Machine Learning (CE 40717)

Fall 2025

Ali Sharifi-Zarchi

CE Department
Sharif University of Technology

October 28, 2025

Evaluation Context

- Model evaluation quantifies predictive performance on unseen data.
- Metrics are derived from the confusion matrix:

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

- TP, TN, FP, FN provide the foundation for classification performance indices.

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:

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

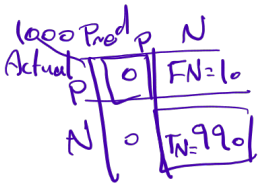
$\frac{990}{1000} = .99$

Error Rate: complement of accuracy

$$\text{Error Rate} = \frac{FP + FN}{TP + TN + FP + FN} = 1 - \text{Accuracy}$$

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

Example: cancer detection 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?

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

Example: cancer detection problem cont.

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).

Sensitivity = تعداد دفعات درست آژیر خطر است / تعداد دفعات درست آمده = TP / (TP + FN)

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

Photo adapted from https://stock.adobe.com/search?k=thief+cartoon

Performance metrics cont.

Sensitivity vs. Specificity

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

Recall = $$\text{Sensitivity} = \frac{TP}{TP + FN}$$ Precision vs. Recall

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.

Performance metrics cont.

Pr / Ac	P	N
P	0	10
N	0	990

Accuracy = 99%
Specificity = 100%
Sensitivity = 0%
Recall = 0%
Precision: Undefined $\frac{0}{0}$

Actual Class \ Predicted Class	Positive	Negative	
Positive	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP+FN)}$ = Recall
Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
	Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

Figure adapted from https://medium.com

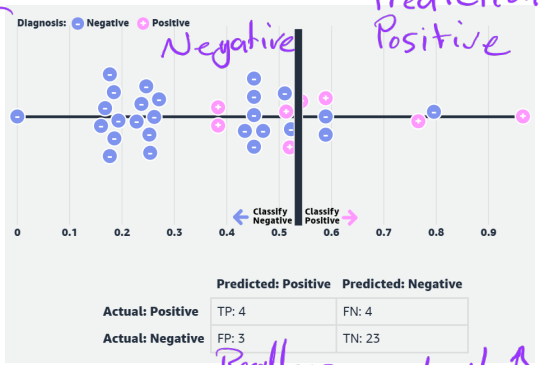
A combined measure: F1

- Combined measure: **F1 measure**
  - allows us to trade off precision and recall
  - Provides a single measure balancing precision and recall.
  - Useful in imbalanced datasets or when both types of error are significant.
  - Trade-off: Increasing recall typically decreases precision and vice versa.
  - harmonic mean of P and R:

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

Precision/recall/F1

- This website could give you a perfect intuition about precision recall trade-off



- Link: https://mlu-explain.github.io/precision-recall/

Why harmonic mean?

- Why don't we use a different mean of P and R as a measure?
  - e.g., the arithmetic mean
- The simple (arithmetic) mean is 50% for "return true for every thing", which is too high.
- Desideratum: Punch really bad performance either on precision or recall
  - Taking the minimum achieves this.
  - F (harmonic mean) is a kind of **smooth minimum**.

F1 and other averages

- Harmonic mean is a conservative average. We can view the harmonic mean as a kind of soft minimum

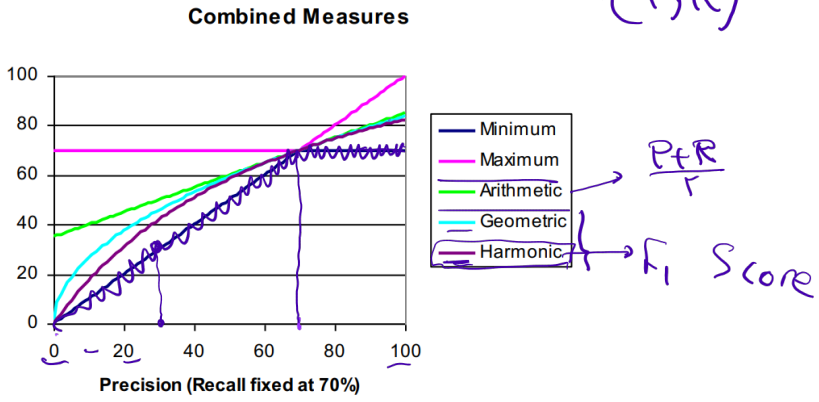


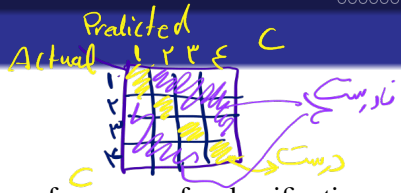
Figure adapted from M.Soleymani ML slides, Sharif University of Technology

