02 Classification Metrics in Machine Learning

Classification Metrics and Confusion Matrices

Introduction to Classification Metrics

Classification metrics are essential tools for evaluating the performance of machine learning models that classify data into categories. These metrics help data scientists determine whether a model is performing well or needs further optimization through hyperparameter tuning and additional training iterations.

Common Classification Metrics

1. Accuracy

Accuracy measures the fraction of total predictions that were correct.

Formula:

```
Accuracy = Number of Correct Predictions / Total Number of Predictions
```

For binary classification, accuracy can also be calculated using:

```
Accuracy = (TP + TN) / (TP + TN + FP + FN)
```

Where:

- **TP (True Positive)**: Model predicts positive, and the actual value is positive
- **TN (True Negative)**: Model predicts negative, and the actual value is negative
- **FP (False Positive)**: Model predicts positive, but the actual value is negative
- **FN (False Negative)**: Model predicts negative, but the actual value is positive

Example Context: In a public safety model predicting arrests:

- **True Positive**: Model predicts an arrest, and an arrest occurs
- **True Negative**: Model predicts no arrest, and no arrest occurs
- **False Positive**: Model predicts an arrest, but no arrest occurs
- **False Negative**: Model predicts no arrest, but an arrest occurs

Accuracy alone is not recommended for imbalanced datasets, where one class significantly outnumbers the other. In such cases, a model could achieve high accuracy simply by always predicting the majority class.

2. Precision

Precision quantifies how many of the positive predictions actually belong to the positive class.

Formula:

```
Precision = TP / (TP + FP)
```

When to use: Use precision when you want to minimize false positives.

Example Use Case: Spam email filtering

- You don't want important emails (non-spam) to be incorrectly classified as spam (false positives)
- A high precision model ensures that when an email is classified as spam, it's likely to be actual spam

3. Recall (Sensitivity)

Recall measures how many of the actual positives your model correctly identified.

Formula:

```
Recall = TP / (TP + FN)
```

When to use: Use recall when false positives are acceptable, but you want to minimize false negatives.

Example Use Case: Credit card fraud detection

- It's better to flag a legitimate transaction as potentially fraudulent (false positive) than to miss an actual fraudulent transaction (false negative)
- The inconvenience of occasionally dealing with a falsely flagged transaction is preferable to the significant harm of allowing fraud
- -- **Which metric measures accuracy while highlighting the sensitivity of your model?**
- -- Ans: *Recall is another way to measure accuracy and highlight your model's sensitivity. Recall quantifies the number of positive predictions made out of all the actual positives in the dataset.*

4. F1 Score

F1 score provides a balance between precision and recall, offering a single metric that combines both considerations.

Formula:

```
F1 = 2 * (Precision * Recall) / (Precision + Recall)
```

When to use: F1 score is particularly effective for imbalanced datasets where a balance between precision and recall is needed.

5. AUC (Area Under the ROC Curve)

AUC measures how well predictions are ranked across different threshold settings by calculating the area under the Receiver Operating Characteristic (ROC) curve.

What it represents: How well the model can distinguish between classes across different classification thresholds.

When to use: When you want to evaluate model performance independent of a specific classification threshold, especially when false positive rates are important.

The Confusion Matrix

A confusion matrix is not a metric itself but a table that summarizes prediction results, showing counts of true positives, true negatives, false positives, and false negatives.

Structure of a Binary Confusion Matrix

...	Predicted: Positive	Predicted: Negative
Actual: Positive	True Positives (TP)	False Negatives (FN)

...	Predicted: Positive	Predicted: Negative
Actual: Negative	False Positives (FP)	True Negatives (TN)

Interpreting the Confusion Matrix

The confusion matrix provides visual insights into the types of errors made by your model:

- **Diagonal elements (TP, TN):** Represent correct predictions
- **Off-diagonal elements (FP, FN):** Represent incorrect predictions

Example Confusion Matrix (Public Safety Model)

```
True Positives (TP): 51,492
True Negatives (TN): 102,583
False Positives (FP): 64,021
False Negatives (FN): 33,720
```

Calculating Metrics from the Confusion Matrix

Using the above example:

- **Accuracy** = (51,492 + 102,583) / (51,492 + 102,583 + 64,021 + 33,720) = 154,075 / 251,816 = 0.61 or 61%
- **Precision** = 51,492 / (51,492 + 64,021) = 51,492 / 115,513 = 0.45 or 45%
- **Recall** = 51,492 / (51,492 + 33,720) = 51,492 / 85,212 = 0.60 or 60%
- **F1 Score** = 2 *(0.45* 0.60) / (0.45 + 0.60) = 2 * 0.27 / 1.05 = 0.54 / 1.05 = 0.51 or 51%

Multi-class Confusion Matrices

For problems with more than two classes, the confusion matrix expands to an n × n matrix, where n is the number of classes. Each row represents the actual class, and each column represents the predicted class.

