Performance Analysis in Machine Learning

Outline

Overview of Key Metrics

Overview of Key Metrics

- MSE: Mean Squared Error
- RMSE: Root Mean Squared Error
- R^2: Coefficient of Determination
- ROC-AUC: Area Under the ROC Curve
- Confusion Matrix
- Precision
- Recall
- F1 Score

MSE (Mean Squared Error)

MSE: Definition and Formula

Definition: Average of squared prediction errors.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- Lower is better.
- Sensitive to large errors.

MSE: Example

Actual values: $y = [3, -0.5, 2, 7]$

Predicted values: $\hat{y} = [2.5, 0.0, 2, 8]$

$$\text{MSE} = \frac{1}{4}[(3 - 2.5)^2 + (-0.5 - 0.0)^2 + (2 - 2)^2 + (7 - 8)^2]$$

$$= \frac{1}{4}[0.25 + 0.25 + 0 + 1] = \frac{1.5}{4} = 0.375$$

RMSE (Root Mean Squared Error)

RMSE: Formula and Intuition

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

- Same units as the target variable.
- Interpretable and commonly reported.

RMSE: Example

Using the same MSE as before:

$$\text{MSE} = 0.375 \quad \Rightarrow \quad \text{RMSE} = \sqrt{0.375} \approx 0.612$$

R^2 (Coefficient of Determination)

R^2: Formula and Meaning

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

- Measures proportion of variance explained.
- $R^2 = 1$: perfect model
- $R^2 = 0$: same as predicting mean

R²: Example

Actual: $y = [1, 2, 3] \quad \hat{y} = [1.1, 1.9, 3.2]$

Step 1: Mean $\bar{y} = 2$

$$\mathrm{SS}_{\mathrm{res}} = (1-1.1)^2+(2-1.9)^2+(3-3.2)^2 = 0.01+0.01+0.04 = 0.06$$

$$\mathrm{SS}_{\mathrm{tot}} = (1-2)^2 + (2-2)^2 + (3-2)^2 = 1 + 0 + 1 = 2$$

$$R^2 = 1 - \frac{0.06}{2} = 0.97$$

Confusion Matrix: Definition

A confusion matrix summarizes prediction results for binary classification:

$$\begin{bmatrix} \mathrm{TP} & \mathrm{FN} \\ \mathrm{FP} & \mathrm{TN} \end{bmatrix}$$

- TP: True Positives
- TN: True Negatives
- FP: False Positives
- FN: False Negatives

Example Dataset

Email	Actual (Spam?)	Predicted (Spam?)
1	1	1
2	0	0
3	1	1
4	1	0
5	0	1
6	1	1
7	0	0
8	0	0
9	1	0
10	0	0

From this we get:

$$TP = 3, \quad FN = 2, \quad FP = 1, \quad TN = 4$$

Precision

Definition: Precision measures the proportion of predicted positives that are actually positive.

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

$$\text{Precision} = \frac{3}{3+1} = \frac{3}{4} = 0.75$$

Interpretation: When the model predicts spam, it's correct 75% of the time.

Recall

Definition: Recall measures the proportion of actual positives that were correctly predicted.

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

$$\text{Recall} = \frac{3}{3+2} = \frac{3}{5} = 0.60$$

Interpretation: The model catches 60% of actual spam emails.

F1 Score

Definition: Harmonic mean of precision and recall.

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$F_1 = 2 \cdot \frac{0.75 \cdot 0.60}{0.75 + 0.60} = \frac{0.90}{1.35} \approx 0.667$$

Interpretation: Balanced measure that considers both false positives and false negatives.

Summary of Metrics

Metric	Value
True Positives (TP)	3
True Negatives (TN)	4
False Positives (FP)	1
False Negatives (FN)	2
Precision	0.75
Recall	0.60
F1 Score	0.667

ROC-AUC (Classification)

ROC-AUC: Concept

- ROC Curve: TPR(True Positive Rate) vs. FPR (False Positive Rate) at different thresholds.
- AUC: Area under the ROC curve.
- Measures the model's ability to separate classes.

