It's fucking good morning! Yes...

Metric depends on business problem,
not on the model we use

Add your own case studies
↳ Reminders already know about AAIC case studies

→ If there's a MAJOR extension to AAIC, it's okay to have it on resume.

→ One unique, detailed case study is better than 3-4 case studies from AAIC.

→ Random Search obviously better than grid search.

→ There's no rule of thumb for stuff in ML/DS. It's new and things change based on the problem we solve.

→ Knowing what metric to choose is a skill, no textbook has this stuff.

→ AAIC case studies have an intro video explaining what's the business case and how to adapt/morph that as as a ML problem.

→ Why fuck with tools when you can write custom code not limited by the software. [Talking about Tableau]

→ For serious ML/DS, it's a shame to fuck with Tableau.

Key Metrics:

1. Accuracy.
2. Confusion Matrix: TPR, FPR, TNR, FNR
3. AUC-ROC
4. Precision, Recall, F1
5. Log-loss
6. Macro averaged & Micro averaged metrics
7. Cross-Entropy.
8. Jaccard Similarity.
9. R2
10. Root Mean Squared Error
11. Mean Absolute error.
12. Percentile Errors.
13. Logarithmic-metrics for regression
14. Explained Variance.

Slack

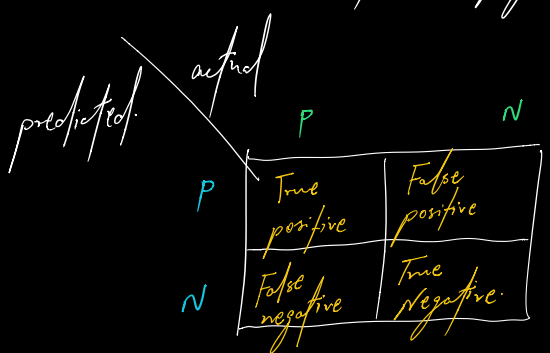
↳ These are the metrics discussed in the course and this session.

Simple man. → "You're not pregnant" → True Negative

"You are pregnant" → False Positive !!

Pregnant Woman → "You're not pregnant" → False negative

"You are pregnant" → True positive.



Confusion Matrix.

Disadvantages :→ Does not take class probability into account.

The model give probability, not the metric.

for ad-click

$$P(y_i = 1 \mid x_i) = 0.\overline{}$$

This is super useful for knowing if the user will click on the ad.

→ Even for carrier prediction, prob is important

Conf. matrix is just a break down of what the fuck our model classified where. It's not supposed to give us probability.

→ CM is useful in imbalanced dataset especially.

if 99% data is "NO CANCER" and my model learns to say "NO CANCER" to all the time get 99% accuracy, that's a problem.

This is where CM will break down what the fuck is going on behind "99%" accuracy.

→ CM just gives a matrix, not a single score. So, it's hard to compare two models easily. [it is possible to compare, but it's not obvious]

→ shit gets crazy when there are more than two classes. try using CM with that.

→ Under/Overfitting can't be detected with this.

→ Researchers publish stuff left and right without

worrying about feasibility of the work. You always find stuff connected to work on some thing else. But it's usually shit.

→ with can do one vs. rest but we have to deal with a dozen metrics that way.

→ If we google search "Multiclass Conf. Matrix", we WILL find results. People WILL have figured out some JUGAAD to get their paper published.

→ Advantages of Conf. matrix.

→ for interpretability. when working with imbalanced data.

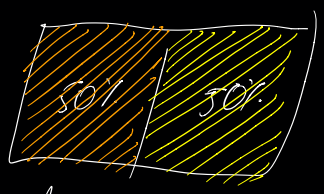
⇒ Accuracy. → its a single figure, easy to compare just one number.

→ Imbalanced data doesn't work with it.

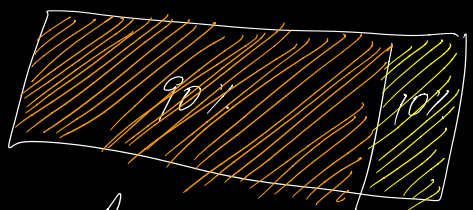
→ In cancer prediction, the cost of making a mistake is high. That's where accuracy fucks up.

