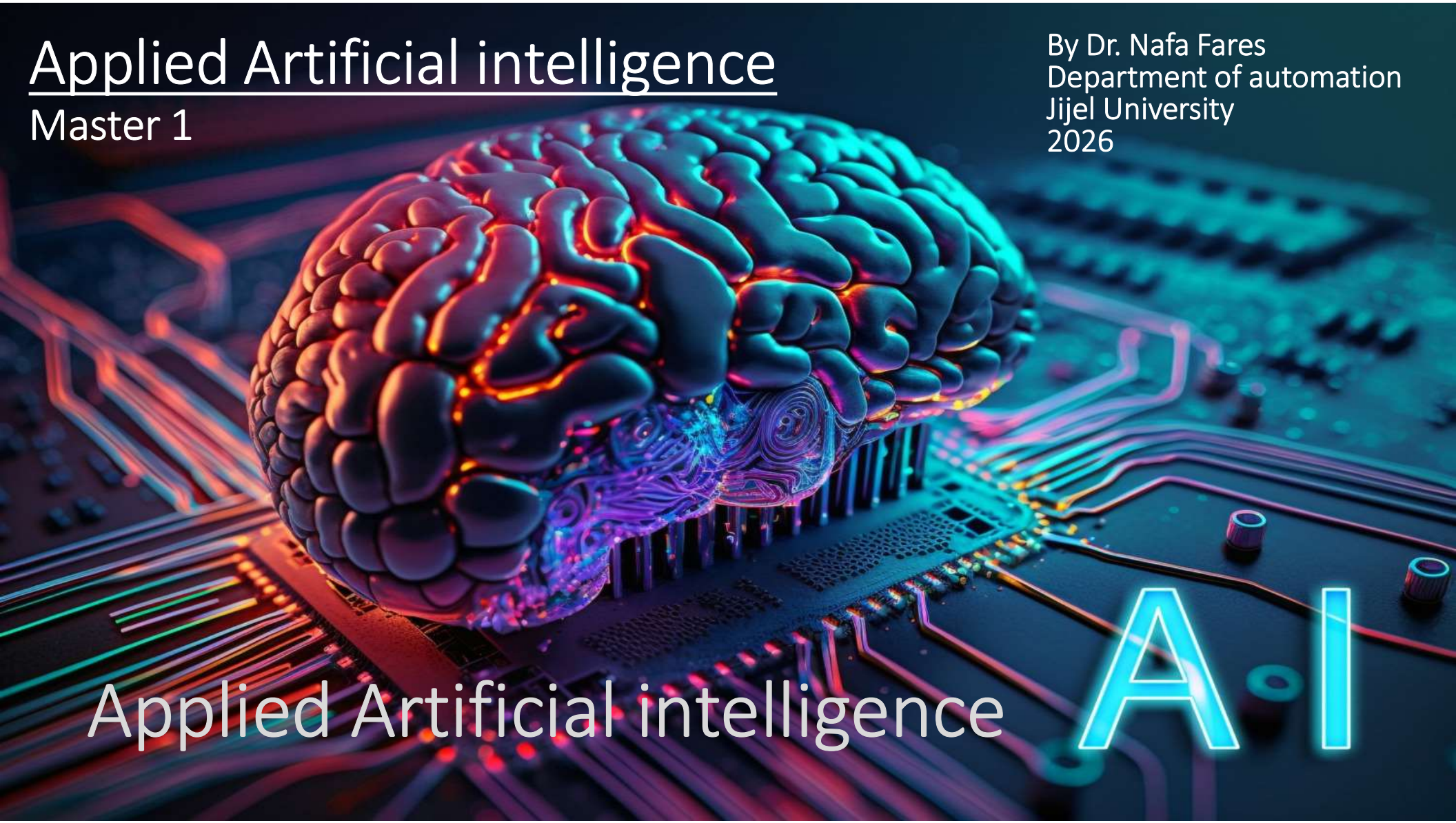
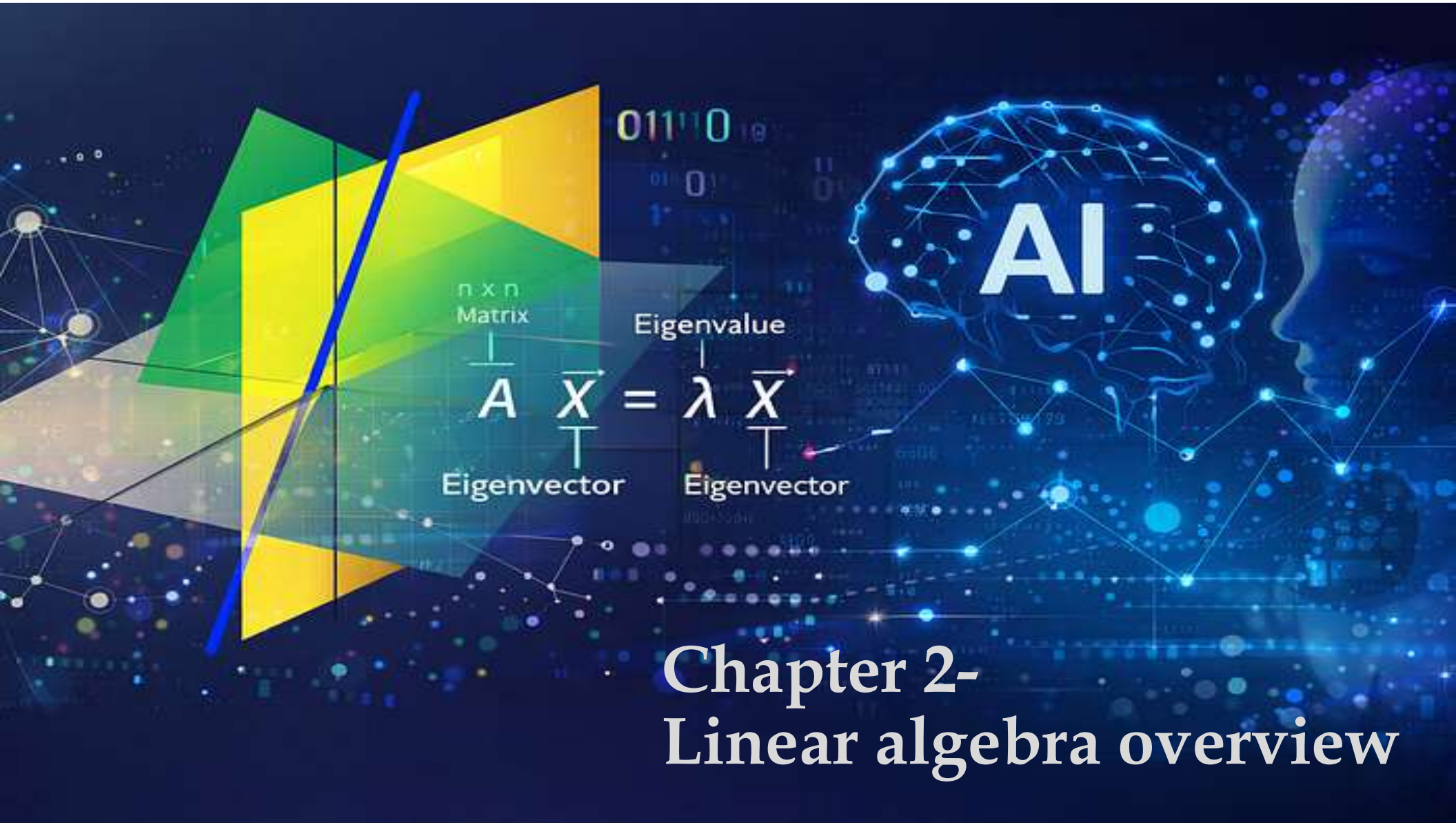
# Applied Artificial intelligence

