

Machine learning

Life cycle of Machine Learning

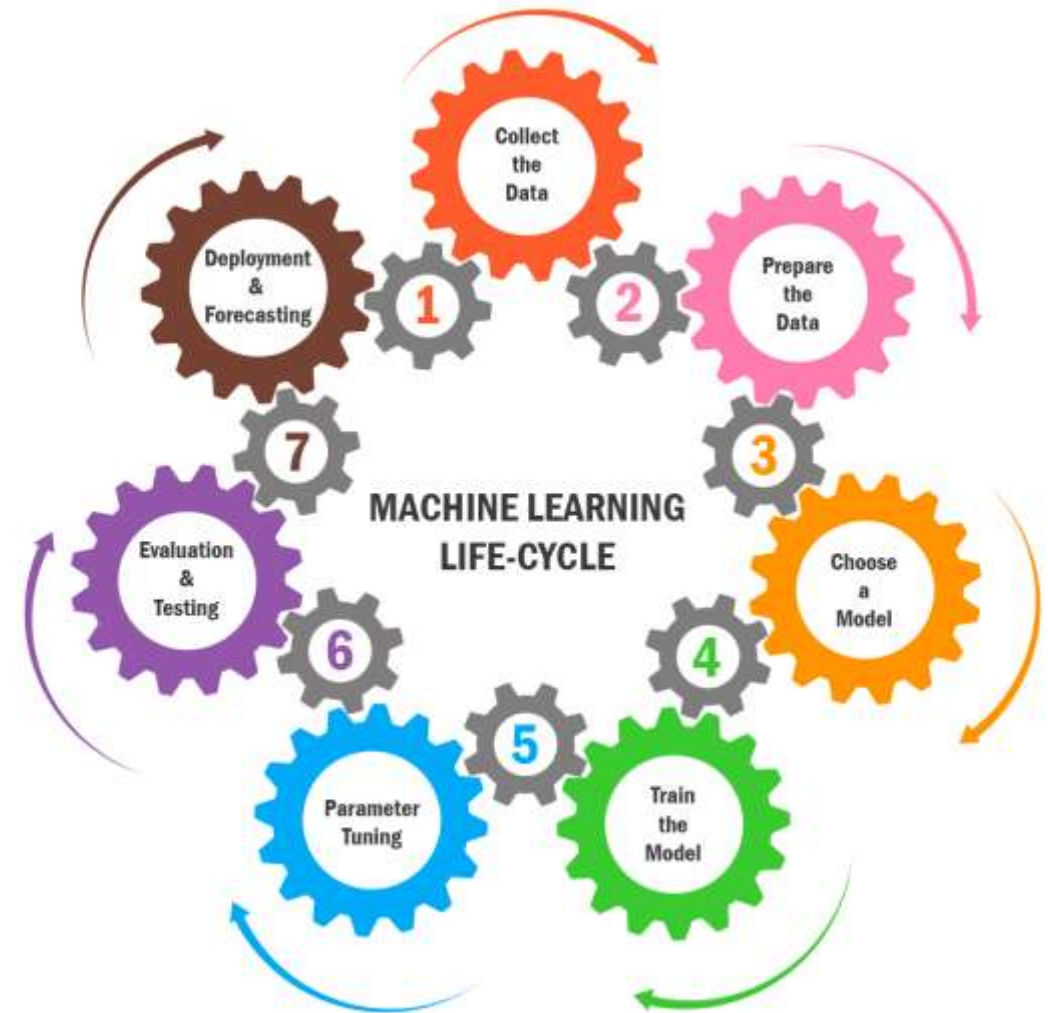
1. Collect the Data

- Depending upon the type of model, find proprietary data, accessing public data or a mixture of both
- Data is a fundamental part of machine learning, the quality and quantity of your data can have direct consequences for model performance.

Gathering the data



Integrating the data



Data collection

Gather Data

- Different sources such as databases, text files, pictures, sound files, or web scraping may be used for data collection
- make sure that the gathering the right data from the right source.

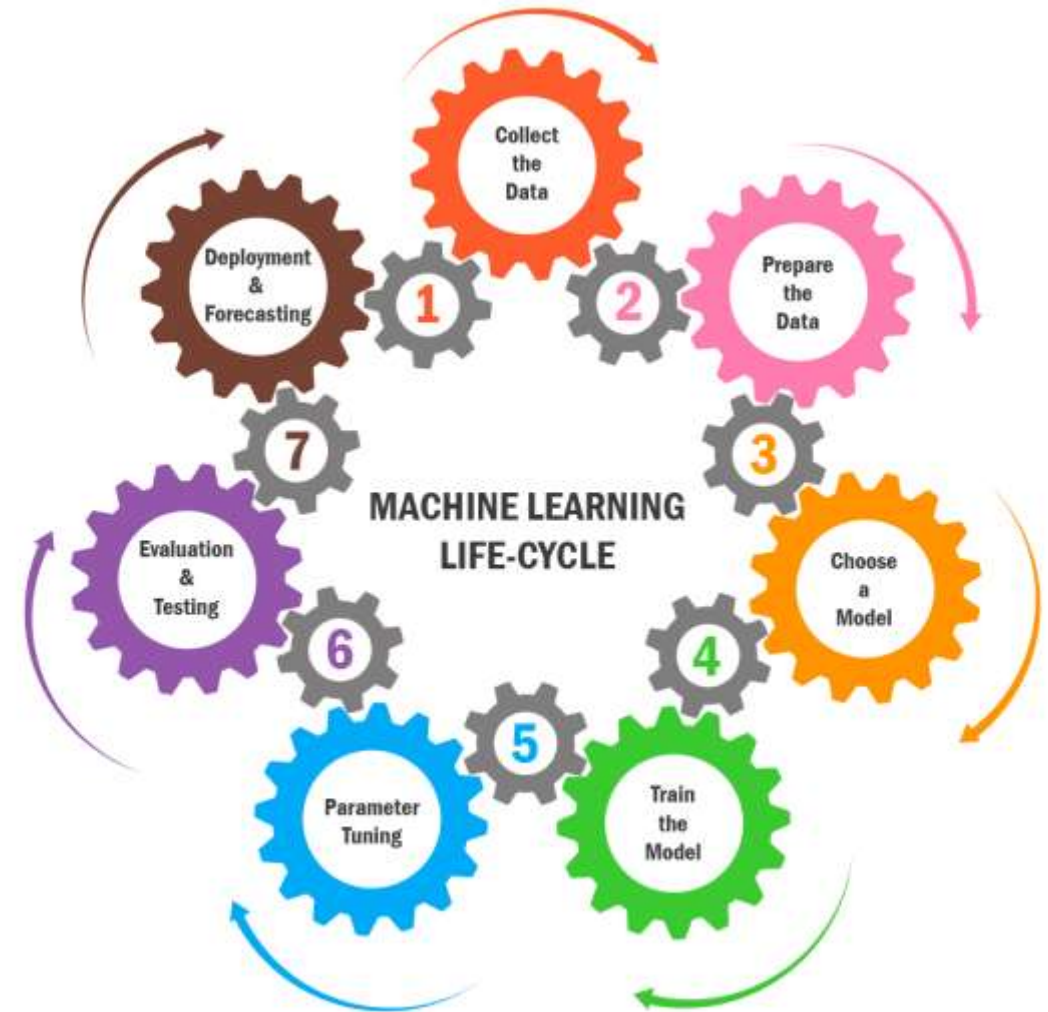
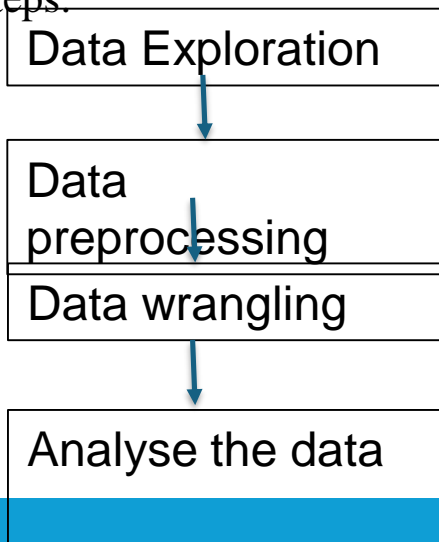
Integrate Data