Implementing Classification Metrics in Python

Creating a Confusion Matrix

```python
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Assuming y_true and y_pred are your actual and predicted values
cm = confusion_matrix(y_true, y_pred)

# Create a visual representation
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

Generating a Complete Classification Report

```
from sklearn.metrics import classification_report

# Get all metrics at once
report = classification_report(y_true, y_pred)
print(report)
```

Practical Considerations for Choosing Metrics Imbalanced Datasets

When working with imbalanced datasets:

- Accuracy can be misleading
- Consider precision, recall, F1 score, or area under the Precision-Recall curve instead
- Techniques like oversampling, undersampling, or using class weights may help improve model performance

Domain-Specific Requirements

Different applications require different optimization priorities:

1. **Medical Diagnosis**:
   - High recall (sensitivity) is critical to avoid missing diseases
   - False positives can be tolerated as they lead to further testing
2. **Legal Applications**:
   - High precision might be prioritized where "innocent until proven guilty" is the standard
   - False accusations have significant consequences
3. **Financial Fraud Detection**:
   - Balance between catching fraud (recall) and minimizing false alarms (precision)
   - Cost of false negatives (missed fraud) vs. cost of investigation for false positives

Advanced Classification Metrics

Beyond the basic metrics covered, consider exploring:

- **Matthews Correlation Coefficient (MCC)**: Especially useful for imbalanced datasets
- **Cohen's Kappa**: Measures agreement between predicted and actual classifications, accounting for chance
- **Log Loss**: Measures the performance of a classification model where the prediction is a probability value
- **Balanced Accuracy**: The average of sensitivity and specificity
- **Specificity**: True negative rate
- **Precision-Recall AUC**: Area under the precision-recall curve, useful for imbalanced datasets

Implementation Example: Public Safety Model Analysis

```
# Calculate metrics manually
accuracy = (TP + TN) / (TP + TN + FP + FN)
precision = TP / (TP + FP)
recall = TP / (TP + FN)
f1 = 2 * (precision * recall) / (precision + recall)

print(f"Manual calculations:")
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1 Score: {f1:.2f}")

# Or use sklearn's functions
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score

accuracy = accuracy_score(y_true, y_pred)
precision = precision_score(y_true, y_pred)
recall = recall_score(y_true, y_pred)
f1 = f1_score(y_true, y_pred)

