

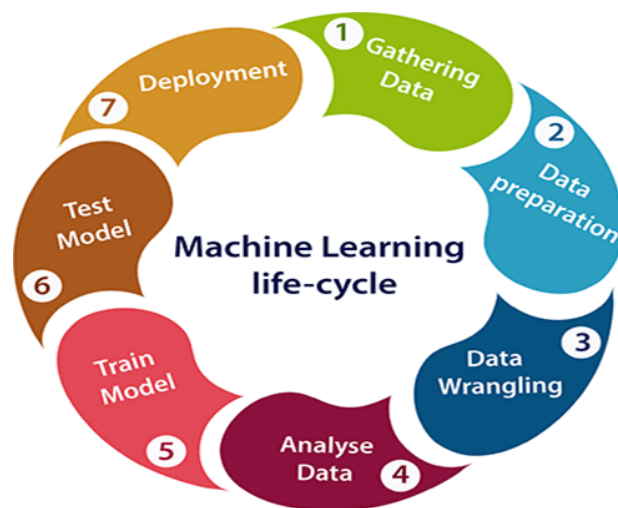
# Machine learning

**Machine Learning (ML):** Machine Learning is a subset of artificial intelligence (AI) that involves the development of algorithms and models that enable computers to learn and make predictions or decisions without being explicitly programmed. The primary goal is for the system to improve its performance on a task over time as it is exposed to more data.

## Types of Machine Learning:

1. **Supervised Learning:**
  - **Definition:** The algorithm is trained on a labeled dataset, where each input is paired with its corresponding output.
  - **Use Case:** Classification and Regression problems.
2. **Unsupervised Learning:**
  - **Definition:** The algorithm is given data without explicit instructions on what to do with it. It learns patterns and structures within the data.
  - **Use Case:** Clustering and Association problems.
3. **Semi-Supervised Learning:**
  - **Definition:** Combines aspects of both supervised and unsupervised learning. The algorithm is trained on a dataset that contains both labeled and unlabeled data.
  - **Use Case:** When acquiring labeled data is expensive but obtaining unlabeled data is easier.
4. **Reinforcement Learning:**
  - **Definition:** The algorithm learns by interacting with its environment. It receives feedback in the form of rewards or penalties and adjusts its strategy to maximize the cumulative reward.
  - **Use Case:** Game playing, robotic control, and autonomous systems.

## Machine learning life cycle



## **Gathering data**

1. Identify various data sources
2. Collect data
3. Integrate the data obtained from different sources

## **Data preparation**

1. **Dataexploration:**

It is used to understand the nature of data that we have to work with.

2. **Datapre-processing:**

Now the next step is preprocessing of data for its analysis.

## **3. Data Wrangling**

1. Missing Values
2. Duplicate data
3. Invalid data
4. Noise

## **4. Data Analysis**

1. Selection of analytical techniques
2. Building models
3. Review the result
4. It starts with the determination of the type of the problems, where we select the machine learning techniques such as **Classification, Regression, Cluster analysis, Association**, etc. then build the model using prepared data, and evaluate the model.

## **5. Train Model**

we train our model to improve its performance for better outcome of the problem.

## **6. Test Model**

we check for the accuracy of our model by providing a test dataset to it.

## **7. Deployment**

The last step of machine learning life cycle is deployment, where we deploy the model in the real-world system.

## **Data Preprocessing**

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model.

1. Getting the dataset
2. Importing libraries
3. Importing datasets
4. Finding Missing Data
5. Encoding Categorical Data
6. Splitting dataset into training and test set
7. Feature scaling

## 1) Get the Dataset

First thing we required is a dataset as a machine learning model completely works on data. we usually put it into a CSV **file**. CSV stands for "**Comma-Separated Values**" files; it is a file format which allows us to save the tabular data, such as spreadsheets. It is useful for huge datasets and can use these datasets in programs.

## 2) Importing Libraries

Three specific libraries that we will use for data preprocessing, which are:

**Numpy:** Numpy Python library is used for including any type of mathematical operation in the code. It is the fundamental package for scientific calculation in Python. It also supports to add large, multidimensional arrays and matrices.