Master 1

By Dr. Nafa Fares
Department of automation
Jijel University
2026

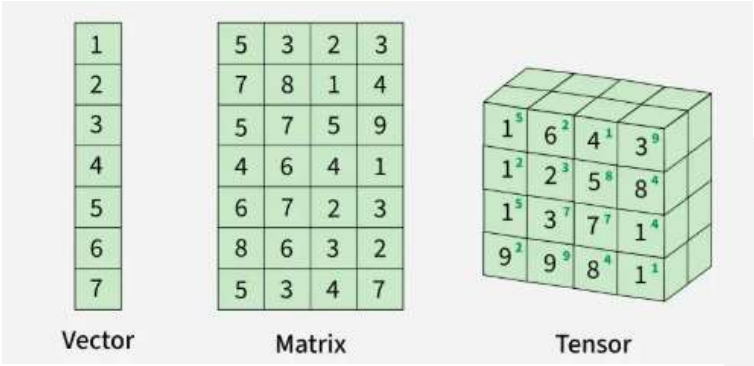


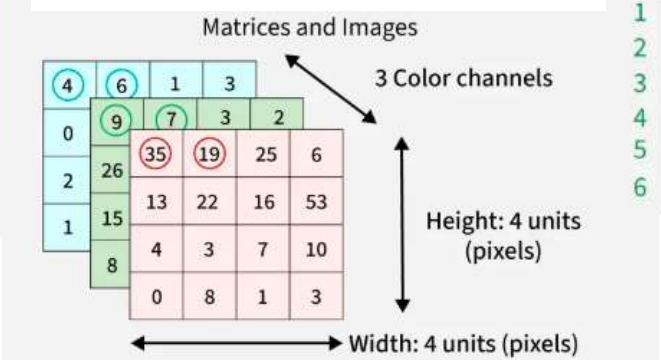


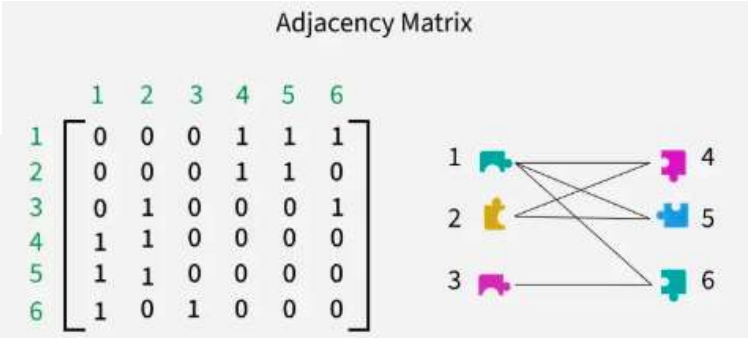

# Chapter 2-
# Linear algebra overview

# 1. Linear Algebra Operations For Machine Learning

- Linear algebra is a core mathematical foundation for machine learning, as most datasets and models are represented using vectors and matrices. It allows efficient computation, data manipulation and optimization, making complex tasks manageable.
- Data in ML is represented as vectors (features) and matrices (datasets).
- Operations like dot product, matrix multiplication and transformations power ML algorithms.
- Key concepts such as eigenvalues, eigenvectors and decompositions simplify dimensionality reduction, optimization and training.
- Algorithms like PCA, SVD, regression, SVMs and neural networks rely heavily on linear algebra.



## 2. Fundamental Concepts in Linear Algebra for Machine Learning

In machine learning, **vectors**, **matrices** and **scalars** play key roles in handling and processing data.

- **Vectors**

Vectors are quantities that have both magnitude and direction, often represented as arrows in space.

$$v = \begin{bmatrix} 2 \\ -1 \\ 4 \end{bmatrix}$$

- **Matrices**

Matrices are rectangular arrays of numbers, arranged in rows and columns. Matrices are used to represent linear transformations, systems of linear equations and data transformations in machine learning.

Example: $M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ u = [3, 4] v = [-1, 2]

**Scalars**

Scalars are single numerical values, without direction, magnitude only. Scalars are just single numbers that can multiply vectors or matrices. In machine learning, they're used to adjust things like the weights in a model or the learning rate during training

**Example:** Let's consider a scalar, k= 3 and a vector $v = \begin{bmatrix} 2 \\ -1 \\ 4 \end{bmatrix}$

Scalar multiplication involves multiplying each component of the vector by the scalar.

## 2. Fundamental Concepts in Linear Algebra for Machine Learning

So, if we multiply the vector v by the scalar k= 3 we get:

$$k.v = 3.\begin{bmatrix} 2 \\ -1 \\ 4 \end{bmatrix} = \begin{bmatrix} 6 \\ -3 \\ 12 \end{bmatrix}$$

### Operations in Linear Algebra

**Addition & Subtraction:** Add or subtract corresponding elements of vectors/matrices.

**Example:**

u = [ 2, −1, 4], v = [ 3, 0, −2]

u +v = [ 5, −1, 2], u −v = [ −1, −1, 6]

**Scalar Multiplication:** Multiply each element by a scalar.

**Example:** 3· [ 2, −1, 4] = [ 6, −3, 12]

**Dot Product:** Measures similarity of directions by multiplying matching elements and summing.

**Example:** $u \cdot v = u_1v_1 + u_2v_2 + u_3v_3$

**Cross Product:** For 3D vectors, produces a new vector perpendicular to both.

**Example:** $u \times v = [u_2v_3 - u_3v_2, u_3v_1 - u_1v_3, u_1v_2 - u_2v_1]$

### Linear Transformations

Linear transformations are basic operations in linear algebra that change vectors and matrices while keeping important properties like straight lines and proportionality. In machine learning, they are key for tasks like preparing data, creating features and training models. This section covers the definition, types and uses of linear transformations.

## 2. Fundamental Concepts in Linear Algebra for Machine Learning

### Definition

A transformation T is linear if it satisfies:

- **Additivity**: T(u+v) = T(u)+T(v)
- **Homogeneity**: T(kv) = k T(v)

### Common Types in ML

- **Translation** : Centering data by subtracting the mean.
- **Scaling** : Normalizing features so no single feature dominates.
- **Rotation** : Turning data, often used in computer vision and robotics.

### Matrix Operations

Matrix operations are central to linear algebra and widely used in machine learning for data handling, transformations and model training. The most common ones are:

- **Matrix Multiplication:** Combines two matrices by taking the dot product of rows and columns. Used in feature transformations, parameter computation and neural network operations.

$$A=\begin{bmatrix}1 & 2\\ 2 & 1\end{bmatrix};\quad B=\begin{bmatrix}3 & 0\\ 1 & 2\end{bmatrix}\qquad A\times B=\begin{bmatrix}7 & 2\\ 5 & 4\end{bmatrix}$$

## 2. Fundamental Concepts in Linear Algebra for Machine Learning

•**Transpose:** Flips a matrix across its diagonal (rows become columns). Denoted by $A^T$.
•**Inverse:** The matrix $A^{-1}$ satisfies $A \cdot A-1=I A \cdot A-1=I$. Exists only if $\det(A) \neq 0$. Used in solving equations and optimization.
•**Determinant:** A scalar value indicating whether a matrix is invertible. If $\det(A) = 0$, the matrix cannot be inverted.

### Eigenvalues and Eigenvectors

Eigenvalues and eigenvectors describe how matrices transform space, making them fundamental in many ML algorithms.
•**Eigenvalues (λ):** Scalars showing how much a transformation stretches or compresses along a direction.
•**Eigenvectors (v):** Non-zero vectors that only scale (not change direction) under transformation.

**Example:**

$$A= \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix};$$

solving $\det(A-\lambda I) = 0$ gives $\lambda_1 = 1, \lambda_2 = 3$.

•$\lambda 1=1 \rightarrow v1=[\ 1,-1]^T$
•$\lambda 2=3 \rightarrow v2=[\ 1,1]^T$

### Eigen Decomposition

$A=Q \Lambda Q^{-1}$
where Q holds eigenvectors and Λ is diagonal with eigenvalues.
**Applications in ML**:
•**Dimensionality Reduction (PCA):** Keeps directions with largest eigenvalues (most variance).
•**Matrix Factorization (SVD, NMF):** Breaks large datasets into smaller, structured parts for feature extraction.

## 2. Fundamental Concepts in Linear Algebra for Machine Learning

### Solving Linear Systems of equations

Linear systems are common in machine learning for parameter estimation and optimization. Key methods include:

1. **Gaussian Elimination:** Transforms a matrix into row-echelon form using row operations. Steps:
   - Forward Elimination -> make entries below diagonal zero
   - Back Substitution -> solve variables from last row upward
   - Pivoting -> swap rows to avoid division by zero
2. **LU Decomposition:** Splits a matrix into Lower (L) and Upper (U) triangular matrices. Solves systems efficiently using forward and back substitution.
3. **QR Decomposition:** Splits a matrix into Orthogonal (Q) and Upper triangular (R). Useful for least squares problems and eigenvalue computation.