→ No one in medical field uses accuracy.

Let's consider a case where data is upsampled to bring it to 50%. →



Acc → 99%



Acc → ?? (not 99%)

Because the distribution is different.

Upsampling / Downsampling does not tie-in well with each other.

Maybe with imbalanced data we can have accuracy for each class separately.

Recall

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

Out of all positive points in dataset, how much did the model get right?

Precision

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

→ Out of all points that model predicted positive, how much are actually positive?

These two are useful in Information Retrieval. Search Engines use something more advanced than these.

If precision ↑ & recall ↓

That means very few simple men are being predicted as pregnant. ⇒ That's good!!

&

Lots of pregnant women are being predicted as "NOT pregnant" ⇒ Not good!!

Model is predicting a lot of thing as negative.

Precision & Recall don't look at True Negatives.

↑
Disadvantage

If one class is more important than another,

that's when we use precision and recall.

↳ depends on business case.

There are extensions for dealing with multiple classes but it's more effective for focusing on a single class

F1 score is the harmonic mean of Pr & Re.

$$F1 = \frac{1}{P} + \frac{1}{R}$$

$$F1 = \frac{R + P}{P \cdot R}$$

Q. why take a harmonic mean? why not geometric? or arithemetic

#Important interview question.

One rule → Having one number is better than

many. we want to care about less stuff.

arithemetic mean values all inputs equally.

So $M = \frac{A + B}{2}$

↳ B can be low.

↳ A can be high.

And the result would still be same.

→ One input can overshadow the other one and cause the result to go up.

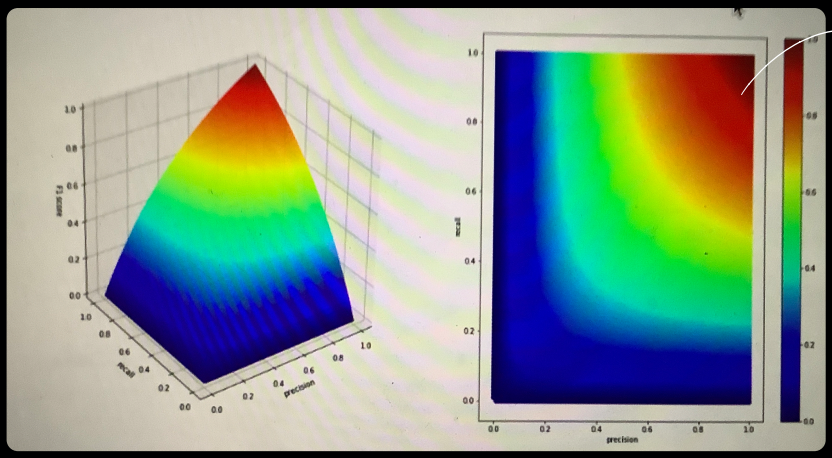
We want our result to be $\min(A, B)$

↓

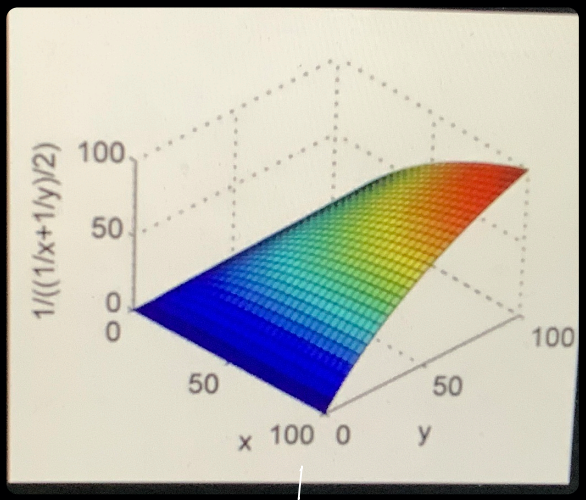
but this is not differentiable.