- This next step is to integrate the data that you've gathered with your workflow and, ultimately, your machine learning model.
- This may mean importing the data into your proprietary database or using APIs to set up an automated feed of data from third-party sources.

Life cycle of Machine Learning

2. Preparing the Data

- Pre-processing of data is a key step in the process of machine learning. It involves **deleting duplicate data**, **fixing errors**, **managing missing data** either by eliminating or filling it in, and **adjusting and formatting the data**.
- There are four steps:



Preparing the Data

Data Exploration

- This is also where you'll identify the approach that you'll take during the next two steps to make sure that you have everything ready for the algorithm.

Data Pre-Processing

- Pre-processing involves cleaning up any formatting that might be in place and stripping out blank entries and other anomalous elements within the data.
- We're talking about actions that you can carry out across the whole dataset to make it ready for further processing rather than focusing on any individual entries.

Data Wrangling

- Data wrangling requires you to manually go through the data that you have and to update any of them that need updating for your company to be able to process it.
- This is also where you'll carry out any changes to the data that are needed to make it readable and easy to process for the model that you build.

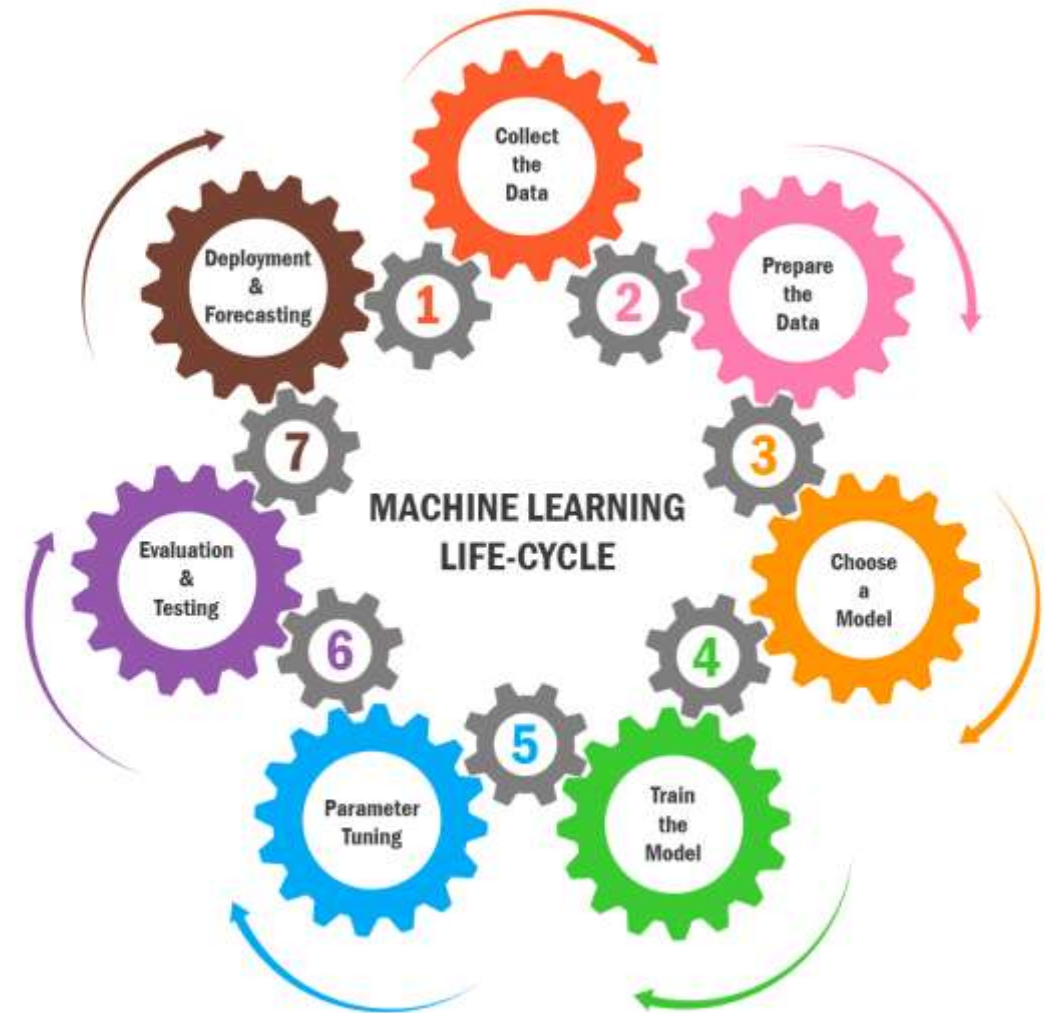
Analyse Data

- By now, your data should be in pretty good shape, so the next step is for you to take a closer look at the data that you have and analyze it to determine how you're going to go about processing it and building your model.

