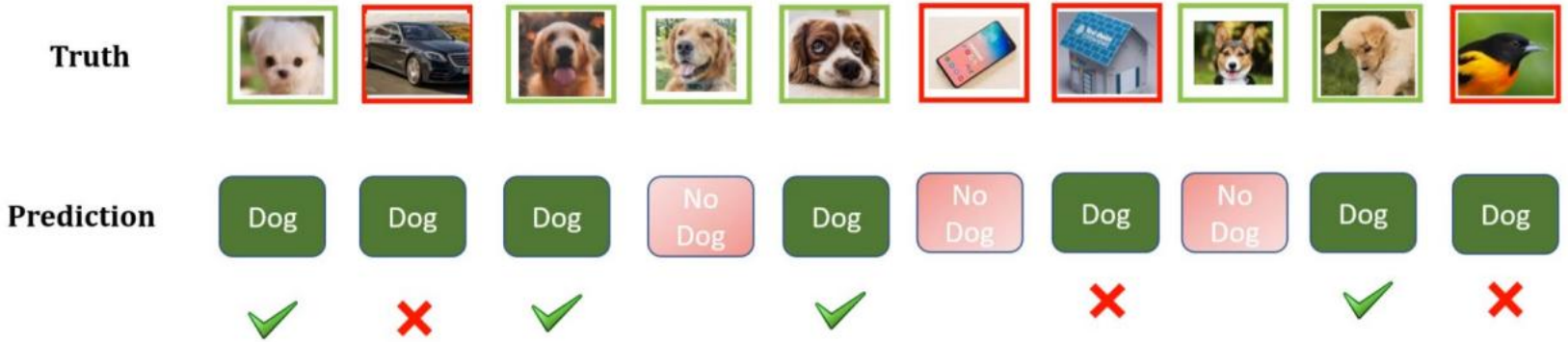
$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions}$$

$$\frac{TP + TN}{TP + FP + TN + FN}$$

Performance Metrics in Machine Learning

- True positives (TP): Predicted positive and are actually positive.
- False positives (FP): Predicted positive and are actually negative.
- True negatives (TN): Predicted negative and are actually negative.
- False negatives (FN): Predicted negative and are actually positive.

Precision



True Positive = 4

False Positive = 3

Precision is out of all dog predictions how many you got it right?

Precision = 4 / 7 = 0.57

Precision = TP / (TP + FP)

Precision

- Percentage of positive instances out of the total predicted positive instances. Here denominator is the model prediction done as positive from the whole given dataset. Take it as to find out 'how much the model is right when it says it is right'.

$$\frac{TP}{TP + FP}$$

RECALL

Recall is out of all dog truth how many you got it right?

Total Dog truth samples = 6

True Positive = 4

Recall = 4 / 6 = 0.67

Recall = TP / (TP + FN)

RECALL

- Percentage of positive instances out of the total actual positive instances. Therefore denominator (TP + FN) here is the actual number of positive instances present in the dataset. Take it as to find out 'how much extra right ones, the model missed when it showed the right ones'.

$$\frac{TP}{TP + FN}$$

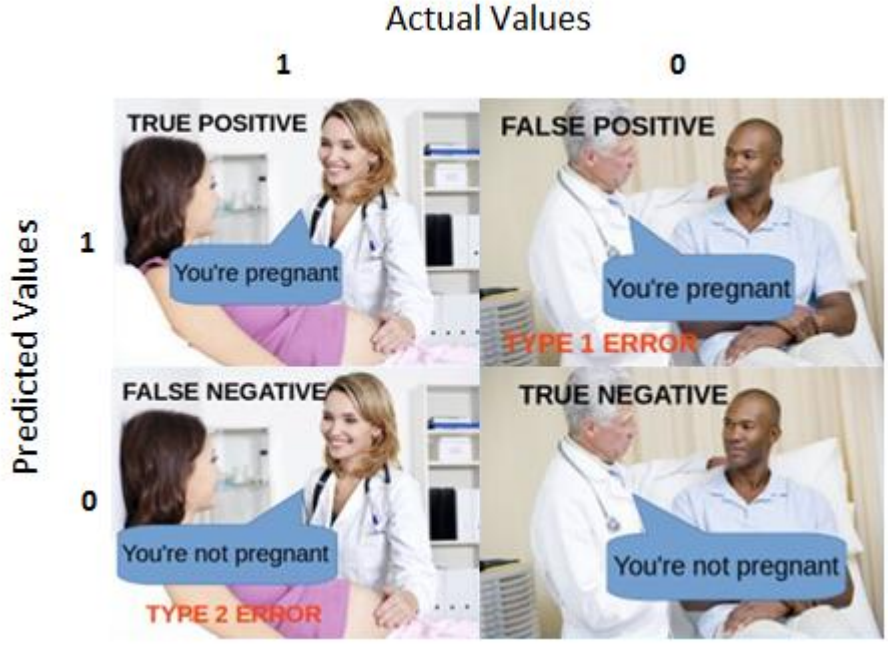
F1 score

- It is the harmonic mean of precision and recall. This takes the contribution of both, so higher the F1 score, the better. See that due to the product in the numerator if one goes low, the final F1 score goes down significantly. So a model does well in F1 score if the positive predicted are actually positives (precision) and doesn't miss out on positives and predicts them negative (recall).

$$\frac{2}{\frac{1}{precision}+\frac{1}{recall}} = \frac{2 * precision * recall}{precision + recall}$$

Confusion Matrix

Predicted Values \ Actual Values	Positive (1)	Negative (0)
Positive (1)	TP	FP
Negative (0)	FN	TN



Q3. Given an AI model to test the collected test sample for covid-positive and covid-negative classes, having the following confusion matrix. Evaluate the model as per its ac curacy, Precision, recall and f-measure.

N=100	Predicted No	Predicted Yes
Actual No 60	50	10
Actual Yes 40	5	35

A hospital uses an AI model to detect a particular disease.

Out of **300 patients**, the following information is observed:

- **70 patients actually had the disease.**
- The model predicted **65 patients as positive**, out of which **55 were truly sick** (correct positive predictions).
- Among the **235 patients predicted as negative**, **15 were actually sick but were missed by the model**.

Using this information, **find the following**:

Using this information, **find the following**:

1. **True Positive (TP)**
2. **True Negative (TN)**
3. **False Positive (FP)**
4. **False Negative (FN)**
5. **Accuracy of the model**
6. **Precision**
7. **Recall (Sensitivity)**
8. **F1 Score**

Total patients = 300

- 70 patients have the disease

That means.

Total positive Case = 70

Here, model has predicted 55 were truly sick (correct positive prediction)

Therefore, TP = 55

So, We can write

$$TP + FN = 70$$

$$55 + FN = 70 \Rightarrow FN = 15$$

Here, Total negative Prediction 235

$$TN + FN = 235$$

$$TN = 235 - 15 = 220$$

- $TP + TN + FP + FN = 300$

$$\Rightarrow 55 + 220 + FP + 15 = 300$$

$$\Rightarrow FP = 300 - (290)$$

$$= 10$$

In other way,

Model has predicted total 65 positive, where 55 is actu positive prediction.

$$TP + FP = 65$$

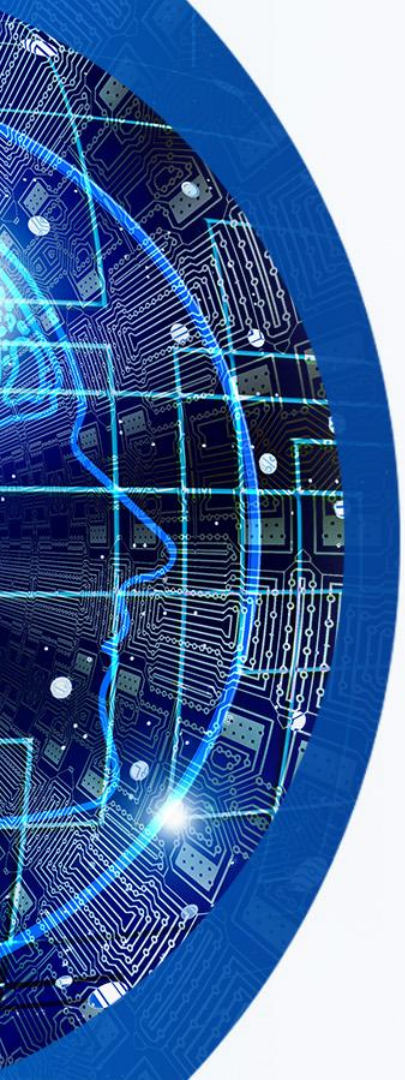
therefore, $FP = 65 - 55 = 10$

A classifier was tested on a dataset. It correctly detected **40 positive cases**, missed **10 positive cases**, and incorrectly labeled **5 negative cases** as positive.

The total number of samples was **100**.

Find:

1. **Accuracy**
2. **Precision**
3. **Recall**



Let’s compute each metric step by step.

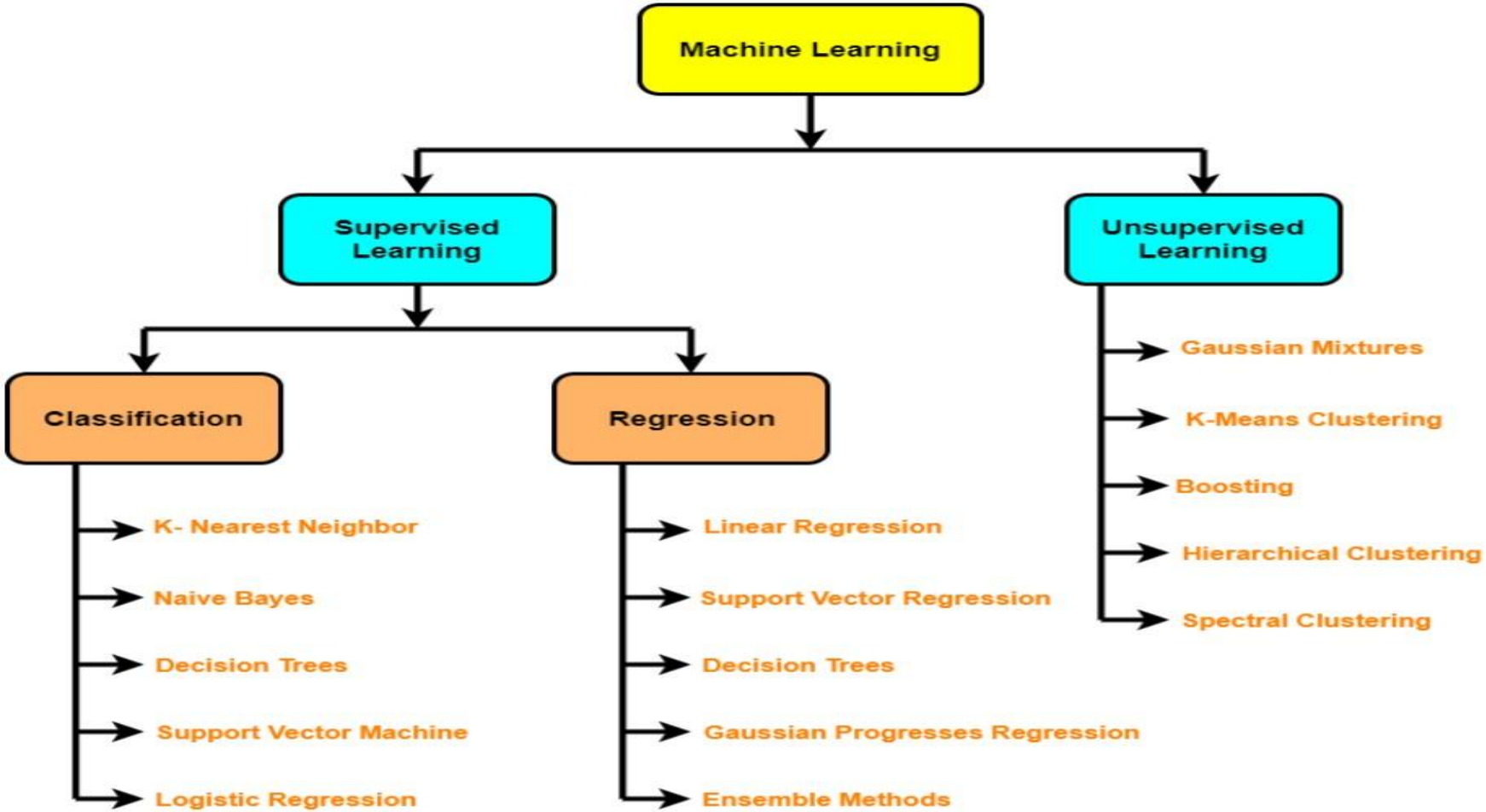
Given:

- True Positives (TP) = 40
- False Negatives (FN) = 10
- False Positives (FP) = 5
- Total samples = 100

First, find **True Negatives (TN)**:

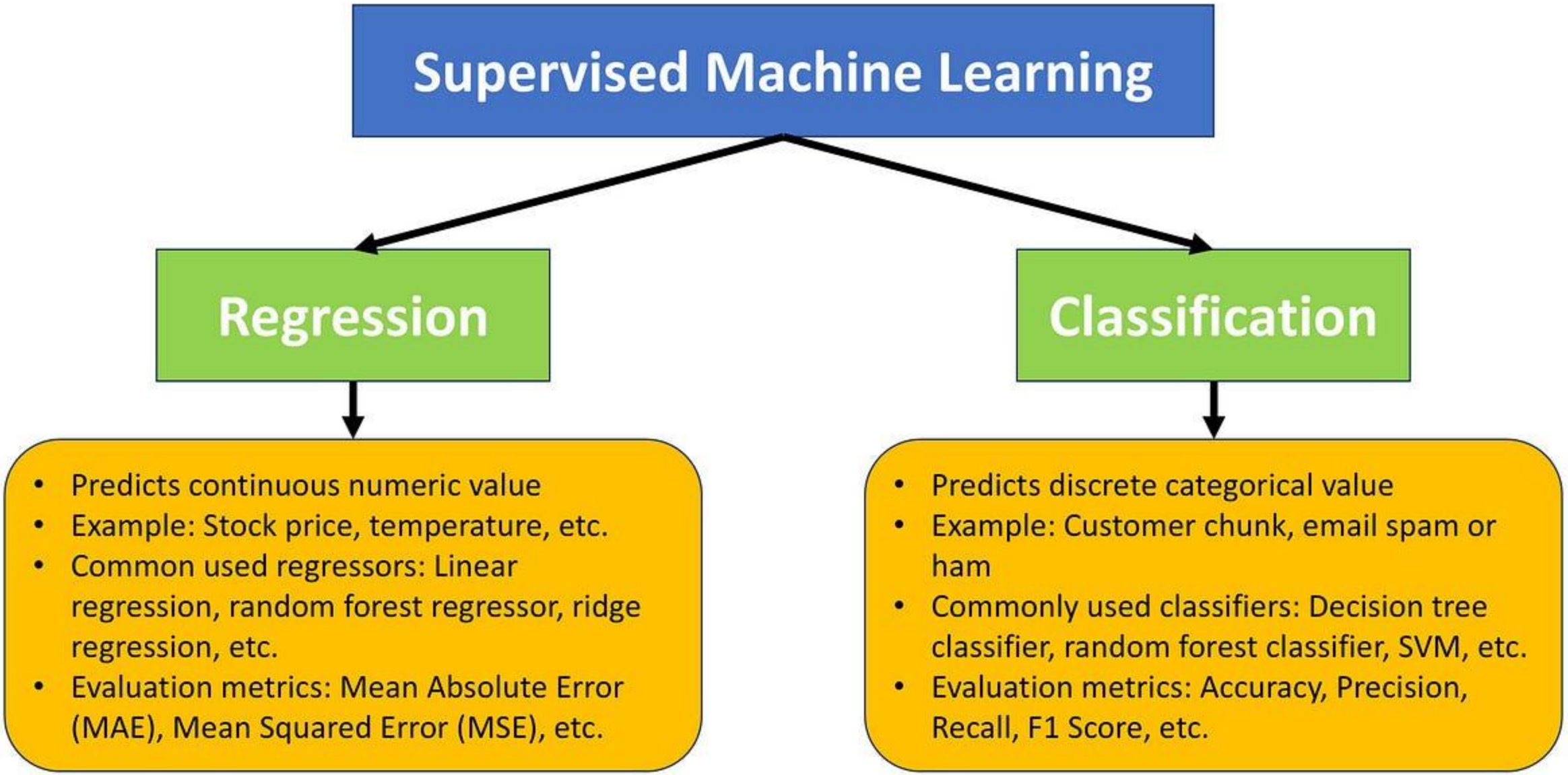
TN = Total – (TP + FN + FP)

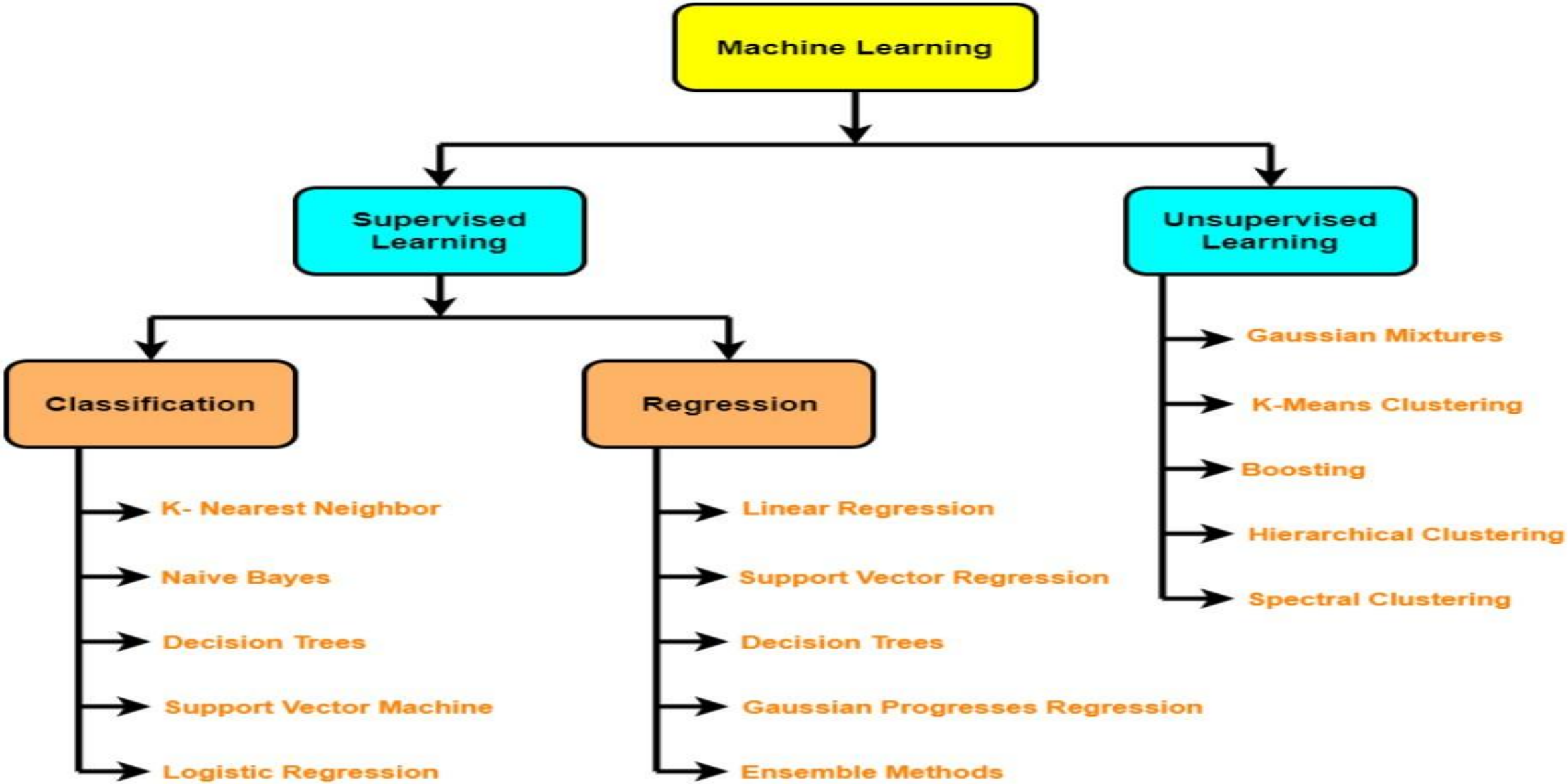
TN = 100 – (40 + 10 + 5) = **45**

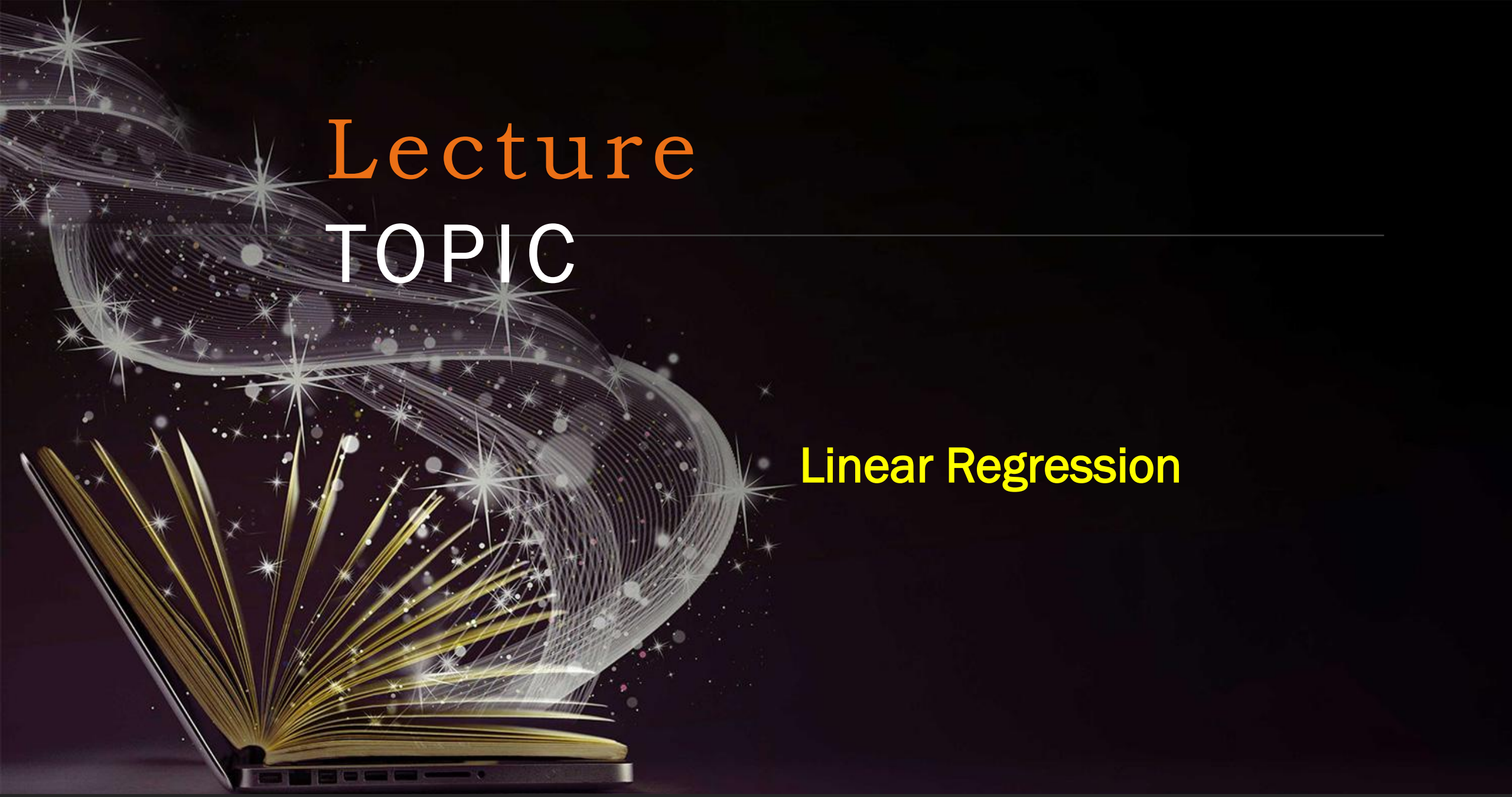

Machine Learning
Supervised Learning
Unsupervised Learning
Classification
Regression
K- Nearest Neighbor
Naive Bayes
Decision Trees
Support Vector Machine
Logistic Regression
Linear Regression
Support Vector Regression
Decision Trees
Gaussian Progresses Regression
Ensemble Methods
Gaussian Mixtures
K-Means Clustering
Boosting
Hierarchical Clustering
Spectral Clustering

Supervised Machine Learning

DR. SUMANA DE


Supervised Machine Learning
Regression
Classification
• Predicts continuous numeric value
• Example: Stock price, temperature, etc.
• Common used regressors: Linear regression, random forest regressor, ridge regression, etc.
• Evaluation metrics: Mean Absolute Error (MAE), Mean Squared Error (MSE), etc.
• Predicts discrete categorical value
• Example: Customer chunk, email spam or ham
• Commonly used classifiers: Decision tree classifier, random forest classifier, SVM, etc.
• Evaluation metrics: Accuracy, Precision, Recall, F1 Score, etc.