print(f"\nScikit-learn calculations:")
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1 Score: {f1:.2f}")
```

Summary and Key Takeaways

1. **Classification metrics** help evaluate model performance for classification problems
2. **Accuracy** measures overall correctness but can be misleading for imbalanced datasets
3. **Precision** focuses on the correctness of positive predictions
4. **Recall** measures how well the model captures all positive instances
5. **F1 score** balances precision and recall
6. **AUC** evaluates model performance across different thresholds
7. **Confusion matrices** provide visual representation of model performance
8. **Choose metrics** based on your specific use case and the costs of different types of errors

Next Steps

- [ ] Research additional classification metrics relevant to your specific use case
- [ ] Explore regression metrics (coming in the next lecture)
- [ ] Practice implementing these metrics in Python with real datasets
- [ ] Consider how to optimize your models based on the most appropriate metrics

Questions for Understanding

1. Why might accuracy be a poor metric for an imbalanced dataset?
2. In what scenarios would you prioritize precision over recall?
3. How would you interpret an F1 score of 0.90 versus 0.30?
4. How can the confusion matrix help you diagnose specific weaknesses in your model?
5. For your specific application, which metric would be most appropriate and why?

1. Why might accuracy be a poor metric for an imbalanced dataset?

Accuracy measures the proportion of correct predictions (both true positives and true negatives) out of all predictions made. In imbalanced datasets, where one class significantly outnumbers the other(s), accuracy can be misleading because it gives equal weight to both types of errors (false positives and false negatives).

For example, in a binary classification task where 95% of instances belong to Class A and 5% to Class B, a naive model that always predicts Class A will achieve 95% accuracy—appearing highly accurate despite failing entirely on the minority class.

This issue arises because:

- Accuracy does not reflect performance on underrepresented classes.
- It doesn't account for cost asymmetry between different types of errors (e.g., missing a cancer case vs. a false alarm).
- It's insensitive to changes in the confusion matrix when the positive class is rare.

Thus, accuracy should be avoided or used cautiously in such settings, and more informative metrics like F1 score, precision-recall curves, or area under the ROC curve (AUC-ROC) are preferred.

2. In what scenarios would you prioritize precision over recall?

Precision quantifies how many selected items are relevant (i.e., of all the predicted positives, how many are actually positive). Recall measures how many relevant items were selected (i.e., of all actual positives, how many were correctly identified).

You would prioritize precision over recall in situations where:

- **False positives are costly or dangerous.** For example:
  - Medical testing for a disease with invasive follow-up procedures: you want high confidence that anyone flagged as positive truly has the condition.
  - Spam detection: marking a legitimate email as spam (false positive) may lead to important information being missed.
  - Legal document retrieval: falsely including irrelevant documents in a discovery process could have legal consequences.

In these cases, minimizing false positives is critical—even if it means missing some true positives (lower recall).

3. How would you interpret an F1 score of 0.90 versus 0.30?

The F1 score is the harmonic mean of precision and recall, balancing the trade-off between them. It ranges from 0 (worst) to 1 (best).

- **F1 = 0.90**: This indicates a very strong balance between precision and recall. Either both metrics are high (~0.9 or above), or at least one is high while the other remains reasonably good. The model is effectively identifying positive instances without introducing too many false positives or missing too many true positives. This is generally considered excellent performance, especially in imbalanced settings.
- **F1 = 0.30**: This suggests poor performance. Either precision or recall (or both) is very low. The model is either:
  - Making many false positives (low precision),
  - Missing most of the actual positives (low recall), or
  - Both.

This score implies the model needs significant improvement, possibly due to class imbalance, poor feature engineering, incorrect thresholding, or inadequate model capacity.

---

4. How can the confusion matrix help you diagnose specific weaknesses in your model?

The confusion matrix provides a granular breakdown of prediction outcomes into four categories: True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN). From this, you can derive numerous diagnostic insights:

- **Class Imbalance Sensitivity**: If the model predicts only the majority class, you'll see high TN/TP but zero or near-zero FP/FN for the minority class.
- **Bias Toward One Class**: High FP indicates over-prediction of positives; high FN indicates under-detection of positives.
- **Cost-Sensitive Errors**: In domains like healthcare or finance, knowing which type of error dominates helps guide recalibration or retraining.
- **Threshold Tuning**: By analyzing how TP/FP rates change across thresholds, you can adjust decision boundaries to optimize for precision or recall.
- **Model Calibration**: Discrepancies between predicted probabilities and actual outcomes can be inferred from patterns in the matrix.

In summary, the confusion matrix enables targeted improvements by highlighting exactly *where* and *how* the model fails.

---

5. For your specific application, which metric would be most appropriate and why?

(Note: Since no specific application was provided, I'll describe how to approach this question in general terms.)

To determine the most appropriate metric, consider the following factors:

a. Nature of the Problem

- Binary classification? Multi-class? Anomaly detection?

- Is the data imbalanced?

b. Cost of Errors

- Are false positives more harmful than false negatives?
- Or vice versa?

c. End Use Case

- Is the goal to rank instances (e.g., risk scoring), or make hard classifications?
- Will domain experts review top-ranked cases, or must the model be fully autonomous?

d. Stakeholder Priorities

- Do stakeholders care about overall correctness (accuracy), catching all positives (recall), or avoiding false alarms (precision)?
- Is there a need for probabilistic outputs (then AUC-ROC or log loss may be useful)?

Example Applications:

- **Fraud Detection**: Precision is key—too many false positives waste resources. But recall matters too—missing fraud is costly. Hence, F1 or AUC-ROC are common choices.
- **Medical Screening**: Recall is often prioritized—to ensure no sick patients are missed—even if it means more false positives.
- **Customer Churn Prediction**: A mix of precision and recall may be needed depending on marketing budget constraints.
- **Image Classification (Balanced)**: Accuracy may suffice, especially if all classes are equally important.

Conclusion: There's no universal best metric. Choose based on domain understanding, error costs, and deployment context. Often, reporting multiple metrics (e.g., precision, recall, F1, AUC) gives a fuller picture than relying on any single one.