Life cycle of Machine Learning

3. Choose the Model

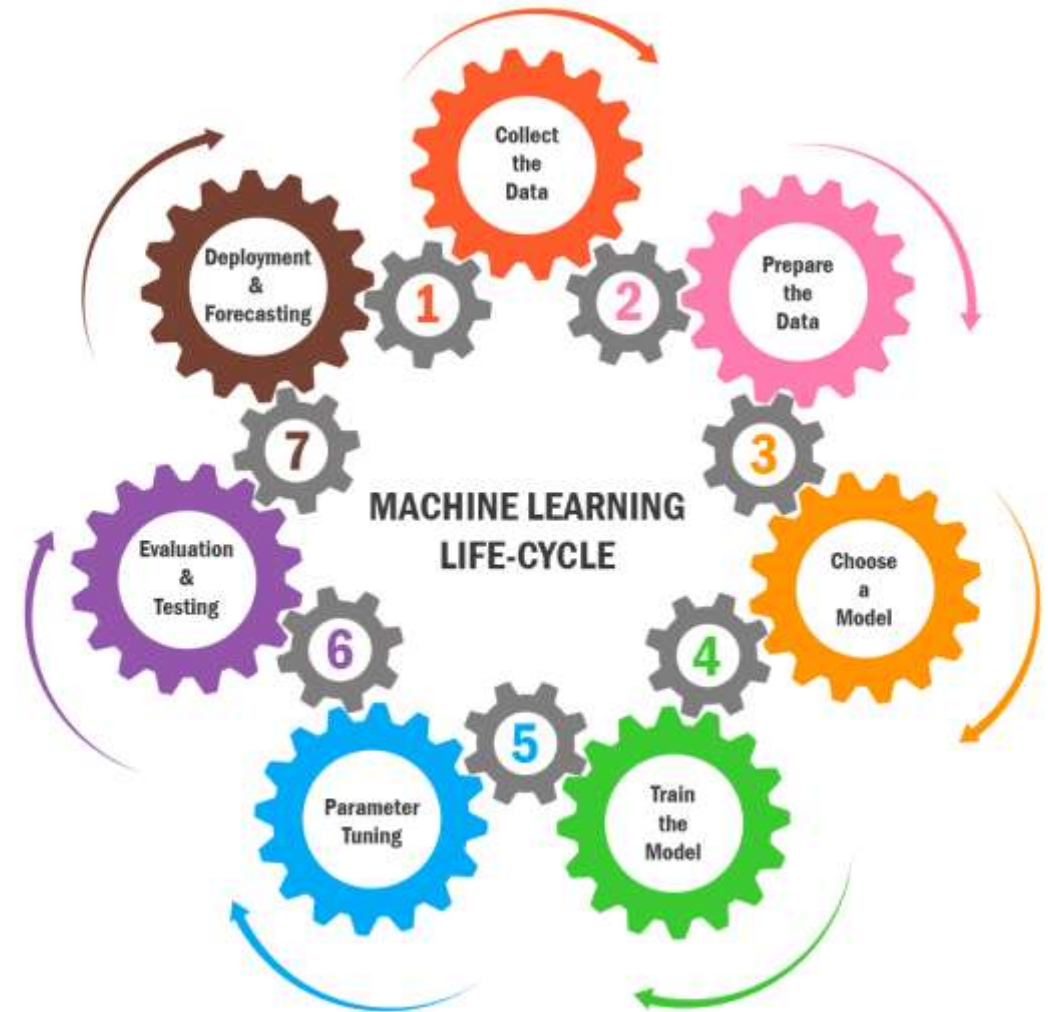
- The next step is to select a machine learning model; once data is prepared then we apply it to ML Models like **Linear regression**, **decision trees**, and **Neural Networks** that may be selected to implement.
- The selection of the model generally depends on what kind of data you're dealing with and your problem.
- The size and type of data, complexity, and computational resources should be taken into account when choosing a model to apply



Life cycle of Machine Learning

4. Train the Model

- The next step is to train it with the data that has been prepared after you have chosen a model.
- Training is about connecting the data to the model and enabling it to adjust its parameters to predict output more accurately.
- Overfitting and underfitting must be avoided during the training.



Overfitting and underfitting

Overfitting

- Overfitting happens when a model is **too complex and learns not just the actual patterns in the training data, but also the noise (random mistakes or details that don't matter)**.
- model does great on the training data but performs poorly on new, unseen data because it's too specific to the training data.
- **Example:** Imagine you're trying to predict house prices based on features like size and number of rooms. If you use a very complex model (like a deep decision tree), it might fit the training data perfectly, even capturing random noise. But it won't do well on new data because it's "too attached" to the original data.

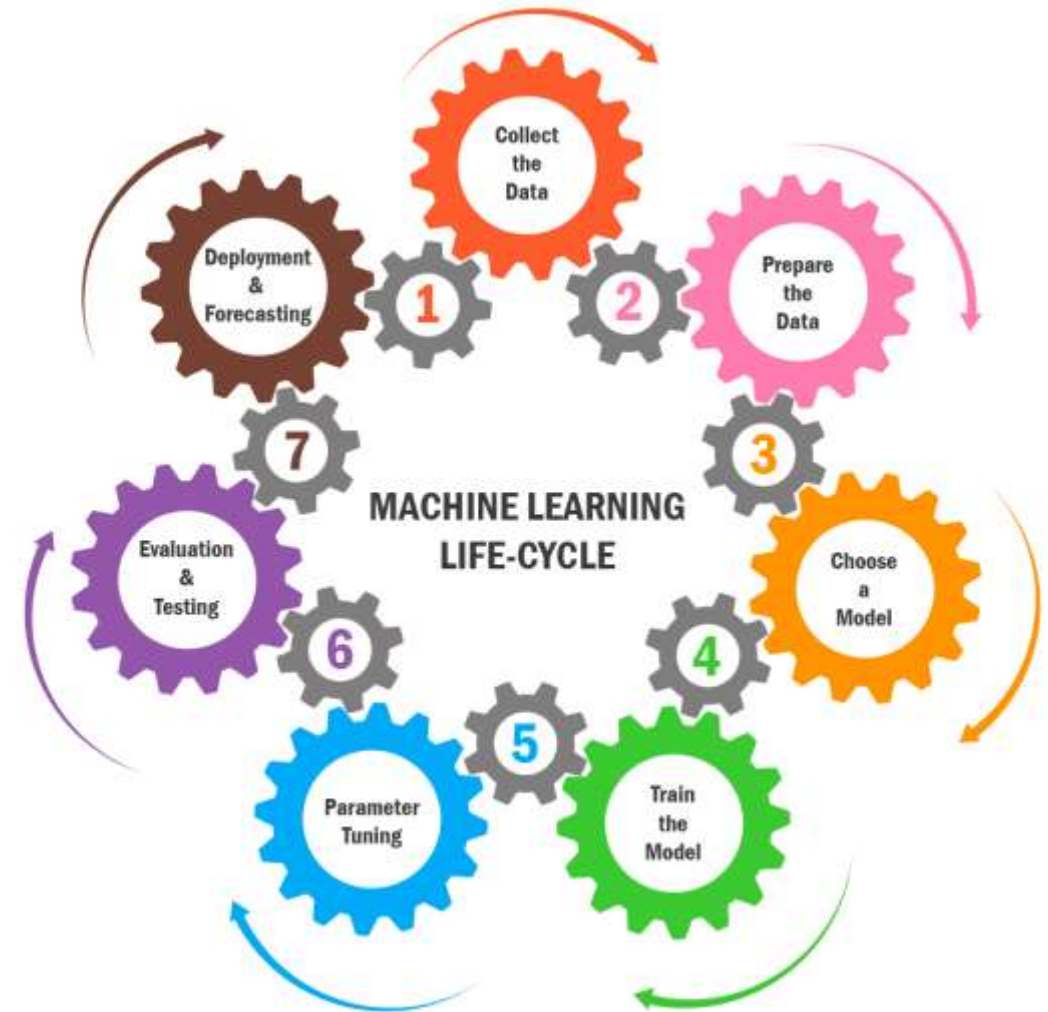
Underfitting

- Underfitting happens **when a model is too simple and can't capture important patterns** in the data.
- The model doesn't perform well on either the training data or new data because it's not learning enough from the data.
- **Example:** If you use a simple linear model to predict house prices based only on the number of rooms, it might not capture other important factors (like location or size), and so it doesn't work well.

