Evaluation Metrics in ML

Evaluation metrics in machine learning are mathematically defined based on the type of predictions a model produces. The formulas below represent the standard industry definitions for classification, regression, and ranking tasks.

Classification Formulas

Classification metrics are derived from the Confusion Matrix, which categorizes predictions into True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).[12][16]

- **Accuracy**: The fraction of total predictions that are correct.

  $\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$

- **Precision**: Measures the quality of positive predictions

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

- **Recall (Sensitivity)**: Measures the ability to find all positive instances.

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

- **F1 Score**: The harmonic mean of precision and recall.

  $F1 = 2 \times \frac{\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}$

Regression Formulas

Regression formulas quantify the difference between the predicted value y_i and the actual value x_i for n data points.[13][20]

- **Mean Absolute Error (MAE)**: The average of absolute residuals.

  $\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |x_i - y_i|$

- **Mean Squared Error (MSE)**: Penalizes larger errors by squaring them.

  $\text{MSE} = \frac{1}{n}\sum_{i=1}^{n} (x_i - y_i)^2$

- **Root Mean Squared Error (RMSE)**: Brings the error back to the original units

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (x_i - y_i)^2}$$

- **R-Squared (R^2)**: Indicates the proportion of variance explained by the model

$$R^2 = 1 - \frac{\sum (x_i - y_i)^2}{\sum (x_i - \bar{x})^2}$$

Ranking and Retrieval Formulas

Ranking metrics evaluate the order of items, often within a set of top k results.[14][22]

- **Average Precision (AP)**: Calculated for a single query as the average of precision values at each relevant position.

$$\text{AP} = \frac{\sum_{k=1}^{n} (P(k) \times rel(k))}{\text{number of relevant items}}$$

- **Mean Average Precision (MAP)**: The mean of AP values across all Q queries.

$$\text{MAP} = \frac{1}{|Q|}\sum_{q=1}^{|Q|} AP(q)$$

- **Discounted Cumulative Gain (DCG)**: Penalizes relevant items appearing lower in the list.

$$\text{DCG}_k = \sum_{i=1}^{k} \frac{rel_i}{\log_2 (i+1)}$$

- **Normalized DCG (NDCG)**: Divides DCG by the Ideal DCG (IDCG) to normalize scores between 0 and 1.

$$\text{NDCG}_k = \frac{\text{DCG}_k}{\text{IDCG}_k}$$

Regularization in ML

Regularization is a technique used in machine learning to prevent **overfitting** by discouraging the model from learning overly complex patterns. It achieves this by adding a penalty term to the standard loss function, which constrains the magnitude of the model's coefficients (weights).

Core Regularization Techniques

Most regularization methods follow the general structure of adding a penalty to the original loss function $L(\theta)$.

- **L1 Regularization (Lasso Regression)**: Adds a penalty equal to the absolute value of the magnitude of coefficients. It can shrink some coefficients exactly to zero, effectively performing automatic feature selection

$$\text{Loss}_{L1} = L(\theta) + \lambda \sum_{j=1}^{m} |w_j|$$

- **L2 Regularization (Ridge Regression)**: Adds a penalty equal to the square of the magnitude of coefficients. It shrinks coefficients toward zero but rarely makes them exactly zero, helping to handle multicollinearity.

$$\text{Loss}_{L2} = L(\theta) + \lambda \sum_{j=1}^{m} w_j^2$$

- **Elastic Net**: A hybrid approach that combines both L1 and L2 penalties. It is particularly useful when there are multiple correlated features.

$$\text{Loss}_{EN} = L(\theta) + \lambda_1 \sum_{j=1}^{m} |w_j| + \lambda_2 \sum_{j=1}^{m} w_j^2$$

Comparison of Methods

Technique	Penalty Term	Effect on Weights	Key Benefit
L1 (Lasso)	Sum of absolute values	Can zero out weights	Feature selection/Sparsity [36][39].
L2 (Ridge)	Sum of squares	Shrunk but non-zero	Handles multicollinearity [36][40].
Elastic Net	Combined L1 & L2	Balanced shrinkage	Robust with correlated features [36][41].

