Performance Measurement of Models

Some classification metrics

$$\text{Accuracy} = \frac{\#\text{pts correctly classified}}{\text{Total } \#\text{pts in } D_{test}}$$

Accuracy $\in [0, 1]$.
(bad) (good)

easy to understand measure.

Measured on test set usually.

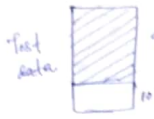
100 pts → 60 +ve → 53 +ve, 7 -ve
100 pts → 40 -ve → 35 +ve, 5 -ve.

error = 12. correctly = 88.

$$\text{Accuracy} = \frac{88}{100} = 0.88.$$

Problems with accuracy

① Doesn't work well with unbalanced data.



dumb model any $x_q \longrightarrow$ -ve.

Accuracy = 0.9.

So, never use accuracy as performance measure for imbalanced data.

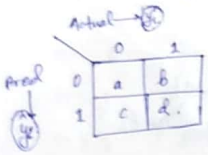
	x	y	Model 1	Model 2	$\hat{y}_1$	$\hat{y}_2$
	x_1	1	0.9	0.6	1	1
	x_2	1	0.8	0.65	1	1
	x_3	0	0.1	0.45	0	0
	x_4	0	0.16	0.49	0	0

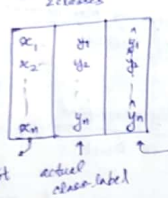
* We can clearly say that M_1 is better than M_2 by looking at the probability values.
* But since predicted class labels are same, we get same accuracy.

* Accuracy — can't use probability scores.

Confusion Matrix (doesn't process probability scores).

Consider a binary classification task ($C_0, 1$).

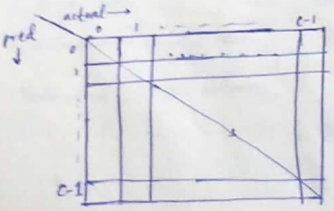




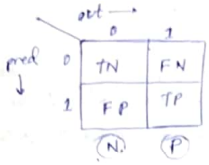
a: #pts s.t. $y_i = 0$ & $\hat{y}_i = 0$.
b: #pts s.t. $y_i = 1$ & $\hat{y}_i = 0$.

This can be also extended to multiclass classifications.

Consider c-class classification.



for a sensible model, principal diagonal elements ↑.
off diagonal elements - small ↓.



$\hat{y}$ — what is the predicted label.
y — is the prediction correct.

N: # negatives.
P: # positives.
n = N+P.

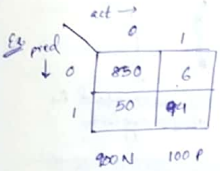
TPR (True Positive Rate) = $\frac{TP}{P}$.

TNR (True Negative Rate) = $\frac{TN}{N}$.

FPR (False Positive Rate) = $\frac{FP}{N}$.

FNR (False Negative Rate) = $\frac{FN}{P}$.

Ex:



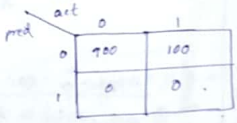
TPR = 94%.
TNR = $\frac{850}{900}$
FPR = $\frac{50}{900}$.
FNR = 6%.

So, we have high TPR & TNR, which is desired.

Test set:- 900 -ve, 100 +ve } imbalanced.

Let us now assume a dumb model which gives o/p label always -ve.

Ideally, TPR ↑ FPR ↓. Then,
FNR ↑ FNR ↓



✗ TPR = 0%
✓ FPR = 0%
✓ TNR = 100%
✗ FNR = 100%

So, we can detect dumb models by using these 4 metrics together.

* Now, which of the 4 metrics is more important?

→ domain specific.

Ex → diagnosing cancer → We must have very low FNR as we can't afford to say an actual patient that he doesn't have cancer.

* little high FPR is acceptable as even if we say a non-patient that he may have cancer, later through powerful tests that can be identified.

③ Precision, Recall and F1-Score (Information Retrieval).

Pred ↓ \ Act →	0	1
0	TN	FN
1	FP	TP
	N	P

Precision (Pr) = $\frac{TP}{TP+FP}$ → of all pts the model declared to be (+)ve, what %age of them are actually (+)ve.

Recall (Re) = TPR = $\frac{TP}{P}$ → of all the ~~multiple~~ pts which actually belong to cls '1', (+)ve, how many are predicted to be (+)ve.

We want both Pr & Re to be high.
[0,1] [0,1].

F1 score combines them both.

$$\text{F1-score} = 2 * \left(\frac{\text{pre} * \text{Re}}{\text{Pr} + \text{Re}}\right)$$

↓ used as metric in many Kaggle competitions.

Harmonic mean of Pr & Re.

$$\frac{1}{F1} = \frac{1}{2}\left(\frac{1}{Pr} + \frac{1}{Re}\right)$$

* Pr & Re — more interpretable. — can understand in simple english.

④ Reciever Operating Characteristic Curve (ROC) & AUC (Area Under Curve).

— for binary classification only.

Assume model outputs a score which can be interpreted similar to probability of class '1' score.

x	y	ŷ
x_1	1	0.95
x_2	1	0.92
x_3	0	0.80
x_4	1	0.76
x_5	1	0.71

Steps for ROC

① sort entries in decreasing order of ŷ

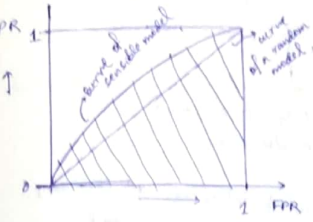
② Thresholding (z).
Serially from top to bottom, take each value of ŷ as z.