**import numpy as nm**

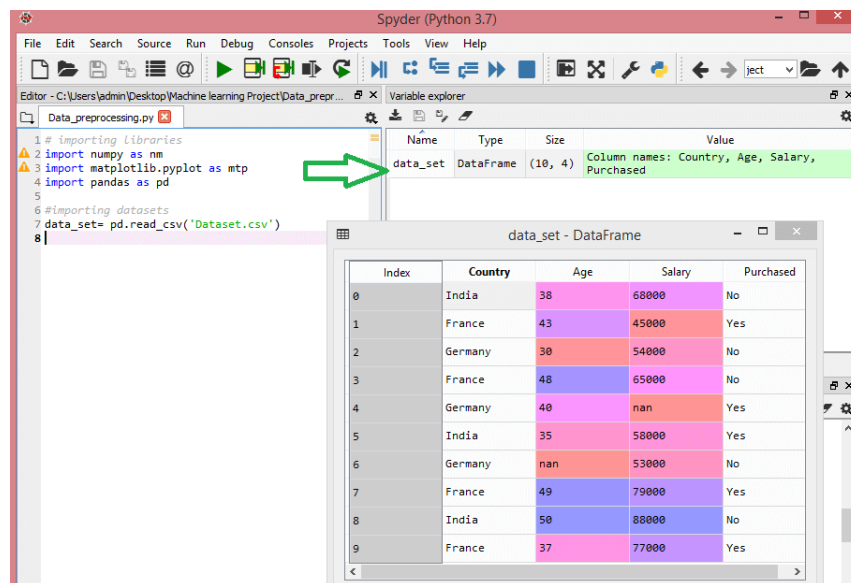
**Matplotlib:** The second library is **matplotlib**, which is a Python 2D plotting library, and with this library, we need to import a sub-library **pyplot**. This library is used to plot any type of charts in Python for the code.

**import matplotlib.pyplot as mpt**

**Pandas:** The last library is the Pandas library, which is one of the most famous Python libraries and used for importing and managing the datasets. It is an open-source data manipulation and analysis library.

**data\_set= pd.read\_csv('Dataset.csv')**

## Extracting dependent and independent variables:



The screenshot shows the Spyder Python IDE interface. The editor window displays a script named 'Data\_preprocessing.py' with the following code:

```
1 # importing libraries
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import pandas as pd
5
6 # importing datasets
7 data_set = pd.read_csv('Dataset.csv')
8
```

The Variable explorer on the right shows a variable named 'data\_set' of type 'DataFrame' with a size of (10, 4). The value of 'data\_set' is displayed in a separate window, showing a DataFrame with the following data:

Index	Country	Age	Salary	Purchased
0	India	38	68000	No
1	France	43	45000	Yes
2	Germany	30	54000	No
3	France	48	65000	No
4	Germany	40	nan	Yes
5	India	35	58000	Yes
6	Germany	nan	53000	No
7	France	49	79000	Yes
8	India	50	88000	No
9	France	37	77000	Yes

There are three independent variables that are **Country**, **Age**, and **Salary**, and one is a dependent variable which is **Purchased**.

**x= data\_set.iloc[:, :-1].values**      First(:)=all rows , Second(:)=all columns, -1=remove last column

To extract dependent variables, again, we will use Pandas .iloc[] method.

**y= data\_set.iloc[:, 3].values**

### 3) Data Cleaning:

Identify and handle missing or irrelevant data.

#### 1. Remove or fill missing values

##### 1. By deleting the particular row

##### 2. By calculating the mean

To handle missing values, we will use **Scikit-learn** library in our code, which contains various libraries for building machine learning models. Here we will use **Imputer** class of **sklearn.preprocessing** library.

```
#handling missing data (Replacing missing data with the mean value)
from sklearn.preprocessing import Imputer
imputer= Imputer(missing_values = 'NaN', strategy='mean', axis = 0)
#Fitting imputer object to the independent variables x.
imputerimputer= imputer.fit(x[:, 1:3])
#Replacing missing data with the calculated mean value
x[:, 1:3]= imputer.transform(x[:, 1:3])
```

3. Eliminate duplicate entries.

4. Address outliers that might negatively impact model performance.

#### 4.) Data Transformation

Convert data into a format suitable for machine learning algorithms.