### Applications of Linear Algebra in Machine Learning

Linear algebra powers many ML algorithms by enabling data manipulation, model representation and optimization. Key applications include:

- **PCA (Principal Component Analysis):** Reduces dimensionality by computing covariance, eigenvalues/eigenvectors and projecting data onto principal components.
- **SVD (Singular Value Decomposition):** Factorizes a matrix into $A = U\Sigma V^T$, used for dimensionality reduction, compression and noise filtering.
- **Linear Regression:** Models relationships via matrix form $Y = X\beta + \epsilon$, solved using the normal equation $X^TX\beta = X^TY$.
- **SVM (Support Vector Machines):** Uses the kernel trick and optimization to find decision boundaries for classification and regression.
- **Neural Networks:** Depend on matrix multiplications, gradient descent and weight initialization for training models

## 2. Fundamental Concepts in Linear Algebra for Machine Learning

### Vector norms

Vector norms in machine learning measure a vector's "size" or magnitude, crucial for calculating distances, defining loss functions
(like Mean Squared Error using L2 norm), and implementing regularization (L1 for sparsity, L2 for preventing overfitting), with the general p-norm formula :

$$(\sum |\mathrm{xi}|^{p})^{1/p}$$

covering common types like L1 (Manhattan), L2 (Euclidean), and L∞ (Max) norms, each serving different analytical purposes.

Key Concepts

- **What it is**: A function mapping a vector to a single non-negative number, representing its length or magnitude from the origin.
- **Why it matters**: Quantifies "how big" a vector is, essential for comparing data points, evaluating model errors, and managing model complexity.

Common Vector Norms (p-norms)

- **L1 Norm (Manhattan/Taxicab Norm)**: $\sum |\mathrm{xi}|$ Sum of absolute values. Encourages sparsity by driving many features to zero; used in Lasso regression.
- **L2 Norm (Euclidean Norm)**: $\sqrt{\sum x_i^2}$ Square root of the sum of squares. Standard for distance; used in Ridge regression, Mean Squared Error..
- **L∞ Norm (Max Norm)**: $\max(|\mathrm{xi}|)$ . Maximum absolute value. Useful for outlier detection or when the largest error matters most.

## 2. Fundamental Concepts in Linear Algebra for Machine Learning

In probability, **expectation (mean)** is the long-run average value of a random variable, while **variance** measures its spread or dispersion around that mean; expectation tells you the center ($E[X]=\mu$), and variance quantifies how much data points typically deviate from it (like $Var(X)=\sigma^2$ ), with standard deviation ($\sigma$) being the square root of variance for units matching the variable. These concepts help characterize random behavior, essential for modeling uncertainty in data science, engineering, and finance.

### Expectation (E[X] or μ)

- **Definition:** The weighted average of all possible outcomes, where weights are their probabilities.
- **Formula (Discrete):** $E[X]=\sum$xi $P(xi)$ , Where: $xi$ is each possible outcome and $P(xi)$ is its probability
- **Meaning:** The central tendency or average value you'd expect over many trials.

### Variance (Var(X) or σ²)

- **Definition:** The expected value of the squared differences from the mean, measuring dispersion.
- **Formula (Discrete):** $Var(X)=\sum_{i=1}^{n}(xi-\mu)^2 P(xi)$ or $E[X^2]-(E[X])^2$
- **Meaning:** How spread out the values are; a higher variance means more spread.

Key Relationships & Concepts

- **Standard Deviation ($\sigma$):** The square root of the variance $\sigma=\sqrt{Var(X)}$ giving a measure of spread in the original units of the variable.
- **Covariance:** Measures how two variables change together; positive means same direction, negative means opposite.
- **Independence:** If variables are independent, calculations (like variance of a sum) simplify significantly.

## 2. Fundamental Concepts in Linear Algebra for Machine Learning

Normal, Binomial, and Uniform are fundamental probability distributions: **Normal** is a symmetric, bell-shaped curve for continuous data (like heights); **Binomial** models discrete success/failure counts (like coin flips over trials); and **Uniform** gives equal probability to all outcomes in a range (like a fair die roll). They differ in data type (discrete/continuous), shape, and real-world applications, with Normal as foundational for many natural phenomena and Binomial for repeated experiments, while Uniform represents pure chance

### Normal Distribution (Continuous)

- **Description:** Symmetric, bell-shaped curve (Gaussian), with mean, median, and mode at the center.
- **Example:** Heights of people, test scores.
- **Formula (PDF):** $P(x)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

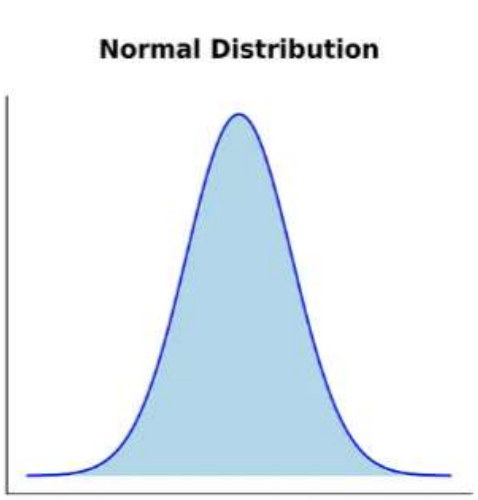
### Binomial Distribution (Discrete)

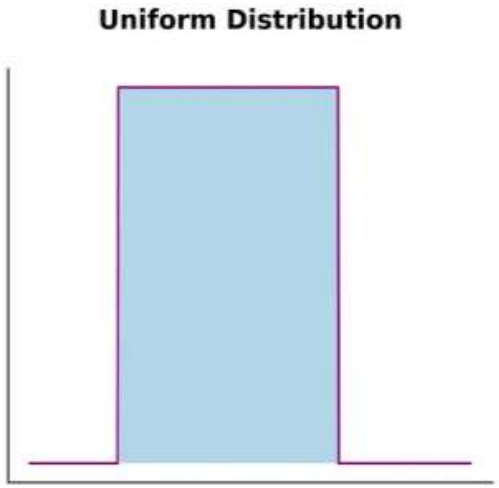
- **Description:** Models the number of successes in a fixed number ($n$) of independent trials, each with two outcomes (success/failure, p/1-p).
- **Key Feature:** Discrete (whole numbers), based on trials and success probability.
- **Example:** Number of heads in 10 coin flips, number of defective items in a batch.
- **Formula:** $P(X=k)=\binom{n}{k}p^k\ (1-p)^k$, with: $\binom{n}{k}=\frac{n!}{k!(n-k)!}$

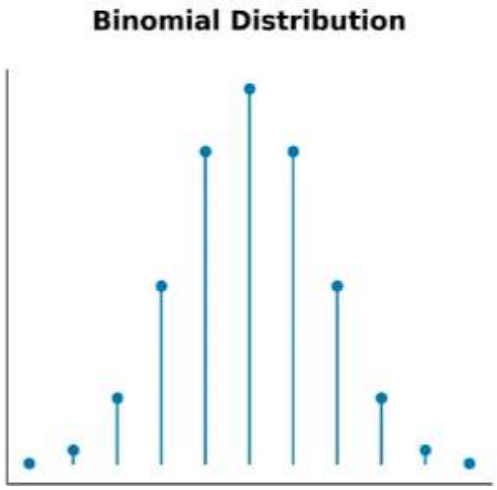
# 2. Fundamental Concepts in Linear Algebra for Machine Learning

## Uniform Distribution (Continuous & Discrete)

- **Description:** All outcomes within a given range are equally likely.
- **Key Feature:** Flat probability, constant density.
- **Continuous Example:** Waiting time for a bus (if arrival is random within an hour).
- **Discrete Example:** Rolling a fair die (each face 1-6 has 1/6 chance).
- **Formula (Continuous):** $P(x)=\frac{1}{b-a}$ for $a \leq x \leq b$



## 3. Linear Regression in Machine learning

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 hobw many hours were studied.

We use the independent variable to predict the dependent variable.