Other Forms of Regularization

Beyond weight penalties, other techniques help improve a model's ability to generalize to unseen data.[42][35]

- **Dropout**: Randomly "drops out" neurons during training in neural networks, preventing them from co-adapting too closely
- **Early Stopping**: Halts the training process as soon as the performance on a validation set starts to degrade, even if the training error continues to decrease
- **Data Augmentation**: Increases the diversity of training data by applying transformations (e.g., rotating images), which acts as a form of regularization by exposing the model to more variations.

Cross Validation in Machine Learning

Cross-validation is a technique used to check how well a machine learning model performs on unseen data while preventing overfitting. It works by:

- Splitting the dataset into several parts.
- Training the model on some parts and testing it on the remaining part.
- Repeating this resampling process multiple times by choosing different parts of the dataset.
- Averaging the results from each validation step to get the final performance.

Types of Cross-Validation

There are several types of cross-validation techniques which are as follows:

1. Holdout Validation

In Holdout Validation method typically 50% data is used for training and 50% for testing. Making it simple and quick to apply. The major drawback of this method is that only 50% data is used for training, the model may miss important patterns in the other half which leads to high bias.

2. LOOCV (Leave One Out Cross Validation)

In this method the model is trained on the entire dataset except for one data point which is used for testing. This process is repeated for each data point in the dataset.

- All data points are used for training, resulting in low bias.
- Testing on a single data point can cause high variance, especially if the point is an outlier.
- It can be very time-consuming for large datasets as it requires one iteration per data point.

3. Stratified Cross-Validation

It is a technique that ensures each fold of the cross-validation process has the same class distribution as the full dataset. This is useful for imbalanced datasets where some classes are underrepresented.

- The dataset is divided into k folds, keeping class proportions consistent in each fold.
- In each iteration, one-fold is used for testing and the remaining folds for training.
- This process is repeated k times so that each fold is used once as the test set.
- It helps classification models generalize better by maintaining balanced class representation.

4. K-Fold Cross Validation

K-Fold Cross Validation splits the dataset into *k* equal-sized folds. The model is trained on *k-1* folds and tested on the remaining fold. This process is repeated *k* times each time using a different fold for testing.

Example of K Fold Cross Validation

The diagram below shows an example of the training subsets and evaluation subsets generated in k-fold cross-validation. Here we have total 25 instances.

K Fold Cross Validation

- Here we will take k as 5.
- **1st iteration:** The first 20% of data [1–5] is used for testing and the remaining 80% [6–25] is used for training.
- **2nd iteration:** The second 20% [6–10] is used for testing and the remaining data [1–5] and [11–25] is used for training.
- This process continues until each fold has been used once as the test set.

Iteration	Training Set Observations	Testing Set Observations
1	[5-24]	[0-4]
2	[0-4, 10-24]	[5-9]
3	[0-9, 15-24]	[10-14]
4	[0-14, 20-24]	[15-19]
5	[0-19]	[20-24]

Each iteration uses different subsets for testing and training, ensuring that all data points are used for both training and testing.

Comparison between K-Fold Cross-Validation and Hold Out Method

K-Fold Cross-Validation and Hold Out Method are used technique and sometimes they are confusing so here is the quick comparison between them:

Feature	K-Fold Cross-Validation	Holdout Method
Data Split	Dataset is divided into k folds, and each fold is used once as test set	Dataset is split once, typically into training and testing sets
Training & Testing	Model is trained and tested k times, each fold serving as test set once	Model is trained once on training set and tested once on test set
Bias & Variance	Lower bias, more reliable performance estimate and variance depend on k	Higher bias if the split is not representative and results can vary significantly
Execution Time	Slower, especially for large datasets because model is trained k times	Faster, only one training and testing cycle
Best Use Case	Small to medium datasets where accuracy estimation is important	Very large datasets or when quick evaluation is needed