Life cycle of Machine Learning

5. Model Parameter Tuning

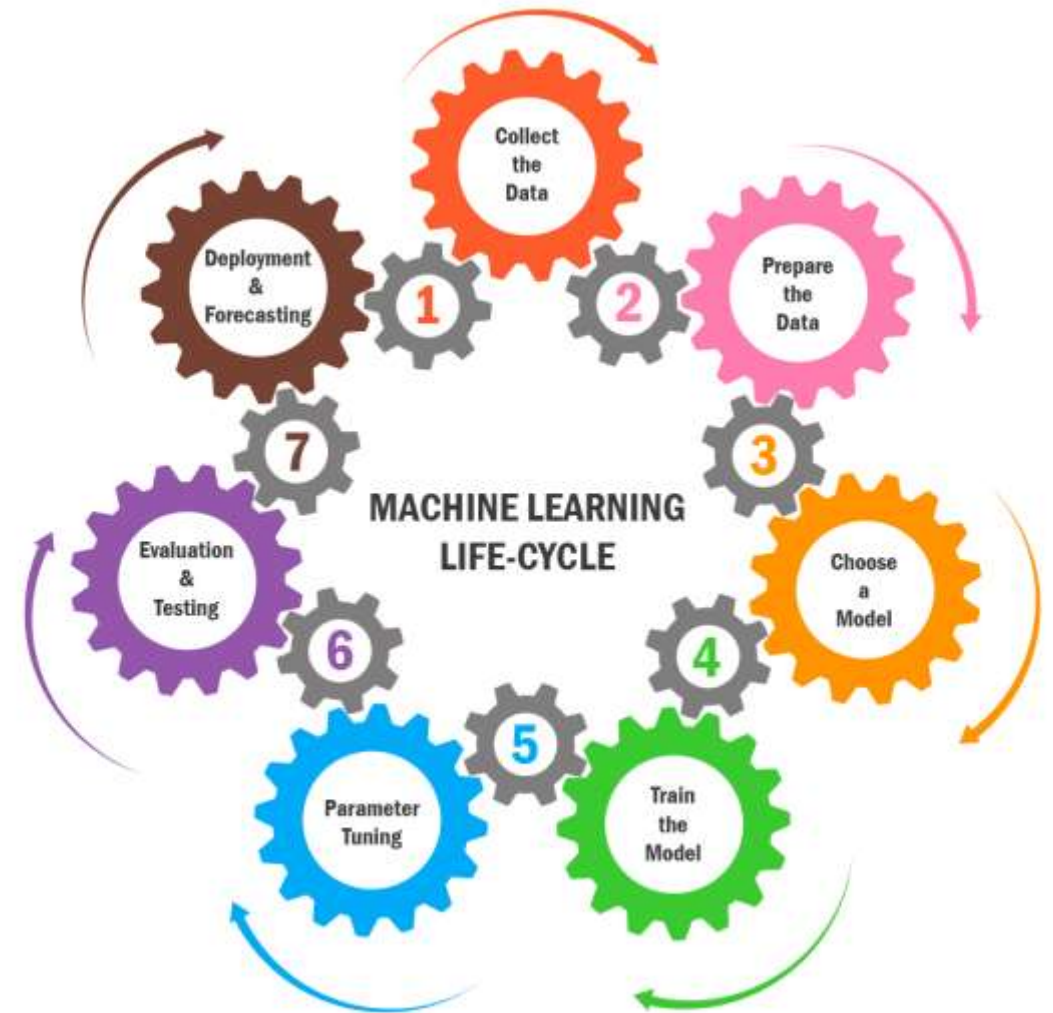
- what changes you need to make to your model to fine-tune it and ensure that it does a better job of taking you toward your goals.
- You can repeat steps five and six over and over again, one after the other, until you're ready to move on to the seventh and final step.



Life cycle of Machine Learning

6. Model Evaluation and Testing

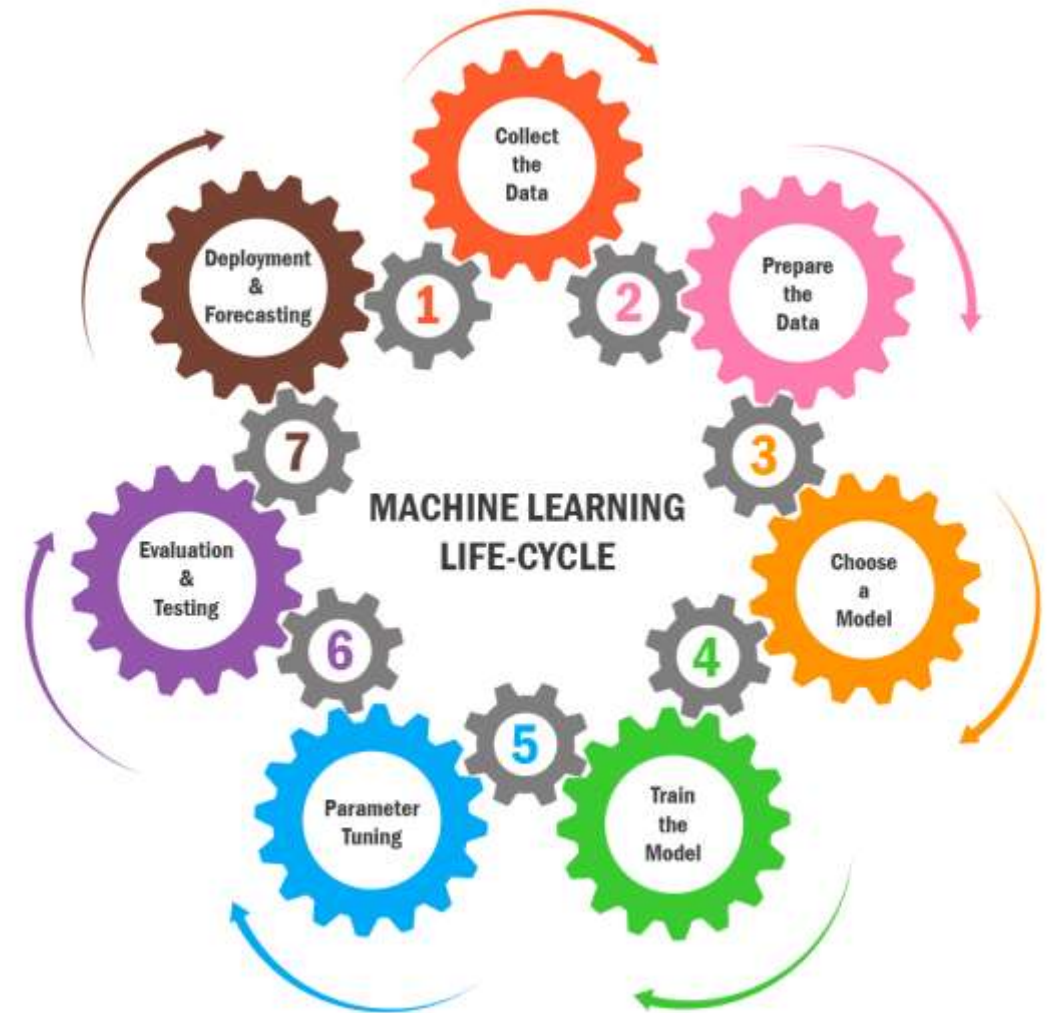
- It is important to assess the model's performance before deployment as soon as a model has been trained.
- This means that the model has to be tested on new data that was not seen during training.