- **Encoding Categorical Variables:** Convert categorical variables into numerical format (e.g., one-hot encoding). Categorical data is data which has some categories such as, in our dataset; there are two categorical variable, **Country**, and **Purchased**. Firstly, we will convert the country variables into categorical data. So to do this, we will use **LabelEncoder()** class from **preprocessing** library.

**For Country variable:**

```
#Categorical data
#for Country Variable
from sklearn.preprocessing import LabelEncoder
label_encoder_x= LabelEncoder()
x[:, 0]= label_encoder_x.fit_transform(x[:, 0])
```

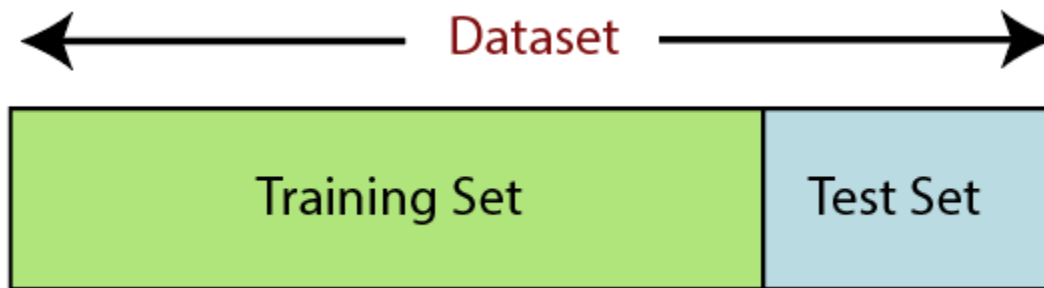
we have imported **LabelEncoder** class of **sklearn library**. This class has successfully encoded the variables into digits.

In our dataset, we have 3 categories so it will produce three columns having 0 and 1 values. For Dummy Encoding, we will use **OneHotEncoder** class of **preprocessing** library.

```
#for Country Variable
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
label_encoder_x= LabelEncoder()
x[:, 0]= label_encoder_x.fit_transform(x[:, 0])
#Encoding for dummy variables
onehot_encoder= OneHotEncoder(categorical_features= [0])
x= onehot_encoder.fit_transform(x).toarray()
```

- **Feature Scaling:**
  - Normalize or standardize numerical features to bring them to a similar scale.
- **Data Binning/Discretization:**
  - Group continuous data into bins or intervals.

## 5) Splitting the Dataset into the Training set and Test set



```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.2, random_state=0)
```

The last parameter **random\_state** is used to set a seed for a random generator so that you always get the same result, and the most used value for this is 42.

## 7) Feature Scaling

we put our variables in the same range and in the same scale so that no any variable dominate the other variable.

```
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
```

## Supervised Learning

### 1. Regression Analysis:

- Regression analysis helps us to understand how the value of the dependent variable is changing corresponding to an independent variable when other independent variables are held fixed.
- Example:** Predicting house prices based on features like size and number of bedrooms. It predicts continuous/real values such as **temperature, age, salary, price**, etc.

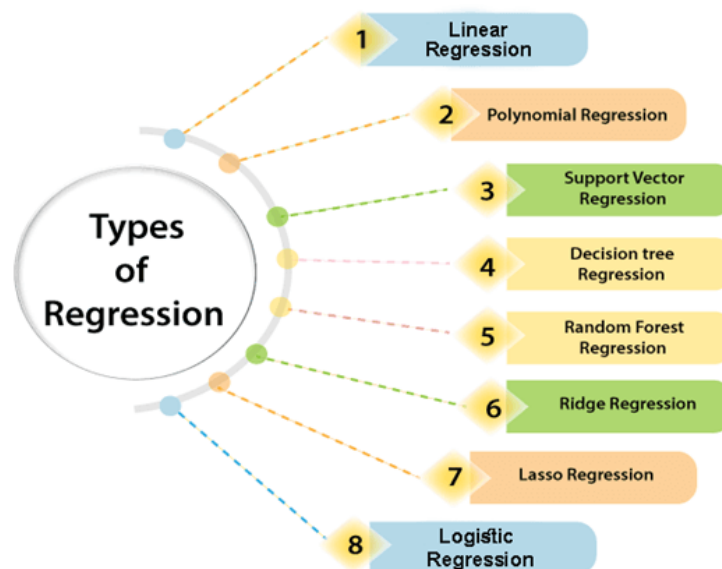
Advertisement	Sales
\$90	\$1000
\$120	\$1300
\$150	\$1800
\$100	\$1200
\$130	\$1380
\$200	??