Comparative Overview

Metric	Formula	Focus	Typical Use-Case
Accuracy	$\frac{TP+TN}{TP+FP+FN+TN}$	Overall correctness	Balanced classes
Precision	$\frac{TP}{TP+FP}$	False positive control	Spam detection, IR
Recall	$\frac{TP}{TP+FN}$	False negative control	Medical diagnosis, anomaly detection
F1-Score	$2\frac{PR}{P+R}$	Balance	Imbalanced or skewed datasets

- Metric choice should reflect operational objectives and misclassification costs.
- Always interpret metrics in conjunction with the confusion matrix.

Confusion matrix



- The **confusion matrix** is a **table** used to evaluate the performance of a classification model.
- It compares the **actual values (true labels)** with the **predicted values** from the model.
- Each **row** of the matrix represents the **actual class**, while each **column** represents the **predicted class**.
- It helps us understand not just how often the model is correct, but also **where it makes mistakes**.

Confusion matrix cont.

- 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)

Confusion matrix cont.

- 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.

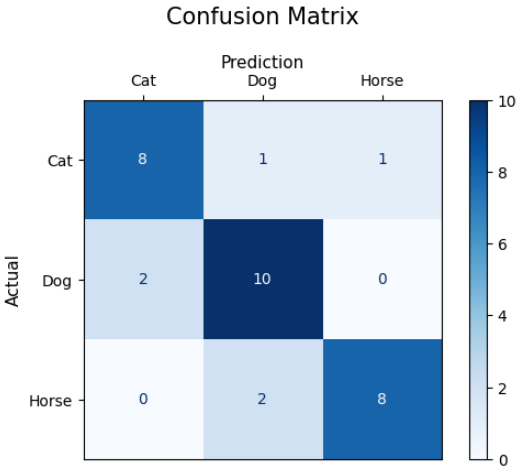


Figure 1: *

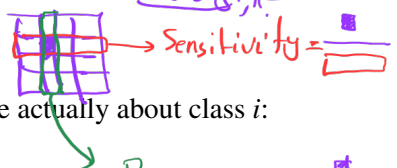
Per class evaluation measures

- **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

---

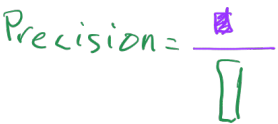
- Recall: Fraction of the samples in class i classified correctly:

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



- 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: macro vs. micro

- 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

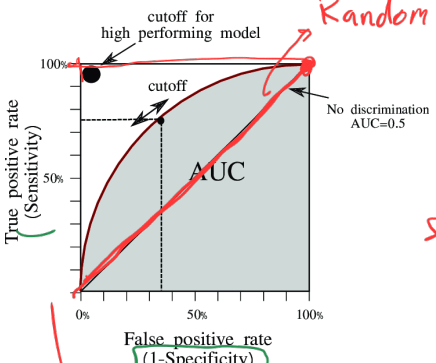
Micro Ave. Table

	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

AUC-ROC

- Area Under the Receiver Operating Characteristic Curve
  - ROC (Receiver Operating Characteristic) is a graphical representation of the performance of a binary classification model.
  - It plots the true positive rate (TPR) against the false positive rate (FPR) at different classification thresholds.



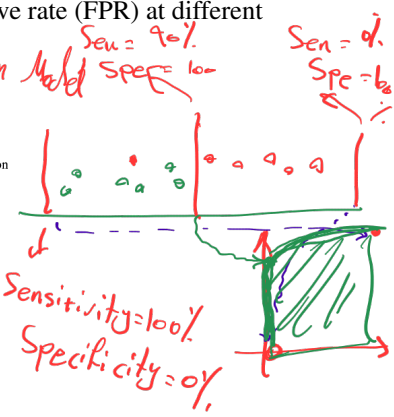


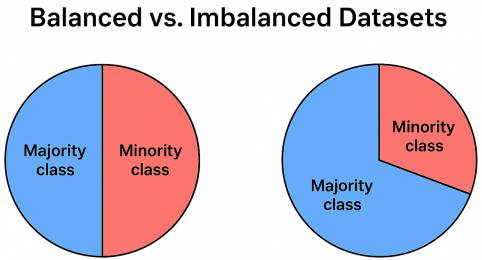
Figure adapted from https://www.researchgate.net/figure/ROC-curves-and-area-under-curve-AUC_fig2_351506473

AUC-ROC cont.