Life cycle of Machine Learning

7. Prediction and Evaluation

- This is done by adding new data to the model and using its output for decision-making or other analysis
- deployment of this model involves its integration into a production environment where it is capable of processing real-world data and providing timely information.



Features of ML

- Machine learning uses data to detect various patterns in a given dataset.
- It can learn from past data and improve automatically.
- It is a data-driven technology.
- Machine learning is much similar to data mining as it also deals with the huge amount of the data.

Relevance of ML

- Rapid increment in the production of data.
- Solving complex problems, which are difficult for a human.
- Decision making in various sector including finance.
- Finding hidden patterns and extracting useful information from data.

Challenges of Machine Learning

- Incomplete, noisy, or unbalanced data can make it difficult for models to learn effectively.
- Striking the right balance between overfitting (model memorizing the training data) and underfitting (model being too simple to learn patterns) is tricky.
- Some machine learning models, especially deep learning models, are like "black boxes," meaning it's hard to understand how they make decisions.
- Machine learning models can inherit biases from the data they are trained on, leading to unfair or biased outcomes

Challenges of Machine Learning

- Training machine learning models, especially deep learning models, **requires a lot of computational power** and time, which can be costly.
- Handling **sensitive data such as personal information** while respecting privacy laws and regulations.
- Ensuring that models trained on one dataset perform well on unseen or new data, especially in real-world environments.

Relevance of Tools in Machine Learning (ML)

- Machine learning tools play a critical role in different stages, from data collection to deployment.
- Choosing the right tools can enhance efficiency, scalability, and model performance.

1. Data Collection & Preprocessing

- Tools **helps in cleaning, transforming, and structuring data before feeding it into models.**

Why They Matter?

- Poor data quality leads to inaccurate predictions, it impacts on the model accuracy.
- Cleaning, transforming, and labeling data can be time-consuming.
- Efficient preprocessing reduces model training time.
- Big data tools helps in handling large-scale ML tasks.

Tools:

- **Pandas & NumPy:** Data manipulation and preprocessing.
- **Dask & Spark:** Processing large-scale datasets in parallel.
- **OpenCV:** Image and video data preprocessing.
- **Hadoop & Google BigQuery:** Distributed data storage and query processing.

Pandas & NumPy: Data manipulation and preprocessing

```
import pandas as pd
import numpy as np

# Create a DataFrame
df = pd.DataFrame({
    'A': np.random.randint(1, 100, 5),
    'B': np.random.rand(5),
    'C': ['apple', 'banana', 'cherry', 'date', 'elderberry']
})

# Convert column A to NumPy array
A_array = df['A'].to_numpy()

# Normalize column A using NumPy
df['A_normalized'] = (A_array - np.mean(A_array)) / np.std(A_array)

print(df)
```

Dask : Processing large-scale datasets in parallel

```
import dask.dataframe as dd

# Convert Pandas DataFrame to Dask
dask_df = dd.from_pandas(df, npartitions=2)

# Compute mean of column A
print(dask_df['A'].mean().compute())
```

Spark (PySpark) → Distributed Data Processing

```
from pyspark.sql import SparkSession

# Initialize Spark session
spark = SparkSession.builder.appName("MLTools").getOrCreate()

# Convert Pandas to Spark DataFrame
spark_df = spark.createDataFrame(df)

# Show data
spark_df.show()
```

OpenCV → Image Processing

```
import cv2
import matplotlib.pyplot as plt
from google.colab import files
uploaded = files.upload()

image_path = "Figure_1.png"
# Read an image
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

# Display the image using Matplotlib
plt.imshow(image, cmap='gray')
plt.axis("off") # Hide axis labels
plt.show()
```

2. Exploratory Data Analysis (EDA) & Visualization

Why It's Important?

- Helps understand patterns, correlations, and data distribution.
- Identifies missing values and outliers.

Relevant Tools:

- **Matplotlib & Seaborn** – Data visualization.
- **Plotly & Bokeh** – Interactive visualizations.
- **Pandas Profiling** – Automated data exploration reports.

3. Model Development & Training

Why It's Important?

- Choosing the right model and training efficiently is crucial for performance.
- Hyperparameter tuning improves accuracy.

Relevant Tools:

- **Scikit-learn** – Traditional ML models (SVM, Random Forest, etc.).
- **TensorFlow & PyTorch** – Deep learning frameworks for neural networks.
- **XGBoost & LightGBM** – Gradient boosting algorithms for structured data.
- **Optuna & Hyperopt** – Automated hyperparameter tuning.

4. Model Evaluation & Validation

Why It's Important?

- Ensures the model generalizes well to new data.
- Prevents overfitting and underfitting.

Relevant Tools:

- **Scikit-learn Metrics** – Accuracy, precision, recall, F1-score.
- **MLflow** – Model tracking and logging.
- **Cross-validation tools (Stratified K-Fold, LOOCV, etc.)** – Improve reliability.

5. Model Deployment & Serving

Why It's Important?

- A model must be efficiently deployed to real-world applications.
- Deployment tools enable integration with APIs and mobile/web applications.

Relevant Tools:

- **Flask & FastAPI** – Lightweight API development for model serving.
- **TensorFlow Serving & TorchServe** – Scalable deep learning model deployment.
- **Docker & Kubernetes** – Containerization and orchestration.
- **AWS SageMaker, Google AI Platform, Azure ML** – Cloud-based deployment.

6. Model Monitoring & Maintenance

Why It's Important?

- Model performance can degrade due to data drift or changing patterns.
- Continuous monitoring helps detect anomalies.

