Metrics and Evaluation

Fatemeh Mansoori

Some images in these slides are selected from the slides of the Machine Learning course by Sharifi zarchi at Sharif University

Confusion Matrix

		Predicted		
		Negative (0)	Positive (1)	
Actual	Negative (0)	**True Negative** **TN**	**False Positive** **FP** (Type I error)	**Specificity** $= \frac{TN}{TN + FP}$
	Positive (1)	**False Negative** **FN** (Type II error)	**True Positive** **TP**	**Recall**, **Sensitivity**, True positive rate (**TPR**) $= \frac{TP}{TP + FN}$
		Accuracy $= \frac{TP + TN}{TP + TN + FP + FN}$	**Precision**, Positive predictive value (**PPV**) $= \frac{TP}{TP + FP}$	**F1-score** $= 2 \times \frac{Recall \times Precision}{Recall + Precision}$

The confusion matrix is a useful tool for analyzing how well your classifier can recognize tuples of different classes

TP and *TN* tell us when the classifier is getting things right, while *FP* and *FN* tell us when the classifier is getting things wrong

Accuracy in classification problems

- **Accuracy** is one of the simplest and most commonly used performance metrics.
- It is defined as the ratio of correctly predicted instances to the total instances:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Samples}}$$

- However, accuracy alone can be misleading, especially with imbalanced datasets.

Class imbalance

- **class imbalance problem**
  - main class of interest is rare.
  - data set distribution reflects a significant majority of the negative class and a minority positive class
  - E.g. in fraud detection applications, the class of interest (or positive class) is *“fraud,”* which occursmuch less frequently than the negative *“nonfraudulant”* class

Class imbalance problem

- Imagine a dataset with 1000 patients:
  - Only 10 have cancer (**positive class**).
  - 990 do not have cancer (**negative class**).
- A classifier predicts that no one has cancer (predicts all as negative).
- What will be the accuracy of this model?

Class imbalance example

Look at this table for our model which predict negative all the time:

	Predicted Negative	**Predicted Positive**
Actual Negative	990 (TN)	0 (FP)
Actual Positive	10 (FN)	0 (TP)

$$\text{Accuracy} = \frac{990 + 0}{1000} = 99\%$$

High accuracy, but the model fails to detect any actual cases of cancer!

Why accuracy can be misleading

- In highly imbalanced datasets (e.g., cancer detection), the **minority class** (positive cases) is often underrepresented.
- A model that always predicts the majority class can still have high accuracy, but poor real-world performance.
- In the cancer detection example, 99% accuracy sounds good, but the model doesn't detect any actual cancer cases.
- We need better metrics to evaluate model performance.

Performance Metrics

- **Scenario:**
  - An alarm system can either ring or not ring when a thief is present.
  - Let's define the outcomes:
    - **True Positive (TP)**: Alarm rings (correctly) when a thief is present.
    - **True Negative (TN)**: Alarm does not ring (correctly) when no thief is present.
    - **False Positive (FP)**: Alarm rings (incorrectly) when no thief is present (a false alarm).
    - **False Negative (FN)**: Alarm does not ring (incorrectly) when a thief is present (a missed alarm).

	Thief Present	**No Thief Present**
Alarm Rings	TP	FP
Alarm Does Not Ring	FN	TN

Performance Metrics

- **Metrics:**
  - **Sensitivity (Recall)**:

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

Indicates the ability of the alarm system to correctly identify a thief. It is the proportion of actual positives (thief present) that are correctly identified.

  - **Specificity**:

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

Measures the ability of the alarm system to correctly identify when no thief is present. It is the proportion of actual negatives that are correctly identified.

  - **Precision**:

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

Indicates the accuracy of the alarm when it rings. It is the proportion of times the alarm rang and a thief was indeed present out of all the times the alarm was activated.