Machine Learning
Supervised Learning
Unsupervised Learning
Classification
Regression
K- Nearest Neighbor
Naive Bayes
Decision Trees
Support Vector Machine
Logistic Regression
Linear Regression
Support Vector Regression
Decision Trees
Gaussian Progresses Regression
Ensemble Methods
Gaussian Mixtures
K-Means Clustering
Boosting
Hierarchical Clustering
Spectral Clustering



Lecture TOPIC

Linear Regression

Linear Regression

Linear Regression is a **supervised learning algorithm** used to model the relationship between a **dependent variable (target)** and one or more **independent variables (features)** by fitting a straight line (or hyperplane) to the data.

In simple terms:

➡️ It **predicts a numeric value** based on input features by finding the **best-fit line** that minimizes the error between predicted and actual values.

Linear Regression

Linear regression is a type of supervised machine-learning algorithm that learns from the labelled datasets and maps the data points with most optimized linear functions which can be used for prediction on new datasets. It assumes that there is a linear relationship between the input and output, meaning the output changes at a constant rate as the input changes. This relationship is represented by a straight line.

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

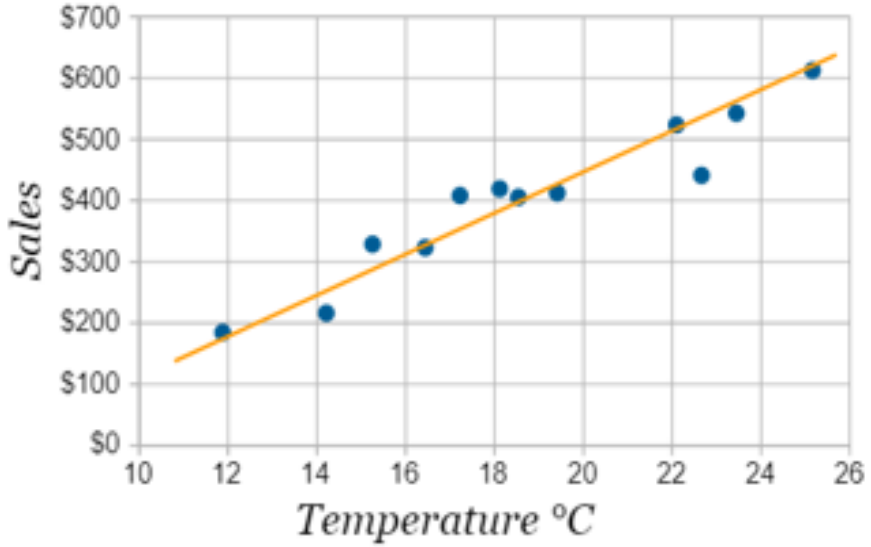
Independent variable (input): Hours studied because it's the factor we control or observe.

Dependent variable (output): Exam score because it depends on how many hours were studied.

Calculate the line using Least Squares Regression.

Line of Best Fit

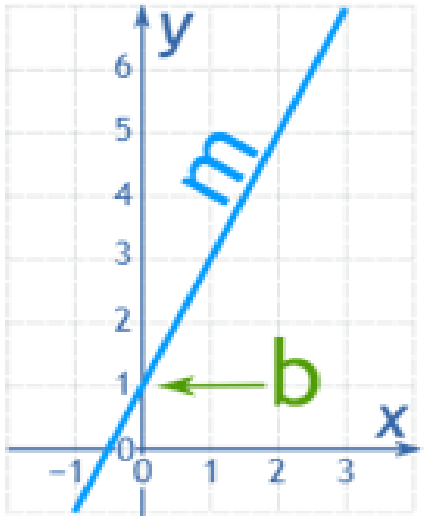
- Imagine you have some points, and want to have a line that best fits them like this:
- Imagine you have some points, and want to have a **line** that best fits them like this:

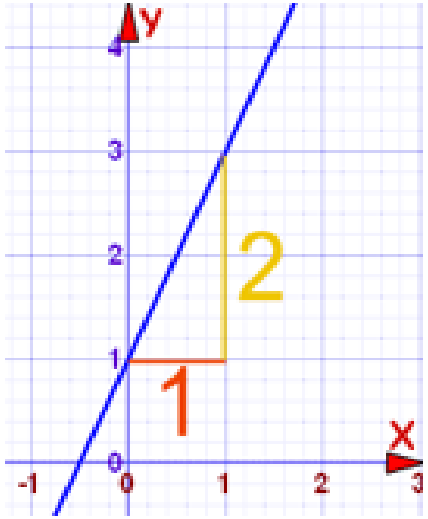


We can place the line "by eye": try to have the line as close as possible to all points, and a similar number of points above and below the line.
But for better accuracy let's see how to calculate the line using **Least Squares Regression**.

How to draw straight line

- Our aim is to calculate the values m (slope) and b (y-intercept) in the equation of a line :
- $y = mx + b$
- Where:
- **y** = how far up
- **x** = how far along
- **m** = Slope or Gradient (how steep the line is)
- **b** = the Y Intercept (where the line crosses the Y axis)





Putting that into **y** = **mx** + **b** gets us:

$$y = 2x + 1$$

$$m = 2/1 = 2$$

$b = 1$ *(value of* ***y*** *when x=0)*

BEST FIT line using Least Squares Regression

To find the line of best fit for **N** points:

Step 1: For each (x,y) point calculate x^2 and xy

Step 2: Sum all x, y, x^2 and xy, which gives us Σx, Σy, Σx^2 and Σxy (Σ means "sum up")

Step 3: Calculate Slope **m**:

$$\mathbf{m} = \frac{N \Sigma(xy) - \Sigma x \, \Sigma y}{N \Sigma(x^2) - (\Sigma x)^2}$$

(N is the number of points.)

Step 4: Calculate Intercept **b**:

$$\mathbf{b} = \frac{\Sigma y - m \, \Sigma x}{N}$$

Step 5: Assemble the equation of a line

$$y = mx + b$$

Done!

EXAMPLE

- Example: Sam found how many hours of sunshine vs how many ice creams were sold at the shop from Monday to Friday:

"x" Hours of Sunshine	"y" Ice Creams Sold
2	4
3	5
5	7
7	10
9	15

Let us find the best **m** (slope) and **b** (y-intercept) that suits that data

$$y = mx + b$$

Step 1: For each (x,y) calculate x^2 and xy:

x	y	x^2	xy
2	4	4	8
3	5	9	15
5	7	25	35
7	10	49	70
9	15	81	135

Step 2: Sum x, y, x^2 and xy (gives us Σx, Σy, Σx^2 and Σxy):

x	y	x^2	xy
2	4	4	8
3	5	9	15
5	7	25	35
7	10	49	70
9	15	81	135
Σx: 26	**Σy: 41**	**Σx^2: 168**	**Σxy: 263**

Also **N** (number of data values) = **5**

Step 3: Calculate Slope **m**:

$$\mathbf{m} = \frac{N\,\Sigma(xy) - \Sigma x\,\Sigma y}{N\,\Sigma(x^2) - (\Sigma x)^2}$$

$$= \frac{5 \times 263 - 26 \times 41}{5 \times 168 - 26^2}$$

$$= \frac{1315 - 1066}{840 - 676}$$

$$= \frac{249}{164} = 1.5183...$$

Step 4: Calculate Intercept **b**:

$$\mathbf{b} = \frac{\Sigma y - m\,\Sigma x}{N}$$

$$= \frac{41 - 1.5183 \times 26}{5}$$

$$= 0.3049...$$

Step 5: Assemble the equation of a line:

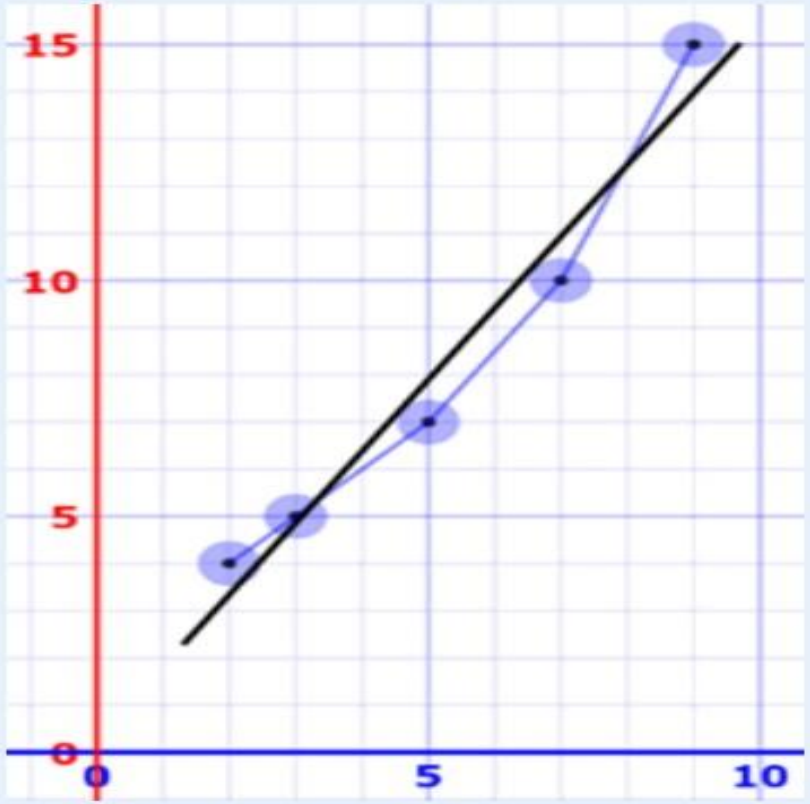
$$y = mx + b$$

$$y = 1.518x + 0.305$$

Let's see how it works out:

x	y	y = 1.518x + 0.305	error
2	4	3.34	−0.66
3	5	4.86	−0.14
5	7	7.89	0.89
7	10	10.93	0.93
9	15	13.97	−1.03

Here are the (x,y) points and the line **y = 1.518x + 0.305** on a graph:



Nice fit!

Sam hears the weather forecast which says "we expect 8 hours of sun tomorrow", so he uses the above equation to estimate that he will sell

$$y = 1.518 \times 8 + 0.305 = 12.45 \text{ Ice Creams}$$

Linear Regression – Solved Example

- Let us consider an example where the five weeks' sales data (in Thousands) is given as shown in Table.
- Apply linear regression technique to predict the 7^{th} and 12^{th} week sales.

x_i $(Week)$	y_j $(Sales\ in\ Thousands)$
1	1.2
2	1.8
3	2.6
4	3.2
5	3.8

COST FUNCTION/LOSS FUNCTION/ERROR FUNCTION

A. NEED OF COST FUNCTION
B. WHAT IS COST FUNCTION
C. WHAT IS LINEAR REGRESSION
D. COST FUNCTION IN LINEAR REGRESSION
E. TYPES OF ERROR IN ML
F. REGRESSION MODEL USING GRADIENT DESCENDENT

COST FUNCTION

Rise in demand of houses

Actual

Predicted

Cost Function

What is Cost Function?

Measure of how badly a model is performing

Expressed as the difference between actual and predicted values.

What Is Cost Function?

Cost function measures the performance of a machine learning model for given data. Cost function quantifies the error between predicted and expected values and present that error in the form of a single real number. Depending on the problem, cost function can be formed in many different ways. The purpose of cost function is to be either:

- Minimized: The returned value is usually called cost, loss or error. The goal is to find the values of model parameters for which cost function return as small a number as possible.
- Maximized: In this case, the value it yields is named a reward. The goal is to find values of model parameters for which the returned number is as large as possible.

How to Tailor a Cost Function

Let's start with a model using the following formula:

$$\hat{y} = wx$$

- $\hat{y}$ = predicted value,
- x = vector of data used for prediction or training
- w = weight.

Notice that we've omitted the bias on purpose. Let's try to find the value of weight parameter, so for the following data samples:

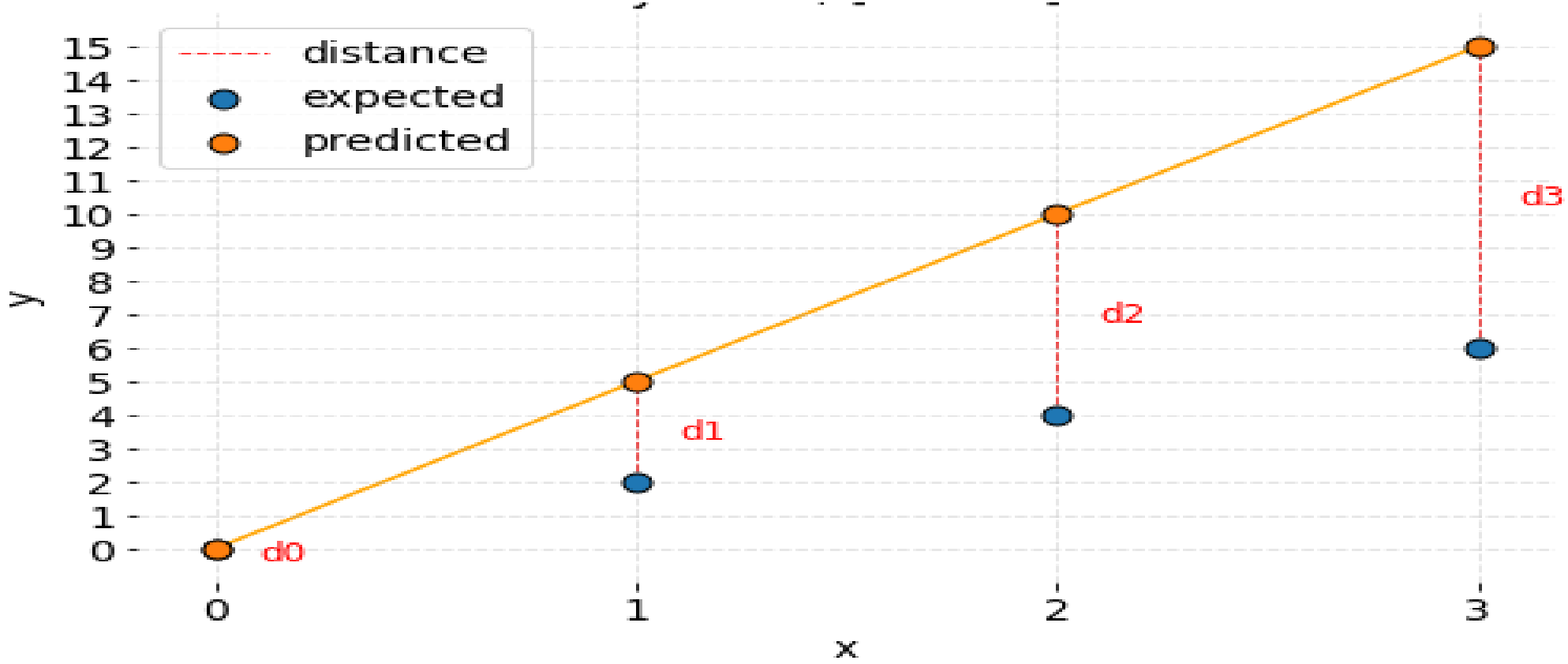
$$x_0 = [0], x_1 = [1], x_2 = [2], x_3 = [3]$$

The outputs of the model are as close as possible to:

$$y_0 = 0, y_1 = 2, y_2 = 4, y_3 = 6$$

Now it's time to assign a random value to the weight parameter and visualize the model's results. Let's pick `w = 5.0` for now.

y = wx, [w=5.0]


distance
expected
predicted
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
y
d0
d1
d2
d3
0
1
2
3
x

We can observe that the model predictions are different than expected values but how can we express that mathematically? The most straightforward idea is to subtract both values from each other and see if the result of that operation equals zero. Any other result means that the values differ. The size of the received number provides information about how significant the error is. From the geometrical perspective, it's possible to state that error is the distance between two points in the coordinate system. Let's define the distance as:

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

According to the formula, calculate the errors between the predictions and expected values:

$$d_0 = 0 - 0 = 0$$

$$d_1 = 5 - 2 = 3$$

$$d_2 = 10 - 4 = 6$$

$$d_3 = 15 - 6 = 9$$

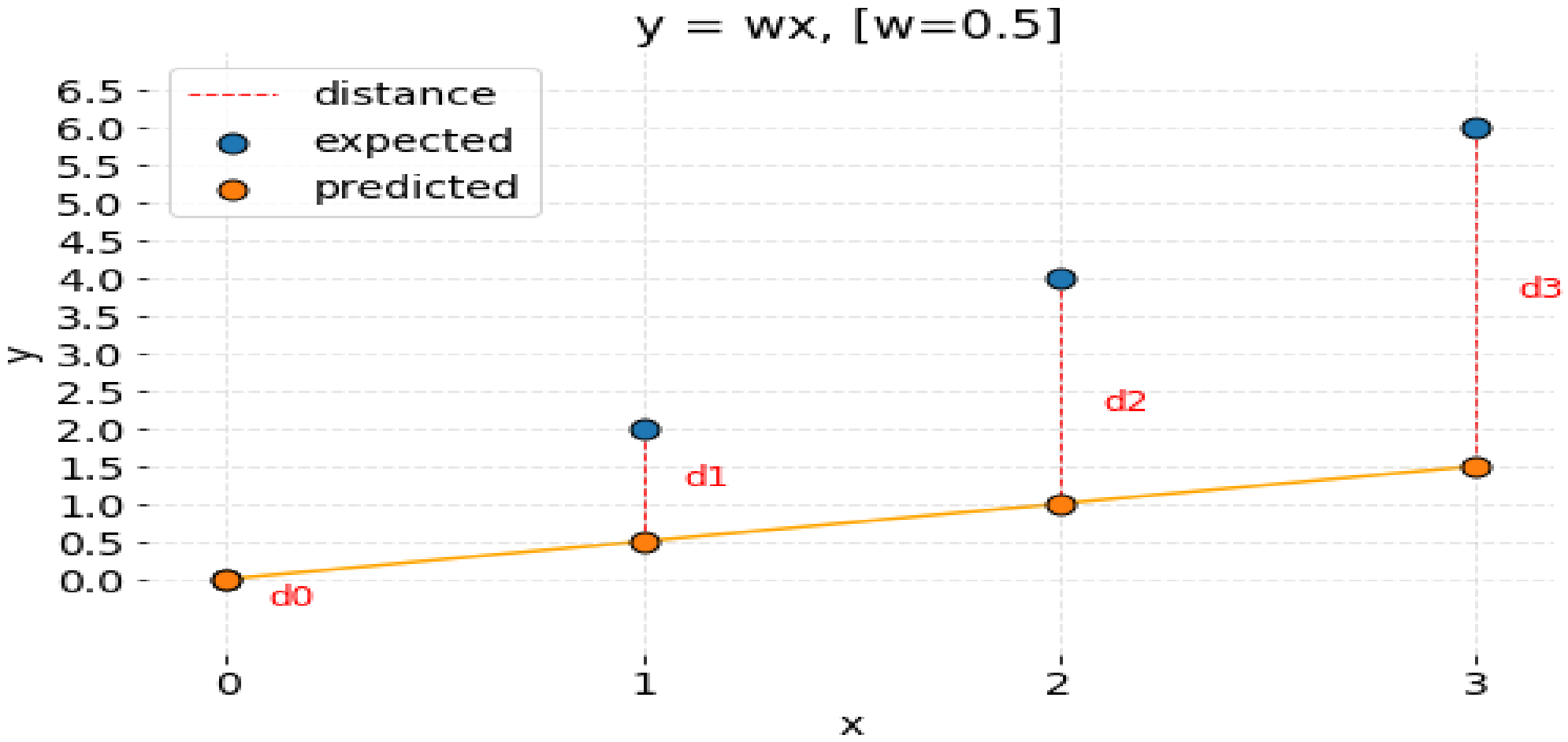
$$cost(5) = 0 + 3 + 6 + 9 = 18$$

However, now imagine there are a million points instead of four. The accumulated errors will become a bigger number for a model making a prediction on a larger data set than on a smaller data set. Consequently, we can't compare those models. That's why we have to scale in some way. The right idea is to divide the accumulated errors by the number of points. Cost stated like that is mean of errors the model made for the given data set.

$$cost(w) = \frac{1}{4}(d_0 + d_1 + d_2 + d_3)$$

$$cost(5) = \frac{1}{4}(0 + 3 + 6 + 9) = \frac{18}{4} = 4.5$$

Unfortunately, the formula isn't complete. We still have to consider all cases so let's try picking smaller weights and see if the created cost function works. We'll set weight to `w = 0.5`.


y = wx, [w=0.5]
distance
expected
predicted
6.5
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
y
d0
d1
d2
d3
0
1
2
3
x

The predictions are off again. However, in comparison to the previous case, that predicted points are below expected points. Numerically, predictions are smaller. The cost formula is going to malfunction because calculated distances have negative values.

$$d_0 = 0 - 0 = 0$$

$$d_1 = 0.5 - 2 = -1.5$$

$$d_2 = 1 - 4 = -3$$

$$d_3 = 1.5 - 6 = -4.5$$

The cost value is also negative:

$$cost(0.5) = \frac{1}{4}[0 + (-1.5) + (-3) + (-4.5)] = \frac{-9}{4} = -2.25$$

Since distance can't have a negative value, we can attach a more substantial penalty to the predictions located above or below the expected results (some cost functions do so, e.g. RMSE), but the value shouldn't be negative because it will cancel out positive errors. It will then become impossible to properly minimize or maximize the cost function.