***"Regression shows a line or curve that passes through all the datapoints on target-predictor graph in such a way that the vertical distance between the datapoints and the regression line is minimum."***

Terminologies Related to the Regression Analysis:

1. **Dependent Variable:** we want to predict or understand is called the dependent variable. It is also called **target variable**.
2. **Independent Variable:** The factors which affect the dependent variables or which are used to predict the values of the dependent variables are called independent variable, also called as a **predictor**.
3. **Outliers:** Outlier is an observation which contains either very low value or very high value in comparison to other observed values. An outlier may hamper the result, so it should be avoided.
4. **Multicollinearity:** If the independent variables are highly correlated with each other than other variables, then such condition is called Multicollinearity. It should not be present in the dataset, because it creates problem while ranking the most affecting variable.
5. **Underfitting and Overfitting:** If our algorithm works well with the training dataset but not well with test dataset, then such problem is called **Overfitting**. And if our algorithm does not perform well even with training dataset, then such problem is called **underfitting**.

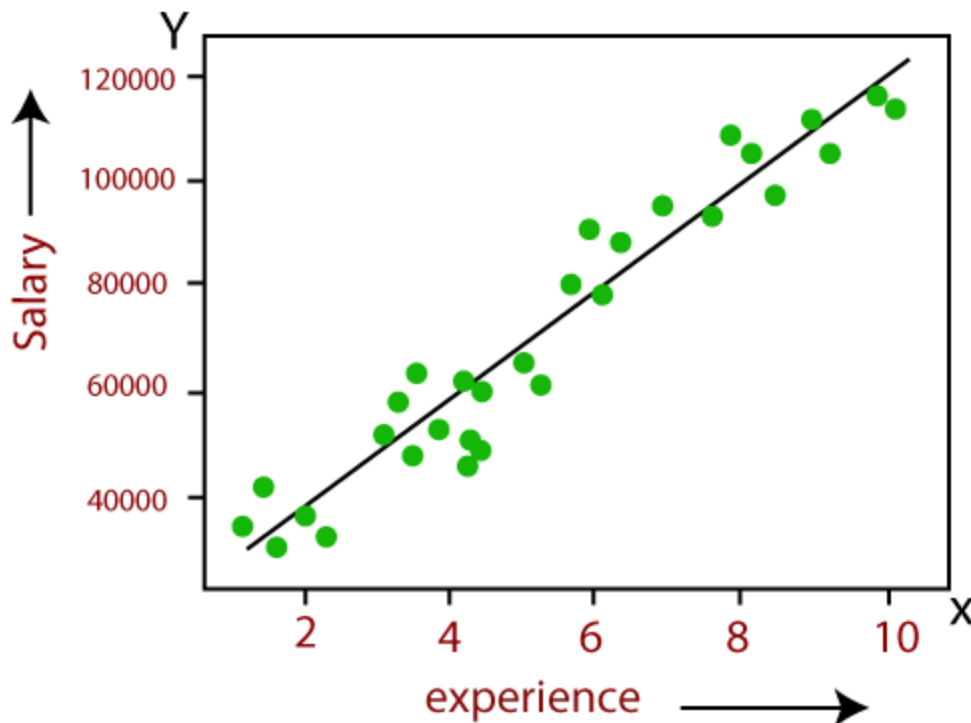
## Types of Regression



### Linear Regression:

- Linear regression is a statistical regression method which is used for predictive analysis.

- If there is only one input variable (x), then such linear regression is called **simple linear regression**. And if there is more than one input variable, then such linear regression is called **multiple linear regression**.



$$Y=ax+b$$

### Multiple Linear Regression

*Multiple Linear Regression is one of the important regression algorithms which models the linear relationship between a single dependent continuous variable and more than one independent variable.*

#### Example:

Prediction of CO<sub>2</sub> emission based on engine size and number of cylinders in a car.