A combined measure : F1

- Combined measure: **F1 measure**
  - allows us to trade off precision and recall
  - harmonic mean of P and R

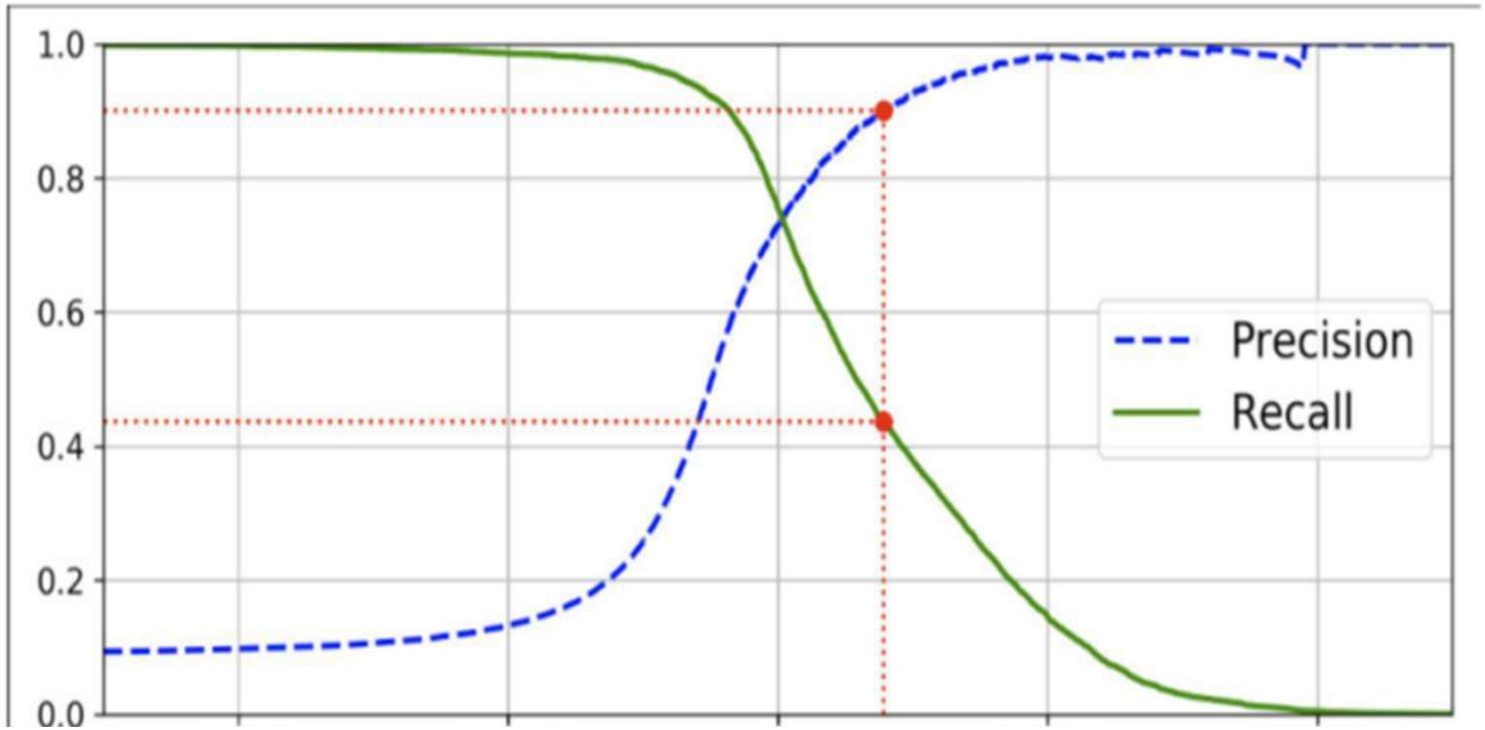
$$F = \frac{1}{\frac{1}{2P} + \frac{1}{2R}} = \frac{2PR}{P+R}$$