### Best Fit Line in Linear Regression

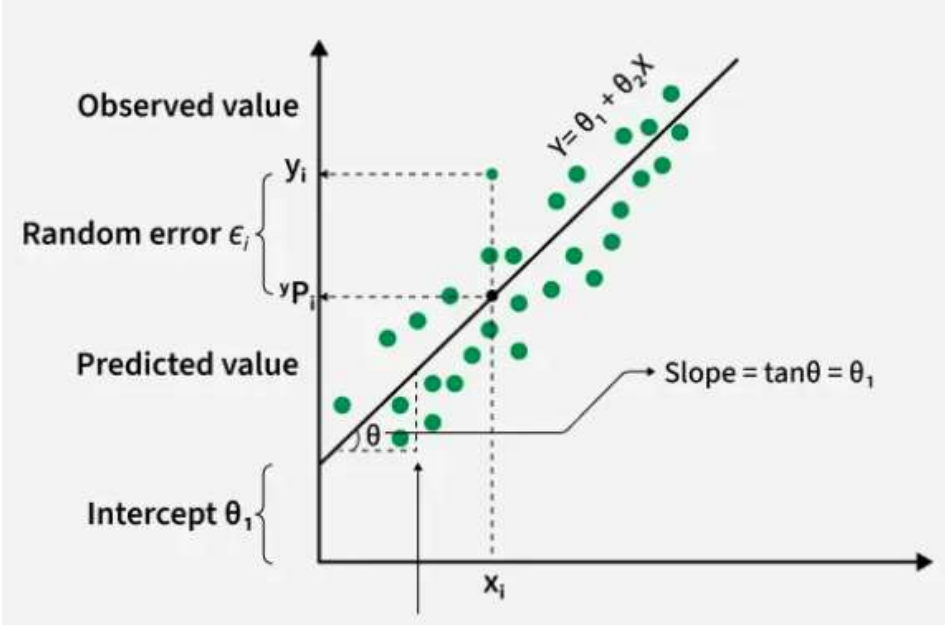
In linear regression, the best-fit line is the straight line that most accurately represents the relationship between the independent variable (input) and the dependent variable (output). It is the line that minimizes the difference between the actual data points and the predicted values from the model.

#### 1. Goal of the Best-Fit Line

The goal of linear regression is to find a straight line that minimizes the error (the difference) between the observed data points and the predicted values. This line helps us predict the dependent variable for new, unseen data.

# 3. Linear Regression in Machine learning

Here Y is called a dependent or target variable and X is called an independent variable also known as the predictor of Y. There are many types of functions or modules that can be used for regression. A linear function is the simplest type of function. Here, X may be a single feature or multiple features representing the problem.




## 2. Equation of the Best-Fit Line

For simple linear regression (with one independent variable), the best-fit line is represented by the equation:

$$y=mx+b \qquad (1)$$

**Where:**

- **y** is the predicted value (dependent variable)
- **x** is the input (independent variable)
- **m** is the slope of the line (how much y changes when x changes)
- **b** is the intercept (the value of y when x = 0)

The best-fit line will be the one that optimizes the values of m (slope) and b (intercept) so that the predicted y values are as close as possible to the actual data points.

# 3. Linear Regression in Machine learning

## 3. Minimizing the Error: 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:

$$\text{Residual} = y_i - \hat{y}_i \qquad (2)$$

**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:

$$\text{Sum\_of\_squared\_errors(SSE)} = \sum(y_i - \hat{y}_i)^2 \qquad (3)$$

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

## 4. Interpretation of the Best-Fit Line

•**Slope (m):** The slope of the best-fit line indicates how much the dependent variable (y) changes with each unit change in the independent variable (x). For example if the slope is 5, it means that for every 1-unit increase in x, the value of y increases by 5 units.

•**Intercept (b):** The intercept represents the predicted value of y when x = 0. It's the point where the line crosses the y-axis.

## 3. Linear Regression in Machine learning

In linear regression some hypothesis are made to ensure reliability of the model's results. ***Limitations***
***Assumes Linearity:*** *The method assumes the relationship between the variables is linear. If the relationship is non-linear, linear regression might not work well.*
***Sensitivity to Outliers:*** *Outliers can significantly affect the slope and intercept, skewing the best-fit line.*

### Hypothesis function in Linear Regression

In linear regression, the hypothesis function is the equation used to make predictions about the dependent variable based on the independent variables. It represents the relationship between the input features and the target output. For a simple case with one independent variable, the hypothesis function is:

$$h(x)=\beta_0 + \beta_1 x \qquad (4)$$

**Where:**
**h(x)**(or $\hat{y}$ ) is the predicted value of the dependent variable (y).
**x** the independent variable.
$\beta_0$ is the intercept, representing the value of y when x is 0.
$\beta_1$ is the slope, indicating how much y changes for each unit change in x.
For **multiple linear regression** (with more than one independent variable), the hypothesis function expands to:

$$h(x_1,x_2,\ldots,x_k)=\beta_0+\beta_1x_1+\beta_2x_2+\ldots+\beta_kx_k \qquad (5)$$

## 3. Linear Regression in Machine learning

**Where:**

•$x_1, x_2, \ldots, x_k$ $x_1, x_2, \ldots, x_k$ are the independent variables.

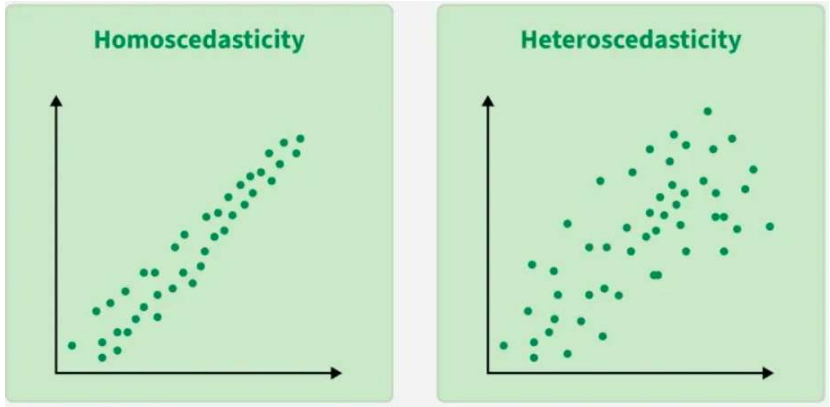
•$\beta_0$ $\beta_0$ is the intercept.

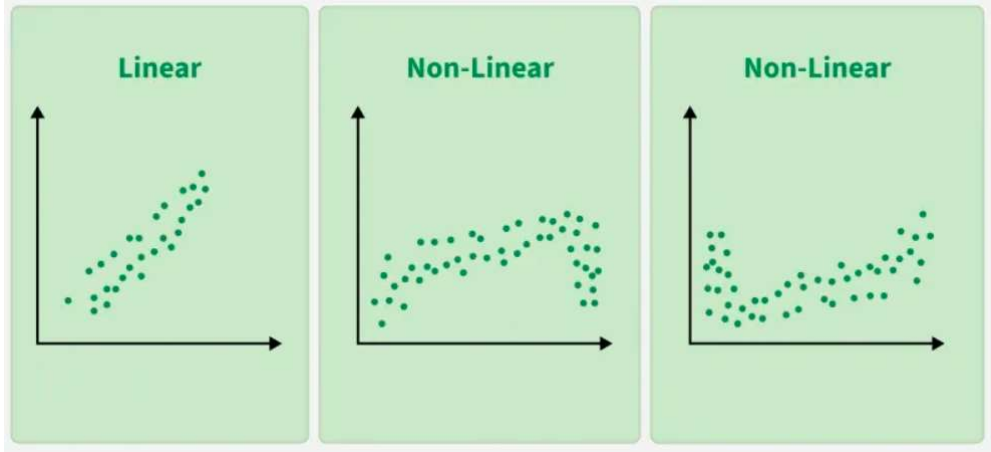
•$\beta_1, \beta_2, \ldots, \beta_k$ $\beta_1, \beta_2, \ldots, \beta_k$ are the coefficients, representing the influence of each respective independent variable on the predicted output.

### Assumptions of the Linear Regression

1. **Linearity**: The relationship between inputs (X) and the output (Y) is a straight line.
2. **Independence of Errors**: The errors in predictions should not affect each other.

