Evaluation Metrics for Supervised ML Models

Case Study & Discussion

Case Study & Discussion: Evaluating Model Performance

Learning Objective

By the end of this lesson, students will be able to:

- Apply classification and regression evaluation metrics to real-world datasets.
- Analyze model performance using computed metrics.
- Recommend improvements based on evaluation results.

Real-World Scenario

Imagine you are a data scientist at a healthcare company developing two models:

1. A **classification model** to detect whether a tumor is malignant or benign.
2. A **regression model** to predict patient recovery time after surgery.

Your task is to evaluate both models and determine if they meet business requirements.

Case Study: Classification Model (Tumor Detection with Imbalanced Data & Multi-Class Scenario)

In a more complex setting, your classification model is now detecting not only **malignant vs. benign** tumors but also differentiating between **different stages of malignancy** (early, mid, advanced). The dataset is highly imbalanced, as most tumors in the dataset are benign.

Predicted vs. Actual Labels

Copy

```
import numpy as np
from sklearn.metrics import classification_report, confusion_matrix

# True labels (0 = Benign, 1 = Early-stage malignant, 2 = Mid-stage malignant, 3
y_true = np.array([0, 1, 2, 2, 3, 0, 1, 1, 2, 0, 3, 3, 1, 0, 2])

# Model predictions
y_pred = np.array([0, 1, 2, 1, 3, 0, 0, 1, 2, 0, 3, 2, 1, 0, 3])

print("Classification Report:")
print(classification_report(y_true, y_pred))
print("Confusion Matrix:")
print(confusion_matrix(y_true, y_pred))
```

Discussion Questions

- How does the **class imbalance** affect precision and recall for each class?
- Why might the **F1-score** be more useful than accuracy in this scenario?
- How can we improve predictions for underrepresented tumor stages?

Case Study: Regression Model (Predicting Recovery Time with Additional Features)

Your team has enhanced the regression model by incorporating additional patient features such as **age, pre-existing conditions, and type of surgery**. Your task is to analyze whether these improvements result in a **better predictive model**.

Copy

```python
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Simulated patient data
patient_data = pd.DataFrame({
    "age": [35, 50, 60, 45, 70, 55, 40, 65, 30, 75],
    "pre_existing_conditions": [1, 2, 3, 2, 3, 1, 0, 2, 1, 3],
    "surgery_type": [0, 1, 1, 0, 2, 1, 0, 2, 0, 2],
    "recovery_days": [10, 12, 15, 14, 18, 20, 25, 22, 30, 28]
})

# Predicting recovery time using multiple features
X = patient_data[["age", "pre_existing_conditions", "surgery_type"]]
y = patient_data["recovery_days"]

model = LinearRegression()
model.fit(X, y)
y_pred = model.predict(X)

mae = mean_absolute_error(y, y_pred)
mse = mean_squared_error(y, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y, y_pred)

print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")
print(f"R-squared (R²): {r2:.2f}")
```

Discussion Questions

- Does adding more features improve model performance based on **R²**?
- Are there any **feature engineering** techniques that might enhance predictive accuracy?
- Would a **non-linear model** (e.g., decision tree, random forest) perform better?

Key Takeaways

- In multi-class classification, **confusion matrices** provide deeper insights than just accuracy.
- **Class imbalance** significantly impacts model performance; techniques like **resampling** or **weighted loss functions** may help.
- In regression, incorporating **more meaningful features** can improve predictive performance, but model complexity must be managed carefully.

< Previous