- Harmonic mean of P and R:

$$\frac{1}{F} = \frac{1}{2}\left(\frac{1}{P} + \frac{1}{R}\right)$$

Precision Recall Trade-off

F measure is the *harmonic mean* of precision and recall

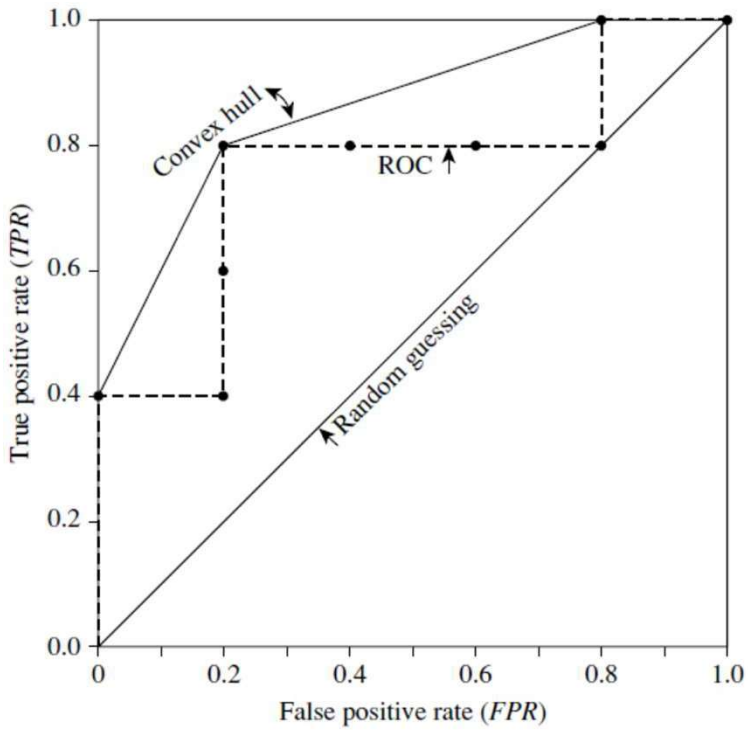


ROC curve

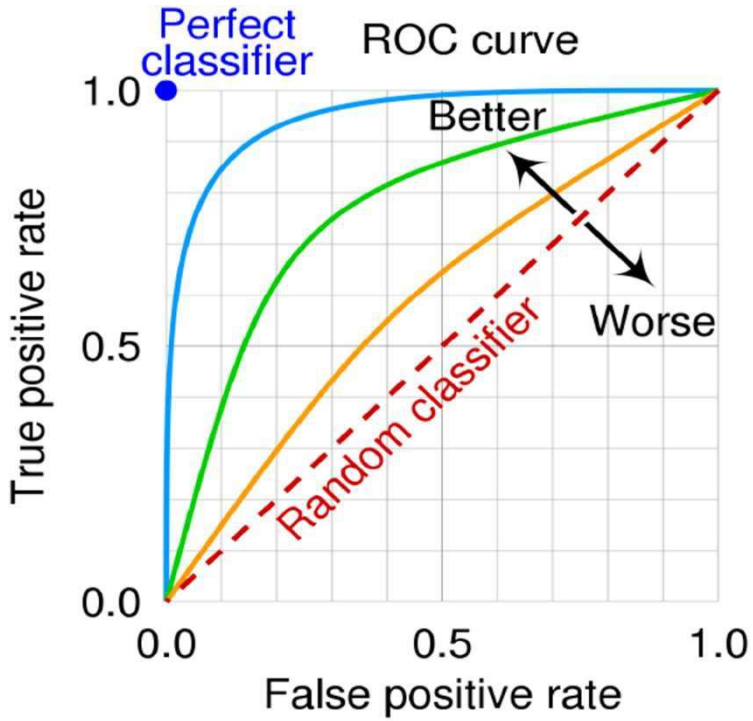
- ROC curve allows us to visualize the trade-off between TPR and FPR
- *TPR* = TP/P and FPR = FP/N
- To plot an ROC curve for a given classification model, *M*, the model must be able to return a probability of the predicted class for each test tuple