**3. Constant Variance (Homoscedasticity):** The errors should have equal spread across all values of the input. If the spread changes (like fans out or shrinks), it's called heteroscedasticity and it's a problem for the model.



## 4. Types of Linear Regression

**4. Normality of Errors**: The errors should follow a normal (bell-shaped) distribution.
**5. No Multicollinearity(for multiple regression)**: Input variables shouldn't be too closely related to each other.
**6. No Autocorrelation**: Errors shouldn't show repeating patterns, especially in time-based data.
**7. Additivity**: The total effect on Y is just the sum of effects from each X, no mixing or interaction between them.

### Types of Linear Regression

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

**1. 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.

**Formula**

$$\hat{y} = \theta_0 + \theta_1 x \qquad (6)$$

**Where:**

$\hat{y}$ is the predicted value
$x$ s the input (independent variable)
$\theta_0$ is the intercept (value of $\hat{y}$ when x=0)
$\theta_1$ is the slope or coefficient (how much $\hat{y}$ changes with one unit of x)

$$\text{with : } \theta_1 = \frac{n\sum xy - (\sum x)(\sum y)}{n\sum x^2 - (\sum x)^2}; \qquad \theta_0 = \frac{\sum y - \theta_1 \sum x}{n}$$

**Example-** Predicting a person's salary (y) based on their years of experience (x).

## 4. Types of Linear Regression

### 2. Multiple Linear Regression

Multiple linear regression involves more than one independent variable and one dependent variable. The equation for multiple linear regression is:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n \qquad (7)$$

**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.

$\theta_0$ is the intercept.

The goal is to find the best-fit line that predicts Y accurately for given inputs X.

**Use Cases**

**Real Estate:** Predict property prices using location, size and other factors.
**Finance:** Forecast stock prices using interest rates and inflation data.
**Agriculture:** Estimate crop yield from rainfall, temperature and soil quality.
**E-commerce:** Analyze how price, promotions and seasons affect sales.
Once you understand linear regression and its types, the next step is building the model in practice.

## 4. Types of Linear Regression

- **Cost function for Linear Regression :** In Linear Regression, the cost function measures how far the predicted values ($\hat{y}_i$) are from the actual values ($y_i$). It helps identify and reduce errors to find the best-fit line. The most common cost function used is Mean Squared Error (MSE), which calculates the average of squared differences between actual and predicted values:

$$\text{Cost function : } J(\theta_i)=\frac{1}{n}\sum_{n}^{i}(\hat{y}_i - y_i)^2 \qquad (8)$$

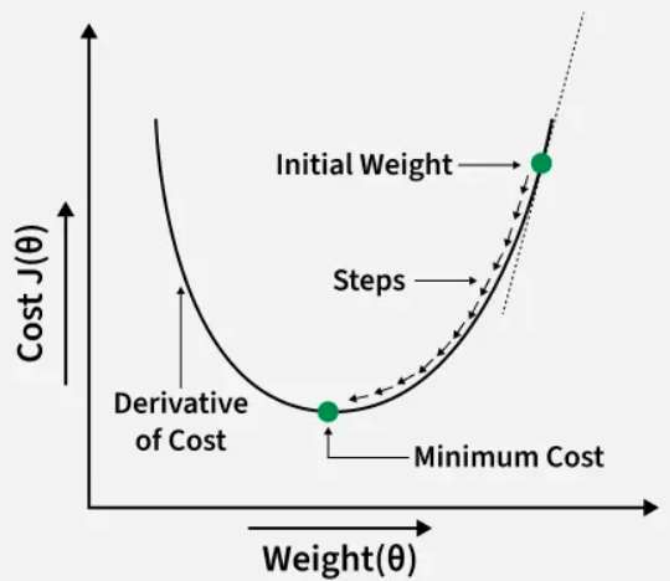
Where : $\hat{y} = \theta_1+\theta_2 x_i$

To minimize this cost, we use **Gradient Descent**, which iteratively updates $\theta_1$ and $\theta_2$ until the MSE reaches its lowest value. This ensures the line fits the data as accurately as possible.

- **Gradient Descent for Linear Regression**

Gradient descent is an optimization technique used to train a linear regression model by minimizing the prediction error. It works by starting with random model parameters and repeatedly adjusting them to reduce the difference between predicted and actual values.

Gradient descent is an iterative algorithm, which means we apply an update repeatedly until some criterion is met. We initialize the weights to something reasonable (e.g. all zeros) and repeatedly adjust them in the direction of steepest descent



# 4. Types of Linear Regression

## Gradient Descent Algorithm

For a cost function J,

- Pick an initial point $x_0$
- Iterate until convergence:

$$\boldsymbol{\theta_{t+1} = \theta_t - \gamma_t \nabla J(\theta_t)} \qquad \textbf{(9)}$$

where $\gamma_t$ is the $t^{th}$ step size (sometimes called learning rate)

Possible Stopping Criteria: iterate until $\|\nabla J(\boldsymbol{\theta_t})\| \leq \epsilon$ for some $\epsilon > 0$

Then, recalling equation (6) , (8) and (9), the parameters $\boldsymbol{\theta_i}$ can be updated so that:

$$\boldsymbol{\theta_1^{t+1} = \theta_1^t - \gamma_1^t \nabla J(\theta_1^t) = \theta_1^t - \gamma_1^t x \frac{2}{n} \sum_n^i (\hat{y}_i - y_i)}$$

$$\boldsymbol{\theta_0^{t+1} = \theta_0^t - \gamma_0^t \nabla J(\theta_0^t) = \theta_0^t - \gamma_0^t \frac{2}{n} \sum_n^i (\hat{y}_i - y_i)} \qquad \textbf{(10)}$$

## 5. Evaluation Metrics for Linear Regression

A variety of evaluation measures can be used to determine the strength of any linear regression model. These assessment metrics often give an indication of how well the model is producing the observed outputs. The most common measurements are:

### 1. Mean Square Error (MSE)

Mean Squared Error (MSE) is an evaluation metric that calculates the average of the squared differences between the actual and predicted values for all the data points. The difference is squared to ensure that negative and positive differences don't cancel each other out.

$$MSE = \frac{1}{n}\sum_{1}^{n}(\hat{y}_i - y_i)^2 \qquad (11)$$

Here,
$n$ is the number of data points.
$y_i$ is the actual or observed value for the *ith* data point.
$\hat{y}_i$ is the predicted value for the *ith* data point.

MSE is a way to quantify the accuracy of a model's predictions. MSE is sensitive to outliers as large errors contribute significantly to the overall score.

### 2. Mean Absolute Error (MAE)

Mean Absolute Error is an evaluation metric used to calculate the accuracy of a regression model. MAE measures the average absolute difference between the predicted values and actual values.
Mathematically MAE is expressed as:

## 5. Evaluation Metrics for Linear Regression

$$MAE = \frac{1}{n}\sum_{1n}|\hat{y}_i - y_i| \qquad (12)$$

Lower MAE value indicates better model performance. It is not sensitive to the outliers as we consider absolute differences.

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

The square root of the residuals' variance is the Root Mean Squared Error. It describes how well the observed data points match the expected values or the model's absolute fit to the data. In mathematical notation, it can be expressed as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{n}^{i}\left(y_i^{actual} - y_i^{predicted}\right)^2} \qquad (13)$$

$n$ is the number of observations

Lower MAE value indicates better model performance. It is not sensitive to the outliers as we consider absolute differences

## 6. Regularization Techniques for Linear Models

### 1. Lasso Regression (L1 Regularization)

Lasso Regression is a technique used for regularizing a linear regression model, it adds a penalty term to the linear regression objective function to prevent overfitting. The objective function after applying lasso regression is:

$$J(\theta) = \frac{1}{2m}\sum_{1}^{m}(\hat{y}_i - y_i)^2 + \lambda\sum_{j=1}^{n}|\theta_j| \qquad (14)$$

the first term is the least squares loss, representing the squared difference between predicted and actual values. the second term is the L1 regularization term, it penalizes the sum of absolute values of the regression coefficient $\theta_j$.