AUC = Probability(model ranks a random positive higher than a random negative)

ROC-AUC: Mini Example

True labels: $y = [0, 0, 1, 1]$

Predicted probs: $\hat{p} = [0.1, 0.4, 0.35, 0.8]$
Threshold $= 0.5$

- Pairwise comparisons of (positive, negative) $\rightarrow$ 4 pairs
- Model correctly ranks $3/4 \rightarrow$ AUC $= 0.75$

Summary

Metric Comparison Summary

- **MSE, RMSE**: Use for regression error.
- **R^2**: Goodness of fit in regression.
- **ROC-AUC**: Classifier performance independent of threshold.
- **Confusion Martix**: Confusion matrix provides the foundation for most classification metrics.
- **Precision**: Use Precision when false positives are costly.
- **Recall**: Use Recall when false negatives are costly.
- **F1Score**: F1 Score balances both — useful in imbalanced datasets.

More Important Metrics

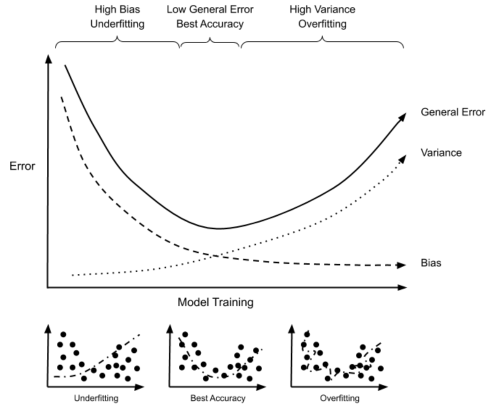
What Are Overfitting and Underfitting?

- Two common modeling errors in machine learning.
- **Overfitting**: Model learns training data (and noise) too well.
- **Underfitting**: Model is too simple to capture patterns.

Goal: Build models that generalize well to unseen data.

Intuition

- **Overfit** $\rightarrow$ low bias, high variance
- **Underfit** $\rightarrow$ high bias, low variance
- We aim for the balance between bias and variance.



Overfitting

Symptoms:

- High training accuracy, low test accuracy
- Model too complex for the data

Causes:

- Too many features or parameters
- Training too long
- Small training set

Fixing Overfitting

Common solutions:

- Use regularization (L1, L2)
- Early stopping
- Cross-validation
- Reduce model complexity
- Data augmentation

Underfitting

Symptoms:

- Poor performance on both training and test sets
- Model fails to capture structure

Causes:

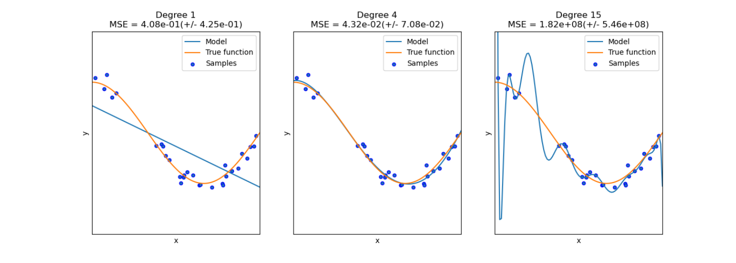
- Model too simple
- Insufficient training
- Too much regularization

Fixing Underfitting

Common solutions:

- Increase model complexity
- Train longer (more epochs)
- Reduce regularization
- Improve features or transformations

Visual Example



Source: Scikit-learn Documentation

Bias-Variance Tradeoff

Metric	Behavior
Bias	High in underfitting
Variance	High in overfitting
Goal	Low bias and low variance

How to Detect Overfitting and Underfitting

- Compare training vs. validation error
- Plot learning curves
- Use cross-validation
- Monitor metrics: accuracy, loss, F1, etc.

Python Example: Overfitting

```
from sklearn.pipeline import make_pipeline
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error

# Overfitting with high-degree polynomial
model = make_pipeline(PolynomialFeatures(15), LinearRegression()
model.fit(X_train, y_train)

print("Train Error:", mean_squared_error(y_train, model.predict(
print("Test Error:", mean_squared_error(y_test, model.predict(X_
```

Best Practices

- Always evaluate on unseen data
- Use regularization wisely
- Visualize performance over time
- Prefer simpler models when possible (Occam's Razor)

Thank you!