So how about fixing the problem by using the absolute value of the distance? After stating the distance as:

$$distance = |\hat{y} - y|$$

The costs for each value of weights are:

$$cost(5) = \frac{1}{4}(|0| + |3| + |6| + |9|) = \frac{18}{4} = 4.5$$

$$cost(0.5) = \frac{1}{4}(|0| + |-1.5| + |-3| + |-4.5|) = \frac{9}{4} = 2.25$$

Now we've correctly calculated the costs for both weights `w = 5.0` and `w = 0.5`. It is possible to compare the parameters. The model achieves better results for `w = 0.5` as the cost value is smaller.

The function we created is mean absolute error.

x	y
5	100
10	200
15	300
20	400

Mean absolute error (MAE)

The **mean absolute error** (MAE) is the simplest regression error metric to understand.

$$\text{Total data points}\frac{1}{\text{Total data points}}\sum \left| \text{Actual output} - \text{predicted output} \right|$$

$$\frac{1}{n}\sum \left| y - \hat{y} \right|$$

x	y
5	100
10	200
15	300
20	400

Mean square error (MSE)

The **mean square error** (MSE) is similar to MAE
but *squares* the difference before summing them all
Cost Function ?

$$\text{Total data points}\frac{1}{\sum}\ (\text{Actual output} - \text{predicted output})^2$$

$$\frac{1}{n}\sum\ (y - \hat{y})^2$$

x	y
5	100
10	200
15	300
20	400

Root Mean square error (RMSE)

The **root mean square error** (RMSE) is similar to MAE but *squares* the difference before summing them all

$$\sqrt{\text{Total data points}\frac{1}{\sum}\quad(\text{Actual output} - \text{predicted output})^2}$$

$$\sqrt{\frac{1}{n}\sum\quad(y-\hat{y})^2}$$

Multiple Linear Regression

Multiple regression is like linear regression, but with more than one independent value, meaning that we try to predict a value based on **two or more** variables.

Multiple Linear Regression attempts to model the relationship between two or more features and a response by fitting a linear equation to observed data. The steps to perform multiple linear Regression are almost similar to that of simple linear Regression. The Difference Lies in the evaluation. We can use it to find out which factor has the highest impact on the predicted output and how different variables relate to each other.

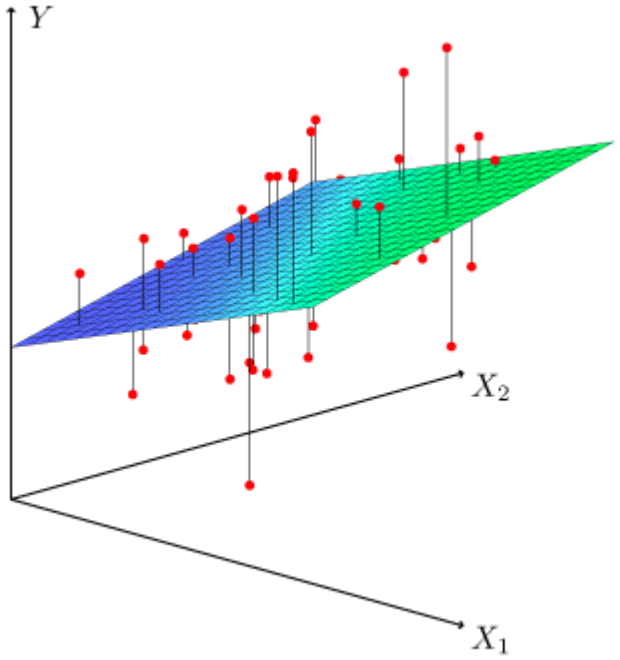
Here : $Y = b0 + b1 * x1 + b2 * x2 + b3 * x3 + \ldots\ldots bn * xn$

Y = Dependent variable and x1, x2, x3, ... xn = multiple independent variables

Two Ways to calculate the multiple regression equation:

1. Using Matrix

2. Using Formula



Using Matrix

- The data in the table relate grams plant dry weight **y**, to present soil organic matter **x1**, and kilograms of supplemental soil nitrogen added per 100 square meters **x2**.

- Obtain the regression equation.
- Also, predict the dry weight given soil organic matter of **5** and soil nitrogen **4**.

Y	X1	X2
78.5	7	2.6
74.3	1	2.9
104.3	11	5.6
87.6	11	3.1
95.9	7	5.2
109.2	11	5.5
102.7	3	7.1

$$y = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2$$

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

Y	X1	X2
78.5	7	2.6
74.3	1	2.9
104.3	11	5.6
87.6	11	3.1
95.9	7	5.2
109.2	11	5.5
102.7	3	7.1

$$Y = \begin{bmatrix} 78.5 \\ 74.3 \\ 104.3 \\ 87.6 \\ 95.6 \\ 109.2 \\ 102.7 \end{bmatrix}$$

$$X = \begin{bmatrix} 1 & 7 & 2.6 \\ 1 & 1 & 2.9 \\ 1 & 11 & 5.6 \\ 1 & 11 & 3.1 \\ 1 & 7 & 5.2 \\ 1 & 11 & 5.5 \\ 1 & 3 & 7.1 \end{bmatrix}$$

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

$$Y = \begin{bmatrix} 78.5 \\ 74.3 \\ 104.3 \\ 87.6 \\ 95.9 \\ 109.2 \\ 102.7 \end{bmatrix} \quad X = \begin{bmatrix} 1 & 7 & 2.6 \\ 1 & 1 & 2.9 \\ 1 & 11 & 5.6 \\ 1 & 11 & 3.1 \\ 1 & 7 & 5.2 \\ 1 & 11 & 5.5 \\ 1 & 3 & 7.1 \end{bmatrix} \quad X^T = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 7 & 1 & 11 & 11 & 7 & 11 & 3 \\ 2.6 & 2.9 & 5.6 & 3.1 & 5.2 & 5.5 & 7.1 \end{bmatrix}$$

$$X^T * X = \begin{bmatrix} 7 & 51 & 32 \\ 51 & 471 & 235 \\ 32 & 235 & 163.84 \end{bmatrix} \quad (X^T * X)^{-1} = \begin{bmatrix} 1.7996 & -0.0685 & -0.2532 \\ -0.0685 & -0.0101 & -0.0011 \\ -0.2532 & -0.0011 & 0.0571 \end{bmatrix}$$

$$(X^T * X)^{-1} X^T = \begin{bmatrix} 0.66178 & 0.99682 & -0.37182 & 0.26118 & 0.00346 & -0.3465 & -0.20362 \\ -0.00066 & -0.06159 & 0.03644 & 0.03919 & -0.00352 & 0.03655 & -0.04601 \\ -0.11244 & -0.08871 & 0.05446 & -0.08829 & 0.03602 & 0.04875 & 0.14891 \end{bmatrix}$$

$$Y = \begin{bmatrix} 78.5 \\ 74.3 \\ 104.3 \\ 87.6 \\ 95.9 \\ 109.2 \\ 102.7 \end{bmatrix}$$

$$(X^T * X)^{-1} X^T = \begin{bmatrix} 0.66178 & 0.99682 & -0.37182 & 0.26118 & 0.00346 & -0.3465 & -0.20362 \\ -0.00066 & -0.06159 & 0.03644 & 0.03919 & -0.00352 & 0.03655 & -0.04601 \\ -0.11244 & -0.08871 & 0.05446 & -0.08829 & 0.03602 & 0.04875 & 0.14891 \end{bmatrix}$$

$$\boldsymbol{\beta} = (X^T * X)^{-1} X^T Y = \begin{bmatrix} \mathbf{51.9} \\ \mathbf{1.5} \\ \mathbf{6.5} \end{bmatrix}$$

$$\boldsymbol{\beta} = (X^T * X)^{-1} \mathrm{X}^{\mathrm{T}} Y = \begin{bmatrix} 51.9 \\ 1.5 \\ 6.5 \end{bmatrix} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix}$$

$$y = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2$$

$$y = 51.9 + 1.5 * x_1 + 6.5 * x_2$$

Plant dry weight given soil organic matter of **5** and soil nitrogen **4**.

$$y = 51.9 + 1.5 * 5 + 6.5 * 4$$

$$y = 85.4$$

x1 Product 1 Sales	x2 Product 2 Sales	Y Weekly Sales
1	4	1
2	5	6
3	8	8
4	2	12

$$a_0 = -1.69$$
$$a_1 = 3.48$$
$$a_2 = -0.05$$

$$y = -1.69 + 3.48x_1 - 0.05x_2$$

Using Formula

Example: Multiple Linear Regression by Hand

Suppose we have the following dataset with one response variable *y* and two predictor variables X_1 and X_2:

y	X_1	X_2
140	60	22
155	62	25
159	67	24
179	70	20
192	71	15
200	72	14
212	75	14
215	78	11

Use the following steps to fit a multiple linear regression model to this dataset.

Step 1: Calculate X_1^2, X_2^2, X_1y, X_2y and X_1X_2.

	y	X_1	X_2
	140	60	22
	155	62	25
	159	67	24
	179	70	20
	192	71	15
	200	72	14
	212	75	14
	215	78	11
Mean	181.5	69.375	18.125
Sum	1452	555	145

X_1^2	X_2^2	X_1y	X_2y	X_1X_2
3600	484	8400	3080	1320
3844	625	9610	3875	1550
4489	576	10653	3816	1608
4900	400	12530	3580	1400
5041	225	13632	2880	1065
5184	196	14400	2800	1008
5625	196	15900	2968	1050
6084	121	16770	2365	858
38767	2823	101895	25364	9859

Sum

Step 2: Calculate Regression Sums.

Next, make the following regression sum calculations:

- $\Sigma x_1^2 = \Sigma X_1^2 - (\Sigma X_1)^2 / n = 38{,}767 - (555)^2 / 8 =$ **263.875**
- $\Sigma x_2^2 = \Sigma X_2^2 - (\Sigma X_2)^2 / n = 2{,}823 - (145)^2 / 8 =$ **194.875**
- $\Sigma x_1 y = \Sigma X_1 y - (\Sigma X_1 \Sigma y) / n = 101{,}895 - (555*1{,}452) / 8 =$ **1,162.5**
- $\Sigma x_2 y = \Sigma X_2 y - (\Sigma X_2 \Sigma y) / n = 25{,}364 - (145*1{,}452) / 8 =$ **-953.5**
- $\Sigma x_1 x_2 = \Sigma X_1 X_2 - (\Sigma X_1 \Sigma X_2) / n = 9{,}859 - (555*145) / 8 =$ **-200.375**

	y	X_1	X_2
	140	60	22
	155	62	25
	159	67	24
	179	70	20
	192	71	15
	200	72	14
	212	75	14
	215	78	11
Mean	181.5	69.375	18.125
Sum	1452	555	145

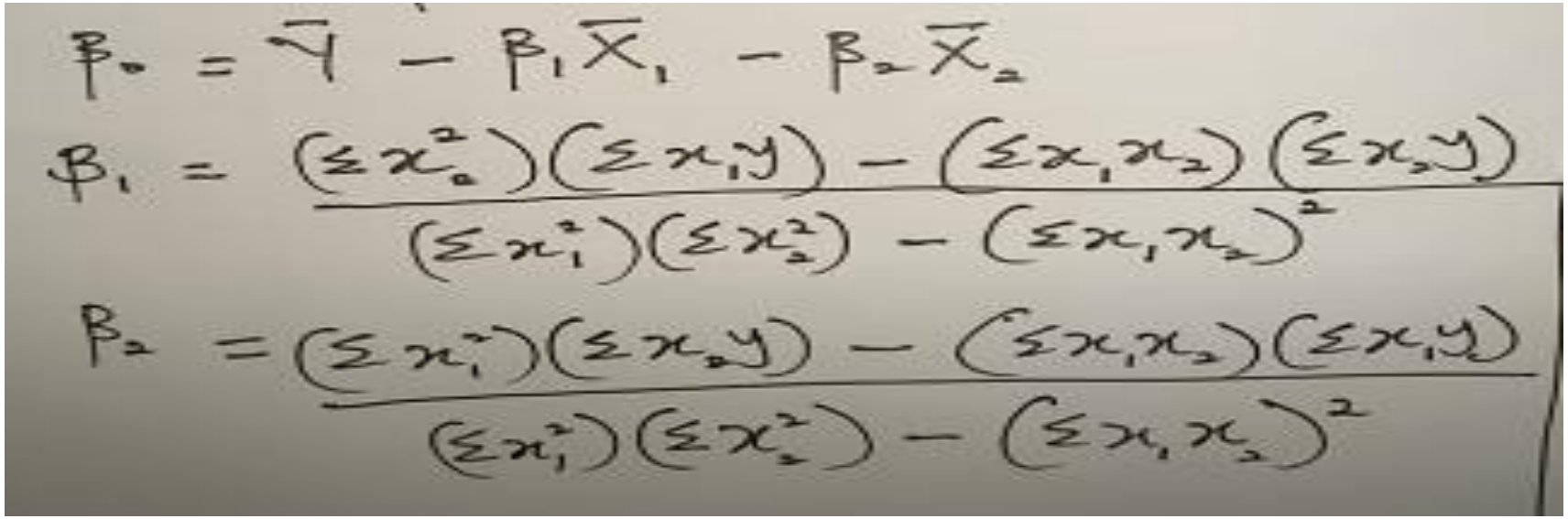
	X_1^2	X_2^2	X_1y	X_2y	X_1X_2
	3600	484	8400	3080	1320
	3844	625	9610	3875	1550
	4489	576	10653	3816	1608
	4900	400	12530	3580	1400
	5041	225	13632	2880	1065
	5184	196	14400	2800	1008
	5625	196	15900	2968	1050
	6084	121	16770	2365	858
Sum	38767	2823	101895	25364	9859

Reg Sums	263.875	194.875	1162.5	-953.5	-200.375

Step 3: Calculate b_0, b_1, and b_2.

The formula to calculate b_1 is: $[(\Sigma x_2^2)(\Sigma x_1 y) - (\Sigma x_1 x_2)(\Sigma x_2 y)] \ / \ [(\Sigma x_1^2)(\Sigma x_2^2) - (\Sigma x_1 x_2)^2]$

Thus, $\mathbf{b_1} = [(194.875)(1162.5) - (-200.375)(-953.5)] \ / \ [(263.875)(194.875) - (-200.375)^2] = \mathbf{3.148}$



The formula to calculate b_2 is: $[(\Sigma x_1^2)(\Sigma x_2 y) - (\Sigma x_1 x_2)(\Sigma x_1 y)] / [(\Sigma x_1^2)(\Sigma x_2^2) - (\Sigma x_1 x_2)^2]$

Thus, $\mathbf{b_2} = [(263.875)(-953.5) - (-200.375)(1152.5)] / [(263.875)(194.875) - (-200.375)^2] = \mathbf{-1.656}$

The formula to calculate b_0 is: $\bar{y} - b_1\bar{X}_1 - b_2\bar{X}_2$

Thus, $\mathbf{b_0} = 181.5 - 3.148(69.375) - (-1.656)(18.125) = \mathbf{-6.867}$

Step 5: Place b_0, b_1, and b_2 in the estimated linear regression equation.

The estimated linear regression equation is: $\hat{y} = b_0 + b_1*x_1 + b_2*x_2$

In our example, it is $\hat{y} = -6.867 + 3.148x_1 - 1.656x_2$

Practice Class

- Let us consider an example where the five weeks' sales data (in Thousands) is given as shown in Table.
- Apply linear regression technique to predict the 7^{th} and 12^{th} week sales.

x_i (Week)	y_j (Sales in Thousands)
1	1.2
2	1.8
3	2.6
4	3.2
5	3.8

Find out Multiple Linear Regression line.

x1 Product 1 Sales	x2 Product 2 Sales	Y Weekly Sales
1	4	1
2	5	6
3	8	8
4	2	12

1. SIMPLE LINEAR REGRESSION

$$\boxed{y = 0.54 + 0.66x}$$

2. MULTIPLE LINEAR REGRESSION

Final Regression Equation

$$\boxed{Y = -1.699 + 3.484X_1 - 0.055X_2}$$

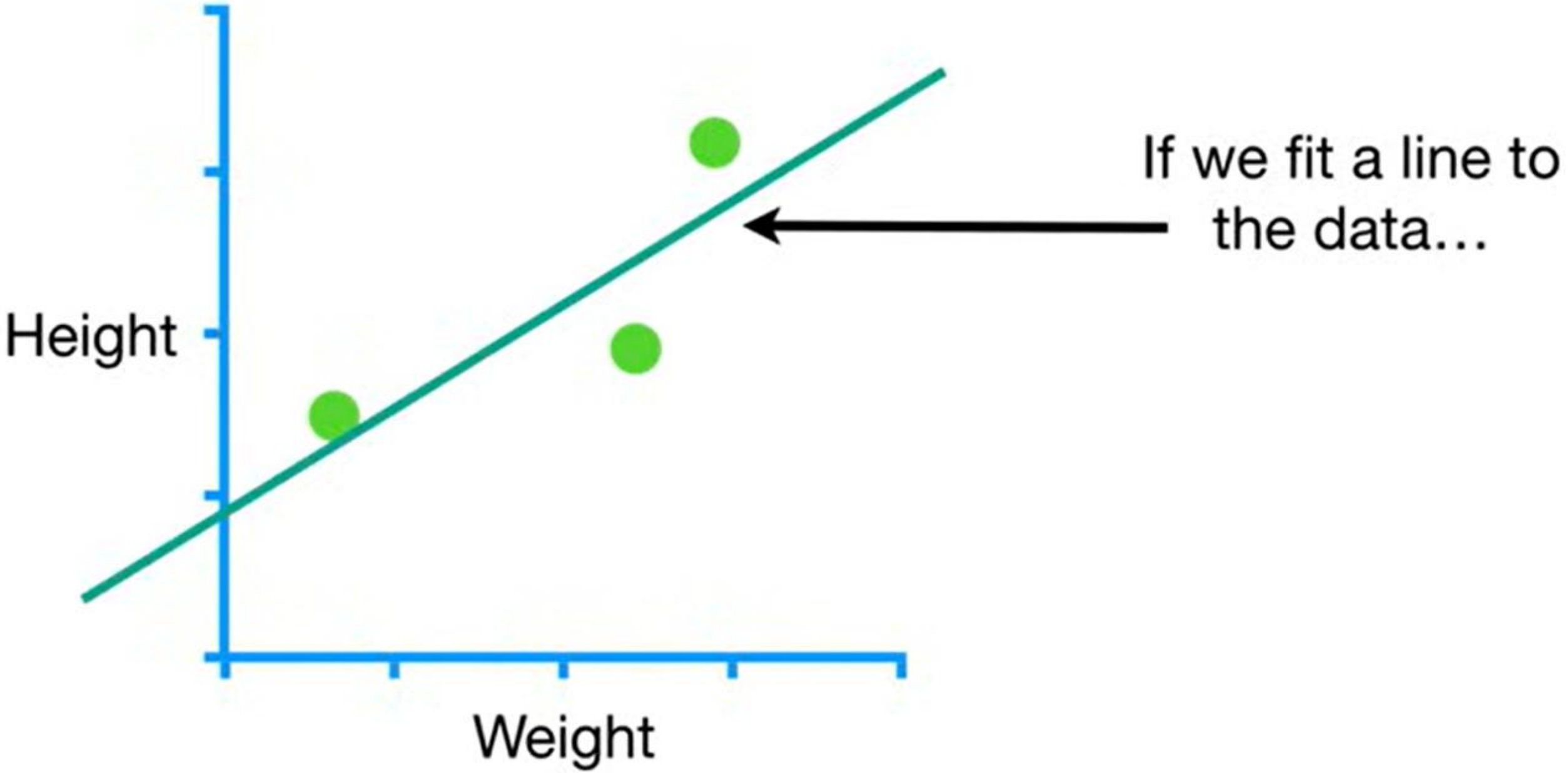
Gradient Descent in Machine Learning

Gradient Descent is known as one of the most commonly used optimization algorithms to train machine learning models by means of minimizing errors between actual and expected results. Further, gradient descent is also used to train Neural Networks.

