

# Configuration Manual

MSc Research Project  
Programme Name

Kishore Nallasivam  
Student ID: X23205962

School of Computing  
National College of Ireland

Supervisor: William Clifford

**National College of Ireland**  
**MSc Project Submission Sheet**  
**School of Computing**



**Student Name:** Kishore Nallasivam  
**Student ID:** X23205962  
**Programme:** MSc. Data Analytics **Year:** 2024  
**Module:** Research project configuration manual  
**Lecturer:** William Clifford  
**Submission Due Date:** 12<sup>th</sup> December 2024  
**Project Title:** Advanced Visa Outcome Predictions for Superior Accuracy and Interpretability

**Word Count:** 869

**Page Count:** 12

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

**Signature:** Kishore Nallasivam

**Date:** 12<sup>th</sup> December 2024

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST**

Attach a completed copy of this sheet to each project (including multiple copies)	<input type="checkbox"/>
<b>Attach a Moodle submission receipt of the online project submission,</b> to each project (including multiple copies).	<input type="checkbox"/>
<b>You must ensure that you retain a HARD COPY of the project,</b> both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

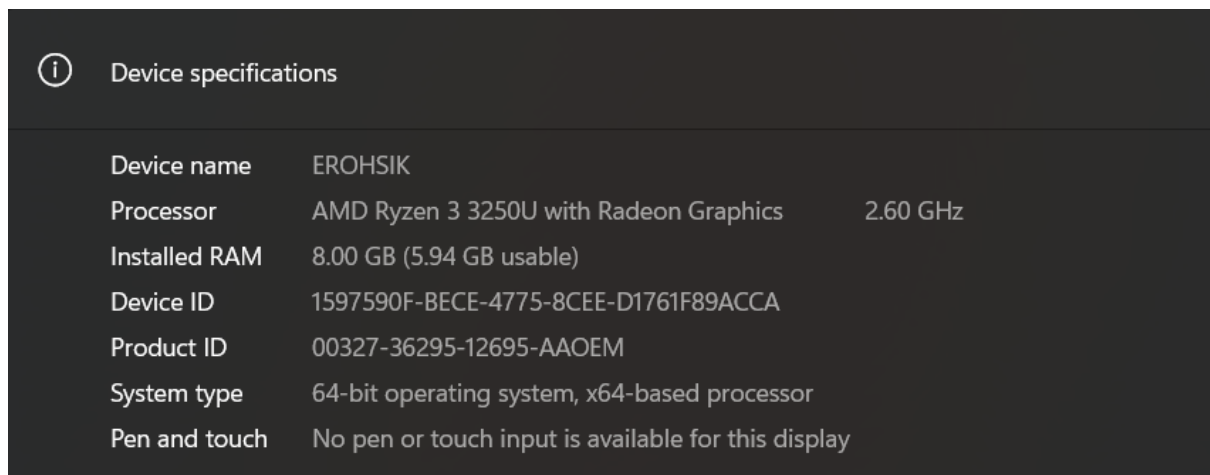
<b>Office Use Only</b>	
Signature:	
Date:	
Penalty Applied (if applicable):	