ROC Curve

Tuple #	Class	Prob.	TP	FP	TN	FN	TPR	FPR
1	*P*	0.90	1	0	5	4	0.2	0
2	*P*	0.80	2	0	5	3	0.4	0
3	*N*	0.70	2	1	4	3	0.4	0.2
4	*P*	0.60	3	1	4	2	0.6	0.2
5	*P*	0.55	4	1	4	1	0.8	0.2
6	*N*	0.54	4	2	3	1	0.8	0.4
7	*N*	0.53	4	3	2	1	0.8	0.6
8	*N*	0.51	4	4	1	1	0.8	0.8
9	*P*	0.50	5	4	0	1	1.0	0.8
10	*N*	0.40	5	5	0	0	1.0	1.0



Precision Recall Trade-off

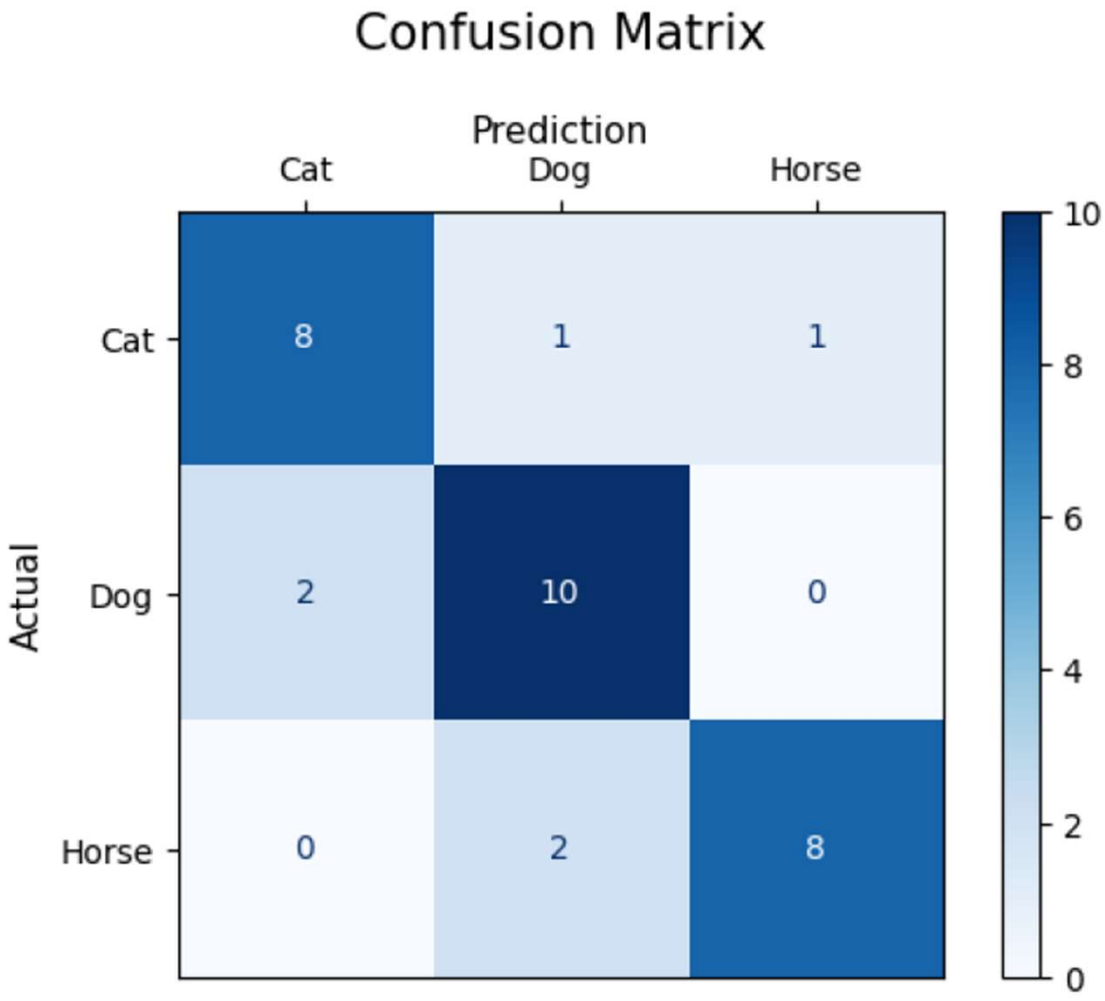


Confusion matrix for more than two class

- Let's consider an image classification task where we classify images into three categories: **Cat**, **Dog**, and **Horse**.
- After training the model, we evaluate its predictions against the actual labels.

