Performance Evaluation

PERFORMANCE EVALUATION MATRIX

Performance Parameters

- Accuracy
- Precision
- Recall
- F1 Score

Dataset

100

1

ML

99%

98.5 - 99.5 %

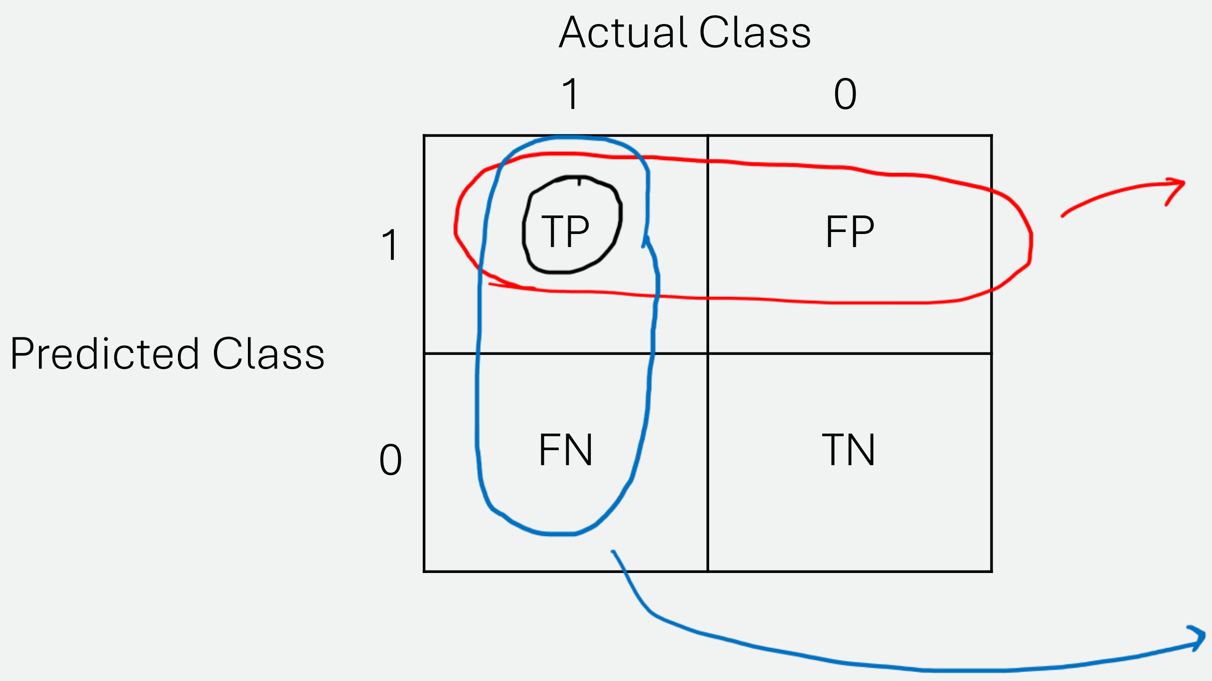
$\hat{y} = 0$

Precision/Recall

		Actual Class	
		1	0
Predicted Class	1		
	0		

		Actual Class	
		1	0
Predicted Class	1	TP	FP
	0	FN	TN

Precision/Recall



Precision = True Pos/#Predicted Pos

Precision = True Pos/True Pos+False Pos

Recall = True Pos/#Actual Pos

Recall = True Pos/True Pos+False Neg

Trade-off between Precision & Recall

Logistic regression: $0 < f_{\vec{w},b}(\vec{x}) < 1$

Predict 1 if $f_{\vec{w},b}(\vec{x}) \geq 0.5$ 0.8 0.2

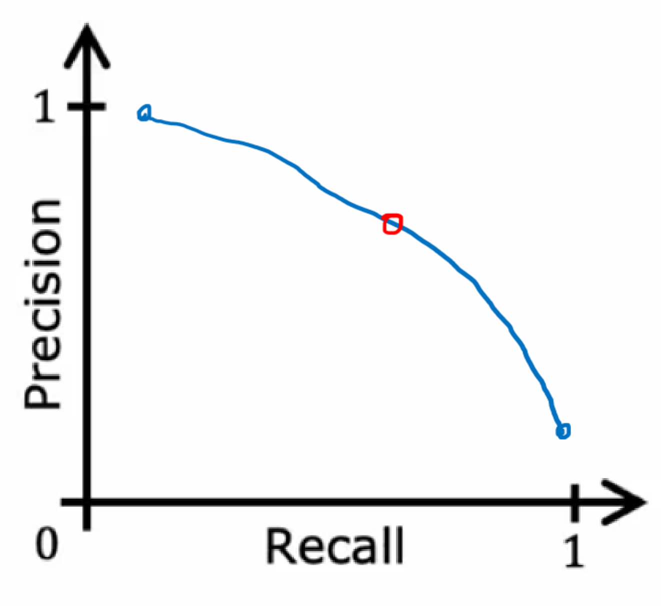
Predict 0 if $f_{\vec{w},b}(\vec{x}) < 0.5$ 0.8 0.2