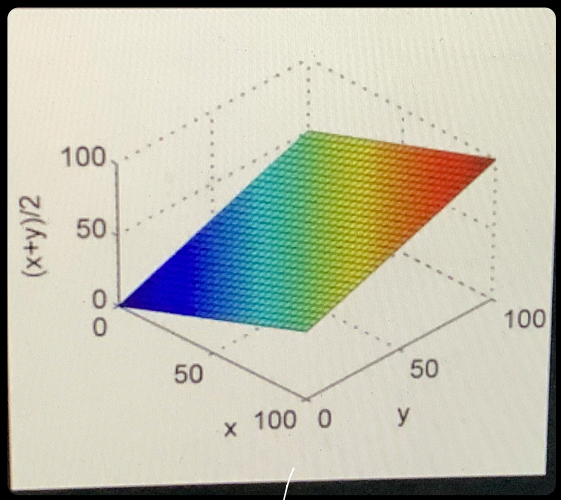
So we use harmonic mean as our function

↳ F1 score



if any one of them is low, it fucks up the result, that's what we want.





$$M = \frac{1}{\frac{1}{x} + \frac{1}{y}}$$

$$M = \frac{x + y}{2}$$

If we need, if the situation calls for it, we can change the metric's formula to whatever the fuck I want. Given that we can justify it.

→ Custom metrics are used a lot in Kaggle.

→ Better stick to std. metrics because it becomes hard to explain to non-tech folks.

Logloss

→ $-\frac{1}{N}\sum_{i=1}^{N} y_i \log p_i + (1-y_i)\log(1-p_i)$

p_i: The probability predicted by model.

y_i: the actual class label.

So, now we care not only about accuracy, we also want the probability to be right.

→ So the confusion matrix can be perfect but logloss will still roast the model for getting the probabilities wrong.

Logloss penalizes even minor deviations. So if

forces the model to confident + concert.

No fucks given about material as long as you can derive stuff.

Tools are not important, mindset is.

You know C, C++, Java, Python at a basic level? Fuck you.
We need someone with detailed knowledge of one language, that person can always pickup syntax of something else.

Any real company won't given a fuck about what language you use.

Where to avoid logloss? Where is it a bad idea?

Note → it can't be used by default if model doesn't give probability.

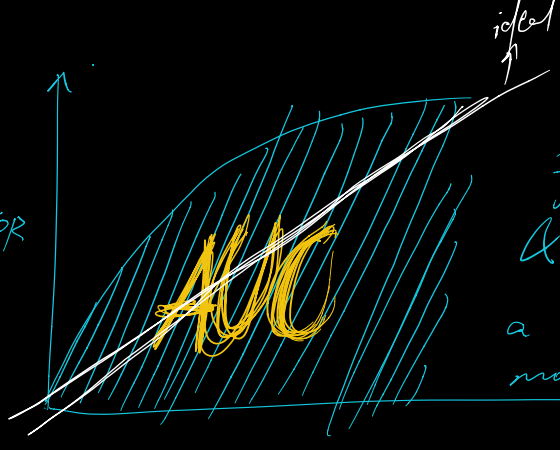
→ Logloss is a black box, no interpretability.
Hard to explain what the fuck is right or wrong with model.

→ We first need a baseline, use a random model to get a baseline score (the worst case, some reference)

→ Models directly optimize logloss, they don't do so for F1, precision, recall.

⇒ Logloss is an overkill for cases where we don't give a fuck about probability.

AUC
Area Under Curve
↳ ROC curve.



So if AUC = 0.9
If you have one (+) & one (-) pt. There's a 90% chance the model will classify correctly.

TLDR → Nothing works for all cases, it all depends.

for LIVE SESSION.

↳ only "correctly."

→ not "positive points correctly"

AUC is scale invariant. as long as the function is monotonic

So give it $\begin{bmatrix} 0.2 \\ 0.3 \\ 0.7 \\ 0.8 \end{bmatrix}$ or $\begin{bmatrix} 0.4 \\ 0.6 \\ 1.4 \\ 1.6 \end{bmatrix}$

↓ datapoints. ↓ datapoints.

The AUC will be same for both.

Learn a concept ⇒ Make notes ⇒ Explain it to someone.

FEYNMANN TECHNIQUE.