	Predicted Cat	**Predicted Dog**	**Predicted Horse**
Actual Cat	True Positive (TP)	False Negative (FN)	False Negative (FN)
Actual Dog	False Negative (FN)	True Positive (TP)	False Negative (FN)
Actual Horse	False Negative (FN)	False Negative (FN)	True Positive (TP)

- Here is an example confusion matrix for a model that classifies images of cats, dogs, and horses:
- We can see that the model classified 8 images of cats correctly, but it classified 1 cat as a dog and 1 cat as a horse (False Negatives).
- Similarly, it made 2 mistakes when predicting dogs and horses.



Evaluation metris for more than one class

- **Definition:** For our **confusion matrix C**, each **element C_{ij}** denotes the number of samples actually in class i that were put in class j by our classifier.
  - Now we could rewrite our performance metrics with confusion matrix view

---

- Recall: Fraction of the samples in class i classified correctly:

$$\frac{C_{ii}}{\sum_j C_{ij}}$$

- Precision: Fraction of the samples assigned class i that are actually about class i:

$$\frac{C_{ii}}{\sum_j C_{ji}}$$

- Accuracy: Fraction of the samples classified correctly:

$$\frac{\sum_i C_{ii}}{\sum_j \sum_i C_{ij}}$$

Averaging : micro vs macro

- We now have an evaluation measure (F1) for one class.
- But we also want a single number that shows aggregate performance over all classes

Micro vs Macro Averaging

- If we have more than one class, how do we combine multiple performance measures into one quantity?
- Macroaveraging: Compute performance for each class, then average
  - Compute F1 for each of the C classes
  - Average these C numbers
- Microaveraging: Collect decisions for all classes, aggregate them and then compute measure.
  - Compute TP, FP, FN for each of the C classes.
  - Sum these C numbers(e.g, all TP to get aggregate TP)
  - Compute F1 for aggregate TP, FP, FN