### 2. Ridge Regression (L2 Regularization)

Ridge regression is a linear regression technique that adds a regularization term to the standard linear objective. Again, the goal is to prevent overfitting by penalizing large coefficient in linear regression equation. It useful when the dataset has multicollinearity where predictor variables are highly correlated. The objective function after applying ridge regression is:

$$J(\theta) = \frac{1}{2m}\sum_{1}^{m}(\hat{y}_i - y_i)^2 + \lambda\sum_{j=1}^{n}\theta_j^2 \qquad (14)$$

the first term is the least squares loss, representing the squared difference between predicted and actual values. The second term is the L1 regularization term, it penalizes the sum of square of values of the regression coefficient $\theta_j$.

## 7. Python Implementation of Linear Regression

### 1. Import the necessary libraries

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
```

### 2. Generating Random Dataset

Fetches the California Housing dataset and separates features (X) and target (y).

```
np.random.seed(42)

X = np.random.rand(50, 1) * 100

Y = 3.5 * X + np.random.randn(50, 1) * 20
```

### 3. Creating and Training Linear Regression Model

```
model = LinearRegression()
model.fit(X, Y)
```

### 4. Predicting Y Values

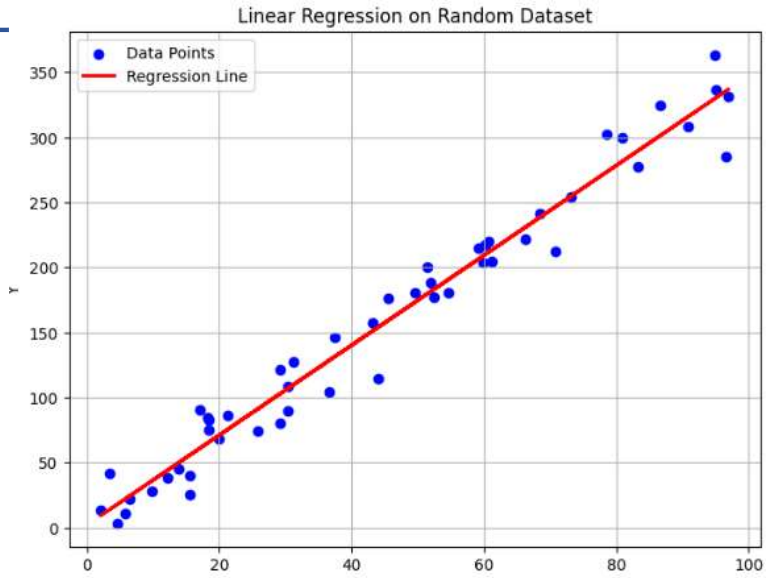
```
Y_pred = model.predict(X)
```

## 7. Python Implementation of Linear Regression

### 5. Visualizing the Regression Line

```
plt.figure(figsize=(8,6))
plt.scatter(X, Y, color='blue', label='Data Points')
plt.plot(X, Y_pred, color='red', linewidth=2,
label='Regression Line')
plt.title('Linear Regression on Random Dataset')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.grid(True)
plt.show()
```

**Output**:




### 6. Slope and Intercept

```
print("Slope (Coefficient):", model.coef_[0][0])
print("Intercept:", model.intercept_[0])
```

```
Slope (Coefficient): 3.4553132007706204
Intercept: 1.9337854893777546
```

## 8. Python Implementation of Linear Regression using Scikit-learn

We will demonstrate a binary linear model as this will be easier to visualize. In this demonstration, the model will use Gradient Descent to learn.

### Step 1: Importing all the required libraries

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing, svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

### Step 2: Reading the dataset:

```
df =
pd.read_csv('https://www.kaggle.com/datasets/dftow001/bot
tle-csv')
df_binary = df[['Salnty', 'T_degC']]

# Taking only the selected two attributes from the
dataset
df_binary.columns = ['Sal', 'Temp']
#display the first 5 rows
df_binary.head()
```

output

| | Sal | Temp |
|---|---|---|
| 0 | 33.440 | 10.50 |
| 1 | 33.440 | 10.46 |
| 2 | 33.437 | 10.46 |
| 3 | 33.420 | 10.45 |
| 4 | 33.421 | 10.45 |

## 8. Python Implementation of Linear Regression using Scikit-learn

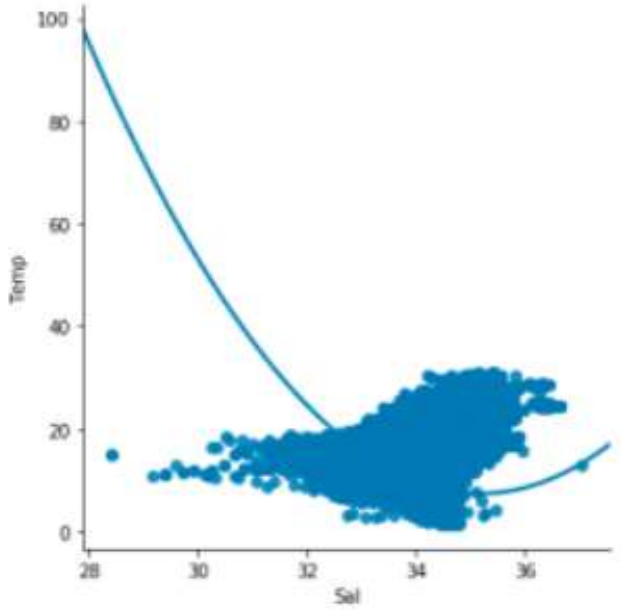
### Step 3: Exploring the data scatter

```
#plotting the Scatter plot to check relationship between
Sal and Temp
sns.lmplot(x ="Sal", y ="Temp", data = df_binary, order =
2, ci = None)
plt.show()
```

```
# Eliminating NaN or missing input numbers
df_binary.fillna(method ='ffill', inplace =
True)
```

### Step 5: Training our model

Output



## 8. Python Implementation of Linear Regression using Scikit-learn

### Step 5: Training our model

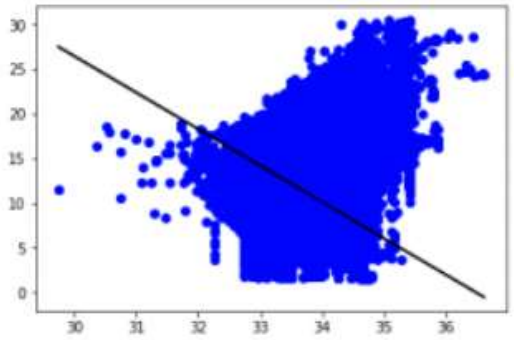
```
X = np.array(df_binary['Sal']).reshape(-1, 1)
y = np.array(df_binary['Temp']).reshape(-1, 1)

# Separating the data into independent and dependent
variables
# Converting each dataframe into a numpy array
# since each dataframe contains only one column
df_binary.dropna(inplace = True)

# Dropping any rows with Nan values
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.25)

# Splitting the data into training and testing data
regr = LinearRegression()

regr.fit(X_train, y_train)
print(regr.score(X_test, y_test))
```

output → `0.20780376990868232`

### Step 6: Exploring our results

```
y_pred = regr.predict(X_test)
plt.scatter(X_test, y_test, color ='b')
plt.plot(X_test, y_pred, color ='k')

plt.show()
# Data scatter of predicted values
```

output →



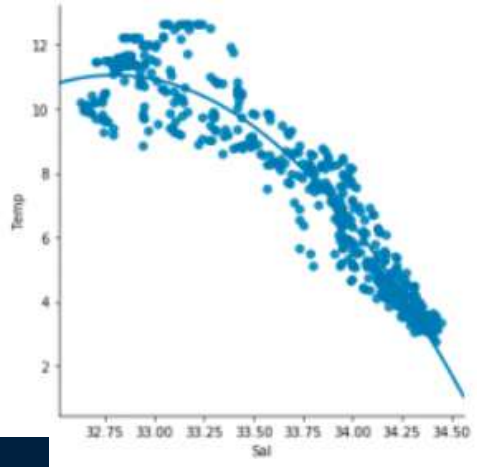
## 8. Python Implementation of Linear Regression using Scikit-learn

The low accuracy score of our model suggests that our regressive model has not fit very well with the existing data. This suggests that our data is not suitable for linear regression. But sometimes, a dataset may accept a linear regressor if we consider only a part of it. Let us check for that possibility.

### Step 7: Working with a smaller dataset

```
df_binary500 = df_binary[:][:500]

# Selecting the 1st 500 rows of the data
sns.lmplot(x ="Sal", y ="Temp", data = df_binary500,
                                order = 2, ci = None)
```

Output



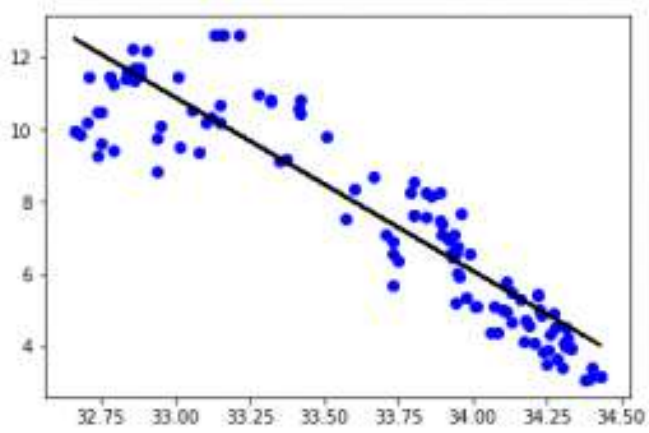

We can already see that the first 500 rows follow a linear model. Continuing with the same steps as before.

