

## Model Evaluation & Metrics

### 1. The Basics

- When you build a classification model (like predicting spam or not spam), you need a way to measure how good it is. The measurements are called as classification metrics.

### 2. Confusion Matrix

- Imagine you predicted if a person has a disease (Yes or No)
- The confusion matrix compares predictions with actual reality

	Predicted : Yes	Predicted : No
Actual : Yes	TP (True Positive)	FN (False Negative)
Actual : No	FP (False Positive)	TN (True Negative)

- TP → Model predicted Yes, and it was actually Yes
- TN → Model predicted No, and it was actually No
- FP → Model predicted Yes, but it was actually No (False alarm)
- FN → Model predicted No, but it was actually Yes (Missed detection)

### 3. Main Metrics

- a) Accuracy :- How often the model is correct

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

- Good if classes are balanced (equal Yes/No cases)
- Bad if classes are imbalanced (like rare disease)

f) ROC - AUC (Receiver operating characteristic - Area Under Curve)

classmate

Date \_\_\_\_\_

Page \_\_\_\_\_

- Plots True Positive Rate vs False Positive rate
- AUC score closer to 1 = better model

h) Precision

- Out of all predicted Yes, how many were actually Yes,

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

$$\text{TP} + \text{FP}$$

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

High precision → fewer false alarms

i) Recall (Sensitivity / True Positive Rate)

- Out of all actual Yes, how many did we catch?

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

High recall → fewer missed cases

j) F1-Score

- Balance between Precision and Recall

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- Good When you want both high precision and high recall

k) Specificity

- Out of all actual No, how many did we correctly predict as No?

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

#### 4. Quick Example

Suppose:

$$\cdot \text{TP} = 50, \text{TN} = 40, \text{FP} = 10, \text{FN} = 5$$

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

$$\text{Accuracy} = \frac{50 + 40}{50 + 40 + 10 + 5} = \frac{90}{105} = 85.7\%$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{50}{50 + 10} = \frac{50}{60} = 83.3\%$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{50}{50 + 5} = \frac{50}{55} = 90.9\%$$

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

$$= \frac{2 \times (0.833 \times 0.909)}{(0.833 + 0.909)}$$

$$= \frac{2 \times 0.757197}{1.742}$$

$$= 2 \times 0.434$$

$$= 0.869$$

$$\text{F1-Score} = 86.9\%$$

If minority class is less than 20%, accuracy will become misleading

- Use Accuracy when data is balanced
- Use Precision & Recall (and F1) when data is imbalanced
- \* How to check our data is balanced or imbalanced

import pandas as pd

# We need to choose our target column based on our need  
for eg label

class\_counts = df['label'].value\_counts()

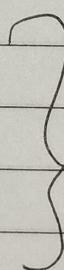
class\_percentage = df['label'].value\_counts(normalize = True) \* 100

print('Class Counts: \n', class\_counts)

print('Class Percentage: \n', class\_percentage)

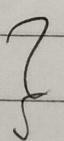
Example Output (Balanced dataset)

Class	Counts :
0	310
1	490



Balanced - both classes have almost equal points

Class	Percentage :
0	51.0
1	49.0



Imbalanced - one class dominates

Class	Counts :
0	990
1	10

Class	Percentage :
0	99.0
1	1.0

- \* Handle the Imbalance in both active learning and -
  - 1) Resample ratio (1:1 bin) based on majority class
    - Oversample (SMOTE)
    - Undersample
  - 2) Use Class Weights
    - Tells our model to give higher weight to the minority class
    - (Works well for tree-based and linear models)
  - 3) Use Algorithms for Imbalance
    - XGBoost
    - LightGBM
    - CatBoost
  - 4) Treat as Anomaly Detection
    - If minority < 1%, treat it as anomaly detection
    - Isolation Forest
    - One-class SVM
    - Autoencoder (deep learning)