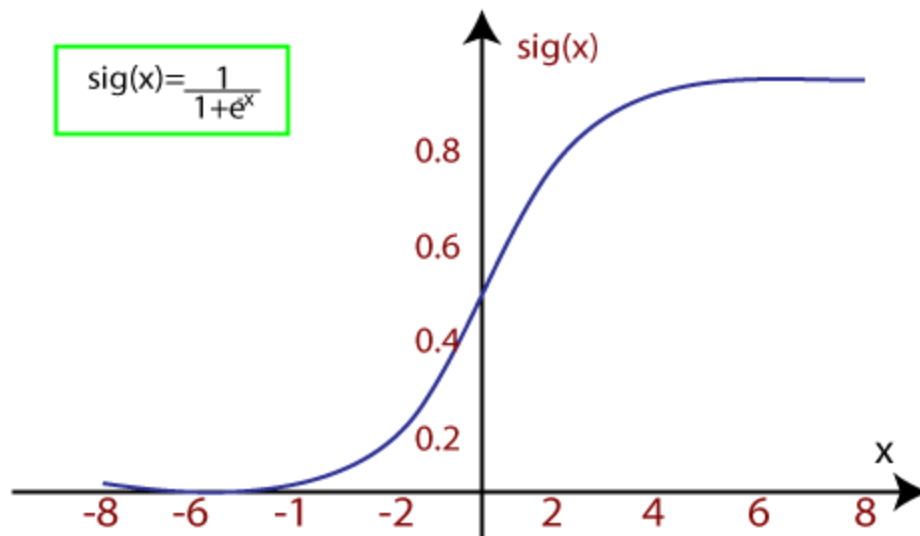
### Logistic Regression:

1. Uses to solve the classification problems. In **classification problems**, we have dependent variables in a binary or discrete format such as 0 or 1.
2. Logistic regression algorithm works with the categorical variable such as 0 or 1, Yes or No, True or False, Spam or not spam, etc.
3. Logistic regression uses **sigmoid function**. This sigmoid function is used to model the data in logistic regression.

$$f(x) = \frac{1}{1+e^{-x}}$$

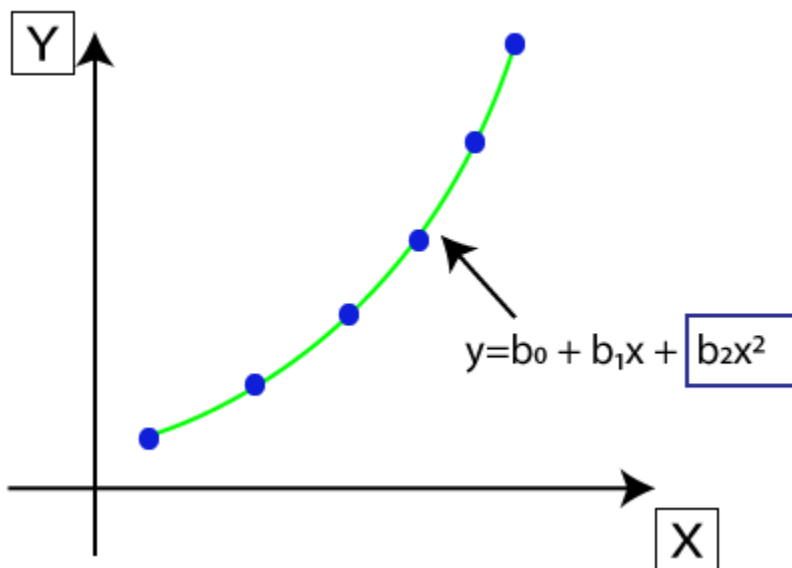


1.  $f(x)$ = Output between the 0 and 1 value.
2.  $x$ = input to the function
3.  $e$ = base of natural logarithm.