Eg:

x	y	ŷ	$\hat{y}_{z=0.93}$	$\hat{y}_{z=0.72}$
x_1	1	0.95	1	1
x_2	1	0.92	0	1
x_3	0	0.80	0	0
x_4	1	0.76	0	0
x_5	1	0.71	0	0
			FPR_1	FPR_2
			FPR1	FPR2.

Then, if $\hat{y} \geq z$: label = 1
else label = 0.

Find TPR & FPR for each such threshold.



Area under curve (AUC) — 0 to 1 (terrible) (v. good).

for a sensible model, this will how the curve will look like.

AUC properties

① Imbalanced data → AUC can be high for dumb-model / simple model.

② AUC is not dependent on the ŷ scores. Only depends on ordering.

Eg →

x_1	1	0.95	0.2
x_2	1	0.92	0.1
x_3	0	0.88	0.08
x_4	1	0.76	0.07
x_5	1	0.71	0.06

Both have completely different values. But the sorted order is same.

③ AUC (random model) = 0.5

If AUC(M) ∈ [0, 0.5], → simply compliment the o/p of the model. (swapping).

⑤ Log-loss (uses probability scores).

consider binary classification case.

x	y	ŷ = p	loss
x_1	1	0.9	→ 0.04457
x_2	1	0.6	→ 0.22
x_3	0	0.1	→ 0.04457
x_4	0	0.41	→ 0.22.

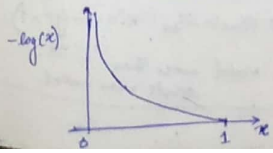
$$\text{Log-loss} = \frac{-1}{n}\sum_{i=1}^{n}\left\{\log(p_i) * y_i\right\} + (1-y_i) * \log(1-p_i)$$

[0, ∞).

Log-loss :- neg. avg. log (prob. correct class label).

very powerful because unlike all previous metrics, it makes use of actual class probabilities.

So, more closer the correct class probability as to 1, lesser is the loss.



for Multiclass classification,

$$\text{log-loss} = \frac{-1}{n}\sum_{i=1}^{n}\sum_{j=1}^{c} y_{ij}\log(p_{ij})$$

p_{ij} → prob. of $x_i \in$ class j.

y_{ij} → 1 if $x_i \in$ class j; 0 otherwise

* One key disadvantage of log-loss is its hard to interpret, as it can go as high as ∞, No upper-bounds. Can't determine how bad.

Some regression metrics

$y \in R$ in regression

Test set i.i.d ton: $x_i, y_i, \hat{y}_i$ ← model o/p

$e_i = (y_i - \hat{y}_i)$.

* Just like in classifications, a simple model would be one that returns the label of the majority class for all i/ps.
* Similarly, for regression, a simple model would be one that returns the mean of y's.

Error for the simple model = $\sum_{i=1}^{n}(y_i - \bar{y})^2$ → also called SS_{tot} or sum of squares (using simple mean model).

residue = $e_i = (y_i - \hat{y}_i)$

$SS_{res} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

$$R^2 = \left(1 - \frac{SS_{res}}{SS_{total}}\right)$$

with R^2 search → coefficient of determination.

Case 1: $SS_{res} = 0 \Rightarrow e_i = 0$ → $R^2 = 1$ best value.

Case 2: $SS_{res} < SS_{tot}$ ⇒ $R^2 \in [0, 1]$

Case 3: $SS_{res} = SS_{tot}$ → $R^2 = 0$ ← model same as simple mean.

Case 4: $SS_{res} > SS_{tot}$ $R^2 = 1 - (\text{gr} > 1)$ ← -ve

Model worse than simple mean model

Median Absolute deviation of errors

R^2 — relies of mean, hence not very robust to outliers.

for each x_i —we have→ $y_i, \hat{y}_i, e_i$

e_i is random variable.

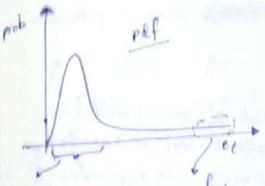
median(e_i) = central value of errors. — defined for a set of pts.

MAD(e_i) = median($|e_i - \text{median}(e_i)|$) — defined for a set of pts.

mean → std dev → if no outliers

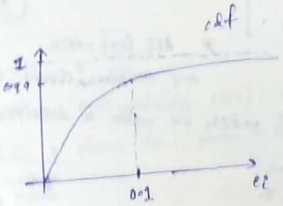
median → MAD → if outliers.

Distribution of errors



most e_i are small

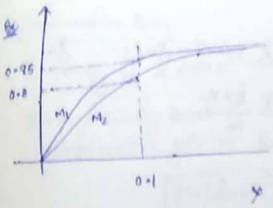
very few e_i are larger



99% of e_i (errors) are < 0.1

1% of errors > 0.1.

So, we can use these standard statistical measures to get an idea of distribution of errors.



M_2 cdf below M_1.

M_1: 95% of errors are below 0.1

M_2: 80% of errors are below 0.1

So, M_1 better model than M_2.

* Ideally, we want as many errors close to 0 as possible.