Micro vs Macro Averaging Example

Class 1

	Truth: yes	Truth: no
Classifier: yes	10	10
Classifier: no	10	970

Class 2

	Truth: yes	Truth: no
Classifier: yes	90	10
Classifier: no	10	890

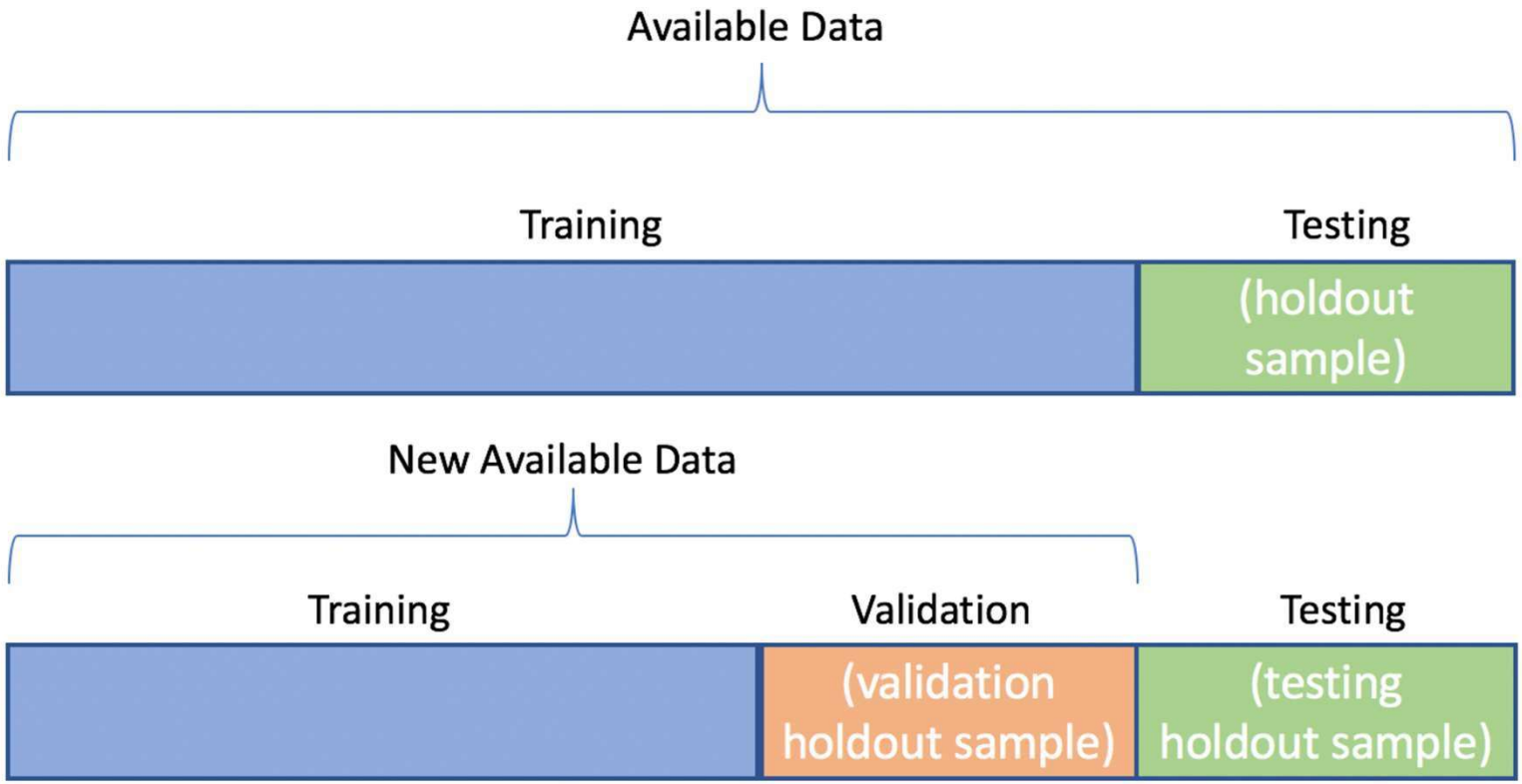
Micro Ave. Table

	Truth: yes	Truth: no
Classifier: yes	100	20
Classifier: no	20	1860

- Macroaveraged precision: $(0.5 + 0.9)/2 = 0.7$
- Microaveraged precision: $100/120 = 0.83$
- Microaveraged score is dominated by score on common classes

Validation

Validation


Available Data
Training
Testing
(holdout sample)
New Available Data
Training
Validation
Testing
(validation holdout sample)
(testing holdout sample)

Type of validation

- Holdout Validation
- K-Fold Cross Validation

Holdout Validation

Pros

- Fully independent of data
- Lower computational costs

Cons

- higher variance

K-Fold Cross Validation

Cross Validation

- **Purpose**: Technique for evaluating how well a model generalizes to unseen data.
- **How It Works**: Split data into k folds; train on $k-1$ folds and validate on the remaining fold.
- **Repeat Process**: Repeat k times, rotating the test fold each time. Average of all scores is the final score of the model.
- Cross-validation reduces overfitting and provides a more reliable estimation of model performance.
- Note that the model must be **retrained** at each iteration to avoid reusing a model that has already seen the test data, ensuring unbiased evaluation.

K-Fold Cross Validation

Cross-Validation for Evaluating Model Performance

- Metrics like accuracy, precision, recall, and F1 score are assessed across different folds.
- Averaging these scores gives a reliable estimate of performance and stability.
- Ensures the model is effective before final testing and use on the test dataset.
- For example, high variance in cross-validation metrics means the model's performance is inconsistent, likely overfitting to specific data subsets.