#### Polynomial Regression:

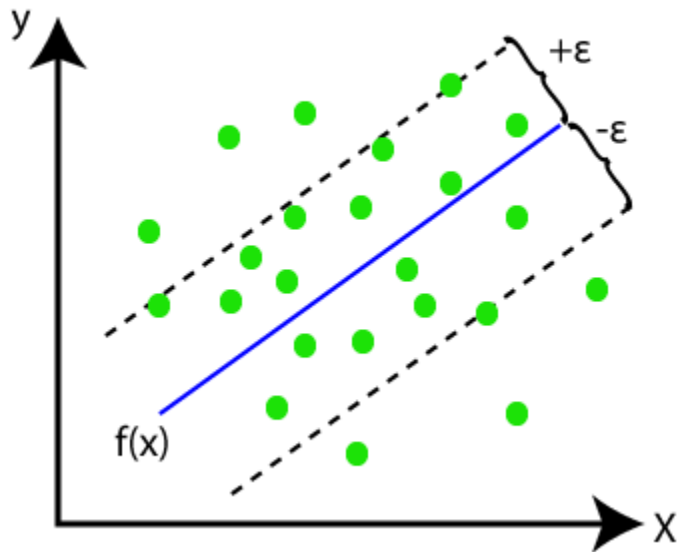
In Polynomial regression, the original features are transformed into polynomial features of given degree and then modeled using a linear model.



### Support Vector Regression:

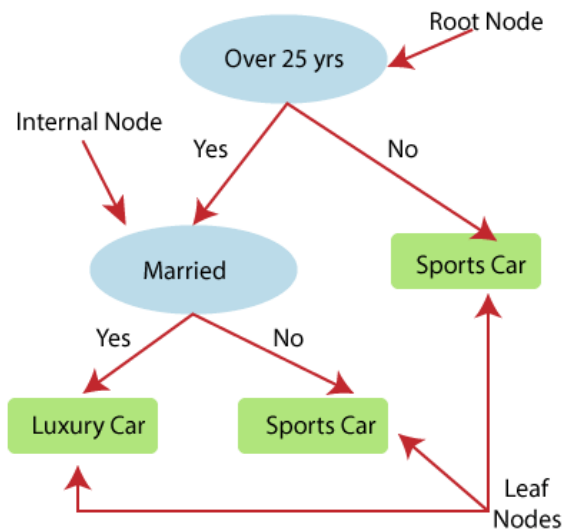
Support Vector Machine is a supervised learning algorithm which can be used for regression as well as classification problems.

*The main goal of SVR is to consider the maximum datapoints within the boundary lines and the hyperplane (best-fit line) must contain a maximum number of datapoints.*



### Decision Tree Regression:

Decision Tree regression builds a tree-like structure in which each internal node represents the "test" for an attribute, each branch represent the result of the test, and each leaf node represents the final decision or result.



**Ridge Regression:**

Ridge regression is one of the most robust versions of linear regression in which a small amount of bias is introduced so that we can get better long term predictions.

**Lasso Regression:**

It is similar to the Ridge Regression except that penalty term contains only the absolute weights instead of a square of weights.

**Cost Function and gradient descent**

the cost function tells you how wrong your predictions are, and gradient descent helps you adjust your model's parameters to minimize these errors, similar to finding the lowest point on a hill.

**Some common ML interview questions:****Q1- What is the curse of dimensionality? How does it affect machine learning algorithms?**

**Ans-**The curse of dimensionality is a bit like that in the world of machine learning. It happens when you have a lot of features or dimensions in your data. Each feature is like a dimension on your map. As you add more dimensions, the "space" your data occupies becomes huge and more difficult to navigate. To address the curse of dimensionality, it's often helpful to perform feature selection or dimensionality reduction. This means choosing the most relevant features or combining them to simplify the data.

**Q2- Explain the bias-variance tradeoff. Why is it important in machine learning?**

It's like training your robot friend to play a game: you want it smart enough to handle different situations but not so overtrained that it can't adapt to new levels. The bias-variance tradeoff is all about striking that right balance for a model that performs well in both familiar and new situations

**Q3- What is regularization, and why is it useful in machine learning models?**

**Regularization is a technique to prevent a model from getting too complex or overfitting to the training data.** Regularization is a way to guide machine learning models to be not too obsessed with the training data and not too complicated. It's about finding that "just right" level of complexity to make the model perform well on both the training data and new, unseen data

#### Q4- What are precision and recall? How do they relate to the concept of false positives and false negatives?

**Ans- Precision:** Imagine you're a detective trying to catch bad guys. Precision is like measuring how accurate you are when you say you've caught a criminal. High precision means you're very certain when you make an arrest—you're not grabbing innocent people by mistake.