Relevant Tools:

- **Prometheus & Grafana** – Real-time model monitoring.
- **Evidently AI** – Detects model drift and performance changes.
- **Kubeflow & MLflow** – Model tracking and lifecycle management.

7. AutoML (Automated Machine Learning)

Why It's Important?

- Speeds up ML model development by automating feature selection, tuning, and training.
- Helps non-experts build ML models.

Relevant Tools:

- **Google AutoML & AutoKeras** – Deep learning automation.
- **H2O.ai & TPOT** – Automated ML pipeline creation.

Concept Representation in Machine Learning

- Concept representation in ML refers to how knowledge, patterns, and relationships within data are structured and learned by models.
- It plays a crucial role in **determining how well a model generalizes to unseen data.**
- Below are key aspects of concept representation in ML:
 1. Feature based Representation
 2. Distributed Representation
 3. Symbolic Representation
 4. Probabilistic Representation

1.Feature-Based Representation (Vector Space Representation)

- Data is represented as feature vectors, where each dimension corresponds to a specific attribute.
- Used in classical ML models like SVMs, Decision Trees, and Neural Networks.

Example:

For a fruit classification problem, a feature vector for an apple could be:

[Red(1),Round(1),Weight=150g,Sweetness=8/10]

Example :Classifying Animals Based on Features

Classifying animals as either a "**Mammal**" or "**Non-Mammal**" based on a few features:

Animal	Fur	Legs	Warm-blooded	Mammal?
Dog	Yes	4	Yes	Yes
Fish	No	0	No	No
Cat	Yes	4	Yes	Yes
Frog	No	4	No	No

Example : Feature-Based Representation (Vector Space Representation)

A **vector representation** simply means converting the features into a fixed-size vector of values.

Animal	Fur (1/0)	Legs (Count)	Warm-blooded (1/0)	Mammal? (Label)
Dog	1	4	1	1
Fish	0	0	0	0
Cat	1	4	1	1
Frog	0	4	0	0

2.Distributed Representation (Embeddings)

- Each concept is represented as a continuous vector in a high-dimensional space.
- Common in deep learning models, especially for NLP and image recognition.

Examples:

- **Word2Vec, GloVe, BERT (NLP):** Represent words as dense vectors capturing semantic meaning.
- **ResNet, CNN Embeddings (Computer Vision):** Represent images as feature maps.

Example of Distributed Representation for Features

Feature	Distributed Vector Representation
Fur (Yes)	[0.9, 0.1, 0.4]
Fur (No)	[0.1, 0.9, 0.6]
Legs (0)	[0.2, 0.5, 0.7]
Legs (4)	[0.8, 0.4, 0.1]
Warm-blooded (Yes)	[0.9, 0.3, 0.7]
Warm-blooded (No)	[0.2, 0.8, 0.4]

Distributed Representation for Animals

To represent the **Dog**, we combine the distributed vectors of **Fur (Yes)**, **Legs (4)**, and **Warm-blooded (Yes)**:

$$\text{Dog} = \text{Fur(Yes)} + \text{Legs(4)} + \text{Warm-blooded(Yes)}$$

$$\text{Dog} = [0.9, 0.1, 0.4] + [0.8, 0.4, 0.1] + [0.9, 0.3, 0.7]$$

$$\text{Dog} = [2.6, 0.8, 1.2]$$

3. Symbolic Representation (Logic-Based & Rule-Based Systems)

- Concepts are defined using rules and symbolic logic.
- Used in Expert Systems and Knowledge Graphs.

Example:

IF color = red **AND** shape = round **THEN** classify as apple.

4. Probabilistic Representation

- Represents uncertainty in concepts using probabilities.
- Used in models like Bayesian Networks and Gaussian Mixture Models.

Example:

- Probability of a patient having a disease given certain symptoms:

$$\mathbf{P(Disease|Symptoms)=P(Symptoms|Disease)P(Disease)/P(Symptoms)}$$

Challenges in Concept Representation

- 1.Feature Selection** – Choosing the right features is critical for performance.
- 2.Dimensionality Reduction** – High-dimensional representations can lead to overfitting .
- 3.Explainability** – Deep learning representations are often difficult to interpret.
- 4.Concept Drift** – Representations may become outdated as data evolves.

Classification Errors in Machine Learning

- Classification errors occur when a machine learning model incorrectly predicts the class of an instance.
- These errors impact model performance and are typically analyzed using evaluation metrics like
 - Accuracy
 - Confusion Matrix
 - Precision
 - Recall or Sensitivity
 - F -Score / F 1 Measure
 - AUC (Area Under the Curve) -ROC

Types of Classification Errors

A. False Positive (FP) – Type I Error

- The model **incorrectly predicts a positive class** when the actual class is negative.

Example: A medical test incorrectly predicts a person has a disease when they are actually healthy.

B. False Negative (FN) – Type II Error

- The model **incorrectly predicts a negative class** when the actual class is positive.

Example: A spam filter fails to classify a spam email as spam and allows it into the inbox.

C. True Positive (TP) & True Negative (TN) (Correct Predictions)

- **True Positive (TP):** The model correctly predicts a positive instance.
- **True Negative (TN):** The model correctly predicts a negative instance.

Evaluating Classification Errors :Accuracy

- The **accuracy** metric is one of the simplest Classification metrics to implement, and it can be determined *as the number of correct predictions to the total number of predictions.*

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

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

- Although it is simple to use and implement, it is suitable only for cases where an equal number of samples belong to each class.
- **Measures overall correctness but can be misleading in imbalanced datasets.**

Confusion Matrix

- A confusion matrix is a **tabular representation of prediction outcomes** of any binary classifier, which is used to **describe the performance of the classification model on a set of test data when true values are known.**
- The confusion matrix is simple to implement
- A typical confusion matrix for a binary classifier looks like the shown image(However, it can be extended to use for classifiers with more than two classes).

Confusion matrix

		Predicted classes	
		Negative 0	Positive 1
Actual classes	Negative 0	TN	FP
	Positive 1	FN	TP

n=165	Predicted: NO	Predicted: YES
Actual: NO	50	10
Actual: YES	5	100

- In the matrix, columns are for the **prediction values**, and rows specify the **Actual values**.

- Here Actual and prediction give two possible classes, Yes or No. So, if we are predicting the presence of a disease in a patient, the Prediction column with Yes means, Patient has the disease, and for NO, the Patient doesn't have the disease.
- In this example, the total number of predictions are 165, out of which 110 time predicted yes, whereas 55 times predicted No.
- However, in reality, 60 cases in which patients don't have the disease, whereas 105 cases in which patients have the disease.

Precision (Positive Predictive Value)

- The precision metric is used to overcome the limitation of Accuracy.
- The precision determines the *proportion of positive prediction that was actually correct*.
- $\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$
- High precision means fewer false positives.
- Important in applications where false positives are costly (e.g., fraud detection).