- A high AUC score indicates that the model has good discrimination ability, i.e., it can effectively differentiate between positive and negative instances at different classification thresholds.
- Conversely, a lower AUC-ROC score suggests that the model struggles to differentiate between the two classes.
- AUC ranges from 0 to 1, with 0.5 indicating random guessing and 1 indicating a perfect classifier.

The Problem: Imbalanced Data

- Real-world datasets often contain classes with unequal representation.
- **Examples:**
  - Fraud detection: 0.1% Fraud, 99.9% Non-Fraud
  - Medical diagnosis: 2% Disease, 98% Healthy
- **Issue:** High accuracy may hide poor performance on the minority class.



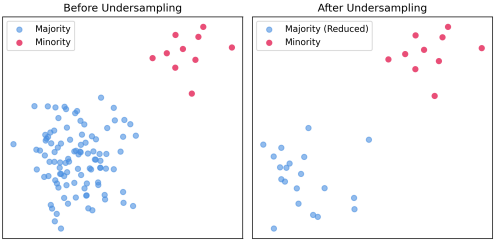
Solution 1: Resampling Techniques

Undersampling

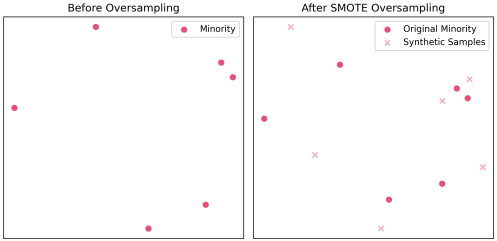
- Remove samples from majority class.
- + Reduces training time.
- – Risk of losing information.

Oversampling (e.g., SMOTE)

- Generate synthetic minority samples.
- + Preserves information.
- – May cause overfitting.



Majority class reduced.



New minority points added via interpolation.

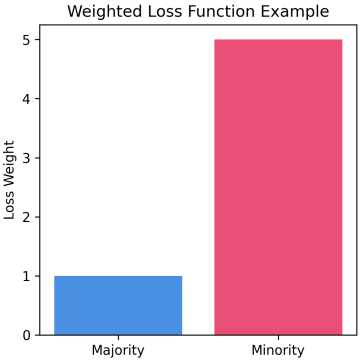
Solution 2: Algorithmic Approaches

Weighted Loss Function

- Assign a higher penalty to errors on the minority class.
- Encourages the model to focus on rare but important cases.

$$J(w) = -\sum_i w_{y^{(i)}} y^{(i)} \log(\hat{y}^{(i)})$$

where $w_{y^{(i)}}$ is larger for the minority class.



Hybrid Approaches

Combining Sampling Strategies

- Integrates both oversampling and undersampling to achieve better class balance.

Typical Pipeline:

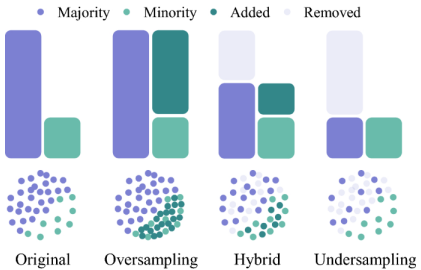
1. Apply **SMOTE** to synthetically augment minority samples.
2. Randomly undersample the majority class to reduce imbalance.
3. Train the model on the newly balanced dataset.

- Provides a practical trade-off between **bias** and **variance**.

Comparison of Methods

Summary

- **Sampling** — modifies data distribution
  (+ Simple, improves balance)
  (– Risk of over/underfitting)
- **Algorithmic** — adjusts model focus
  (+ No data change, interpretable)
  (– Needs careful weight tuning)
- **Hybrid** — combines both strategies
  (+ Balanced, strong results)
  (– More complex, slower training)



Visual summary of trade-offs

Model Selection via 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.

