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 actually 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

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

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

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

→ 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,

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

High precision → fewer false alarms

e) Recall (sensitivity / True Positive Rate)

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

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

High Recall → fewer missed cases

f) 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

g) Specificity

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

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

4. Quick Example

Suppose:

- TP = 50, TN = 40, FP = 10, FN = 5

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

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

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

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

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

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

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

$$= 2 \times 0.434$$

$$= 0.869$$

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('\n Class Percentage:\n', class_percentage)
```

Example output (Balanced dataset)

Class	Counts:
0	510
1	490

Class	Percentage:
0	51.0
1	49.0

} Balanced — both classes have almost equal points

Class	Counts:
0	990
1	10

Class	Percentage:
0	99.0
1	1.0

} Imbalanced - one class dominates

* Handle the Imbalance

1) Resample
   - 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)