Recall (Sensitivity, True Positive Rate)

- High recall means fewer false negatives.
- It is also similar to the precision metric; however, it aims to calculate the proportion of *actual positive that was identified incorrectly*.
- It can be calculated as *a true positive or a prediction that are true to the total number of positives*, either correctly predicted as positive or incorrectly predicted as negative (true positive and false negative)

$$\text{Recall} = \frac{TP}{TP + FN}$$

- Important when missing positive cases is critical (e.G., Disease detection).

F-Scores

- **F-score** or **F1 Score** is a metric to evaluate a binary classification model on the basis of predictions that are made for the positive class.
- It is calculated with the help of Precision and Recall.
- It is a type of single score that represents both Precision and Recall. So, *the F1 Score can be calculated as the harmonic mean of both precision and Recall, assigning equal weight to each of them.*

$$F1 - score = 2 * \frac{precision * recall}{precision + recall}$$

AUC-ROC

- Sometimes we need to visualize the performance of the classification model on charts; then, we can use the **AUC-ROC curve**.
- It is one of the popular and important metrics for evaluating the performance of the classification model.
- Firstly, let's understand ROC (Receiver Operating Characteristic curve) curve.

ROC

•**ROC** represents a graph to show the performance of a classification model at different threshold levels. The curve is plotted between two parameters, which are:

True Positive Rate
False Positive Rate

•**TPR** or **True Positive Rate** is a synonym for Recall, hence can be calculated as:

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

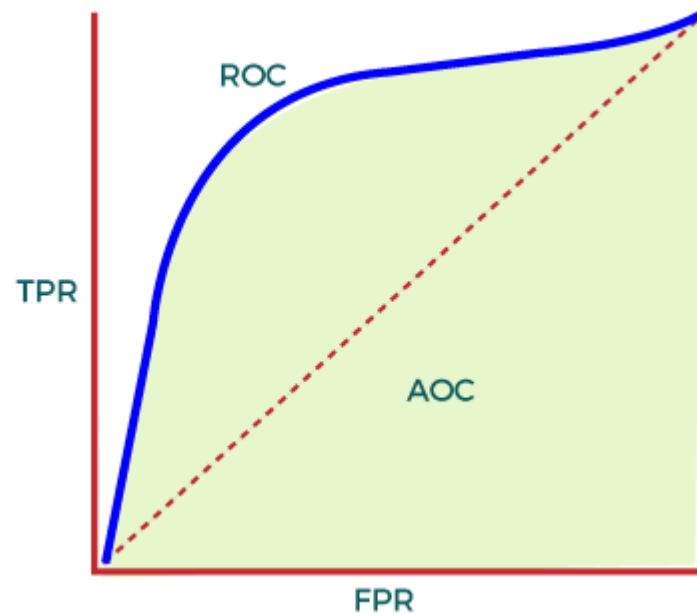
FPR or False Positive Rate can be calculated as:

$$FPR = \frac{FP}{FP + TN}$$

Evaluating a logistic regression model multiple times with different classification thresholds is inefficient. Instead, use **ROC (Receiver Operating Characteristic) curves** and **AUC (Area Under the Curve)**, which efficiently summarize the model's performance across different threshold values.

AUC: Area Under the ROC curve

AUC is known for **Area Under the ROC curve**. As its name suggests, AUC calculates the two-dimensional area under the entire ROC curve, as shown below image:



- **AUC** calculates the performance across all the thresholds and provides an aggregate measure.
- **The value of AUC ranges from 0 to 1.**
- It means a model with 100% wrong prediction will have an AUC of **0.0**, whereas models with 100% correct predictions will have an AUC of **1.0**

The Equations of 4 Key Classification Metrics

Accuracy:

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

Recall:

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

Precision:

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

F₁ score:

$$F_1 = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}}$$

Misclassification Rate in Machine Learning

Misclassification rate is the proportion of incorrect predictions made by a classification model.

$$\text{Misclassification Rate} = \frac{\text{Number of Incorrect Predictions}}{\text{Total Number of Predictions}}$$

Or:

$$\text{Misclassification Rate} = 1 - \text{Accuracy}$$

Reducing Classification Errors

- **Feature Engineering:** Improve data representation.
- **Balanced Dataset:** Handle class imbalance using resampling techniques.
- **Threshold Tuning:** Adjust decision thresholds to optimize precision or recall.
- **Model Selection:** Use more suitable algorithms (e.g., boosting models for better classification).
- **Regularization:** Prevent overfitting with L1/L2 regularization.

Problem 1:

• Given a confusion matrix, calculate the following:

1. Accuracy
2. Precision
3. Recall
4. F1 Score

	Predicted No	Predicted Yes
Actual No	45	5
Actual Yes	5	95

- Consider the confusion matrix given below for a binary classifier predicting the presence of a disease
- The classifier made a total of 150 predictions
Out of those 150 cases, the classifier predicted "yes" 100 times, and "no" 50 times.
- In reality, 100 patients in the sample have the disease, and 50 patients do not.

	Predicted No	Predicted Yes
Actual No	45	5
Actual Yes	5	95

Calculations:

- **Accuracy:** Overall, how often is the classifier correct?

$$\begin{aligned} \text{Accuracy} &= \frac{TN+TP}{TN+FP+FN+TP} \\ &= \frac{45 + 95}{150} = 93.33\% \end{aligned}$$

	Predicted No	Predicted Yes
Actual No	TN = 45	FP = 5
Actual Yes	FN = 5	TP = 95

- **Misclassification Rate:** Overall, how often is it wrong?

- $$\text{Missclassification Rate} = \frac{FN+FP}{TN+FP+FN+TP}$$

$$= \frac{5 + 5}{150} = 6.67\%$$

	Predicted No	Predicted Yes
Actual No	TN = 45	FP = 5
Actual Yes	FN = 5	TP = 95

Calculation of Recall/Sensitivity/True Positive Rate

- **True Positive Rate:** When it's actually yes, how often does it predict yes?
- also known as "Sensitivity" or "Recall"

$$\text{True Positive rate} = \frac{TP}{\text{Actual Yes}}$$

$$= \frac{95}{100} = 95\%$$

	Predicted No	Predicted Yes
Actual No	TN = 45	FP = 5
Actual Yes	FN = 5	<u>TP = 95</u>

Calculation of True Negative Rate

- **True Negative Rate:** When it's actually no, how often does it predict no?
- also known as "Specificity"

$$\text{True Negative rate} = \frac{TN}{\text{Actual No}}$$

$$= \frac{45}{50} = 90\%$$

	Predicted No	Predicted Yes
Actual No	<u>TN = 45</u>	FP = 5
Actual Yes	FN = 5	TP = 95

- **Precision:** When it predicts yes, how often is it correct?

$$\text{Precision} = \frac{TP}{\text{Predicted Yes}}$$

$$= \frac{95}{100} = 95\%$$

	Predicted No	Predicted Yes
Actual No	TN = 45	FP = 5
Actual Yes	FN = 5	TP = 95

- **Prevalence:** How often does the yes condition actually occur in our sample?