K-Fold Cross Validation

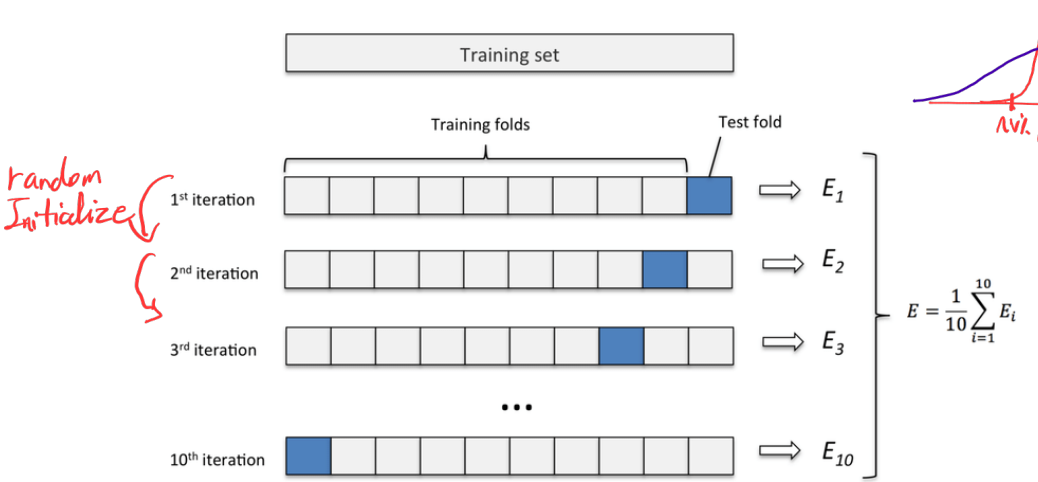


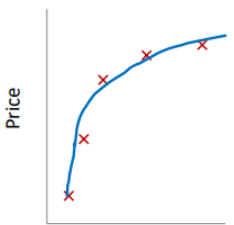
Figure adapted from Introduction to Support Vector Machines and Kernel Methods, J.M. Ashfaque.

Leave-One-Out Cross-Validation (LOOCV)

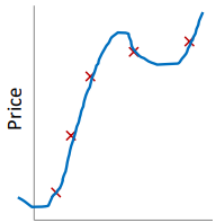
- **Leave-One-Out Cross-Validation (LOOCV)**
  - **How It Works**: Uses a single data point as the validation set ($k = 1$) and the rest as the training set. Repeat for all data points.
  - **Properties:**
    - **No Data Wastage**: Every data point is used for both training and validation.
    - **High Variance, Low Bias**.
    - **Computationally Expensive**: Requires training the model N times for N data points, making it slow for large datasets.
    - **Best for small datasets**.

Cross-Validation for Choosing Regularization Term

Large λ
(Prefer to more simple models)



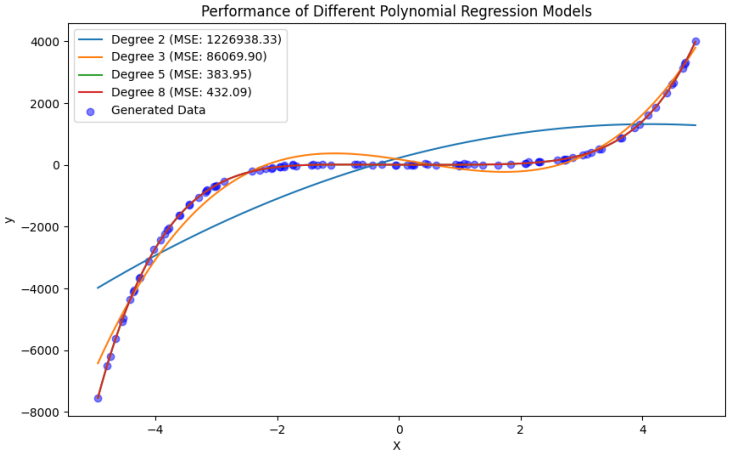
Intermediate λ



Small λ
(Prefer to more complex models)

Figures adapted from slides of F.Salehi, Machine Learning course, Sharif University of Technology.

Cross-Validation for Choosing Model Complexity



[1] C. M. Bishop, *Pattern Recognition and Machine Learning*.

[2] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. 2001.

[3] M. Soleymani, "Machine learning." Sharif University of Technology.

[4] S. F. S. Salehi, "Machine learning." Sharif University of Technology.

[5] Y. S. Abu-Mostafa, "Machine learning." California Institute of Technology, 2012.

[6] L. G. Serrano, *Grokking Machine Learning*. Manning Publications, 2020.

[7] J. M. Ashfaque, "Introduction to support vector machines and kernel methods." April 2019.