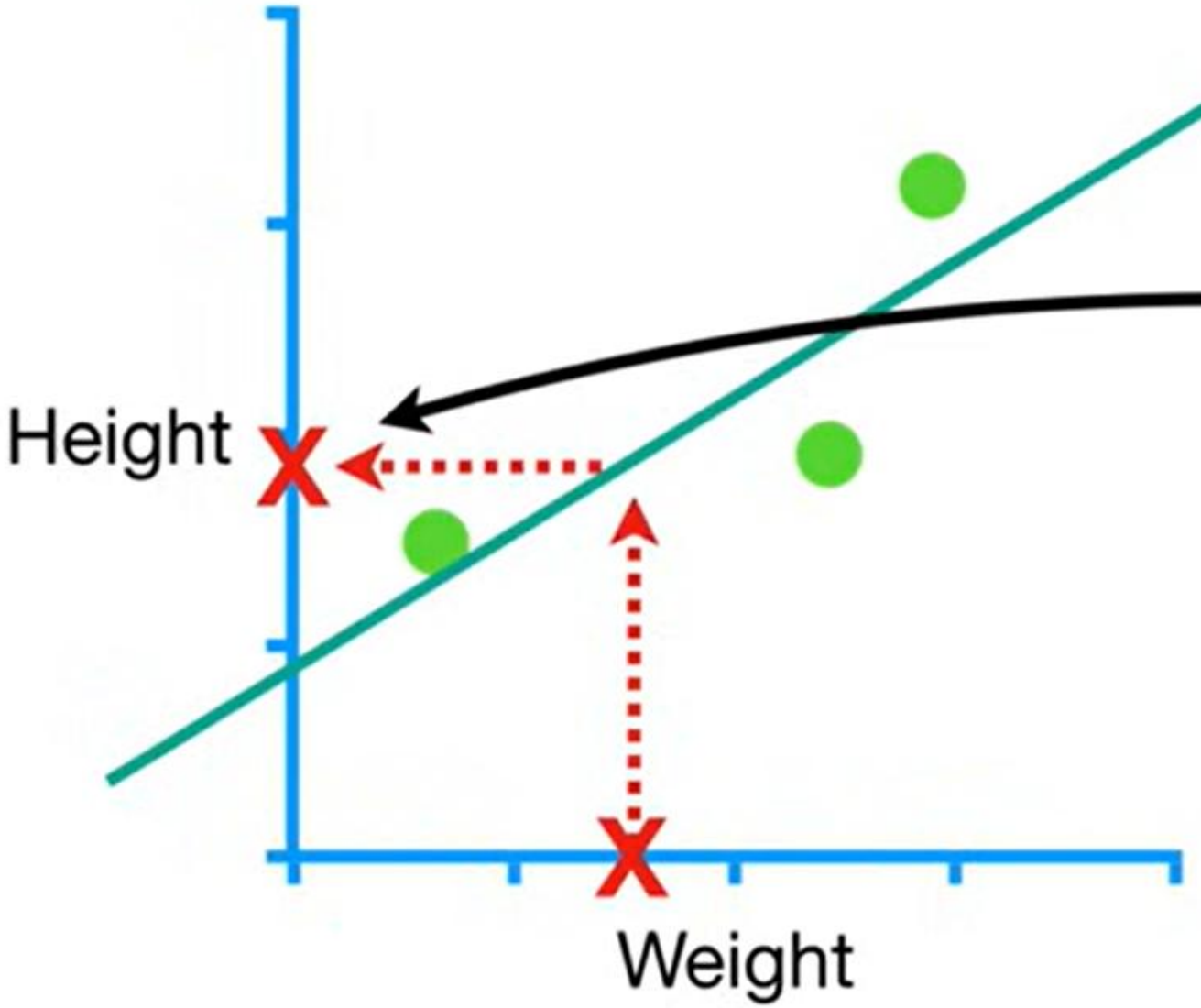
The main objective of using a gradient descent algorithm is to minimize the cost function using iteration

How It Works:

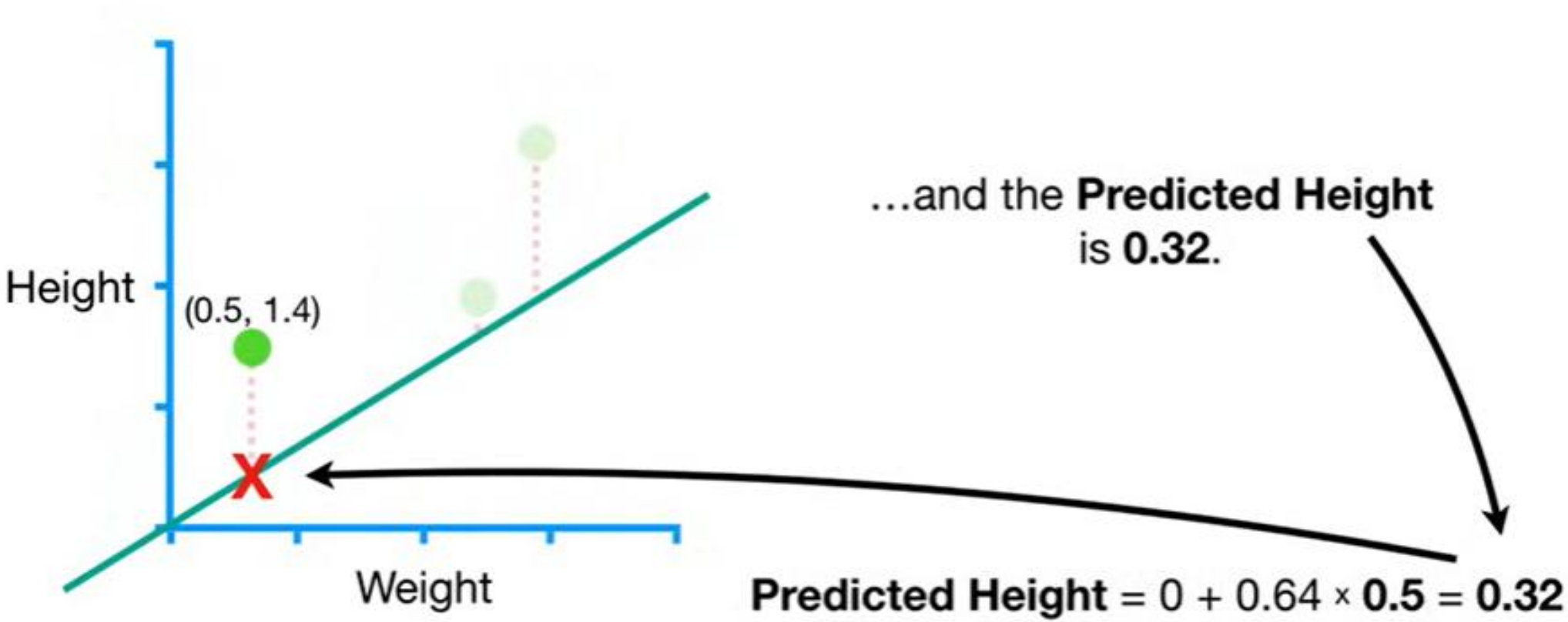
1.Objective Function: In machine learning, we usually have a loss function (like mean squared error, cross-entropy, etc.) that measures how well a model is performing. The goal is to minimize this loss function.
2.Gradient: The gradient of the loss function with respect to the model's parameters indicates the direction and rate of the steepest increase in the function. To minimize the loss function, we move in the opposite direction of the gradient.


If we fit a line to the data...
Height
Weight

Predicted Height = intercept + slope × **Weight**



…we can use the line to predict that they will be **1.9** tall.


Height
(0.5, 1.4)
Weight
…and the **Predicted Height** is **0.32**.
Predicted Height = 0 + 0.64 × **0.5** = **0.32**

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

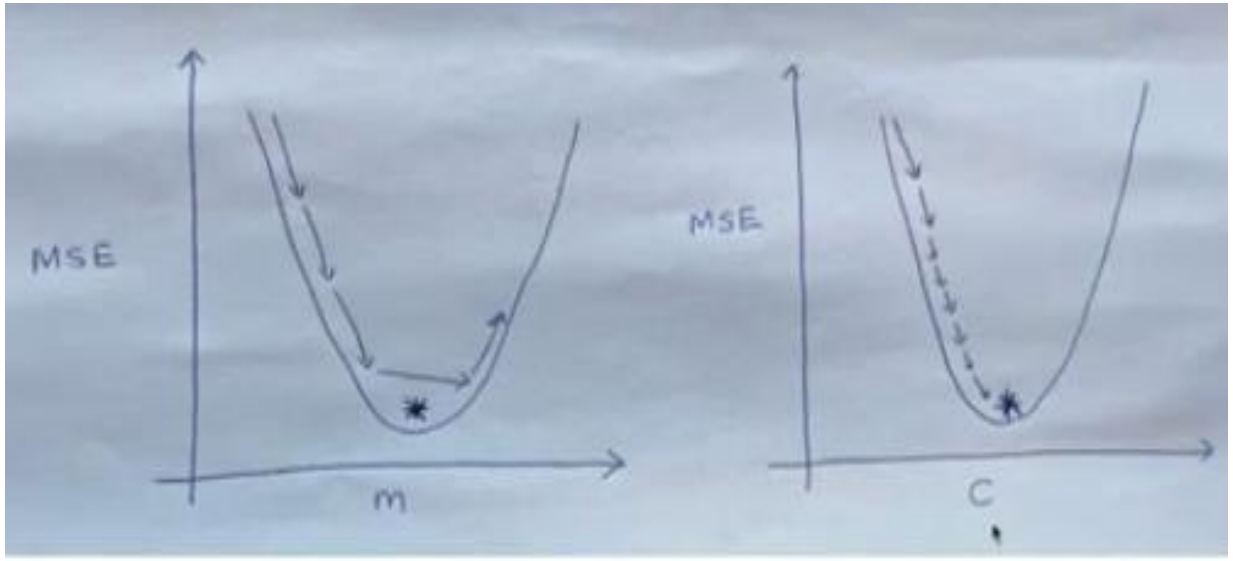
$$Y = mx + c$$

$$m = m - LR * PD(m)$$

$$c = c - LR * PD(c)$$

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - (mx_i + c))^2$$

How to minimize Cost Function?



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

$$PD(c) = -\frac{2}{n} \sum_{i=1}^{n} (y_i - (mx_i + b))$$

```python
import numpy as np
def gradient_descent(x,y):
    m=c= 0
    rate = 0.001
    iteration = 100
    n = len(x)
    for i in range(iteration):
        y_predicted = m * x + c
        cost = (1/n) * sum([val**2 for val in (y-y_predicted)])
        mp = -(2/n)*sum(x*(y-y_predicted))
        cp = -(2/n)*sum(y-y_predicted)
        m = m - rate * mp
        c = c - rate * cp
        print("m : ",m)
        print("c : ",c)
        print("cost : ",cost)
        print("\n")

X = np.array([2,4,6,8,10,12,14,16,18,20])
Y = np.array([10,20,30,40,50,60,70,80,90,100])
gradient_descent(X,Y)
```

```
m :  1.54
c :  0.11
cost :  3850.0
```

```
m :  1.54
c :  0.11
cost :  3850.0


m :  2.60326
c :  0.1859
cost :  1835.2653000000003


m :  3.3373661200000004
c :  0.23825648000000002
cost :  874.8642162084002


m :  3.8442157124800005
c :  0.2743579124
cost :  417.0519521574295
```

```
m :  4.975601438410646
c :  0.34110698159770203
cost :  0.024954446152350263


m :  4.975611841785017 5
c :  0.3409615359894724 3
cost :  0.024933169890094045
```

How It Works:

1. **Objective Function:** In machine learning, we usually have a loss function (like mean squared error, cross-entropy, etc.) that measures how well a model is performing. The goal is to minimize this loss function.
2. **Gradient:** The gradient of the loss function with respect to the model's parameters indicates the direction and rate of the steepest increase in the function. To minimize the loss function, we move in the opposite direction of the gradient.
3. **Descent Process:**
    - Start with initial random values for the parameters (weights, biases).
    - Calculate the gradient of the loss function with respect to each parameter.
    - Update the parameters by moving in the direction opposite to the gradient by a small step (controlled by the learning rate).
    - Repeat this process iteratively until the algorithm converges (i.e., the changes in loss become very small).

Formula:

The update rule for each parameter θ is:

$$\theta = \theta - \alpha \cdot \nabla J(\theta)$$

Where:

- θ represents the model parameters.
- α is the learning rate, a small positive value that controls how big each update step is.
- $\nabla J(\theta)$ is the gradient of the loss function $J(\theta)$ with respect to θ.

Visualization:

Imagine a mountain (the loss function), and you are trying to descend to the lowest point (global minimum). The gradient tells you the direction of the steepest slope, and gradient descent takes steps down this slope until you reach the bottom.

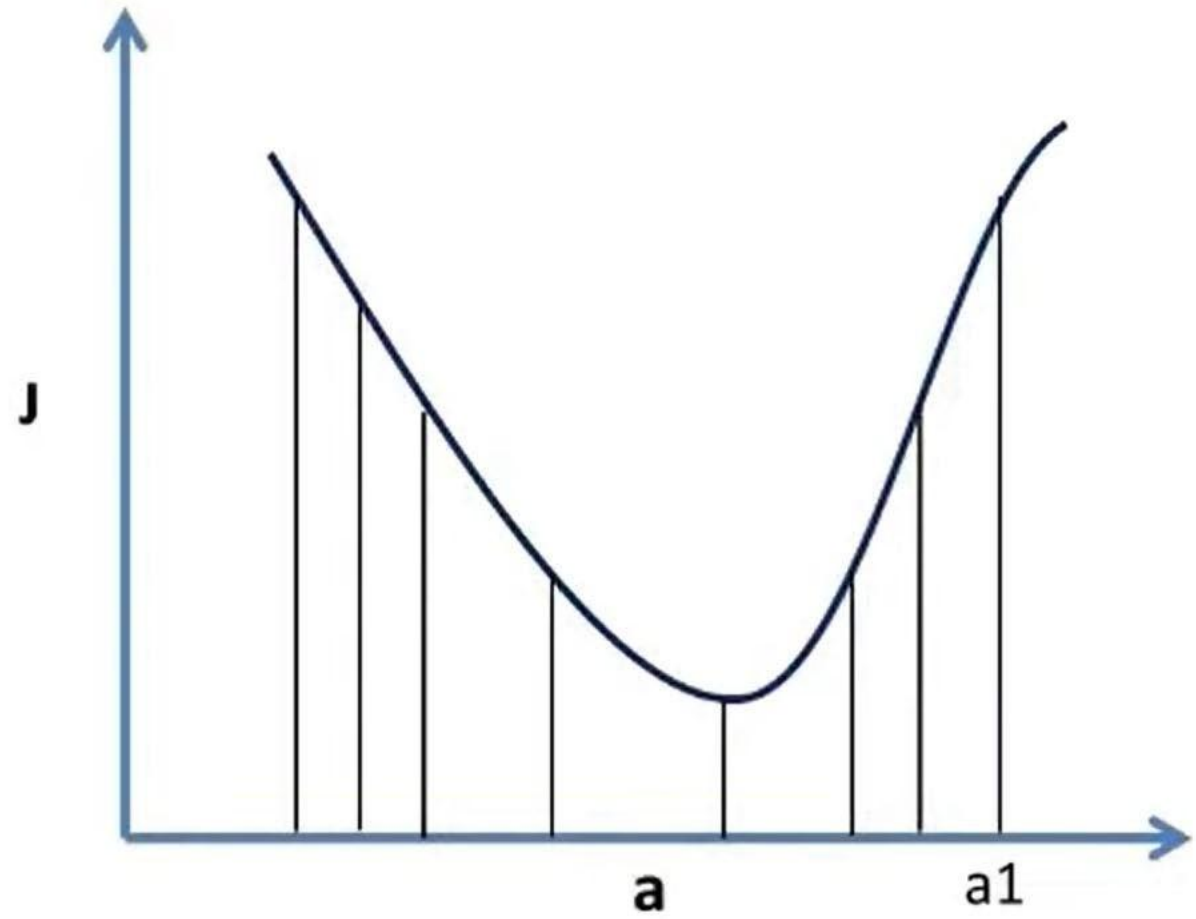
Challenges:

- **Learning Rate**: Choosing the right learning rate is crucial. Too small can make convergence slow, while too large can lead to overshooting and divergence.
- **Local Minima**: In non-convex loss functions, gradient descent might get stuck in local minima instead of finding the global minimum.
- **Saddle Points**: Flat regions can slow down the learning process.

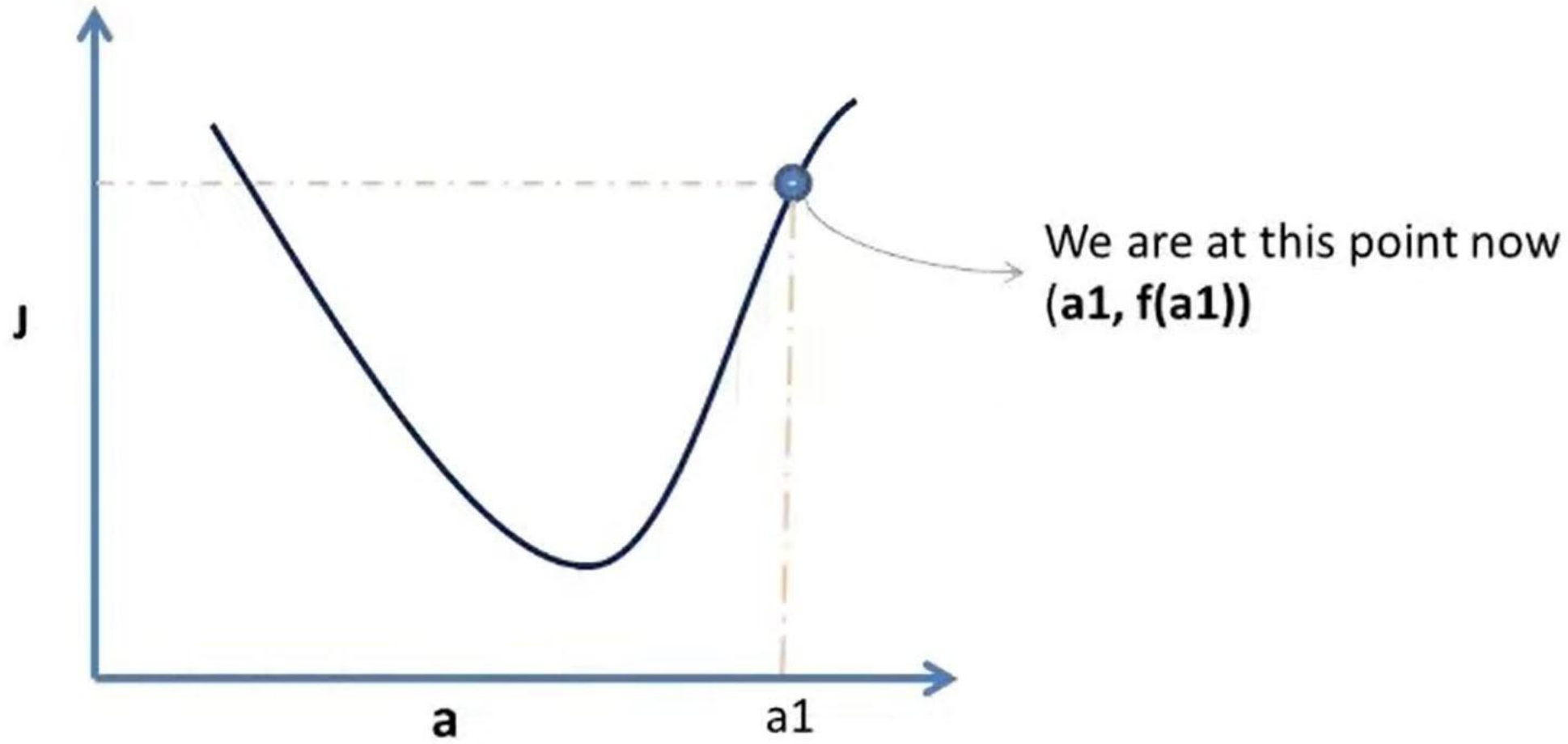
Gradient descent is fundamental to training many machine learning models, including linear regression, logistic regression, and neural networks.