$$\text{Prevalence} = \frac{\text{Actual Yes}}{\text{Total}}$$

$$= \frac{100}{150} = 66.67\%$$

	Predicted No	Predicted Yes
Actual No	TN = 45	FP = 5
Actual Yes	FN = 5	TP = 95

Formula for Remember:

$$TPR = \frac{TP}{\text{Actual Positive}} = \frac{TP}{TP + FN}$$

$$FNR = \frac{FN}{\text{Actual Positive}} = \frac{FN}{TP + FN}$$

$$TNR = \frac{TN}{\text{Actual Negative}} = \frac{TN}{TN + FP}$$

$$FPR = \frac{FP}{\text{Actual Negative}} = \frac{FP}{TN + FP}$$

- Even if data is imbalanced, we can figure out that our model is working well or not.
- For that, **the values of TPR and TNR should be high, and FPR and FNR should be as low as possible.**

Example - 1

ID	Target	Prediction	ID	Target	Prediction	ID	Target	Prediction
1	False	False	8	True	True	15	False	False
2	False	False	9	False	False	16	False	False
3	False	False	10	False	False	17	True	False
4	False	False	11	False	False	18	True	True
5	True	True	12	True	True	19	True	True
6	False	False	13	False	False	20	True	True
7	True	True	14	True	True			

- **Calculate – Accuracy, Misclassification, Precision, Recall, and F1 Score by constructing a confusion matrix.**

Solution

- **STEP – 1**

- **Construction of Confusion Matrix**

ID	Target	Prediction	ID	Target	Prediction	ID	Target	Prediction
1	False	False	8	True	True	15	False	False
2	False	False	9	False	False	16	False	False
3	False	False	10	False	False	17	True	False
4	False	False	11	False	False	18	True	True
5	True	True	12	True	True	19	True	True
6	False	False	13	False	False	20	True	True
7	True	True	14	True	True			

Actual Class	Predicted Class	
	True	False
True	TP	FN
False	FP	TN

Actual Class	Predicted Class	
	True	False
True	8	1
False	0	11

- Calculation of Accuracy

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} = \frac{8+11}{8+11+0+1} = 0.95$$

- Calculation of Misclassification

$$Missclassification Rate = \frac{FP+FN}{TP+TN+FP+FN} = 0.05$$

- Calculate Precision, Recall and F1 Score

$$Precision = \frac{TP}{TP + FP} = \frac{8}{8 + 0} = \mathbf{1}$$

$$Recall = \frac{TP}{TP + FN} = \frac{8}{8 + 1} = \mathbf{0.889}$$

$$F_1 \text{ Score} = \frac{2 * Precision * Recall}{Precision + Recall}$$

$$F_1 \text{ Score} = \frac{2 * 1 * 0.889}{1 + 0.889} = \mathbf{\underline{0.941}}$$

Example -2

- Suppose 10000 patients get tested for flu; out of them, 9000 are healthy and 1000 are actually sick.
- For the sick people, a test was positive for 620 and negative for 380.
- For the healthy people, the same test was positive for 180 and negative for 8820.
- Construct a confusion matrix for the data and compute the accuracy, precision and recall for the data.

Solution

- **True Positive (TP)**: positive test result matches reality - person is actually sick and tested positive.
- **False Positive (FP)**: positive test result doesn't match reality - test is positive but the person is not actually sick.
- **True Negative (TN)**: negative test result matches reality - person is not sick and tested negative.
- **False Negative (FN)**: negative test result doesn't match reality - test is negative but the person is actually sick.

		Predicted ✓	
		Yes	No
Actual ✓	Yes	TP	FN
	No	FP	TN

- Suppose 10000 patients get tested for flu; out of them, 9000 are healthy and 1000 are actually sick.
- For the sick people, a test was positive for 620 and negative for 380.
- For the healthy people, the same test was positive for 180 and negative for 8820.
- Construct a confusion matrix for the data and compute the accuracy, precision and recall for the data.

		Predicted	
		Yes	No
Actual	Yes	TP=620	FN=380
	No	FP=180	TN=8820

		Predicted	
		Yes	No
Actual	Yes	TP=620	FN=380
	No	FP=180	TN=8820

- **Accuracy:** How often is the test correct?

$$- \text{Accuracy} = \frac{TP+TN}{N}$$

$$- \text{Accuracy} = \frac{620+8820}{10000} = 0.944$$

		Predicted	
		Yes	No
Actual	Yes	TP=620	FN=380
	No	FP=180	TN=8820

- **Misclassification rate:** How often the test is wrong?

$$- \text{Misclassification Rate} = 1 - \text{Accuracy} = \frac{FP + FN}{N}$$


$$- \text{Misclassification Rate} = \frac{180 + 380}{10000} = 0.056$$

		Predicted	
		Yes	No
Actual	Yes	TP=620	FN=380
	No	FP=180	TN=8820

- **Precision or Positive Predictive Value (PPV):** When the prediction is positive, how often is it correct?

$$-Precision = \frac{TP}{TP+FP}$$

$$-Precision = \frac{620}{620+180} = 0.775$$

		Predicted 	
		Yes	No
Actual	Yes	TP=620	FN=380
	No	FP=180	TN=8820

- **Negative Predictive Value (NPV):** When the prediction is negative, how often is it correct?

$$- \text{Negative Predictive Value (NPV)} = \frac{TN}{TN+FN}$$

$$- \text{Negative Predictive Value (NPV)} = \frac{8820}{8820+380} = 0.959$$

		Predicted	
		Yes	No
Actual	Yes	TP=620	FN=380
	No	FP=180	TN=8820

- **Positive Likelihood Ratio:** Odds of a positive prediction given that the person is sick (used with odds formulations of probability).

$$- \text{Positive Likelihood Ratio} = \frac{TPR}{FPR}$$

$$- \text{Positive Likelihood Ratio} = \frac{0.62}{0.02} = 31$$

		Predicted	
		Yes	No
Actual	Yes	TP=620	FN=380
	No	FP=180	TN=8820

- **Negative Likelihood Ratio:** Odds of a positive prediction given that the person is not sick.

$$- \text{Negative Likelihood Ratio} = \frac{FNR}{TNR}$$

$$- \text{Negative Likelihood Ratio} = \frac{0.38}{0.98} = 0.388$$

Example

Consider a two-class classification problem of predicting whether a photograph contains a man or a woman. Suppose we have a test dataset of 10 records with expected outcomes and set of predictions from our classification algorithm.

	Expected	Predicted
1	man	woman
2	man	man
3	woman	woman
4	man	man
5	woman	man
6	woman	woman
7	woman	woman
8	man	man
9	man	woman
10	woman	woman

- (a) Compute the confusion matrix for the data.
- (b) Compute the accuracy, precision, recall, sensitivity and specificity of the data.

Problems to Refer

- Numerical Problems - Classification II Confusion Matrix, Precision, Recall Numerical Problems