$$\text{precision} = \frac{\text{true positives}}{\text{total predicted positive}}$$

$$\text{recall} = \frac{\text{true positives}}{\text{total actual positive}}$$

Suppose we want to predict $y = 1$ (rare disease) only if very confident.

Suppose we want to avoid missing too many case of rare disease (when in doubt predict $y = 1$)



F1 Score

- How to compare Precision/Recall numbers?

F1 Score = Harmonic mean of P & R

F1 Score = 2PR/(P+R)

	Precision	Recall	f_1
Logostic Regression	0.4	0.5	0.44
Decision Tree	0.1	0.7	0.175
Support Vector Machine	1	0.02	0.039

Trade-off between Precision & Recall (Dataset)

ID	X1	X2	Actual Class	Logistic Regression (Yhat)	Predicted Class (Threshold >0.5)	Predicted Class (Threshold >0.8)	Predicted Class (Threshold >0.3)
1	1	2	0	0.1	0	0	0
2	1.5	2.5	0	0.2	0	0	0
3	2	3	0	0.55	1	0	1
4	2.5	3.5	0	0.4	0	0	1
5	3	4	0	0.49	0	0	1
6	7	8	1	0.6	1	0	1
7	7.5	8.5	1	0.7	1	0	1
8	8	9	1	0.85	1	1	1
9	8.5	8.5	1	0.2	0	0	0
10	9	9	1	0.95	1	1	1

	Threshold>0.5	Threshold>0.8	Threshold>0.3
Accuracy	0.8	0.6	0.6
Precision	0.8	1	0.57
Recall	0.8	0.4	0.8
F1 Score	0.8	0.57	0.67

Dataset (Diabetes Detection)

Sl No.	Pregnancies	Glucose	Blood Pressure	Skin Thickness	Insulin	BMI	Diabetes Pedigree Function	Age	Outcome
1	6	148	72	35	0	33.6	0.627	50	0
2	1	85	66	29	0	26.6	0.351	31	0
3	8	183	64	0	0	23.3	0.672	32	0
4	1	89	66	23	94	28.1	0.167	21	0
5	0	137	40	35	168	43.1	2.288	33	1
6	5	116	74	0	0	25.6	0.201	30	0
7	3	78	50	32	88	31	0.248	26	0
8	10	115	0	0	0	35.3	0.134	29	0
9	2	197	70	45	543	30.5	0.158	53	0
10	8	125	96	0	0	0	0.232	54	1

Skewed Dataset

- One class training & testing dominates over other

- Solution
  - Oversampling the minority class
    - Duplicate the existing entries of minority class
  - Undersampling the majority class
    - Remove randomly selected entries of majority class
  - SMOTE (Synthetic Minority Over-sampling Technique)
    - Generate Synthetic data

SMOTE (Synthetic Minority Over-sampling Technique)

1. **Select a Minority Class Sample**
2. **Find k Nearest Neighbors**
3. **Generate a Synthetic Sample**
   a) **Select a random neighbor x_{nn} from the k nearest neighbors.**
   b) **Create a new synthetic sample along the line between x and x_{nn}**

$$x_{new} = x + \lambda \times (x_{nn} - x)$$

   c) **where λ is a random number between 0 and 1**
4. **Repeat Until Desired Balance is Achieved**

Dataset

ID	X1	X2	Class
1	1	2	0 (Majority)
2	1.5	2.5	0 (Majority)
3	2	3	0 (Majority)
4	2.5	3.5	0 (Majority)
5	3	4	0 (Majority)
6	7	8	1 (Minority)
7	7.5	8.5	1 (Minority)
8	8	9	1 (Minority)

Data Balancing techniques (comparison)

Method	What it does?	Pros	Cons
Random Oversampling	Duplicates minority class samples	Easy to implement	Can cause overfitting
SMOTE	Generates synthetic samples for the minority class	Prevents overfitting	Can introduce noise
Undersampling	Removes majority class samples	Reduces computation time	Can lose useful data

Lack of Data availability

- Data Augmentation
  - Images (2D transformations, Intensity transformation)
- Data Synthesis
  - GANs

- New Data must resemble with the property of available/expected data