Assignment 3

Assignment on Model Evaluation

1. Accuracy Metrics Calculation

Let's train a classification model (Logistic Regression in this case) on the Breast Cancer dataset and calculate accuracy, precision, recall, and F1-score on the test set.

Python code

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, confusion_matrix
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

# Load the Breast Cancer dataset
data = load_breast_cancer()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = pd.Series(data.target)

# Splitting the 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)

# Training a Logistic Regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Predicting on the test set
y_pred = model.predict(X_test)

# Calculating 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)

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

Analysis:

- **Accuracy**: Percentage of correctly classified instances out of total instances.

- **Precision**: Proportion of correctly predicted positive instances (malignant tumors) out of all predicted positive instances.
- **Recall**: Proportion of correctly predicted positive instances out of all actual positive instances.
- **F1-Score**: Harmonic mean of precision and recall, providing a balance between them.

These metrics collectively provide insights into how well the model is performing in terms of correctly identifying malignant tumors without missing too many (high recall) or incorrectly classifying benign tumors as malignant (high precision).

2. Confusion Matrix Interpretation

Let's create a confusion matrix to further understand the model's performance.

Python code

```
# Creating a confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)
```

Analysis:

- **True Positive (TP)**: Number of malignant tumors correctly predicted.
- **True Negative (TN)**: Number of benign tumors correctly predicted.
- **False Positive (FP)**: Number of benign tumors incorrectly predicted as malignant.
- **False Negative (FN)**: Number of malignant tumors incorrectly predicted as benign.

The confusion matrix helps in understanding the distribution of predictions and provides a clear count of correct and incorrect predictions. It's especially useful for understanding the types of errors the model is making (e.g., false positives vs. false negatives) and assessing the balance between sensitivity (recall) and specificity.

3. ROC/AUC Calculation

Let's plot the ROC curve and calculate the AUC to evaluate the model's performance further.

Python code

```
# Calculating probabilities for ROC curve
y_pred_proba = model.predict_proba(X_test)[:, 1]

# Calculating ROC curve and AUC
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
roc_auc = auc(fpr, tpr)

# Plotting ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (AUC = %0.2f)'
% roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()

print(f"AUC: {roc_auc:.2f}")
```

Analysis:

- **ROC Curve**: Plots the true positive rate (Sensitivity) against the false positive rate (1 - Specificity) across different thresholds.
- **AUC (Area Under the Curve)**: Provides a single value to summarize the ROC curve's performance, where higher values (closer to 1) indicate better discrimination between classes.

The ROC curve and AUC provide insights into how well the model distinguishes between benign and malignant tumors. A higher AUC indicates better overall performance in classification.

4. Cross-Validation Reporting

Lastly, let's perform k-fold cross-validation (k=5) to evaluate the model's performance across different subsets of the data.

Python code

```
# Performing k-fold cross-validation
k = 5
scores = cross_val_score(model, X, y, cv=k, scoring='accuracy')

# Calculating mean and standard deviation of accuracy
mean_accuracy = scores.mean()
std_accuracy = scores.std()

print(f"Mean Accuracy: {mean_accuracy:.2f}")
print(f"Std of Accuracy: {std_accuracy:.2f}")
```

Analysis:

- **Mean Accuracy**: Average accuracy across k folds, providing an estimate of model performance.
- **Standard Deviation of Accuracy**: Measure of variability or consistency across different folds.

Cross-validation helps in assessing the model's robustness and generalization ability by using different subsets of data for training and validation. It provides a more reliable estimate of model performance compared to a single train-test split.