DAY 19 **[16 - JULY - 2025]**

Model Evaluation and Validation

Introduction

Model evaluation and validation are critical steps in the machine learning pipeline to assess the performance of predictive models and ensure they generalize well to unseen data. Evaluation metrics such as accuracy, precision, recall, F1 score, and ROC curves provide quantitative measures of a model's effectiveness. This document explains these metrics, their use cases, and how to implement them using Python.

Evaluation Metrics

1. Accuracy

Definition: Accuracy measures the proportion of correct predictions made by the model out of all predictions.

Use Case: Suitable for balanced datasets where classes are equally represented.

Python Example:

```
from sklearn.metrics import accuracy_score

# Example: True and predicted labels
y_true = [1, 0, 1, 1, 0]
y_pred = [1, 0, 1, 0, 0]

accuracy = accuracy_score(y_true, y_pred)
print("Accuracy:", accuracy)
```

Output:

```
Accuracy: 0.8
```

Limitations: Can be misleading for imbalanced datasets, where a high accuracy may not reflect good performance on the minority class.

2. Precision

Definition: Precision measures the proportion of true positive predictions out of all positive predictions made by the model.

Use Case: Useful when the cost of false positives is high (e.g., spam detection).

Python Example:

```
from sklearn.metrics import precision_score
```

```
precision = precision_score(y_true, y_pred)
print("Precision:", precision)
```

Output:

```
Precision: 1.0
```

3. Recall (Sensitivity or True Positive Rate)

Definition: Recall measures the proportion of true positives identified out of all actual positive instances.

Use Case: Important when the cost of false negatives is high (e.g., disease detection).

Python Example:

```
from sklearn.metrics import recall_score

recall = recall_score(y_true, y_pred)
print("Recall:", recall)
```

Output:

```
Recall: 0.6666666666666666
```

4. F1 Score

Definition: The F1 score is the harmonic mean of precision and recall, balancing the trade-off between the two.

Use Case: Ideal for imbalanced datasets where both precision and recall are important.

Python Example:

```
from sklearn.metrics import f1_score

f1 = f1_score(y_true, y_pred)
print("F1 Score:", f1)
```

Output:

```
F1 Score: 0.8
```

5. ROC Curve and AUC

Definition: The Receiver Operating Characteristic (ROC) curve plots the True Positive Rate (Recall) against the False Positive Rate (FPR) at various threshold settings. The Area Under the Curve (AUC) summarizes the ROC curve, with values closer to 1 indicating better model performance.

Use Case: Useful for evaluating binary classifiers, especially when comparing models or assessing performance across thresholds.

Python Example:

```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

# Example: True labels and predicted probabilities
y_true = [1, 0, 1, 1, 0]
y_scores = [0.9, 0.1, 0.8, 0.4, 0.2]

# Compute ROC curve and AUC
fpr, tpr, thresholds = roc_curve(y_true, y_scores)
roc_auc = auc(fpr, tpr)

# Plot ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f'ROC curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], 'k--')  # Diagonal line
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.show()
```

Output: (A plot would be generated, showing the ROC curve with AUC value.)

6. Confusion Matrix

Definition: A confusion matrix is a table that summarizes the performance of a classification model by showing TP, TN, FP, and FN.

Python Example:

```
from sklearn.metrics import confusion_matrix
import seaborn as sns

# Compute confusion matrix
cm = confusion_matrix(y_true, y_pred)

# Visualize confusion matrix
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

Output: (A heatmap would be generated, showing the counts of TP, TN, FP, and FN.)