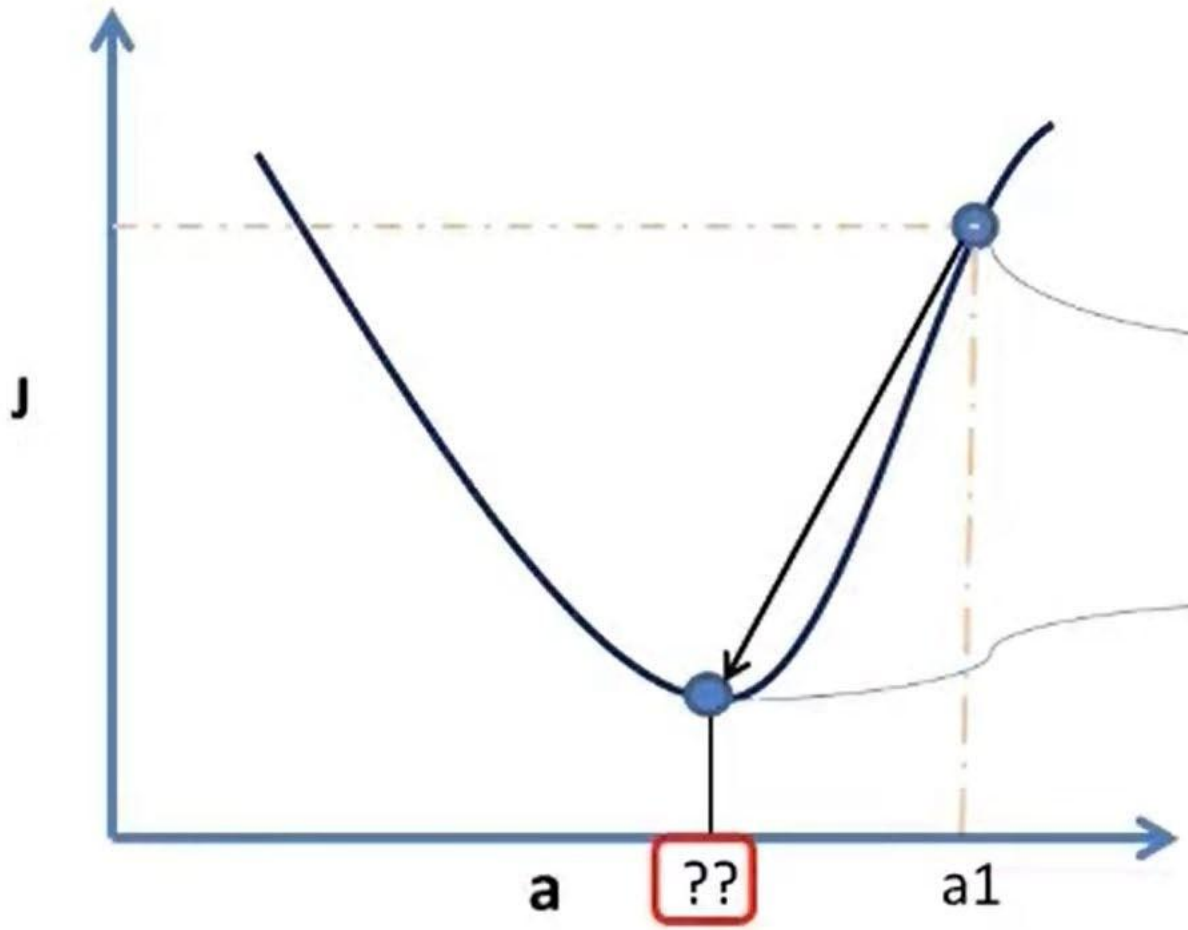
Forget **Cost function** for sometime.

Assume a function J = f(a)



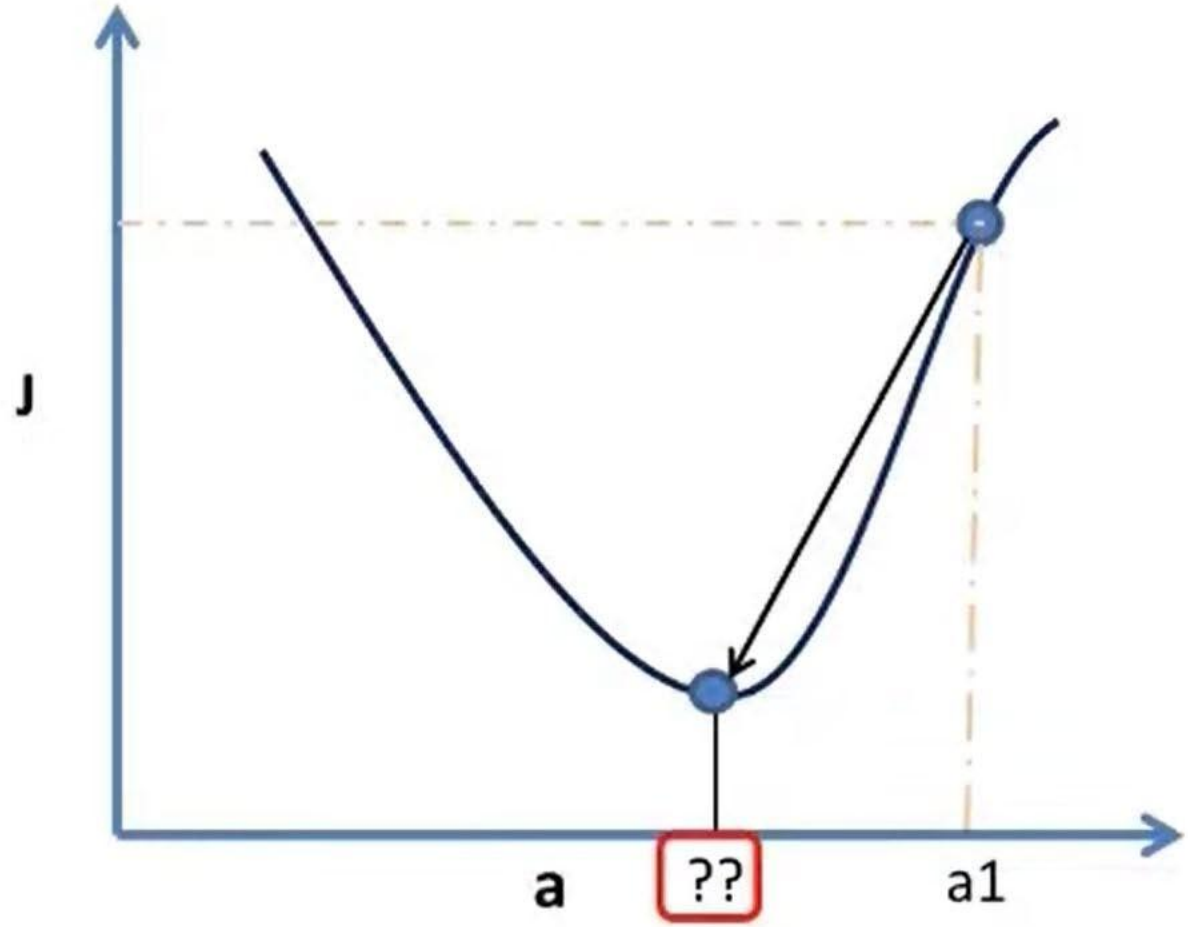
This curve J = f(a) represents the values **J** will assume for different values of '**a**'


J
a
a1
We are at this point now
(a1, f(a1))

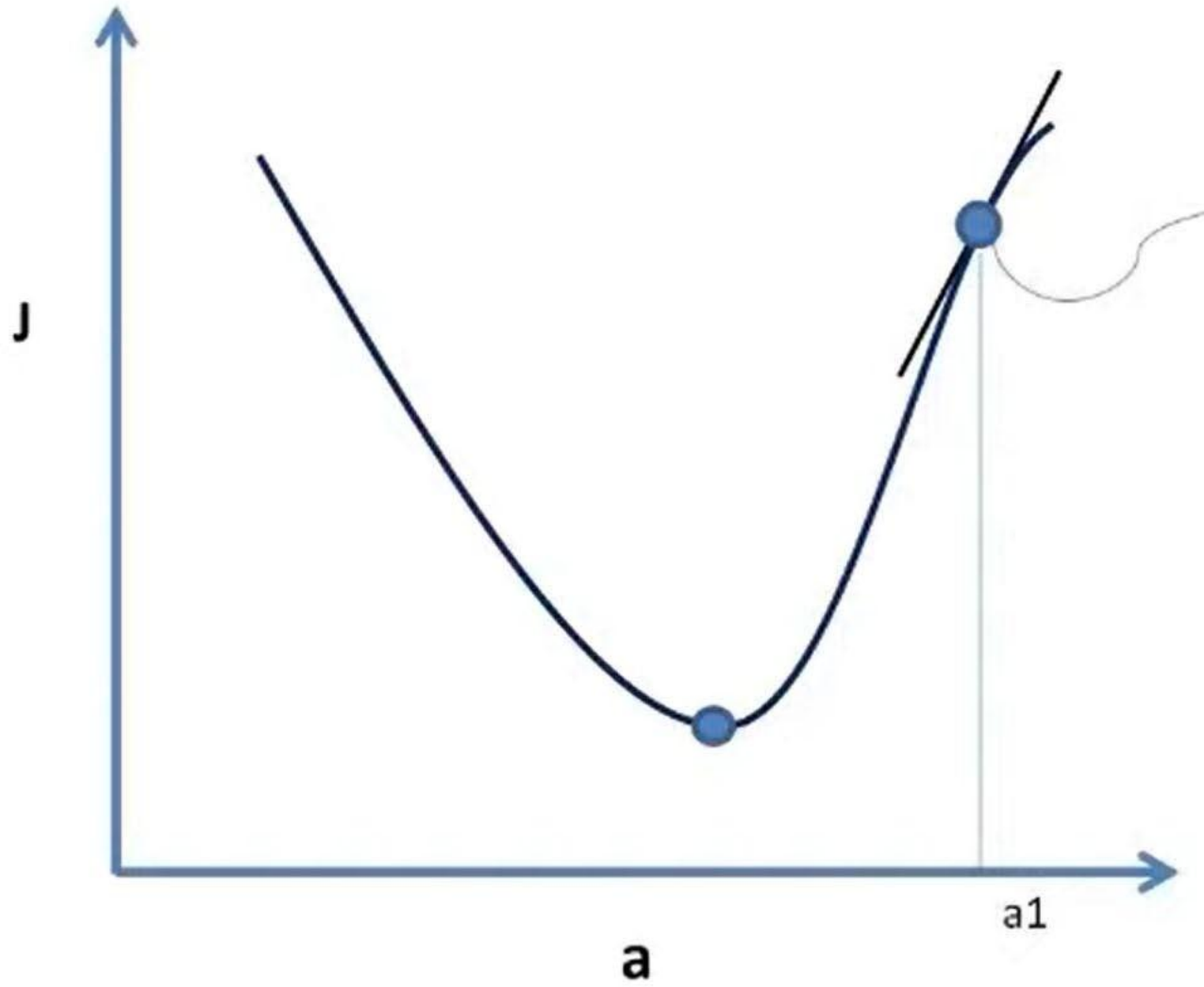


We are at this point now
(a1, f(a1))

We want to reach here, we want to know **for what value of 'a'** our **J = f(a) will assume minimum value**.

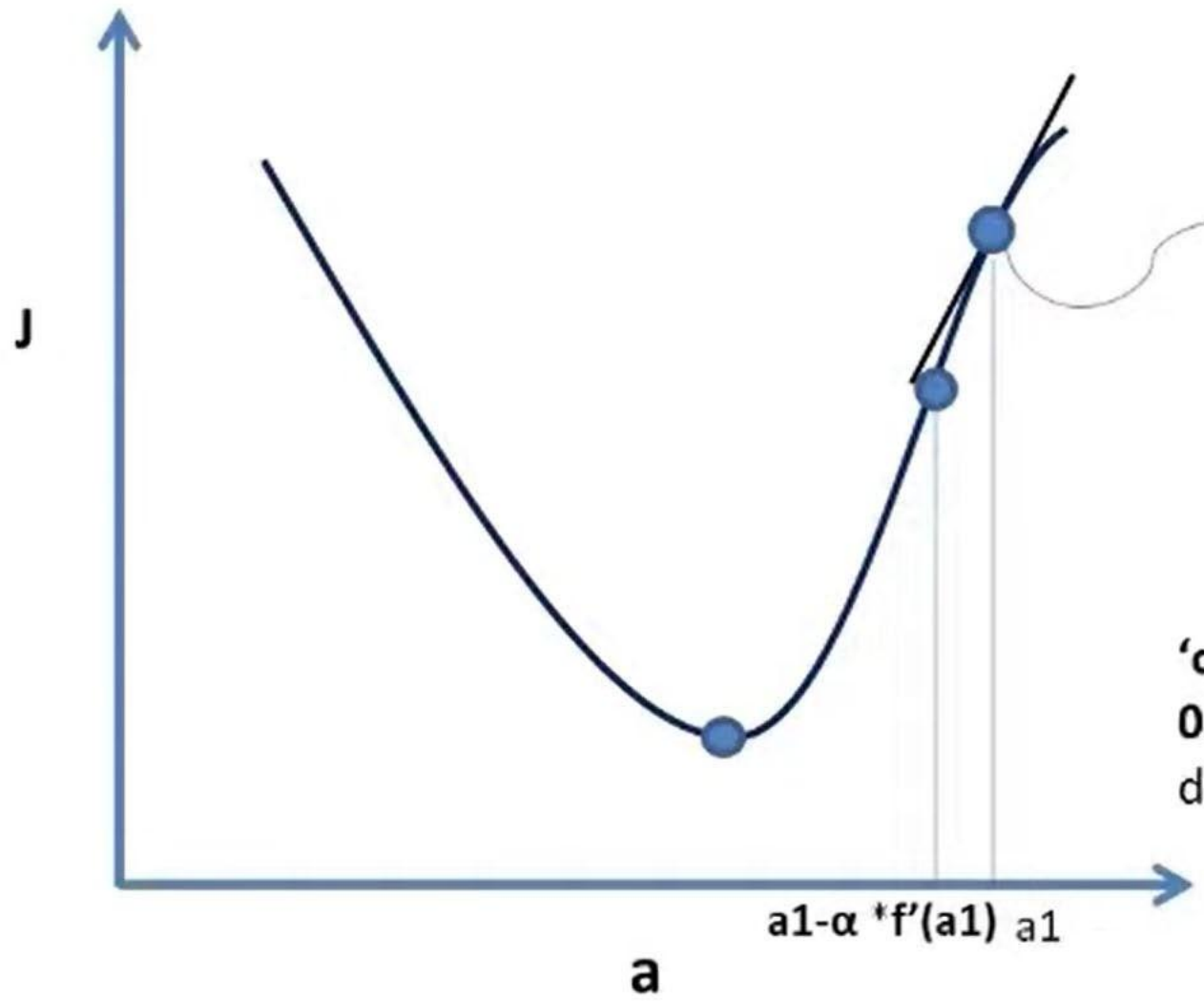


How can we reach the **minimum** starting from '**a1**' ?



Lets calculate slope at this point.

It is $\frac{dJ}{da}$ **at a1** or f'(a1)

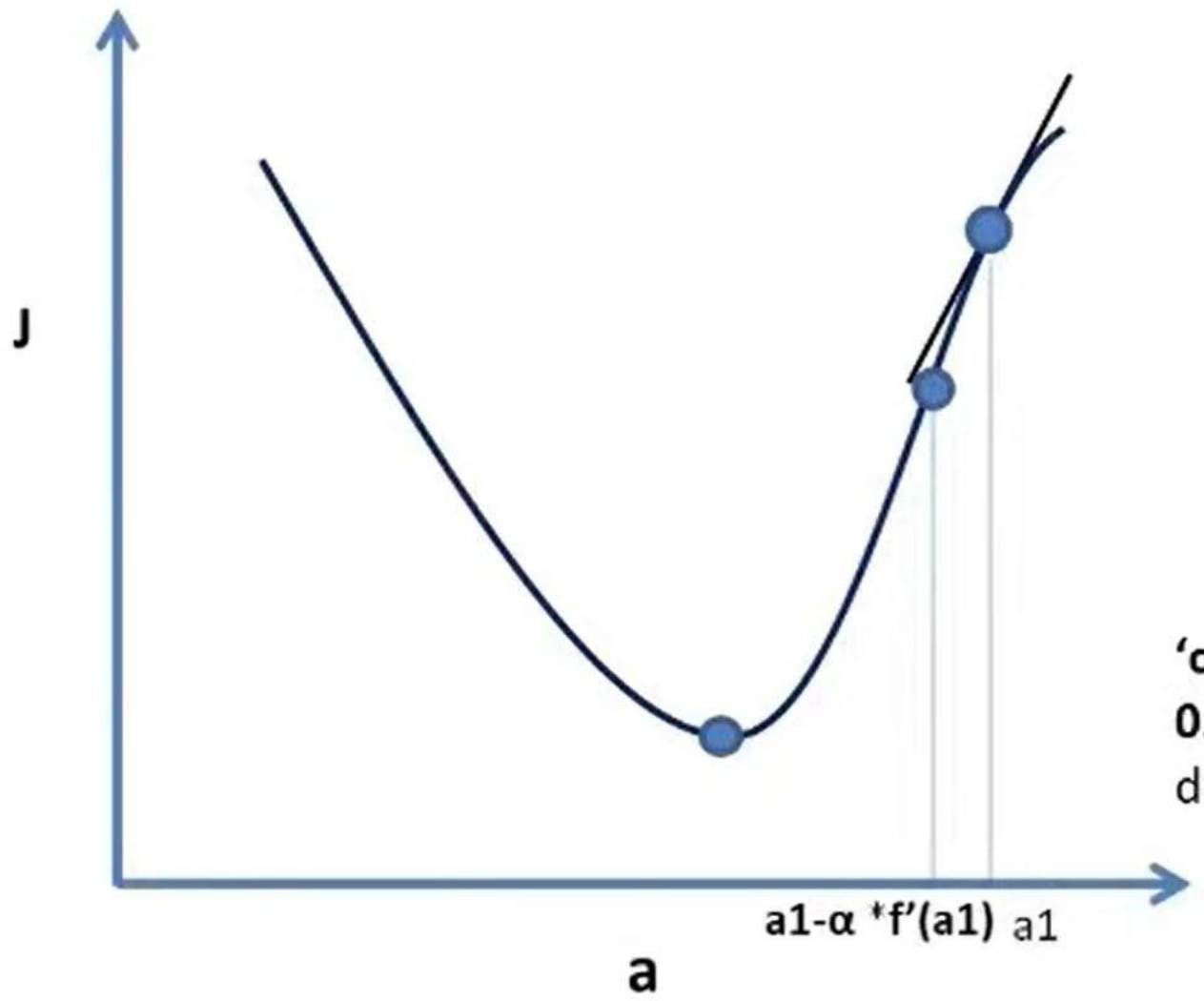


Lets calculate slope at this point.

It is $\frac{dJ}{da}$ **at a1** or f'(a1)

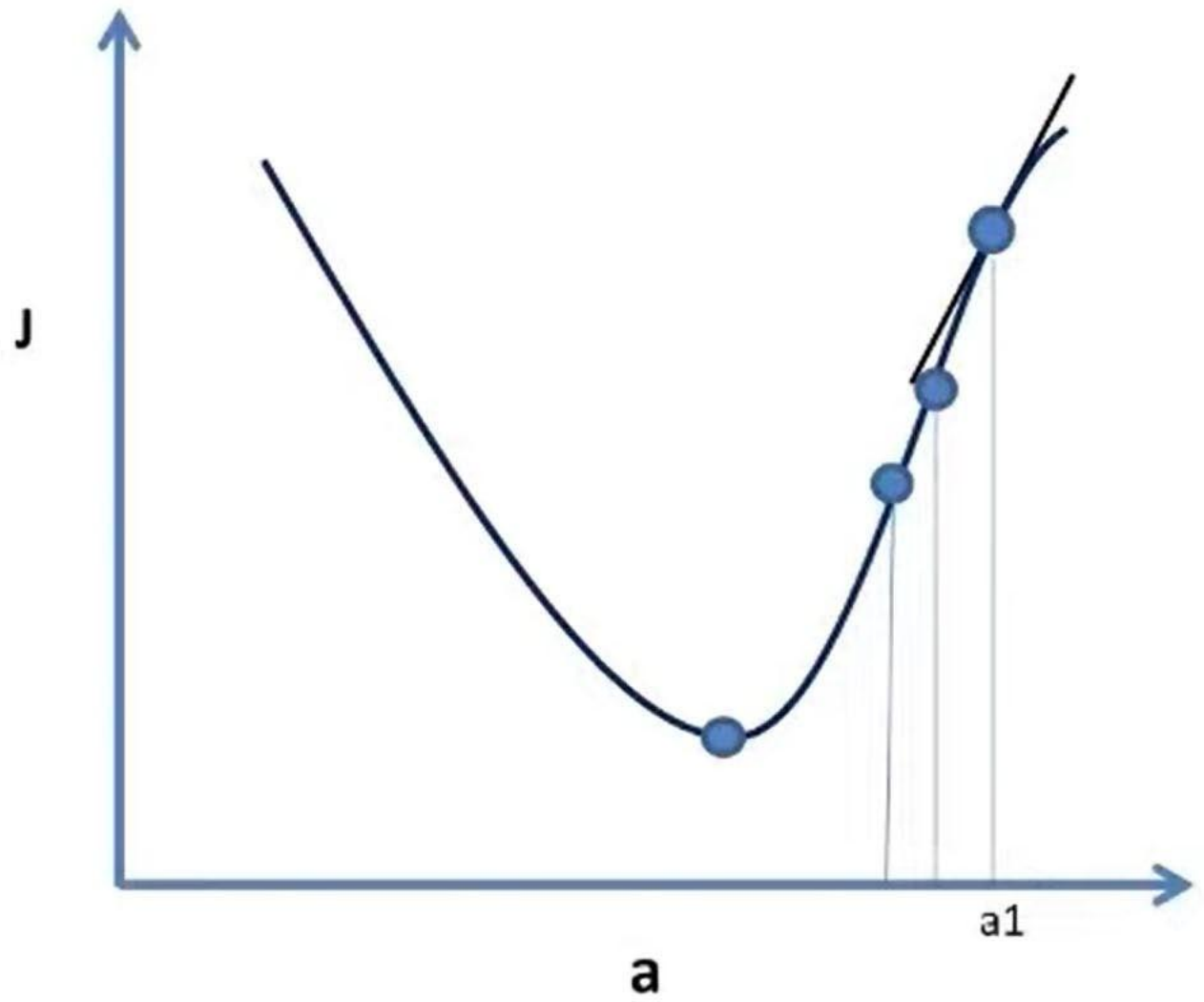
Lets move a step **'α'** in this direction to reach **a1- α *f'(a1)**

'α' is a small fixed quantity in the range of **0.01**. Therefore, the size of our steps is dependent on the slope **f'(a1)**.