```
df_binary500.fillna(method ='fill', inplace = True)

X = np.array(df_binary500['Sal']).reshape(-1, 1)
y = np.array(df_binary500['Temp']).reshape(-1, 1)

df_binary500.dropna(inplace = True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25)

regr = LinearRegression()
regr.fit(X_train, y_train)
print(regr.score(X_test, y_test))
```

Output `0.8475943139663558`

## 8. Python Implementation of Linear Regression using Scikit-learn

```
y_pred = regr.predict(X_test)
plt.scatter(X_test, y_test, color ='b')
plt.plot(X_test, y_pred, color ='k')

plt.show()
```




### Step 8: Evaluation Metrics For Regression

At last, we check the performance of the Linear Regression model with help of evaluation metrics. For Regression algorithms we widely use mean_absolute_error, and mean_squared_error metrics to check the model performance.

```
from sklearn.metrics import
mean_absolute_error,mean_squared_error

mae = mean_absolute_error(y_true=y_test,y_pred=y_pred)
#squared True returns MSE value, False returns RMSE
value.
mse = mean_squared_error(y_true=y_test,y_pred=y_pred)
#default=True
rmse =
mean_squared_error(y_true=y_test,y_pred=y_pred,squared=Fa
lse)

print("MAE:",mae)
print("MSE:",mse)
print("RMSE:",rmse)
```

```
MAE: 0.7927322046360309
MSE: 1.0251137190180517
RMSE: 1.0124789968281078
```

## 8. Why Linear Regression is Important

•**Simplicity and Interpretability:** It's easy to understand and interpret, making it a starting point for learning about machine learning.

•**Predictive Ability**: Helps predict future outcomes based on past data, making it useful in various fields like finance, healthcare and marketing.

•**Basis for Other Models:** Many advanced algorithms, like logistic regression or neural networks, build on the concepts of linear regression.

•**Efficiency:** It's computationally efficient and works well for problems with a linear relationship.
•**Widely Used:** It's one of the most widely used techniques in both statistics and machine learning for regression tasks.

•**Analysis:** It provides insights into relationships between variables (e.g., how much one variable influences another).

## 8. Why Linear Regression is Important

**Advantages**

•**Linear regression** is a relatively simple algorithm, making it easy to understand and implement. The coefficients of the linear regression model can be interpreted as the change in the dependent variable for a one-unit change in the independent variable, providing insights into the relationships between variables.

•**Linear regression** is computationally efficient and can handle large datasets effectively. It can be trained quickly on large datasets, making it suitable for real-time applications.

•**Linear regression** is relatively robust to outliers compared to other machine learning algorithms. Outliers may have a smaller impact on the overall model performance.

•**Linear regression** often serves as a good baseline model for comparison with more complex machine learning algorithms.

•Linear regression is a well-established algorithm with a rich history and is widely available in various machine learning libraries and software packages.

## 8. Why Linear Regression is Important

**Limitations**

•**Linear regression** assumes a linear relationship between the dependent and independent variables. If the relationship is not linear, the model may not perform well.

•**Linear regression** is sensitive to multicollinearity, which occurs when there is a high correlation between independent variables. Multicollinearity can inflate the variance of the coefficients and lead to unstable model predictions.

•**Linear regression** assumes that the features are already in a suitable form for the model. Feature engineering may be required to transform features into a format that can be effectively used by the model.

•**Linear regression** is susceptible to both overfitting and underfitting. Overfitting occurs when the model learns the training data too well and fails to generalize to unseen data. Underfitting occurs when the model is too simple to capture the underlying relationships in the data.

•**Linear regression** provides limited explanatory power for complex relationships between variables. More advanced machine learning techniques may be necessary for deeper insights

# 9. Exercises

## Exercise#0

The following example describes the expenditure (in dollars) on recreation per month by employees at a certain company, and their corresponding monthly incomes.

Using the equation for the linear regression that you calculated, estimate the monthly income of an employee at this company

who spends 5000 dollars per month on recreation.

(a) Calculate the linear regression line for the data (b) What is the

(b) slope? What is the y-intercept? Provide the units of each.

(c) Is this interpolation or extrapolation?

| Expenditure ($) | Income ($) |
|---|---|
| 2400 | 41200 |
| 2650 | 50100 |
| 2350 | 52000 |
| 4950 | 66000 |
| 3100 | 44500 |
| 2500 | 37700 |
| 5106 | 73500 |
| 3100 | 37500 |
| 2900 | 56700 |
| 1750 | 35600 |

## Exercise#1

Consider a company that markets and repairs small computers. The following table illustrates the relationship between the length of a service call, in minutes, and the number of electronic components in the computer that must be repaired or replaced Answer the following questions about the above data
(e)Using the linear regression model, estimate the number of components that need to be repaired, if a customer spends an hour on the phone for a service call(round to the nearest whole number). Is this interpolation or extrapolation? (f)Suppose that a certain customer has an issue that requires 12 components to be fixed. How long does your regression model suggest that the customer will have to spend on the phone during a service call addressing this issue?

| Length of Service Call | Number of Components |
|---|---|
| 23 | 1 |
| 29 | 2 |
| 49 | 3 |
| 64 | 4 |
| 74 | 4 |
| 87 | 5 |
| 96 | 6 |
| 97 | 6 |
| 109 | 7 |
| 119 | 8 |
| 149 | 9 |
| 145 | 9 |
| 154 | 10 |
| 166 | 10 |

# 9. Exercises

## Exercise#2

The table in side shows the two-dimensional frequency distribution of a sample of 80 persons in a study about the relation between the blood cholesterol (X) in mg/dl and the high blood pressure (Y).