# Configuration Manual

Kishore Nallasivam  
X23205962

## 1. Hardware & Software

### 1.1 Device Specifications:

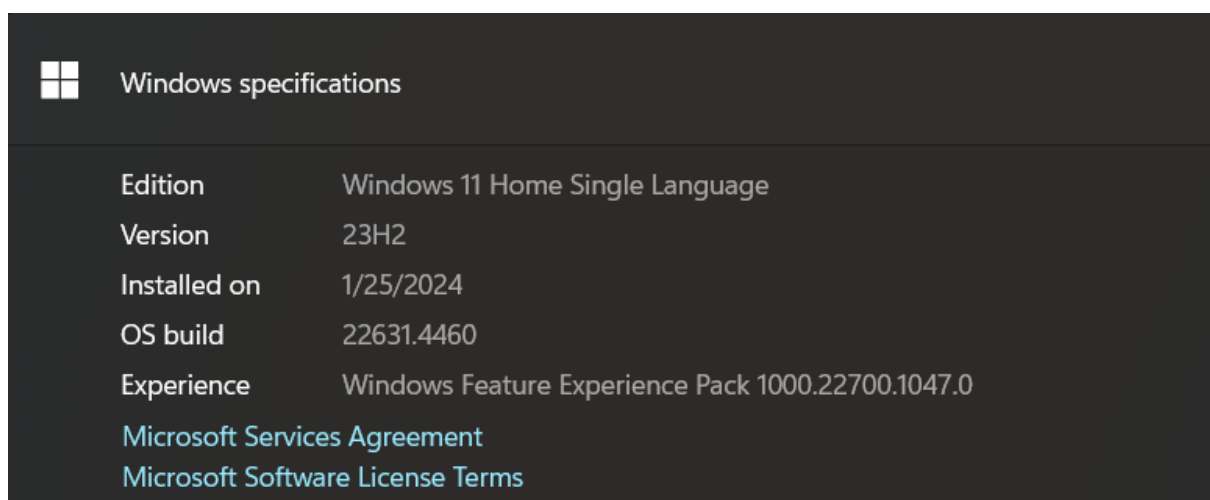


The screenshot shows the 'Device specifications' window in Windows. It contains a table with the following information:

Device name	EROHSIK		
Processor	AMD Ryzen 3 3250U with Radeon Graphics	2.60 GHz	
Installed RAM	8.00 GB (5.94 GB usable)		
Device ID	1597590F-BECE-4775-8CEE-D1761F89ACCA		
Product ID	00327-36295-12695-AAOEM		
System type	64-bit operating system, x64-based processor		
Pen and touch	No pen or touch input is available for this display		

**Figure:1- Device Specification**

### 1.2 Windows Specification



The screenshot shows the 'Windows specifications' window in Windows Settings. It contains a table with the following information:

Edition	Windows 11 Home Single Language
Version	23H2
Installed on	1/25/2024
OS build	22631.4460
Experience	Windows Feature Experience Pack 1000.22700.1047.0
<a href="#">Microsoft Services Agreement</a>	
<a href="#">Microsoft Software License Terms</a>	

**Figure:2- Windows Specification**

## 1.3 Software Requirements

The software tools which we have used in this study are Anaconda Navigator, Jupyter Notebook and Python.

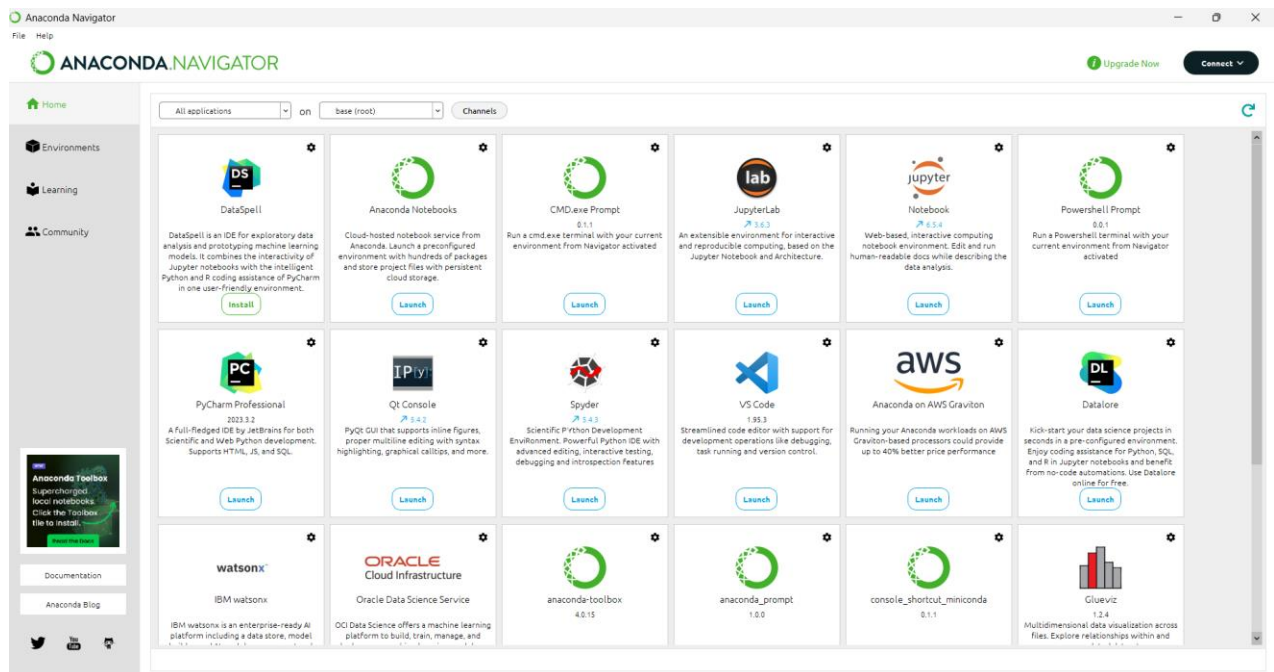


Figure -3: Anaconda Navigator

## 2. Data preprocessing

This part involves the various processes which involved in data preparation for the machine learning and deep learning model.