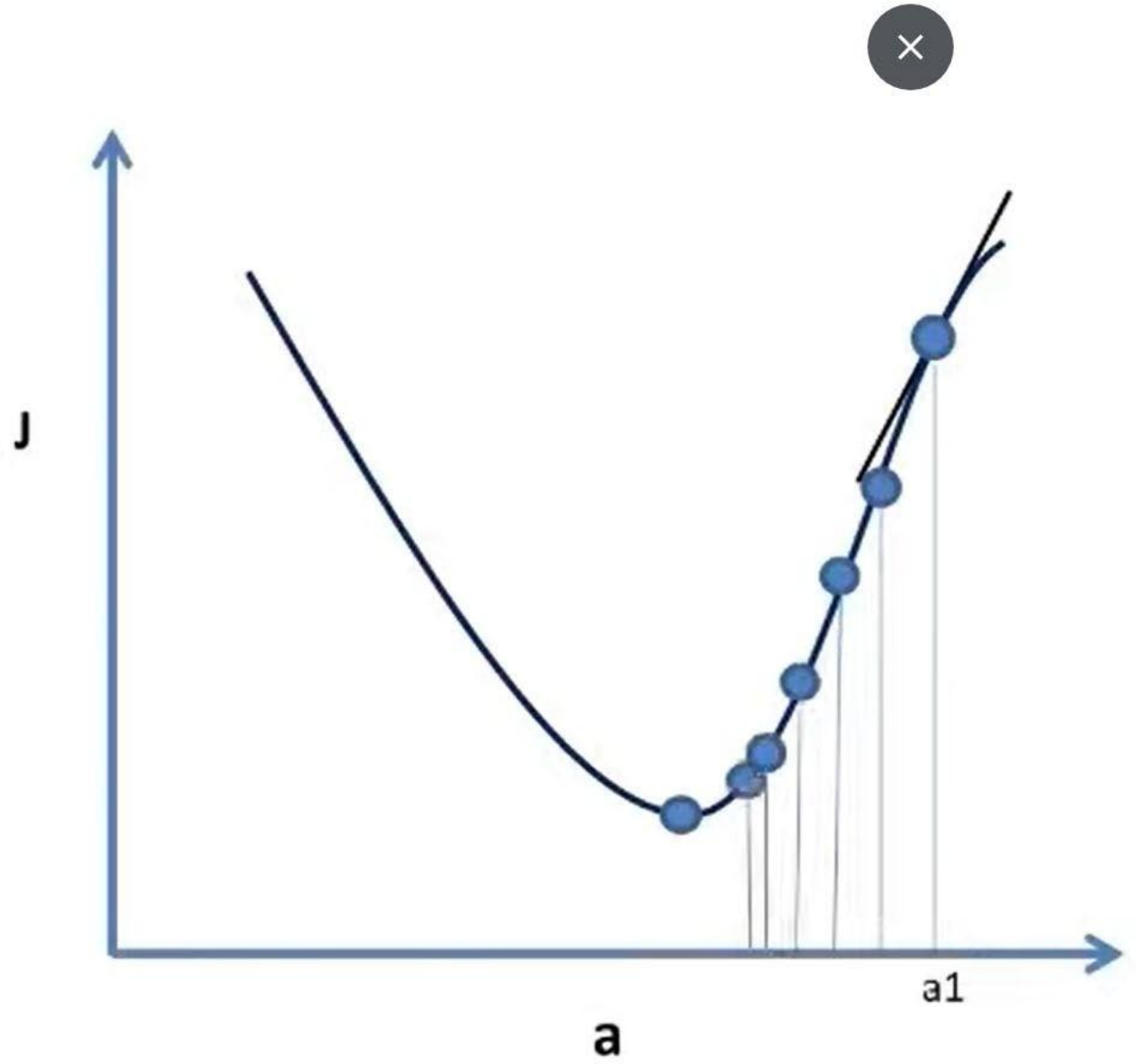


Higher the value of slope (which occurs away from minima) **larger the steps and vice versa.**

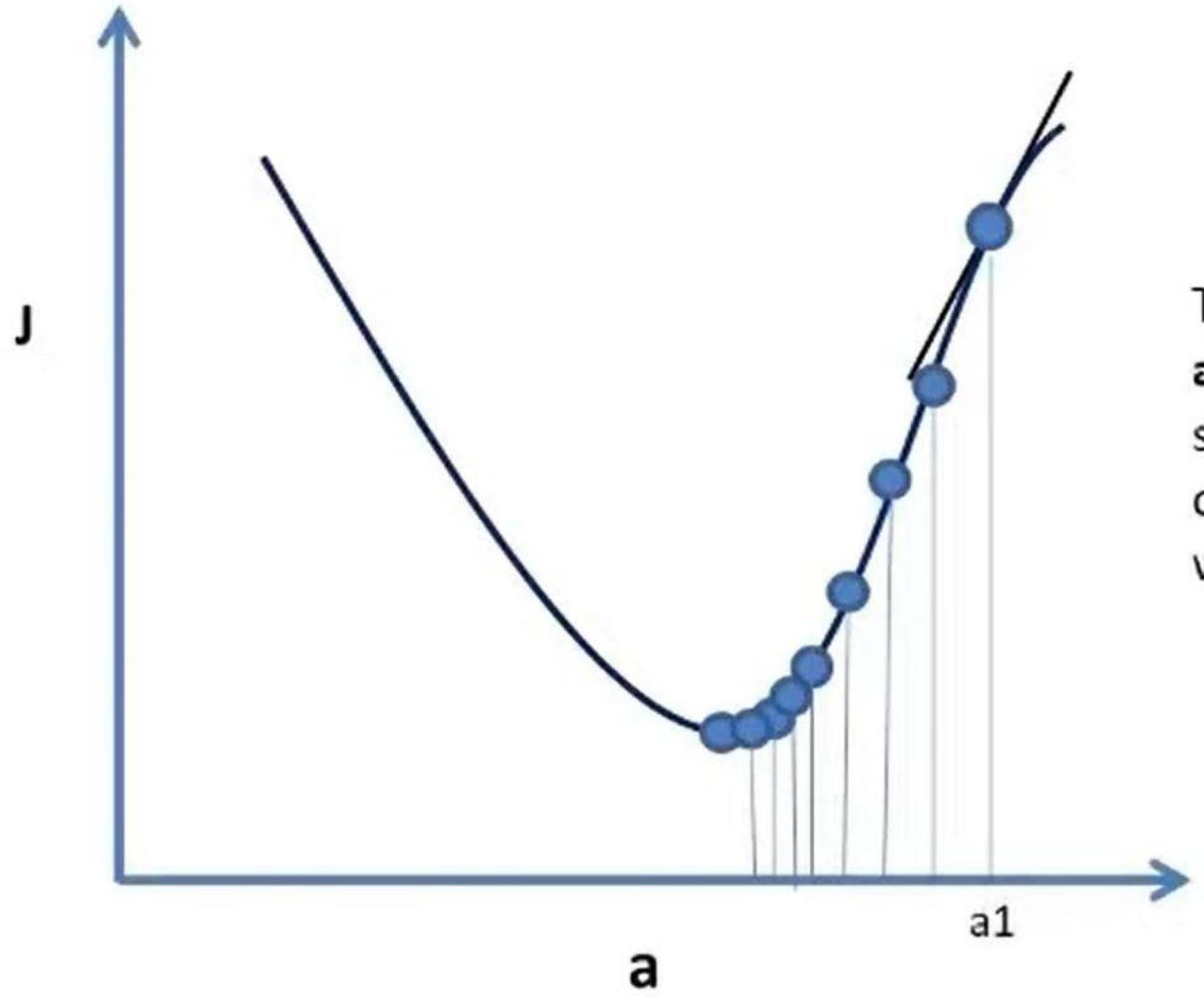
'α' is a small fixed quantity in the range of **0.01**. Therefore, the size of our steps is dependent on the slope **f'(a1)**.



As we move closer to minimum, **the slope decreases** and hence our **steps towards minimum keeps on getting smaller**. At minimum, slope becomes **zero**...



As we move closer to minimum, **the slope decreases** and hence our **steps towards minimum keeps on getting smaller**. At minimum, slope becomes **zero**...



These iteration of **Gradient Descent algorithm** can run in 1000s or 10000s of steps depending on **nature of function** and our '**α**' **(learning rate)** and of course where we start from, **'a1'** in this case

We will use the same methodology to **minimize cost function (J)** which is a function of model parameters $\boldsymbol{a_0}$ and $\boldsymbol{a_1}$ by changing them through iterations of **Gradient Descent Algorithm**.

We will use the same methodology to **minimize cost function (J)** which is a function of model parameters $\boldsymbol{a_0}$ and $\boldsymbol{a_1}$ by changing them through iterations of **Gradient Descent Algorithm**.

This is the part where it can get too mathematical for some people.

Step 1:

Calculate $\frac{\partial J}{\partial a_0}$ **(slope)** at the current value of parameter $\boldsymbol{a_0}$.

Calculate $\frac{\partial J}{\partial a_1}$ **(slope)** at the current value of parameter $\boldsymbol{a_1}$.

Step 2:

$$(\boldsymbol{new})\boldsymbol{a_0} = a_0 - \alpha\left(\frac{\partial J}{\partial a_0}\right)$$

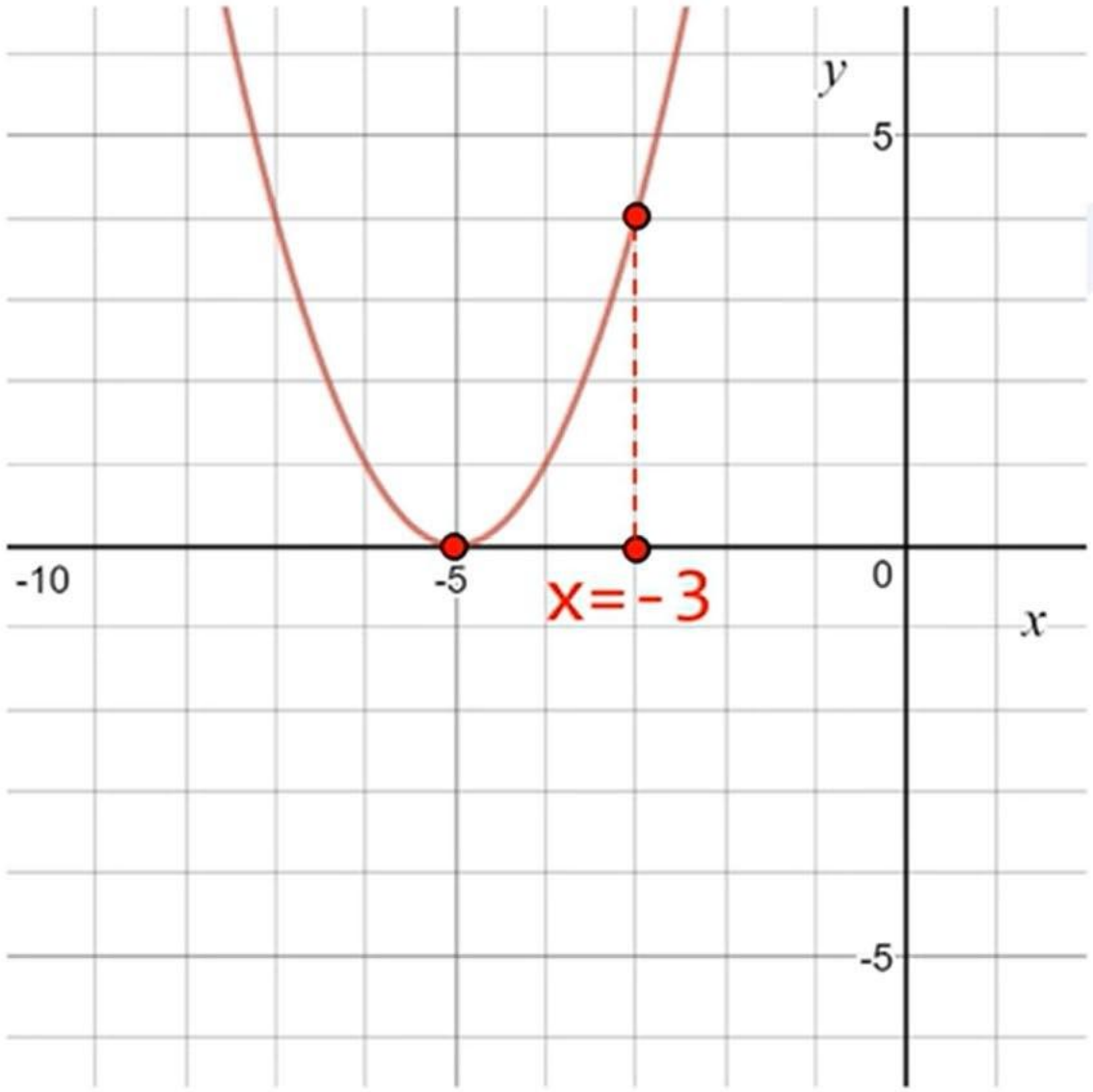
$$(\boldsymbol{new})\boldsymbol{a_1} = a_1 - \alpha\left(\frac{\partial J}{\partial a_1}\right)$$

Step 3:

update Cost Function **J** with **new ($\boldsymbol{a_0}$ and $\boldsymbol{a_1}$)**

Repeat Step 1

Example.



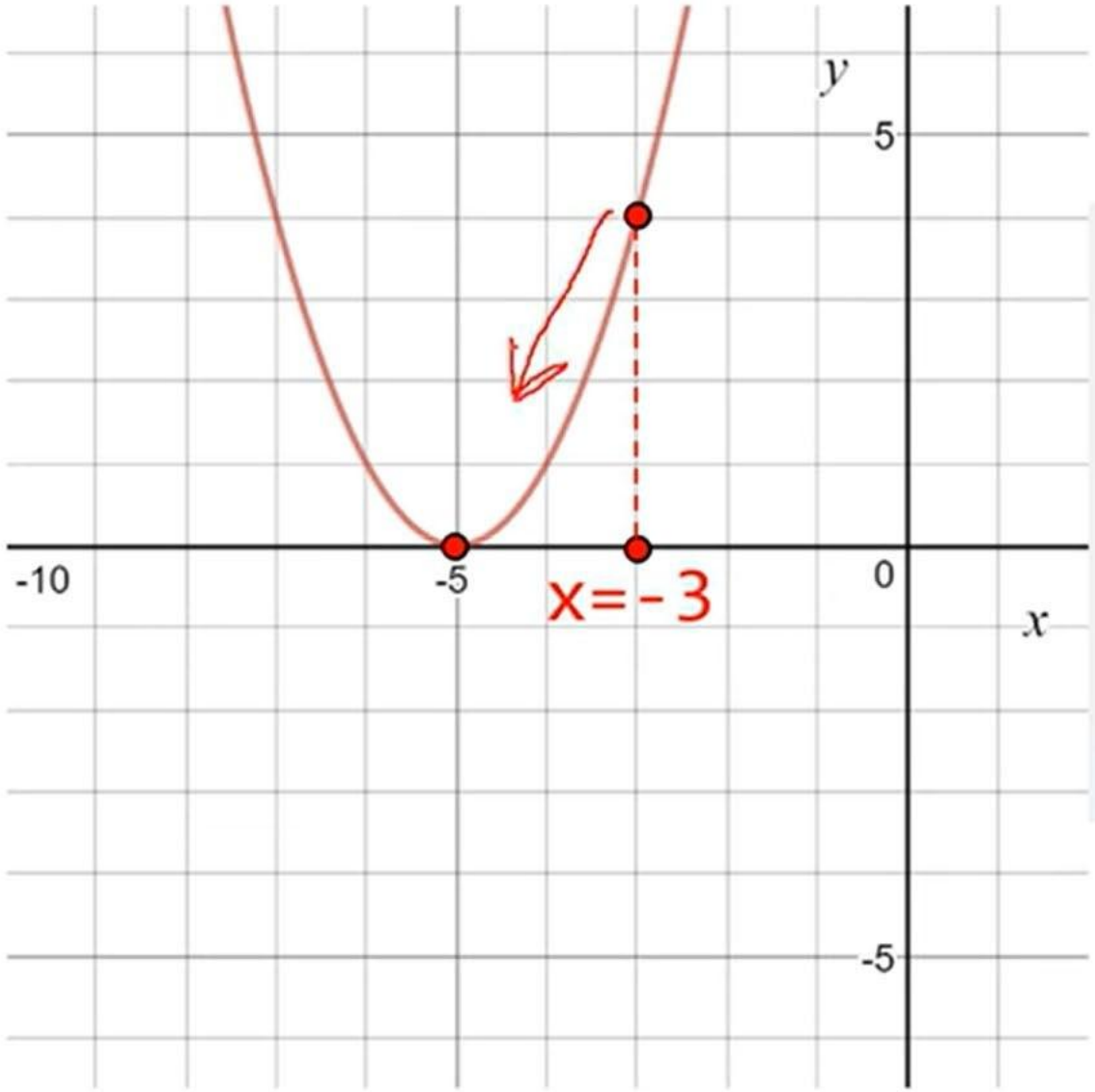
$y=(x+5)^2$

#Step1

Let‘s start from random point **x=-3**.

Then, find the gradient of the function, **dy/dx = 2×(x+5)**.

Example.



$y=(x+5)^2$

#Step1

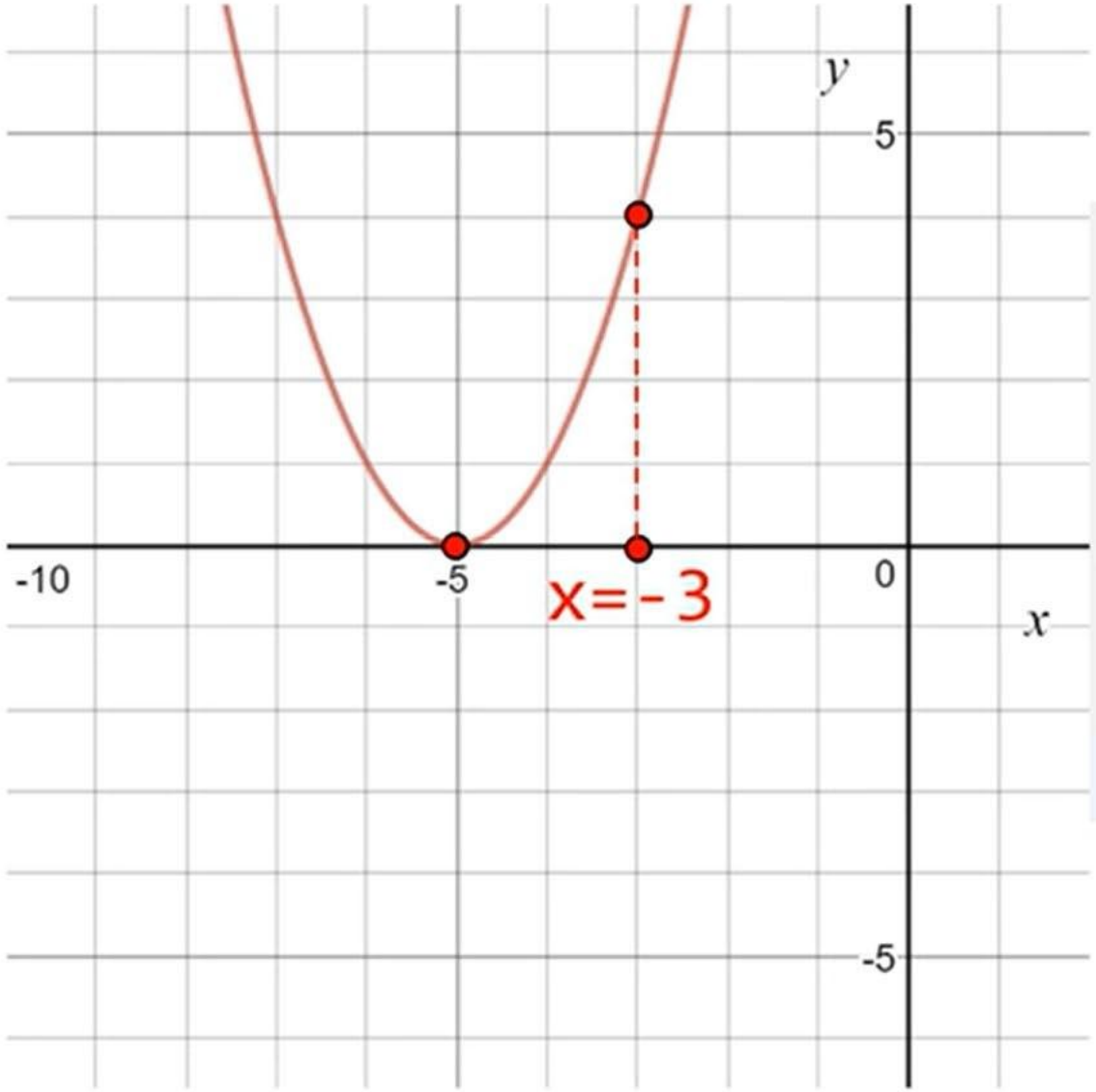
Let's start from random point **x=-3**.

Then, find the gradient of the function, **dy/dx = 2×(x+5)**.

#Step2

Move in the direction of the **negative of the gradient.**

Example.



$y=(x+5)^2$

#Step1

Let's start from random point **x=-3**.

