Task 4: Evaluating the Model Performance

Objective

In Task 4 we will assess the performance of this trained Artificial Neural Network (ANN) model on several important metrics. Accuracy, Precision, Recall (True Positive Rate), F1-Score, Confusion Matrix And Classification ReportsROC Curve and AUC (Area under the curve) This will assist us in gauging the model performance on test dataset and scope of improvement.

Steps Taken

1. Loading the Test Data and Predictions

- **Test Data:**
  - The test dataset is loaded using the same method as in Task 3.
  - **Why:**
    - Since we will be evaluating the model's performance, we need the actual labels (`y_test`) and the predicted labels (`y_pred`) generated in the previous task.
  - **Implementation:**

```
import pandas as pd



# Load the test dataset

test_data_path = config['test_data_path']

test_data = pd.read_csv(test_data_path)



# Extract the features (X) and target variable
(y)

X_test = test_data.drop(columns=['Churn_Yes'])  #
features

y_test = test_data['Churn_Yes']  # target
```

- **Loading Predictions:**
    - The predicted churn values saved in Task 3 are loaded to be compared against the actual values for evaluation.
    - **Why:**
        - To evaluate the performance, we need both the actual labels and the predicted labels to compute metrics like accuracy, precision, and recall.
    - **Implementation:**

```
# Load predictions

predictions_path = config['predictions_path']

predictions_df = pd.read_csv(predictions_path)



# Extract actual and predicted labels

y_test = predictions_df['Actual']

y_pred = predictions_df['Predicted']
```

2. Evaluating Model Performance Using Metrics

Several performance metrics were calculated to evaluate the model's effectiveness:

a. Accuracy

- **Description:**
    - Accuracy measures the proportion of correctly predicted instances (both churn and non-churn) over the total number of instances.
    - **Formula:**

      Where:

        - TP: True Positives
        - TN: True Negatives
        - FP: False Positives
        - FN: False Negatives
    - **Implementation:**

```
from sklearn.metrics import accuracy_score


accuracy = accuracy_score(y_test, y_pred)

print(f'Accuracy: {accuracy:.2f}')
```

b. Precision, Recall, and F1-Score

- **Precision:**
  - Precision measures the proportion of true positive churn cases among all predicted positive churn cases.
  - **Formula:**

- **Recall:**
  - Recall (or Sensitivity) measures the proportion of true positive churn cases correctly identified by the model.
  - **Formula:**
- **F1-Score:**
  - The F1-score is the harmonic mean of precision and recall, providing a single metric that balances both.
  - **Formula:**
  - **Implementation:**

```
from sklearn.metrics import precision_score,
recall_score, f1_score



precision = precision_score(y_test, y_pred)

recall = recall_score(y_test, y_pred)

f1 = f1_score(y_test, y_pred)



print(f'Precision: {precision:.2f}')

print(f'Recall: {recall:.2f}')

print(f'F1-Score: {f1:.2f}')
```

c. Confusion Matrix

- **Description:**
  - A confusion matrix provides a more detailed breakdown of the model's performance by showing the counts of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).
  - **Why:**
    - The confusion matrix helps in understanding the balance of correctly and incorrectly predicted cases, particularly in the context of imbalanced datasets.
  - **Implementation:**

```
from sklearn.metrics import confusion_matrix

import seaborn as sns

import matplotlib.pyplot as plt



cm = confusion_matrix(y_test, y_pred)



# Plot confusion matrix

plt.figure(figsize=(6, 6))

sns.heatmap(cm, annot=True, fmt='d',
cmap='Blues')

plt.title('Confusion Matrix')

plt.ylabel('Actual')

plt.xlabel('Predicted')

plt.show()
```

d. ROC Curve and AUC

- **Description:**
  - The Receiver Operating Characteristic (ROC) curve shows the trade-off between True Positive Rate (TPR) and False Positive Rate (FPR) at various threshold levels.
  - **AUC (Area Under the Curve):**
    - AUC measures the entire area under the ROC curve. It provides a single number summarizing the model's ability to distinguish between positive and negative classes.
  - **Implementation:**

```
from sklearn.metrics import roc_curve, auc


# Get predicted probabilities

y_pred_prob = best_model.predict(X_test).ravel()


# Compute ROC curve

fpr, tpr, _ = roc_curve(y_test, y_pred_prob)

roc_auc = auc(fpr, tpr)


# Plot ROC curve

plt.figure()

plt.plot(fpr, tpr, color='blue', lw=2,
label=f'ROC curve (area = {roc_auc:.2f})')

plt.plot([0, 1], [0, 1], color='black', 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()
```

Results Obtained

- **Accuracy:** The model achieved an accuracy of **79%** on the test data. This indicates that 79% of the total predictions made by the model were correct.
- **Precision, Recall, and F1-Score:**
    - **Precision:** The precision score was **68%**, meaning that 68% of the predicted churn cases were actually correct.

    - **Recall:** The recall score was **32%**, showing that the model correctly identified 32% of the total churn cases.
    - **F1-Score:** The F1-score was **43%**, reflecting the balance between precision and recall.
- **Confusion Matrix:**
    - The confusion matrix provided a detailed breakdown of the predictions:
        - **True Negatives (TN):** 978 customers were correctly predicted to not churn.
        - **False Positives (FP):** 57 customers were incorrectly predicted to churn.
        - **False Negatives (FN):** 253 customers were incorrectly predicted to not churn.
        - **True Positives (TP):** 121 customers were correctly predicted to churn.
- **ROC Curve and AUC:**
    - The ROC curve demonstrated that the model had an AUC of **0.79**, indicating good performance in distinguishing between churn and non-churn customers.

Challenges Faced

- **Data Imbalance:**
    - The dataset might have been imbalanced with more non-churn cases than churn cases, which could affect the performance of the model, particularly for recall. This is evident from the relatively low recall score (32%).
    - **Proposed Solution:** Techniques such as oversampling the minority class (churn cases) or applying class weights during model training could help improve the model's ability to detect churn cases.
- **Threshold Adjustment:**
    - Using a default threshold of 0.5 for classification may not always lead to optimal results, particularly for recall. A more tailored threshold based on business requirements (e.g., prioritizing recall to minimize customer churn) could be explored.

Conclusion

In this task, the trained ANN model was evaluated using various metrics to assess its performance. While the accuracy and precision scores are reasonably good, the recall score suggests that the model struggles to identify churn cases effectively. This may be due to data imbalance or the choice of threshold for binary classification.

The ROC curve and AUC score of 0.79 indicate that the model has a good ability to distinguish between churned and non-churned customers. However, further improvements, such as handling data imbalance and optimizing the classification threshold, could enhance its performance.