- Complete the table.
- Construct the linear regression model of cholesterol on pressure.
- Use the linear model to calculate the expected cholesterol for a person with pressure 160 mmHg.
- According to the linear model, what is the expected pressure for a person with cholesterol 270 mg/dl?

| $X \setminus Y$ | $[110, 130)$ | $[130, 150)$ | $[150, 170)$ | $n_x$ |
|---|---|---|---|---|
| $[170, 190)$ | | 4 | | 12 |
| $[190, 210)$ | 10 | 12 | 4 | |
| $[210, 230)$ | 7 | | 8 | |
| $[230, 250)$ | 1 | | | 18 |
| $n_y$ | | 30 | 24 | |

Use the following sums: $\sum x_i = 16960$ mg/dl, $\sum y_j = 11160$ mmHg, $\sum x_i^2 = 3627200$ (mg/dl)$^2$, $\sum y_j^2 = 1576800$ mmHg$^2$ y $\sum x_i y_j = 2378800$ mg/dl·mmHg.

## Exercise#2_Solution

2.

$\bar{x} = 212$ mg/dl, $s_x^2 = 396$ (mg/dl)$^2$.

$\bar{y} = 139.5$ mmHg, $s_y^2 = 249.75$ mmHg$^2$.

$s_{xy} = 161$ mg/dl·mmHg.

Regression line of cholesterol on blood pressure: $x = 122.0721 + 0.6446y$.

3. $x(160) = 225.2152$ mg/dl.

4.

Regression line of blood pressure on cholesterol: $y = 53.3081 + 0.4066x$.

$y(270) = 163.0808$ mmHg.

| $X \setminus Y$ | $[110, 130)$ | $[130, 150)$ | $[150, 170)$ | $n_x$ |
|---|---|---|---|---|
| $[170, 190)$ | 8 | 4 | 0 | 12 |
| $[190, 210)$ | 10 | 12 | 4 | 26 |
| $[210, 230)$ | 7 | 9 | 8 | 24 |
| $[230, 250)$ | 1 | 5 | 12 | 18 |
| $n_y$ | 26 | 30 | 24 | 80 |

# 9. Exercises

## Exercise#3

A research study has been conducted to determine the loss of activity of a drug. The table below shows the results of the experiment.

| Time (in years) | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Activity (%) | 96 | 84 | 70 | 58 | 52 |

1.Construct the linear regression model of activity on time.

2.According to the linear model, when will the activity be 80%? When will the drug have lost all activity?

## Exercise#3_Solution

1. $\bar{x} = 3$ years, $s_x^2 = 2$ years$^2$.
   $\bar{y} = 72\,\%$, $s_y^2 = 264\,\%^2$.
   $s_{xy} = -22.8$ years·%.
   Regression line of activity on time: $y = 106.2 + -11.4x$.
2.

Regression line of time on activity: $x = 9.2182 + -0.0864y$.

$x(80) = 2.3091$ years and $x(0) = 9.2182$ years.

# 9. Exercises

## Exercise#4

A Basketball team is testing a new stretching program to reduce the injuries during the league. The data below show the daily number of minutes doing stretching exercises and the number of injuries along the league.

- Construct the regression line of the number of injuries on the time of stretching.
- How much is the reduction of injuries for every minute of stretching?
- How many minutes of stretching are require for having no injuries? Is reliable this prediction?

| Stretching minutes | 0 | 30 | 10 | 15 | 5 | 25 | 35 | 40 |
|---|---|---|---|---|---|---|---|---|
| Injuries | 4 | 1 | 2 | 2 | 3 | 1 | 0 | 1 |

Use the following sums ($X$=Number of minutes stretching, and $Y$=Number of injuries):
$\sum x_i = 160$ min, $\sum y_j = 14$ injuries, $\sum x_i^2 = 4700$ min$^2$, $\sum y_j^2 = 36$ injuries$^2$ and $\sum x_i y_j = 160$ min·injuries.

## Exercise#4_Solution

1. $\bar{x} = 20$ min, $s_x^2 = 187.5$ min$^2$.
   $\bar{y} = 1.75$ injuries, $s_y^2 = 1.4375$ injuries$^2$.
   $s_{xy} = -15$ min·injuries.
   Regression line of injuries on time of stetching: $y = 3.35 + -0.08x$.
2. $0.08$ injuries/min.
3.

Regression line of time of stretching on injuries: $x = 38.2609 + -10.4348y$.
$x(0) = 38.2609$ min.
$r^2 = 0.8348$.

# 9. Exercises

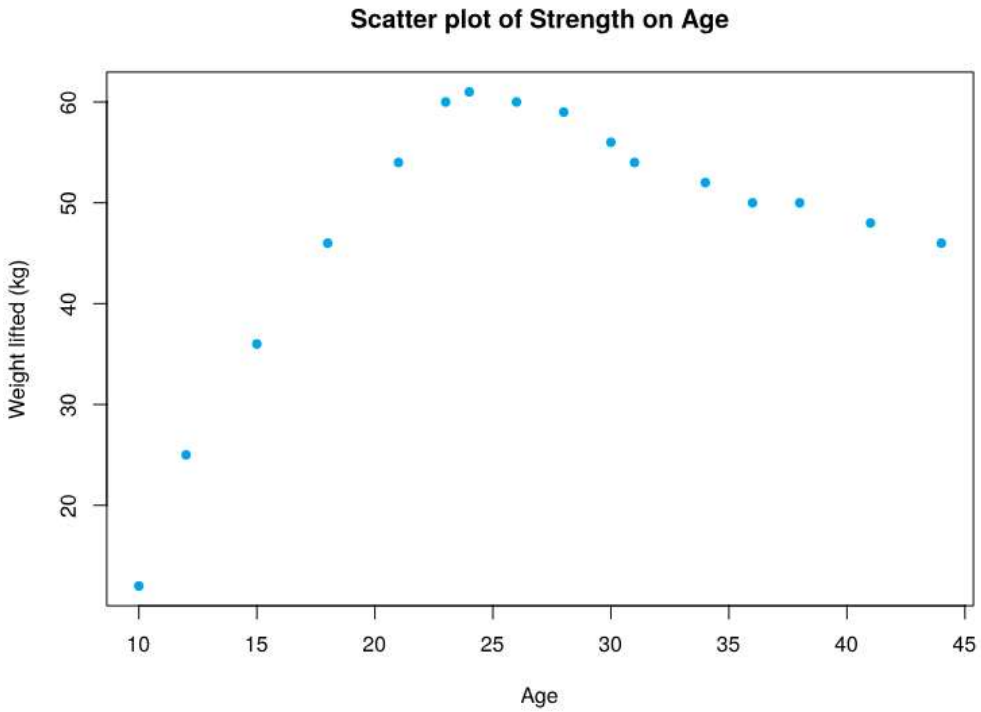
### Exercise#5

A study to determine the relation between the age and the physical strength gave the scatter plot below.

1.Calculate the linear coefficient of determination for the whole sample.

2.Calculate the linear coefficient of determination for the sample of people younger than 25 years old.

3.Calculate the linear coefficient of determination for the sample of people older than 25 years old



# 10. Annex_how to read scv file in python

**Using csv.reader()**

**Example:** This code reads and prints the contents of a CSV file named **'Giants.csv'** using the csv module in Python. It opens the file in read mode, reads the lines, and prints them one by one using a for loop. The **csv.reader()** function is used to read the CSV file, and the data from each row is printed to the console.

```
import csv
with open('Giants.csv', mode ='r')as file:
  csvFile = csv.reader(file)
  for lines in csvFile:
        print(lines)
```

```
['Organization', 'CEO', 'Established']
['Alphabet', 'Sundar Pichai', '02-Oct-15']
['Microsoft', 'Satya Nadella', '04-Apr-75']
['Amazon', 'Jeff Bezos', '05-Jul-94']
```

**Using csv.DictReader() class**

**Example:** This code reads and prints the contents of a CSV file named 'Giants.csv' using the csv module with DictReader. It opens the file in read mode, reads the lines
and prints them one by one. csv.DictReader() reads the CSV file and treats the first row as headers, creating a dictionary for each row where the header values are the keys. The code prints each row as a dictionary, making it easier to work with structured CSV data

```
import csv
with open('Giants.csv', mode ='r') as file:
      csvFile = csv.DictReader(file)
      for lines in csvFile:
            print(lines)
```

```
OrderedDict([('Organization', 'Alphabet'), ('CEO', 'Sundar Pichai'), ('Established', '02-Oct-15')])
OrderedDict([('Organization', 'Microsoft'), ('CEO', 'Satya Nadella'), ('Established', '04-Apr-75')])
OrderedDict([('Organization', 'Amazon'), ('CEO', 'Jeff Bezos'), ('Established', '05-Jul-94')])
```

**Using pandas.read_csv() method**

It is very easy and simple to read a CSV file using pandas library functions. Here read_csv() method of pandas library is used to read data from CSV files.

```
import pandas
csvFile = pandas.read_csv('Giants.csv')
print(csvFile)
```

```
  Organization            CEO Established
0    Alphabet  Sundar Pichai   02-Oct-15
1   Microsoft  Satya Nadella    04-Apr-75
2    Amazon     Jeff Bezos   05-Jul-94
```

## 10. Annex  Read specific columns using read_csv

```
df = pd.read_csv("people.csv", usecols=["First Name",
"Email"])
print(df)
```

| | First Name | Email |
|---|---|---|
| 0 | Shelby | elijah57@example.net |
| 1 | Phillip | bethany14@example.com |
| 2 | Kristine | bthompson@example.com |
| 3 | Yesenia | kaitlinkaiser@example.com |
| 4 | Lori | buchananmanuel@example.net |