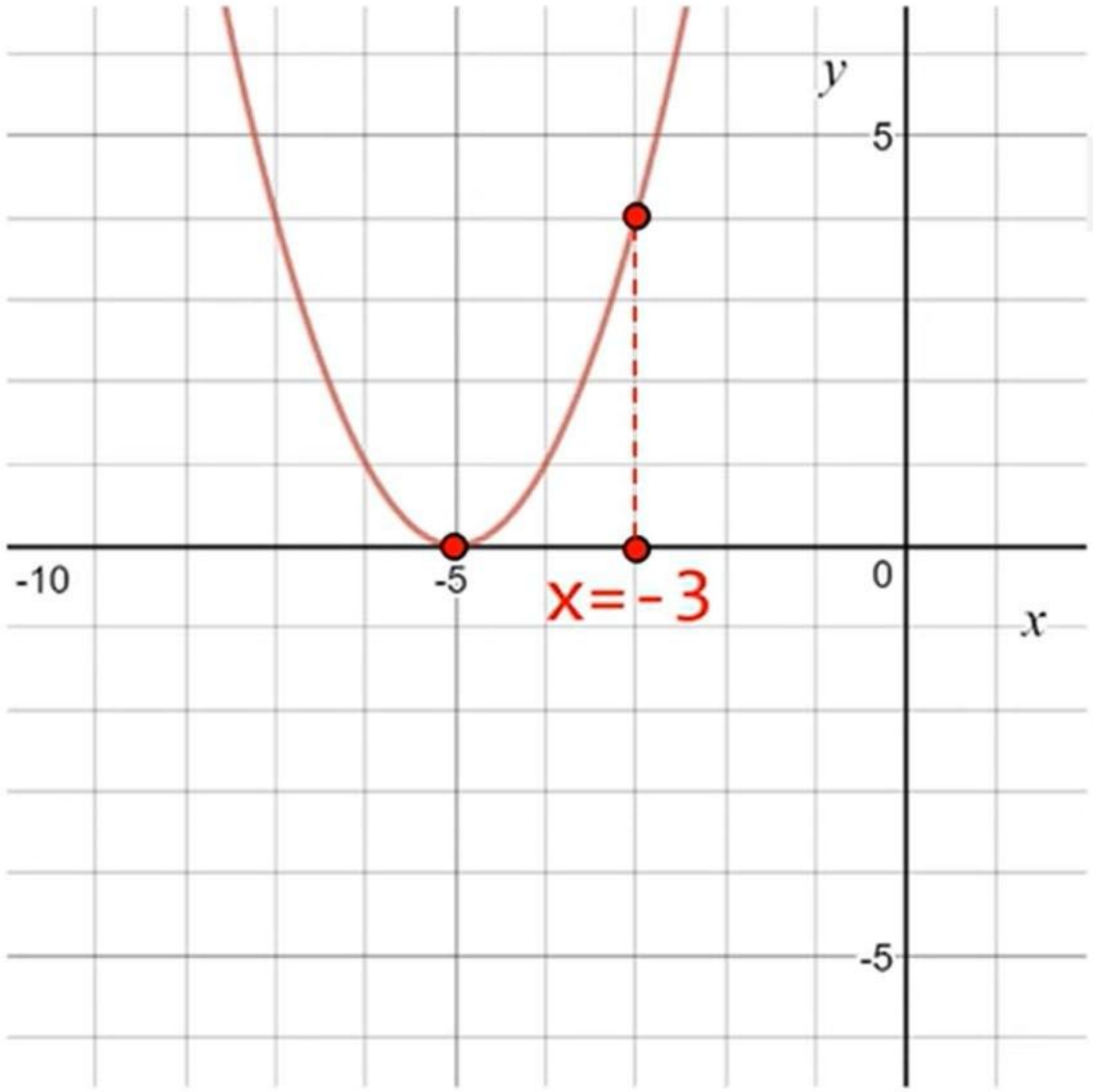
Then, find the gradient of the function, **dy/dx = 2×(x+5)**.

#Step2

Move in the direction of the **negative of the gradient.**

But: HOW MUCH to move? For that, we define a **learning rate: learning_rate = 0.01**

Example.



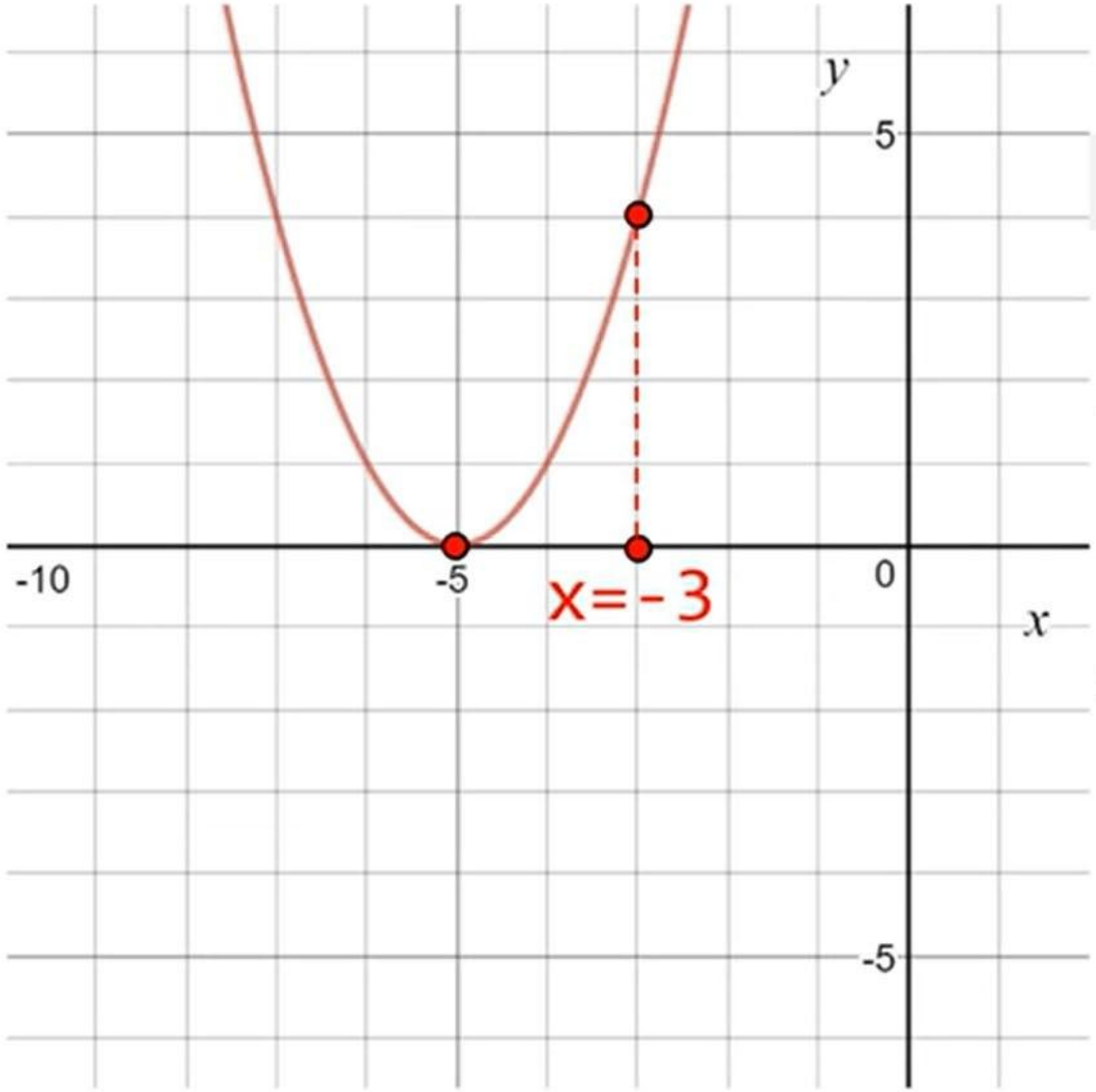
$y=(x+5)^2$

#Step3

Perform **2** iterations of gradient descent.

 # Initialize Parameters

Example.



$y=(x+5)^2$

#Step3

Perform **2** iterations of gradient descent.

 # Initialize Parameters

$X_0 = -3$

learning_rate = 0.01

$dy/dx = 2\times(x+5)$

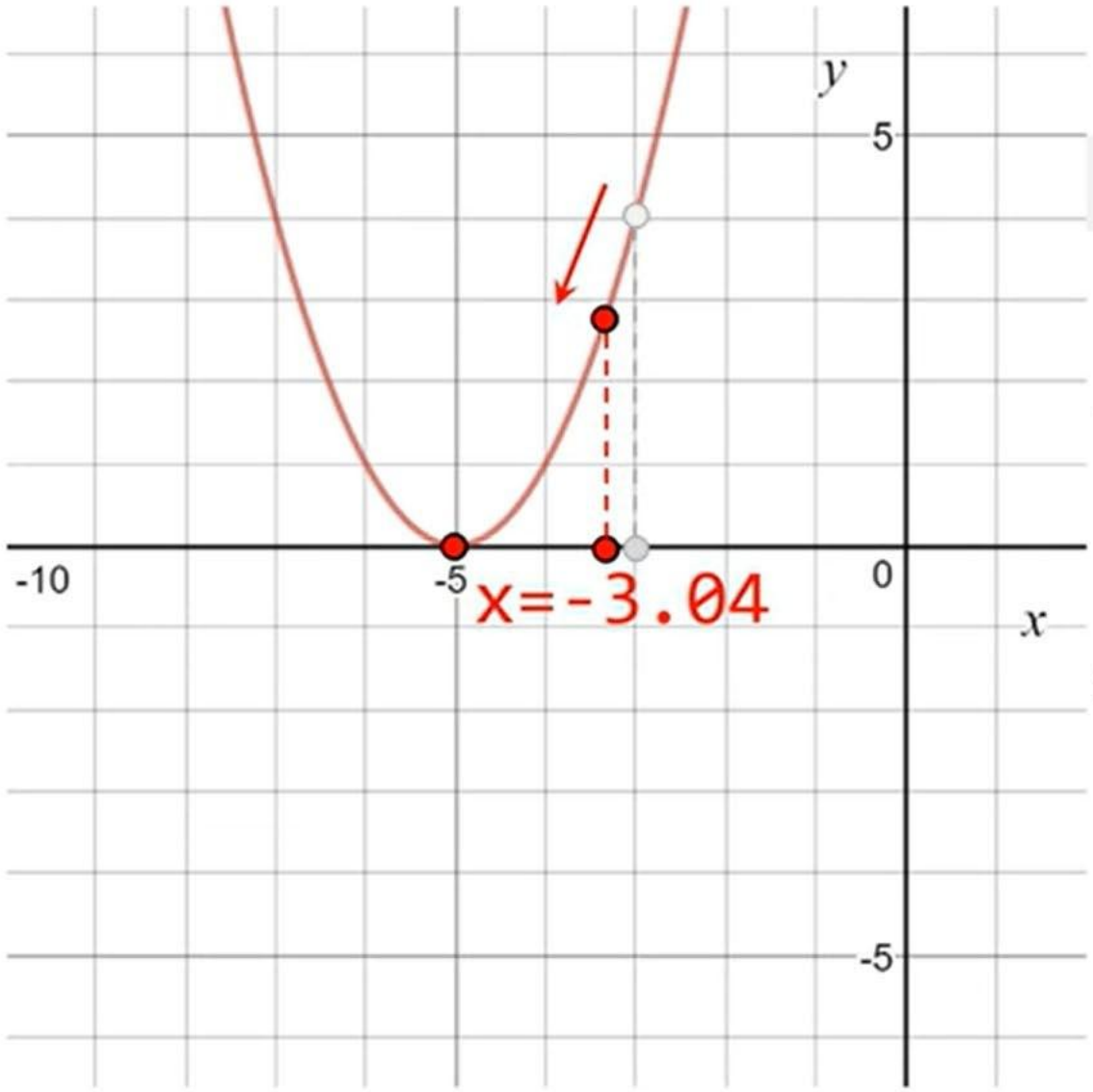
 # Iteration 1

$X_1 = X_0 - (\text{learning\_rate}) \times (dy/dx)$

$X_1 = (-3) - (0.01) \times (2 \times ((-3)+5)$

random_point　learning_rate　dy/dx

Example.



$y=(x+5)^2$

#Step3

Perform **2** iterations of gradient descent.

 # Initialize Parameters

$X_0 = -3$

learning_rate = 0.01

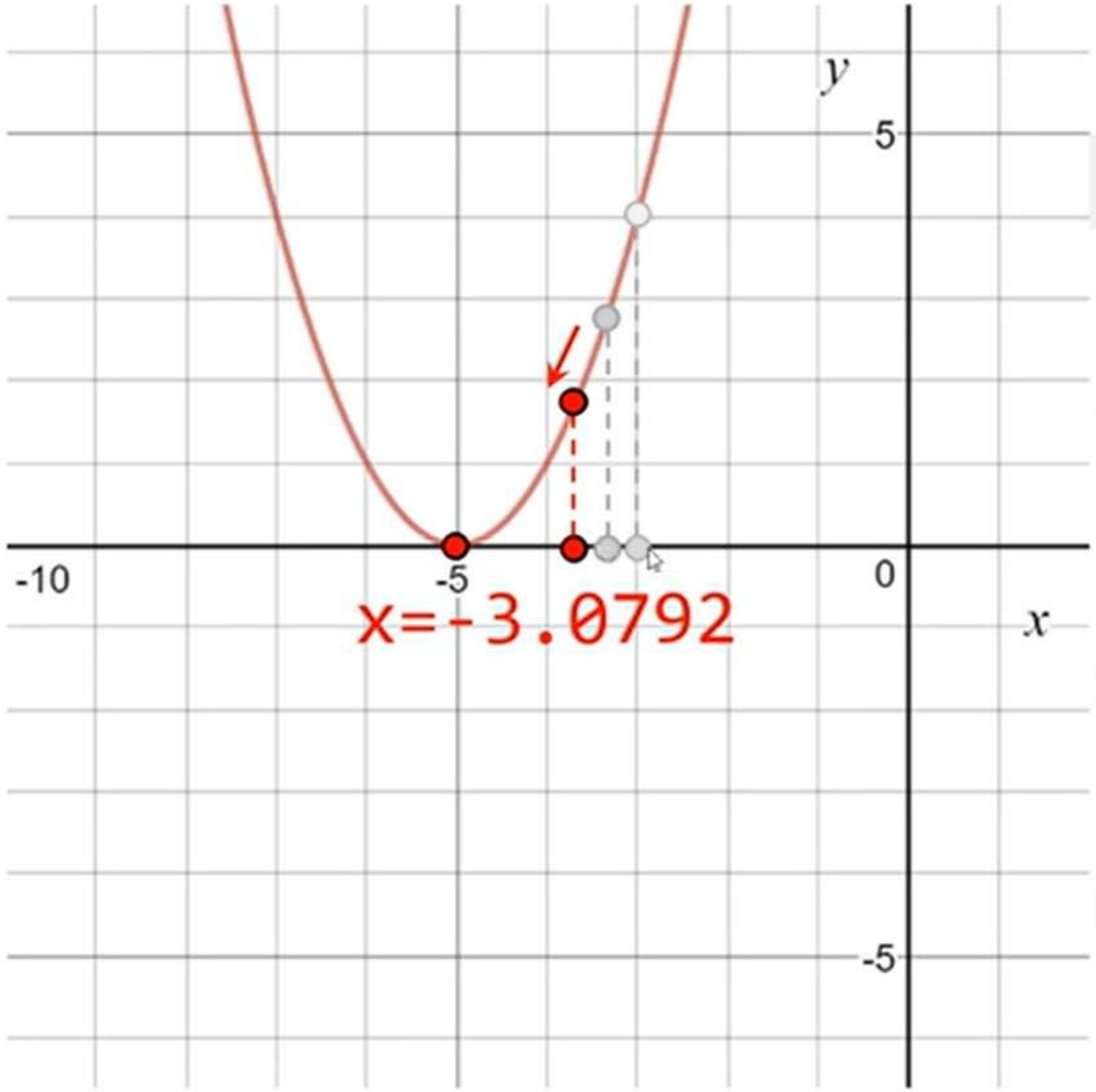
$dy/dx = 2\times(x+5)$

 # Iteration 1

$X_1 = X_0 - (\text{learning\_rate}) \times (dy/dx)$

$X_1 = (-3) - (0.01) \times (2 \times ((-3)+5)) = -3.04$

Example.



$y=(x+5)^2$

#Step3

Perform **2** iterations of gradient descent.

 # Initialize Parameters

$X_0 = -3$

learning_rate = 0.01

$dy/dx = 2\times(x+5)$

 # Iteration 1

$X_1 = X_0 - (\underline{Learning\ rate}) \times (\underline{dy/dx})$

$X_1 = (-3) - (0.01) \times (2 \times ((-3)+5)) = -3.04$

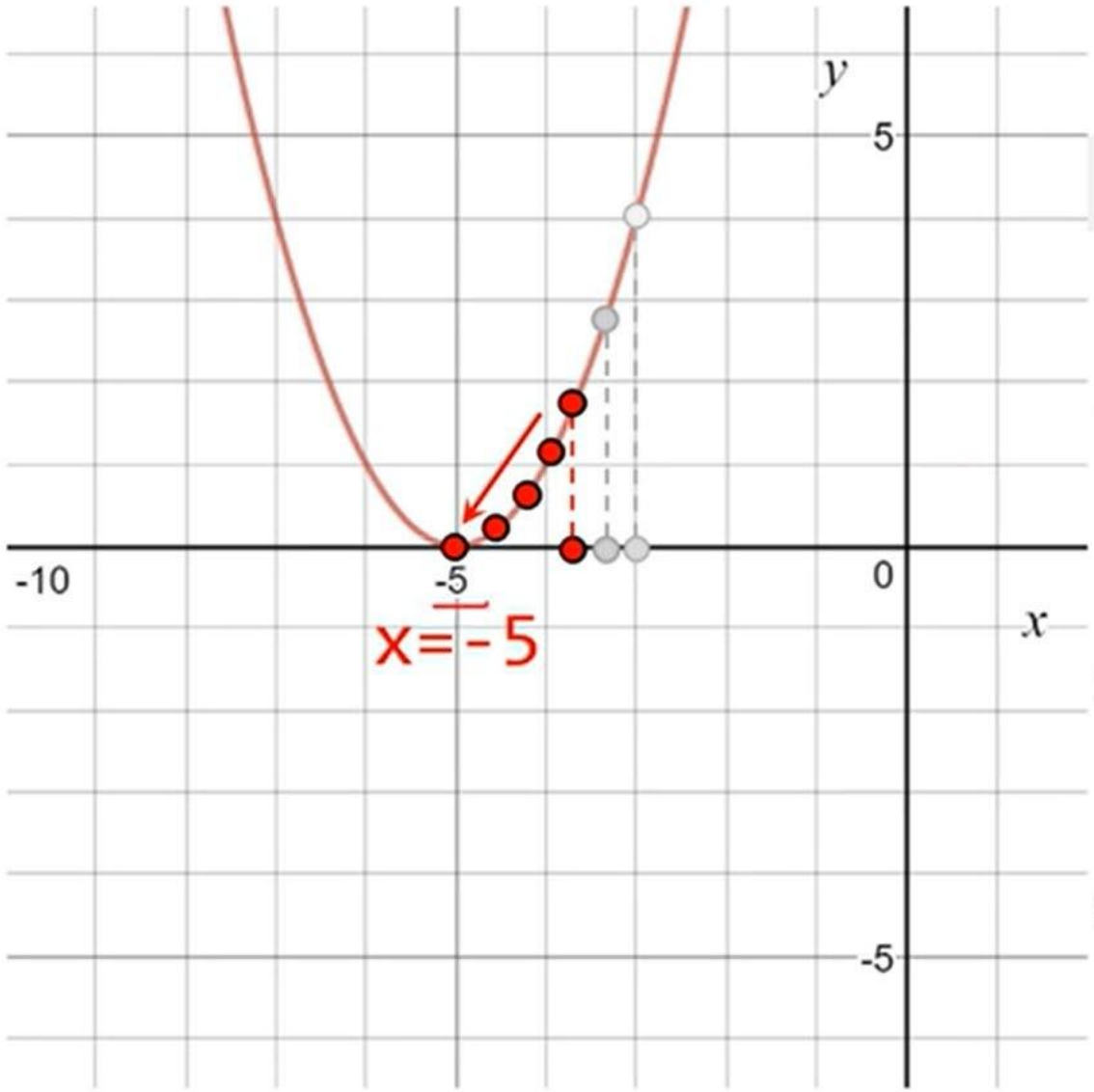
 # Iteration 2

$X_2 = X_1 - (\underline{Learning\ rate}) \times (\underline{dy/dx})$

$X_1 = (-3.04) - (0.01) \times (2 \times ((-3.04)+5)) =$

$= -3.0792$

Example.



$y=(x+5)^2$

#Step3

Perform **2** iterations of gradient descent.

 # Initialize Parameters

$X_0 = -3$

learning_rate = 0.01

$dy/dx = 2\times(x+5)$

 # Iteration 1

$X_1 = X_0 - (\underline{\text{Learning rate}}) \times (\underline{dy/dx})$

$X_1 = (-3) - (0.01) \times (2 \times ((-3)+5)) = -3.04$

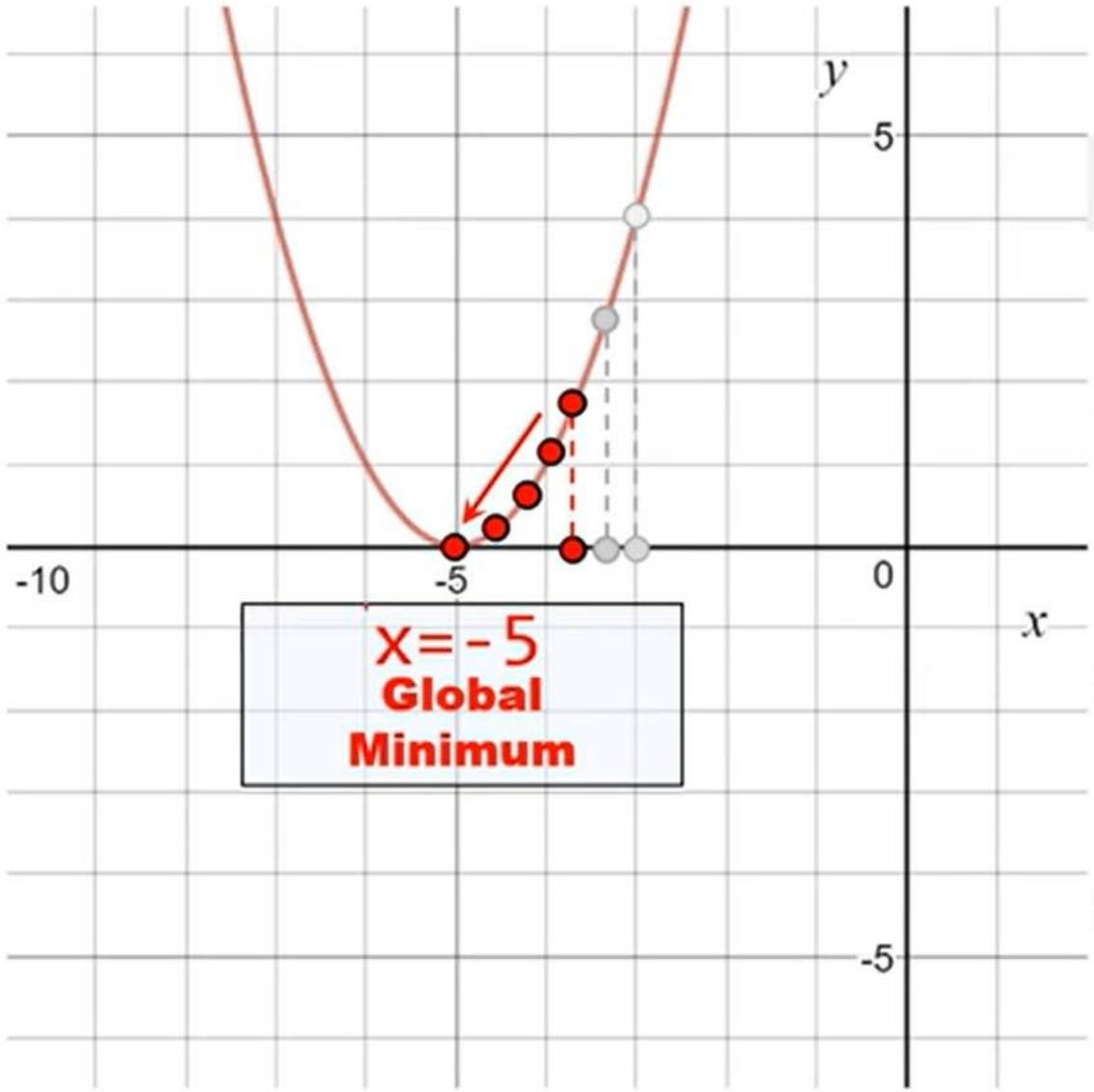
 # Iteration 2

$X_2 = X_1 - (\underline{\text{Learning rate}}) \times (\underline{dy/dx})$

$X_1 = (-3.04) - (0.01) \times (2 \times ((-3.04)+5)) =$

$= -3.0792$

Example.



