**By**

**NAME: Zainab Imran**

**ROLL NO #: 1220100382**

**BS.SE (4th)**

**Fall 2024**

**COURSE TITLE: Artificial Intelligence**

**SUBMITTED TO: Sir Zubair**

**ASSIGMENT NO.3**

To complete the assignment on model evaluation, we will follow these steps:

1. Choose a dataset for classification.
2. Train a classification model on the dataset.
3. Evaluate the model using accuracy, precision, recall, and F1-score.
4. Generate and interpret a confusion matrix.
5. Plot the ROC curve and calculate the AUC.
6. Perform k-fold cross-validation and report the results.

We will use Python's scikit-learn library for model training and evaluation. For this example, let's use the Iris dataset, which is a well-known dataset in the machine learning community.

Here is the step-by-step process:

**Step 1: Choose a Dataset and Train a Classification Model**

We'll use the Iris dataset and train a Logistic Regression model.

python

Copy code

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix, roc\_curve, auc

from sklearn.model\_selection import cross\_val\_score

import matplotlib.pyplot as plt

import numpy as np

# Load dataset

iris = load\_iris()

X, y = iris.data, iris.target

# Binary classification: Let's classify whether the flower is Iris-Setosa or not

y = (y == 0).astype(int)

# Split dataset into training and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Train a Logistic Regression model

model = LogisticRegression()

model.fit(X\_train, y\_train)

# Predict on the test set

y\_pred = model.predict(X\_test)

y\_proba = model.predict\_proba(X\_test)[:, 1]

**Step 2: Calculate Accuracy Metrics**

python

Copy code

# Calculate accuracy, precision, recall, and F1-score

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)

accuracy, precision, recall, f1

**Step 3: Create and Interpret a Confusion Matrix**

python

Copy code

# Create confusion matrix

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

conf\_matrix

**Step 4: Plot ROC Curve and Calculate AUC**

python

Copy code

# Calculate ROC curve and AUC

fpr, tpr, \_ = roc\_curve(y\_test, y\_proba)

roc\_auc = auc(fpr, tpr)

# Plot ROC curve

plt.figure()

plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc\_auc)

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic')

plt.legend(loc="lower right")

plt.show()

roc\_auc

**Step 5: Perform k-Fold Cross-Validation**

python

Copy code

# Perform 5-fold cross-validation

cv\_scores = cross\_val\_score(model, X, y, cv=5)

# Calculate mean and standard deviation of cross-validation scores

mean\_cv\_score = np.mean(cv\_scores)

std\_cv\_score = np.std(cv\_scores)

mean\_cv\_score, std\_cv\_score

**Results and Interpretation**

1. **Accuracy Metrics Calculation**
   * Accuracy:
   * Precision:
   * Recall:
   * F1-Score:

These metrics indicate the following:

* + **Accuracy**: The proportion of correctly classified instances among the total instances.
  + **Precision**: The proportion of positive identifications that were actually correct.
  + **Recall**: The proportion of actual positives that were correctly identified.
  + **F1-Score**: The harmonic mean of precision and recall, providing a single metric that balances both concerns.

1. **Confusion Matrix Interpretation**

plaintext

Copy code

[[TP, FP],

[FN, TN]]

* + TP: True Positives
  + FP: False Positives
  + FN: False Negatives
  + TN: True Negatives

The confusion matrix helps us understand the types of errors our model is making.

1. **ROC/AUC Calculation**
   * ROC Curve: (Graphically shows the trade-off between sensitivity and specificity)
   * AUC: (Represents the likelihood that the model will rank a randomly chosen positive instance higher than a randomly chosen negative one)
2. **Cross-Validation Reporting**
   * Mean Cross-Validation Accuracy:
   * Standard Deviation of Cross-Validation Accuracy:

Cross-validation is important because it provides a more reliable estimate of the model's performance by training and testing the model on different subsets of the data.

I will now run this code to obtain the exact values for the metrics.

Analyzed

python

Always show details

Copy code

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix, roc\_curve, auc

from sklearn.model\_selection import cross\_val\_score

import matplotlib.pyplot as plt

import numpy as np

# Load dataset

iris = load\_iris()

X, y = iris.data, iris.target

# Binary classification: Let's classify whether the flower is Iris-Setosa or not

y = (y == 0).astype(int)

# Split dataset into training and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Train a Logistic Regression model

model = LogisticRegression()

model.fit(X\_train, y\_train)

# Predict on the test set

y\_pred = model.predict(X\_test)

y\_proba = model.predict\_proba(X\_test)[:, 1]

# Calculate accuracy, precision, recall, and F1-score

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)

# Create confusion matrix

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

# Calculate ROC curve and AUC

fpr, tpr, \_ = roc\_curve(y\_test, y\_proba)

roc\_auc = auc(fpr, tpr)

# Plot ROC curve

plt.figure()

plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc\_auc)

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic')

plt.legend(loc="lower right")

plt.show()

# Perform 5-fold cross-validation

cv\_scores = cross\_val\_score(model, X, y, cv=5)

# Calculate mean and standard deviation of cross-validation scores

mean\_cv\_score = np.mean(cv\_scores)

std\_cv\_score = np.std(cv\_scores)

(accuracy, precision, recall, f1), conf\_matrix, roc\_auc, (mean\_cv\_score, std\_cv\_score)

Result

((1.0, 1.0, 1.0, 1.0),

array([[26, 0],

[ 0, 19]]),

1.0,

(1.0, 0.0))

**Results and Interpretation**

1. **Accuracy Metrics Calculation**
   * **Accuracy**: 1.0
   * **Precision**: 1.0
   * **Recall**: 1.0
   * **F1-Score**: 1.0

These metrics indicate that the model has perfectly classified all instances in the test set. Such high scores suggest that the model is extremely well-suited for this particular task, possibly due to the simplicity or structure of the dataset.

1. **Confusion Matrix Interpretation**

plaintext

Copy code

[[26, 0],

[ 0, 19]]

* + **TP (True Positives)**: 26 (Iris-Setosa correctly classified)
  + **FP (False Positives)**: 0 (Other classes incorrectly classified as Iris-Setosa)
  + **FN (False Negatives)**: 0 (Iris-Setosa incorrectly classified as other classes)
  + **TN (True Negatives)**: 19 (Other classes correctly classified)

The confusion matrix shows that the model has no misclassifications. This confirms the high values of accuracy, precision, recall, and F1-score.

1. **ROC/AUC Calculation**
   * **ROC Curve**: The ROC curve shows a perfect separation between the positive and negative classes.
   * **AUC (Area Under Curve)**: 1.0

The ROC curve and AUC value indicate that the model is perfect in distinguishing between the positive and negative classes. An AUC of 1.0 means the model has a perfect ability to rank positive instances higher than negative ones.

1. **Cross-Validation Reporting**
   * **Mean Cross-Validation Accuracy**: 1.0
   * **Standard Deviation of Cross-Validation Accuracy**: 0.0

Cross-validation results confirm that the model's performance is consistently perfect across different subsets of the dataset. Cross-validation is important because it provides a more reliable estimate of the model's performance by training and testing the model on different subsets of the data.

Overall, the model demonstrates excellent performance on the Iris dataset for the task of classifying Iris-Setosa vs. other classes. This could be due to the simplicity of the classification task or the nature of the dataset.