Confusion Matrix in Machine Learning

Type	@datasciencebrain

1. Introduction to Confusion Matrix

A **Confusion Matrix** is a performance measurement tool for classification models. It is used to evaluate the accuracy of a classification algorithm by comparing the actual and predicted labels.

Why is it Important?

- Provides insight into **true positives, false positives, true negatives, and false negatives**.
- Helps in calculating various performance metrics such as **Accuracy, Precision, Recall, and F1-score**.
- Useful for handling **imbalanced datasets**.

---

2. Structure of a Confusion Matrix

A confusion matrix for a **binary classification problem** is represented as:

Actual / Predicted	Predicted Positive (1)	Predicted Negative (0)
Actual Positive (1)	True Positives (TP)	False Negatives (FN)
Actual Negative (0)	False Positives (FP)	True Negatives (TN)

Where:

- **TP (True Positive)**: The model correctly predicted the positive class.
- **FN (False Negative)**: The model incorrectly predicted the negative class for a positive instance.

- **FP (False Positive)**: The model incorrectly predicted the positive class for a negative instance.
- **TN (True Negative)**: The model correctly predicted the negative class.

For **multi-class classification**, the confusion matrix extends to multiple rows and columns where each row represents the actual class and each column represents the predicted class.

---

3. Performance Metrics Derived from Confusion Matrix

From the confusion matrix, we can derive several important evaluation metrics.

1. Accuracy

The proportion of correctly classified instances out of the total instances.

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

2. Precision (Positive Predictive Value)

The proportion of correctly predicted positive instances among all predicted positives.

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

3. Recall (Sensitivity, True Positive Rate)

The proportion of actual positive instances that were correctly predicted.

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

4. F1 Score

The harmonic mean of Precision and Recall, useful when the dataset is imbalanced.

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

5. Specificity (True Negative Rate)

The proportion of actual negative instances that were correctly predicted.

Specificity = $\frac{TN}{TN+FP}$

6. False Positive Rate (FPR)

FPR = $\frac{FP}{FP+TN}$

---

4. Implementing Confusion Matrix in Python

Let's implement a confusion matrix using **Scikit-Learn** in Python.

Step 1: Import Libraries

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, ConfusionMatrixDisplay
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_classification
from sklearn.ensemble import RandomForestClassifier
```

Step 2: Generate Data and Train Model

```
# Generate synthetic dataset
X, y = make_classification(n_samples=1000, n_features=10, n_classes=2, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a classifier
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

Step 3: Compute Confusion Matrix

```
# Compute confusion matrix
cm = confusion_matrix(y_test, y_pred)
```

```
print("Confusion Matrix:\n", cm)
```

Step 4: Visualize Confusion Matrix

```
# Plot confusion matrix
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(cmap='Blues')
plt.show()
```

Step 5: Compute Performance Metrics

```
# Calculate evaluation metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print(f'Accuracy: {accuracy:.2f}')
print(f'Precision: {precision:.2f}')
print(f'Recall: {recall:.2f}')
print(f'F1 Score: {f1:.2f}')
```

---

5. Interpretation of Confusion Matrix Results

Scenario 1: High TP, Low FP & FN

- **Good Model**: The classifier is performing well.
- **High Precision and Recall**

Scenario 2: High FP (False Positives)

- The model is predicting positives incorrectly.
- High **false alarm** rate, which is problematic in applications like **fraud detection**.

Scenario 3: High FN (False Negatives)

- The model is missing actual positives.
- Dangerous in **medical diagnosis** where missing a disease can be fatal.

Scenario 4: Balanced FP & FN but High TN

- The model may be **biased towards the negative class**.
- Useful in scenarios like spam detection where negative class (non-spam) dominates.