**Recall:** Now, recall is like making sure you don't miss any bad guys. High recall means you're capturing most of the actual criminals, and very few are slipping through the cracks.

#### Q5- Can you explain the concept of a confusion matrix and how it is used to evaluate the performance of a classification model?

**Ans-**Imagine you're a detective solving a case, and you've classified suspects as guilty or innocent based on evidence. The confusion matrix helps you understand how well you did in classifying suspects.

##### Confusion Matrix Basics:

1. **True Positives (TP):**
  - Suspects correctly classified as guilty. You got it right!
2. **True Negatives (TN):**
  - Innocent suspects correctly classified as innocent. Another correct decision!
3. **False Positives (FP):**
  - Innocent suspects incorrectly classified as guilty. Oops! A false accusation.
4. **False Negatives (FN):**
  - Guilty suspects incorrectly classified as innocent. A missed opportunity.

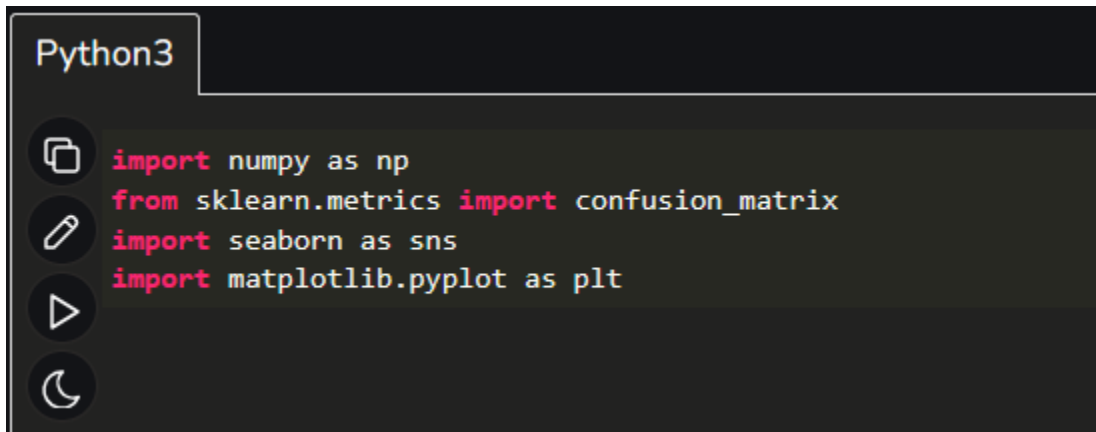
##### Building a Confusion Matrix:

- **Let's say you've investigated 100 suspects:**
    - 30 are guilty, and 70 are innocent.
  - **Your model predicts their guilt or innocence:**
    - **True Positives (TP):** 20 suspects correctly identified as guilty.
    - **True Negatives (TN):** 60 suspects correctly identified as innocent.
    - **False Positives (FP):** 10 innocent suspects incorrectly identified as guilty.
    - **False Negatives (FN):** 10 guilty suspects incorrectly identified as innocent.
1. **Accuracy:**
    - $\text{Accuracy} = \frac{TP + TN}{FP + FN + TP + TN}$
    - Measures the overall correctness of your predictions.
  2. **Precision (Positive Predictive Value):**
    - $\text{Precision} = \frac{TP}{FP + TP}$
    - Measures the accuracy of positive predictions. How many predicted positives are actually positive.
  3. **Recall (Sensitivity, True Positive Rate):**
    - $\text{Recall} = \frac{TP}{FN + TP}$
    - Measures the ability to capture all the actual positives. Out of all actual positives, how many did you catch.

#### 4. **F1-score**

is used to evaluate the overall performance of a classification model. It is the harmonic mean of precision and recall

$$\text{F1-Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$



```
Python3
import numpy as np
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
```

#### **Q6- What is the role of the learning rate in gradient descent optimization?**

**Ans-** In the context of training machine learning models, the learning rate is a critical parameter in gradient-based optimization algorithms like stochastic gradient descent (SGD), Adam, and others. It influences how the model's parameters are updated during training.

Imagine you're blindfolded on a foggy mountain, trying to find the lowest point. The learning rate determines the size of the steps you take while feeling the slope under your feet.