$y=(x+5)^2$

#Step3

Perform **2** iterations of gradient descent.

 # Initialize Parameters

$X_0 = -3$

learning_rate = 0.01

$dy/dx = 2\times(x+5)$

 # Iteration 1

$X_1 = X_0 - (\underline{Learning\ rate}) \times (\underline{dy/dx})$

$X_1 = (-3) - (0.01) \times (2 \times ((-3)+5)) = -3.04$

 # Iteration 2

$X_2 = X_1 - (\underline{Learning\ rate}) \times (\underline{dy/dx})$

$X_1 = (-3.04) - (0.01) \times (2 \times ((-3.04)+5)) =$

$= -3.0792$

Question: Finding Global Minimum using Gradient Descent

Given the function:

$$f(x) = x^2 + 4x + 5$$

Use **Gradient Descent** to find the **global minimum** of the function.

Given:

- Initial value: $x_0 = 0$
- Learning rate (α) = 0.1
- Number of iterations = **3**

Given Function

$$f(x) = x^2 + 4x + 5$$

Step 1: First Derivative

$$f'(x) = 2x + 4$$

Step 2: Gradient Descent Update Rule

$$x_{n+1} = x_n - \alpha f'(x_n)$$

Given:

- Initial value $x_0 = 0$
- Learning rate $\alpha = 0.1$

Iteration 1

$$f'(0) = 2(0) + 4 = 4$$

$$x_1 = 0 - 0.1(4) = -0.4$$

Iteration 2

$$f'(-0.4) = 2(-0.4) + 4 = 3.2$$

$$x_2 = -0.4 - 0.1(3.2) = -0.72$$

Iteration 3

$$f'(-0.72) = 2(-0.72) + 4 = 2.56$$

$$x_3 = -0.72 - 0.1(2.56) = -0.976$$

Approximate Minimum after 3 Iterations

$$x \approx -0.976$$

6. What is the gradient of the function $2x^2 - 3y^2 + 4y - 10$ at point (0, 0)?

a) 0i + 4j

b) 1i + 10j

c) 2i – 3j

d) -3i + 4j

View Answer

Answer: a

Explanation: Given the function $f = 2x^2-3y^2+4y-10$ at point (0, 0). Then the gradient of the function can be calculated as:

$$\frac{\partial f}{\partial x} = \frac{\partial(2x^2-3y^2+4y-10)}{\partial x}$$

= 4x

= 4 * 0

= 0

$$\frac{\partial f}{\partial y} = \frac{\partial(2x^2-3y^2+4y-10)}{\partial y}$$

= -6y + 4

= (-6 * 0) + 4

= 0 + 4

= 4

Gradient, ∇f = 0i + 4j

Gradient descent is an optimization algorithm used to minimize a function by iteratively moving towards the minimum of that function. It is widely used in machine learning, especially in training models like linear regression, logistic regression, neural networks, etc.

Let's go through a numerical example to illustrate how gradient descent works:

Example: Minimizing a Simple Quadratic Function

Consider the simple quadratic function:

$$f(x) = x^2 + 4x + 4$$

We want to find the value of x that minimizes this function.

1. **Compute the Gradient:**

   The gradient (or derivative) of $f(x)$ with respect to x is:

   $$f'(x) = 2x + 4$$

2. **Initialize Starting Point:**

   Let's start with an initial guess, $x_0 = 0$.

3. **Set Learning Rate:**

   The learning rate (α) is a small positive number that determines the size of the steps we take towards the minimum. Let's set $\alpha = 0.1$.

4. **Iterative Update Rule:**

   The update rule for gradient descent is:

   $$x_{\text{new}} = x_{\text{old}} - \alpha \cdot f'(x_{\text{old}})$$

   We will iterate this process several times to move x closer to the minimum.

Numerical Iterations:

Iteration	x_{old}	$f'(x_{\text{old}})$	Update Step ($\alpha \cdot f'(x_{\text{old}})$)	x_{new}
0	0.0	4	$0.1 \times 4 = 0.4$	0.0 - 0.4 = -0.4
1	-0.4	3.2	$0.1 \times 3.2 = 0.32$	-0.4 - 0.32 = -0.72
2	-0.72	2.56	$0.1 \times 2.56 = 0.256$	-0.72 - 0.256 = -0.976
3	-0.976	2.048	$0.1 \times 2.048 = 0.2048$	-0.976 - 0.2048 = -1.1808
4	-1.1808	1.6384	$0.1 \times 1.6384 = 0.16384$	-1.1808 - 0.16384 = -1.34464

Observations:

- With each iteration, the value of x moves closer to the minimum.
- The step size decreases as the gradient decreases, leading to convergence.

TYPES OF GRADIENT DESCENT

1. Batch Gradient Descent (BGD)

Definition:

Uses the **entire training dataset** to compute the gradient in each iteration.

Update Rule:

$$\theta = \theta - \alpha \frac{1}{n} \sum_{i=1}^{n} \nabla J(\theta)$$

Advantages:

- Stable and smooth convergence
- Guaranteed convergence for convex functions

Disadvantages:

- Computationally expensive for large datasets
- Slow for big data

Use Case:

Small datasets where accuracy is more important than speed.

2. Stochastic Gradient Descent (SGD)

Definition:

Updates parameters using **only one training example** at a time.

Update Rule:

$$\theta = \theta - \alpha \nabla J(\theta; x_i, y_i)$$

Advantages:

- Very fast updates
- Can escape local minima due to noise

Disadvantages:

- Noisy updates
- Loss function fluctuates instead of smoothly decreasing

Use Case:

Large datasets and online learning.

3. Mini-Batch Gradient Descent

Definition:

Uses a **small batch of samples** (e.g., 32, 64, 128) for each update.

Advantages:

- Faster than batch GD
- More stable than SGD
- Efficient GPU utilization

Disadvantages:

- Requires batch size tuning

Use Case:

Most commonly used in deep learning.

Lecture

01 Supervised Machine Learning

Feature Scaling

Feature Scaling

Feature scaling is a data preprocessing technique used to standardize the range of independent numerical features in a dataset. It is crucial for many machine learning algorithms to ensure that no single feature dominates the model due to its magnitude or unit differences, leading to better performance and faster convergence.

Prevents Feature Dominance: Without scaling, features with large numerical ranges (e.g., income in thousands of dollars) can disproportionately influence distance calculations and model outcomes compared to features with small ranges (e.g., age in years).

Improves Algorithm Performance: Many algorithms perform better when features are on a similar scale.

Faster Convergence: Gradient descent-based algorithms (like linear/logistic regression and neural networks) converge more quickly to the optimal solution when features are scaled, as it prevents inefficient oscillations in the learning process.

Essential for Distance-Based Algorithms: Algorithms that rely on distance metrics (e.g., k-Nearest Neighbors (k-NN), K-Means Clustering, Support Vector Machines (SVMs) with RBF kernels, PCA) are highly sensitive to the scale of features and require scaling for accurate results.

Common Feature Scaling Techniques

The two primary methods are Normalization and Standardization, along with other robust methods.

Standardization

- **Standardization or Z-Score Normalization** is the transformation of features by subtracting from mean and dividing by standard deviation.
- This is often called as Z-score

$$x_{\text{stand}} = \frac{x - \text{mean}(x)}{\text{standard deviation } (x)}$$

Standardization

- Standardization can be helpful in cases where the data follows a Gaussian distribution, however, this does not have to be necessarily true
- Geometrically - it translates the data to the mean vector of original data to the origin and squishes or expands the points if std dev is 1 respectively
- The shape of the distribution is not affected – as this transforms a normal distribution to a standard normal distribution
- Standardization - not affected by outliers as there is no predefined range of transformed features

Normalization

- **Normalization or Min-Max Scaling** is used to transform features to be on a similar scale
- This technique is to re-scales features between 0 and 1.
- For every feature, the minimum value gets transformed into 0, and the maximum value gets transformed into 1

$$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Standardization vs. Normalization

Normalization	Standardization
Minimum and maximum value of features are used for scaling	Mean and standard deviation is used for scaling.
It is used when features are of different scales.	It is used when we want to ensure zero mean and unit standard deviation.
Scales values usually between [0, 1]	It is not bounded to a certain range.
It is really affected by outliers.	It is much less affected by outliers.
This transformation squishes the n-dimensional data into an n-dimensional unit hypercube.	It translates the data to the mean vector of original data to the origin and squishes or expands.
It is useful when we don't know about the distribution	It is useful when the feature distribution is Normal or Gaussian.
It is a often called as Scaling Normalization	It is a often called as Z-Score Normalization.

Numerical

Normalization Example (Min-Max Scaling)

Problem: Scale the dataset `[10, 20, 30, 40, 50]` to a range of [0].

Formula: $X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$

Standardization Example (Z-Score)

Problem: Standardize the dataset `[10, 20, 30, 40, 50]` (assume it's a sample).

Formula: $Z = \frac{X - \mu}{\sigma}$ (where μ is mean, σ is standard deviation).

normalization

1. **Identify Min/Max:** $X_{min} = 10$, $X_{max} = 50$.
2. **Apply Formula:**
    1. For 10: $(10 - 10)/(50 - 10) = 0/40 = 0.0$
    2. For 20: $(20 - 10)/(50 - 10) = 10/40 = 0.25$
    3. For 30: $(30 - 10)/(50 - 10) = 20/40 = 0.50$
    4. For 40: $(40 - 10)/(50 - 10) = 30/40 = 0.75$
    5. For 50: $(50 - 10)/(50 - 10) = 40/40 = 1.0$

Standardization Example (Z-Score)

Problem: Standardize the dataset `[10, 20, 30, 40, 50]` (assume it's a sample).

Formula: $Z = \frac{X - \mu}{\sigma}$ (where μ is mean, σ is standard deviation).

1. **Calculate Mean (μ):** $(10 + 20 + 30 + 40 + 50)/5 = 150/5 = 30$.
2. **Calculate Standard Deviation (σ):**
    1. Squared Differences from Mean:
       $(10 - 30)^2 = 400, (20 - 30)^2 = 100, (30 - 30)^2 = 0, (40 - 30)^2 = 100, (50 - 30)^2 = 400$
       .
    2. Sum of Squares: $400 + 100 + 0 + 100 + 400 = 1000$.
    3. Variance (σ^2): $1000/(5 - 1) = 1000/4 = 250$. (Using sample variance)
    4. Standard Deviation (σ): $\sqrt{250} \approx 15.81$.
3. **Apply Formula:**
    1. For 10: $(10 - 30)/15.81 \approx -1.26$
    2. For 20: $(20 - 30)/15.81 \approx -0.63$
    3. For 30: $(30 - 30)/15.81 = 0.00$
    4. For 40: $(40 - 30)/15.81 \approx 0.63$
    5. For 50: $(50 - 30)/15.81 \approx 1.26$

Numerical Question: Gradient Descent

Consider the cost function:

$$J(w) = w^2 - 4w + 5$$

---

where w is a model parameter.

Given:

- Initial value of parameter $w_0 = 0$
- Learning rate $\alpha = 0.1$

Tasks:

(a) Find the gradient of the cost function.

(b) Perform **two iterations** of **gradient descent** to compute w_1 and w_2.

(c) State whether the cost is decreasing after each iteration.

Overfitting and underfitting

Overfitting and underfitting are key machine learning problems: **Overfitting** is when a model learns training data *too* well, including noise, leading to high training accuracy but poor performance on new data (high variance). **Underfitting** is when a model is too simple and fails to capture basic patterns, performing poorly on both training and new data (high bias). The goal is the **ideal fit**: a moderately complex model that balances capturing trends without memorizing noise, performing well generally.

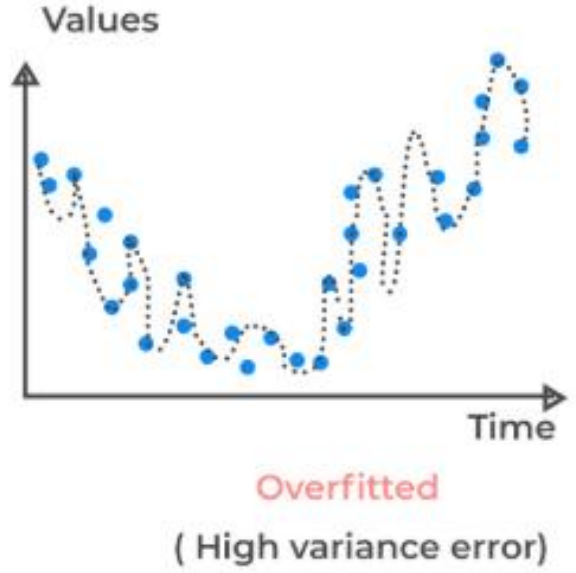
Overfitting (High Variance)

What it is: Model complexity is too high; it memorizes the training data and noise.

Symptoms: Very low error on training data, but high error on test/new data.

Analogy: A student who memorizes answers for one test but can't apply knowledge to new problems.

Solutions: Regularization, more data, cross-validation, dropout, pruning (for trees).



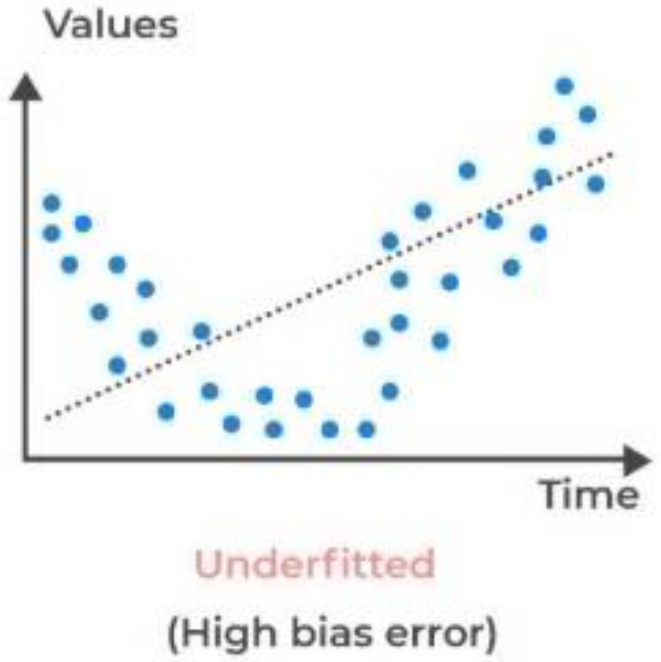
Underfitting (High Bias)

What it is: Model complexity is too low; it doesn't learn the underlying patterns.

Symptoms: High error on *both* training and test data.

Analogy: A student who didn't study enough and performs poorly everywhere.

Solutions: More complex model (more features, higher degree polynomial), longer training, different algorithm.



BIAS

Bias

Bias is the error introduced when a model is **too simple** to capture the underlying pattern in the data.

A high-bias model makes strong assumptions and may **underfit** the data.

Example: Linear regression applied to a highly nonlinear dataset.

Real-life example:

A student studies only **one chapter** and assumes the entire exam will come from that chapter. If the exam includes questions from many chapters, the student performs poorly.

👉 This is **high bias** because the assumption is too simple and causes **underperformance (underfitting)**.

VARIANCE

Variance

Variance is the error introduced when a model is **too complex** and is highly sensitive to small changes in the training data.
A high-variance model may **overfit** the data.

Example: A deep decision tree fitting noise in the training data.

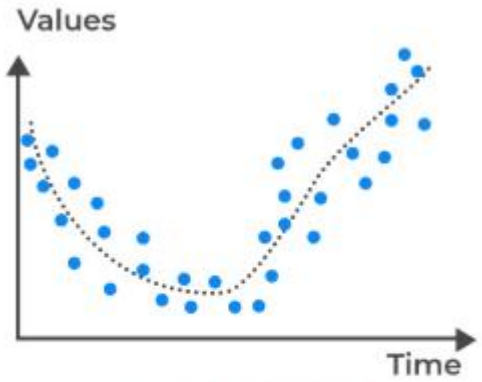
Real-life example:
A student changes their **study strategy every day** based on a single mock test result.
Because the approach keeps changing, the student performs well in some tests but poorly in others.
👉 This is **high variance** because the behavior depends too much on small variations (**overfitting**).

The Bias-Variance Trade-off

This is the balance between the two extremes.

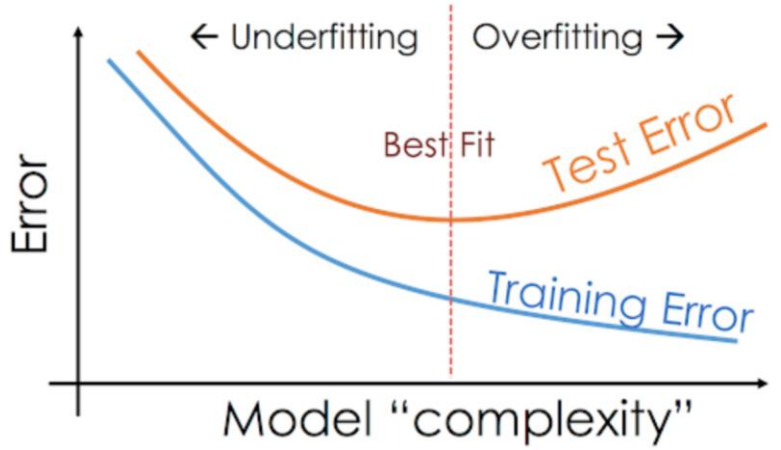
Total Error = Bias² + Variance + Irreducible Error.

You adjust model complexity (e.g., polynomial degree, tree depth, k in KNN) to find the sweet spot that minimizes total error, ensuring good generalization.



Good Fit/Robust

(Balance between bias and variance)



Regularization

Regularization is a technique used in machine learning to prevent **overfitting** by adding a penalty term to the loss function. This penalty discourages overly complex models by constraining the magnitude of model parameters (weights), thus improving generalization on unseen data.

How it works

Penalizes complexity: Regularization adds a penalty term to the model's loss function, making complex models with large weights less favorable during training

Shrinks coefficients: By penalizing large coefficients, it forces the model to use fewer features or smaller weights, reducing reliance on specific training data noise

Improves generalization: A simpler model is less likely to "memorize" the training data, leading to better performance on new data

Types of Regularization

- **L1 Regularization (Lasso)**: Adds the absolute value of the weights to the loss, which can shrink some weights to exactly zero, performing feature selection.
- **L2 Regularization (Ridge)**: Adds the squared value of the weights, shrinking weights but rarely to zero, preventing multicollinearity.

- **Elastic Net**: Combines both L1 and L2 regularization.

Ridge Regression (L2 Regularization) adds a penalty proportional to the **square of the coefficients** to the loss function. It shrinks coefficients toward zero but does **not make them exactly zero**. Ridge is useful when all input features contribute to the output and multicollinearity exists among features.

Ridge Regression (L2 Regularization) – Simple Form

The **Ridge Regression formula** is:

$$\text{Loss} = \text{Sum of Squared Errors} + \lambda \times \text{Sum of Squares of Weights}$$

Mathematically,

$$\boxed{\sum (y - \hat{y})^2 + \lambda \sum w^2}$$

Ridge Regression (L2 Regularization) – Simple Form

The **Ridge Regression formula** is:

$$\text{Loss} = \text{Sum of Squared Errors} + \lambda \times \text{Sum of Squares of Weights}$$

Mathematically,

$$\boxed{\sum (y - \hat{y})^2 + \lambda \sum w^2}$$

where:

- y = actual value
- $\hat{y}$ = predicted value
- w = model weights (coefficients)
- λ = regularization parameter

Meaning:

Ridge Regression reduces overfitting by **penalizing large weights**, but it does **not make them zero**.

Lasso Regression (L1 Regularization) adds a penalty proportional to the **absolute value of the coefficients**. This can force some coefficients to become **exactly zero**, effectively performing **feature selection**. Lasso is preferred when only a subset of features is expected to be important.

Lasso Regression (L1 Regularization) – Simple Form

The **Lasso Regression formula** is:

$$\text{Loss} = \text{Sum of Squared Errors} + \lambda \times \text{Sum of Absolute Weights}$$

Mathematically,

$$\boxed{\sum (y - \hat{y})^2 + \lambda \sum |w|}$$

Lasso Regression (L1 Regularization) – Simple Form

The **Lasso Regression formula** is:

$$\text{Loss} = \text{Sum of Squared Errors} + \lambda \times \text{Sum of Absolute Weights}$$

Mathematically,

$$\boxed{\sum (y - \hat{y})^2 + \lambda \sum |w|}$$

where:

- y = actual value
- $\hat{y}$ = predicted value
- w = model weights (coefficients)
- λ = regularization parameter

Meaning:

Lasso Regression reduces overfitting and can make some weights **exactly zero**, so it also performs **feature selection**.