Task 3: Predicting the Model on the Provided Dataset

Objective

The main aim of this task is to predict customer churn using the trained Artificial Neural Network (ANN) model on unseen test dataset. After predictions were made, the results are stored for later analysis and evaluation.

Steps Taken

1. Loading the Trained Model

- **Loading the Best Model:**
  - After training, the best model (saved as `best_model.h5`) was loaded from disk.
  - **Why we loaded this model:**
    - The saved model represents the best-performing ANN during training. By loading it, we ensure that we are using the version of the model that has the highest validation accuracy and lowest validation loss.
  - **Implementation:**

```
from tensorflow.keras.models import load_model



# Load the saved model

best_model =
load_model('path_to_saved_model/best_model.h5')
```

    - **Path Configuration:** The path to the saved model was set using the `config.json` file, ensuring that file locations are dynamically configured based on the project's structure.

2. Making Predictions on Test Data

- **Loading Test Data:**
  - The test dataset was loaded from the path specified in the `config.json`. This dataset contains customers whose churn status was unknown to the model.

- **Why we used this test data:**
  - The test data was not used during training, making it suitable for evaluating how well the model generalizes to new, unseen data.

```
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
```

- **Prediction Process:**
  - The trained ANN model was used to make predictions on the test dataset. Since we were dealing with a binary classification problem (churn/no churn), the model returned probabilities indicating how likely each customer was to churn.
  - The prediction output is in the form of probabilities between 0 and 1. A threshold of **0.5** was applied to classify whether a customer churned (`1`) or not (`0`).
  - **Implementation:**

```
# Use the trained model to predict on the test
set

y_pred_prob = best_model.predict(X_test)



# Convert probabilities to binary classification
(0 or 1)

y_pred = (y_pred_prob > 0.5).astype('int')
```

- **Why we applied a threshold:**
  - Model predictions are probabilities. Here we are applying a threshold of 0.5 to classify customers which churned and not-churned If probabilities are over 0.5, customer is considered as churn and under that it will be not- churn case.

3. Saving Prediction Results

- **Saving Predictions to CSV:**
  - The predicted churn values were saved to a CSV file (`predictions.csv`) alongside the actual test data labels for comparison and further analysis.
  - **Implementation:**

```
# Create a DataFrame to store actual vs predicted
values

predictions_df = pd.DataFrame({

    'Actual': y_test,

    'Predicted': y_pred.flatten()

})



# Save the predictions to a CSV file

predictions_path = config['predictions_path']

predictions_df.to_csv(predictions_path,
index=False)



print(f'Predictions saved to {predictions_path}')
```

- **Why we saved predictions:**
  - Storing the predictions can be used for additional analysis — such as comparing model output to actual labels, and calculations of evaluation metrics including accuracy, precision & recall.

Results Obtained

- **Prediction Results:**
  - The ANN model predicted the churn status for each customer in the test dataset, returning the binary classification values (i.e. `1` for churn, `0` for no churn).
- **Files Generated:**
  - **predictions.csv**: Contains the predicted churn outcomes and actual churn labels, making it easy to analyze the accuracy of the model.

Sample of the CSV:

Actual	Predicted	Prediction_Probability
1	0	0.437518
0	0	0.183251
0	0	0.183251
1	1	0.756294
0	0	0.183251
1	0	0.207229
0	0	0.195859
0	0	0.183251
…		

- This CSV file will be used in the next task to evaluate the model's performance.

Challenges Faced

- **Prediction Threshold:**
  - It was one of the challenges to identify the right cut-off for classification. This is often a good default choice, but in some specific problems (e.g trying to minimize false positives or reducing harm that comes from being falsely negative) 0.5 might be less appropriate.
  - **Solution:** While 0.5 was used as the default threshold for binary classification, adjusting the threshold based on the project's priorities (e.g., sensitivity vs. specificity) could further improve the results.

- **Data Imbalance:**

- As we saw in earlier tasks, the data set could have an imbalance between non-churned customers and churned one. This might compromise the performance of a model at identifying churn cases (any type of doubt with those comprising relatively small percentages.)
- **Solution:** Techniques like oversampling, undersampling, or applying class weights could be explored in future iterations of the project to address this issue.

Conclusion

The predictive model successfully processed the test dataset and generated churn predictions for each customer. These predictions were saved for further analysis in the next step, where the model's performance will be evaluated using various metrics such as accuracy, precision, recall, and the ROC curve.

By saving the predictions, we now have a clear view of the model's output, which will guide us in evaluating its effectiveness and identifying areas for potential improvement.