#### **Q7- What is the difference between deep learning and machine learning?**

**Ans-** Machine learning is like teaching a computer with explicit instructions and features. Deep learning is like letting the computer learn on its own, discovering features and patterns in the process.

##### **When to Use Each:**

##### **Use Machine Learning When:**

- You have a good understanding of the relevant features.
- The problem can be solved with explicit rules and feature engineering.

##### **Use Deep Learning When:**

- The problem involves complex patterns and representations.
- The amount of data is large, and feature engineering might be challenging

#### **Q8- What is the purpose of activation functions in a neural network? Can you give examples of activation functions?**

**Ans-** Imagine you're trying to predict whether a photo contains a cat or a dog using a neural network. Each neuron in the network makes a decision, and activation functions help add complexity and flexibility to these decisions.

1. **Sigmoid:**

- The sigmoid function squashes input values between 0 and 1. It's often used in the output layer of binary classification models to produce probabilities.

$$\text{Sigmoid}(x) = 1 / (1 + e^{-x})$$

2. **ReLU (Rectified Linear Unit):**

- ReLU is a popular activation function that returns the input for positive values and zero for negative values. It's simple and effective, often used in hidden layers.

$$\text{ReLU}(x) = \max(0, x)$$

3. **Tanh (Hyperbolic Tangent):**

- Tanh squashes input values between -1 and 1. Like the sigmoid, it's used in certain situations, especially in the hidden layers.

$$\text{Tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

4. **Softmax:**

- Softmax is commonly used in the output layer for multi-class classification problems. It converts raw scores into probability distributions.

$$\text{Softmax}(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

## TensorFlow

Imagine you're a chef, and you need to follow a recipe. TensorFlow is like your kitchen assistant, helping you perform complex cooking tasks efficiently. In the world of programming and machine learning, TensorFlow is a powerful library that makes it easier to build and train machine learning models.

### Why We Use TensorFlow:

1. **Building Models:**

- TensorFlow helps you create and define the structure of your machine learning models, whether it's for image recognition, language processing, or other tasks.

2. **Training Models:**

- It provides tools to train your models on data, helping them learn patterns and make predictions.

3. **Scalability:**

- TensorFlow can scale from running on a single device to distributed computing across multiple machines, making it suitable for both small and large projects.

4. **Deep Learning:**

- It's particularly popular for deep learning tasks, where neural networks with many layers are used to understand and learn from complex data.

### When to Use TensorFlow:

- **You might use TensorFlow when:**

- You need to build and train complex machine learning models.
- Your project involves deep learning and neural networks.
- You want scalability and flexibility in your machine learning tasks.

**1. Installing TensorFlow:**

pythonCopy code

```
pip install tensorflow
```

**2. Importing TensorFlow in Python:**

pythonCopy code

```
import tensorflow as tf
```

**3. Creating a Simple Tensor (Number):**

pythonCopy code

```
x = tf.constant(5)
print(x)
```

**4. Performing Operations:**

pythonCopy code

```
y = x + 10
print(y)
```

**5. Building a Simple Neural Network:**

pythonCopy code

```
model = tf.keras.Sequential([
    tf.keras.layers.Dense(128, activation='relu', input_shape=(input_s:
    tf.keras.layers.Dense(10, activation='softmax')
])
```

↓

**6. Compiling and Training the Model:**

pythonCopy code

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=
model.fit(train_data, train_labels, epochs=10, batch_size=32)
```

## Numpy

NumPy is a powerful library for numerical operations.

### Why We Use NumPy:

1. **Efficient Numerical Operations:**
2. **Multidimensional Arrays:**
3. **Mathematical Functions:**
4. **Linear Algebra Operations:**

#### 1. Installing NumPy:

```
python
```

[Copy code](#)

```
pip install numpy
```

#### 2. Importing NumPy in Python:

```
python
```

[Copy code](#)

```
import numpy as np
```

#### 3. Creating NumPy Arrays:

```
python
```

[Copy code](#)

```
arr = np.array([1, 2, 3, 4, 5])
```

#### 4. Performing Operations:

##### \* Element-wise Operations:

```
python
```

[Copy code](#)

```
result = arr + 2
```

##### \* Array Operations:

```
python
```

[Copy code](#)

```
sum_array = np.sum(arr)
```



Thank you

Manu Chaudhary