Day 8 Training Report

2 July 2025

Model Evaluation — Accuracy, Confusion Matrix, Precision, Recall

On **Day 8**, the focus was on understanding **how to evaluate machine learning models effectively**. While building models is essential, evaluating their performance is equally critical to ensure they make **reliable and meaningful predictions**. Students explored key **classification metrics** such as **accuracy**, **confusion matrix**, **precision**, and **recall** using practical examples.

---

1. Introduction to Model Evaluation

Model evaluation helps determine **how well a machine learning model performs on unseen data**. Relying on a single metric can be misleading; hence, multiple metrics are used to assess different aspects of performance.

- **Training Accuracy** → How well the model fits the training data
- **Testing Accuracy** → How well the model generalizes to new data

In classification tasks, metrics like **accuracy**, **precision**, **recall**, and **F1-score** provide deeper insights, especially when the dataset is **imbalanced** (e.g., spam vs. non-spam emails, disease detection).

---

2. Accuracy

Accuracy is the simplest evaluation metric. It measures the **overall correctness** of the model's predictions.

Accuracy=Number of Correct PredictionsTotal Number of Predictions\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}Accuracy=Total Number of PredictionsNumber of Correct Predictions

✅ **Pros:** Easy to understand and calculate.
⚠️ **Cons:** Can be **misleading with imbalanced datasets** (e.g., if 95% of data is "No Disease", predicting all as "No Disease" gives 95% accuracy but is useless).

Example:

```
from sklearn.metrics import accuracy_score

y_true = [1, 0, 1, 1, 0, 1, 0]
y_pred = [1, 0, 1, 0, 0, 1, 1]

acc = accuracy_score(y_true, y_pred)
```

```
print("Accuracy:", acc)
```

---

3. Confusion Matrix

The **confusion matrix** provides a **detailed breakdown of prediction outcomes**. It shows how many instances were correctly or incorrectly classified.

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

Terms Explained:

- **TP (True Positive)** → Correctly predicted positive cases
- **TN (True Negative)** → Correctly predicted negative cases
- **FP (False Positive)** → Incorrectly predicted as positive
- **FN (False Negative)** → Incorrectly predicted as negative

Example:

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_true, y_pred)
print("Confusion Matrix:\n", cm)
```

Sample Output:

```
Confusion Matrix:
[[2 1]
 [1 3]]
```

Interpretation:

- 2 TN, 1 FP, 1 FN, 3 TP

☑ **Key Insight:** Confusion matrices are extremely helpful for identifying **what types of errors** a model is making.

---

4. Precision and Recall

When accuracy is not enough (e.g., detecting rare diseases or fraudulent transactions), **precision** and **recall** become more important.

Precision:

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

It answers: “Out of all positive predictions, how many were actually correct?”

- High precision = low false positives
- Useful in scenarios like **spam detection** (don’t wrongly classify important emails as spam).

Recall (Sensitivity):

Recall=TPTP+FN\text{Recall} = \frac{TP}{TP + FN}Recall=TP+FNTP

It answers: “Out of all actual positives, how many did the model correctly identify?”

- High recall = low false negatives
- Useful in **medical diagnosis**, where missing a positive case is dangerous.

Example:

```
from sklearn.metrics import precision_score, recall_score

precision = precision_score(y_true, y_pred)
recall = recall_score(y_true, y_pred)

print("Precision:", precision)
print("Recall:", recall)
```

Sample Output:

```
Precision: 0.75
Recall: 0.80
```

☑ **Key Points:**

- Precision and recall often **trade off** with each other.
- Depending on the application, one may be prioritized over the other.

---

5. Visualization of Confusion Matrix

Visualizing the confusion matrix helps interpret the results more intuitively.

```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(5,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```