The data comprised “H-1B, H-1B1, E-3 Visa Petitions 2017 – 2022” applications collected from Kaggle for the years 2017 to 2022. The raw data

was provided in CSV format, including variables such as Visa Type, Employer Name and Visa Case Status.

	Visa_Class	Employer_Name	SOC_Title	Job_Title	Full_Time_Position	Worksite	Prevailing_Wage	Unit_Of_Pay	Employer_Location	Employer_Country
0	H-1B	DISCOVER PRODUCTS INC.	Computer Systems Analysts	ASSOCIATE DATA INTEGRATION	Y	Riverwoods, Illinois	59197.0	Year	Riverwoods, Illinois	United States Of America
1	H-1B	DFS SERVICES LLC	Operations Research Analysts	SENIOR ASSOCIATE	Y	Riverwoods, Illinois	49800.0	Year	Riverwoods, Illinois	United States Of America
2	H-1B	EASTBANC TECHNOLOGIES LLC	Computer Programmers	.NET SOFTWARE PROGRAMMER	Y	Washington, District of Columbia	76502.0	Year	Washington, District of Columbia	United States Of America
3	H-1B	INFO SERVICES LLC	Computer Occupations, All Other	PROJECT MANAGER	Y	Jersey City, New Jersey	90376.0	Year	Livonia, Michigan	United States Of America
4	H-1B	BB&T CORPORATION	Credit Analysts	ASSOCIATE - ESOTERIC ASSET BACKED SECURITIES	Y	New York, New York	116605.0	Year	Wilson, North Carolina	United States Of America

**Figure- 5: H1B visa dataset**

## 2.1 Importing Libraries

```
#Importing libraries

import pandas as pd
import numpy as np
import os
import tensorflow as tf
import xgboost as xgb
import keras_tuner as kt
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

from sklearn.preprocessing import LabelEncoder, StandardScaler
from imblearn.under_sampling import RandomUnderSampler
from sklearn.model_selection import train_test_split
from tensorflow.keras import layers
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint, LearningRateScheduler
from tensorflow.keras.layers import Input, Dense, LSTM, Dropout, BatchNormalization, Bidirectional, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from sklearn.metrics import accuracy_score, f1_score, recall_score, confusion_matrix, classification_report
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, Dropout, Bidirectional, LSTM, Input
```

**Figure - 6: libraries**

## 2.2 Loading data

```
# Loading the datasets

H1B_visa_2022 = pd.read_csv(r"C:\Users\Kisho\Downloads\archive(20)\LCA_FY_2022.csv")
H1B_visa_2021 = pd.read_csv(r"C:\Users\Kisho\Downloads\archive(20)\LCA_FY_2021.csv")
H1B_visa_2020 = pd.read_csv(r"C:\Users\Kisho\Downloads\archive(20)\LCA_FY_2020.csv")
H1B_visa_2019 = pd.read_csv(r"C:\Users\Kisho\Downloads\archive(20)\LCA_FY_2019.csv")
H1B_visa_2018 = pd.read_csv(r"C:\Users\Kisho\Downloads\archive(20)\LCA_FY_2018.csv")
H1B_visa_2017 = pd.read_csv(r"C:\Users\Kisho\Downloads\archive(20)\LCA_FY_2017.csv")
```

```
# Adding 'Year' column to each dataset
```

```
H1B_visa_2022['Year'] = 2022
H1B_visa_2021['Year'] = 2021
H1B_visa_2020['Year'] = 2020
H1B_visa_2019['Year'] = 2019
H1B_visa_2018['Year'] = 2018
H1B_visa_2017['Year'] = 2017
```

**Figure – 7: Python code for Loading Data**

### 3. Data Visualization

As for the distribution of the visa case status we can see that 95% of all identified applications are Certified with more than 3.7 million approved applications. In the Denied status, there were almost 1,25,485 cases. suggesting that H1B visa applications are not frequently rejected by the American government.

```
# Getting the counts of each Visa Case Status
visa_case_counts = combined_df['Visa Case Status'].value_counts().reset_index()
visa_case_counts.columns = ['Visa Case Status', 'Count']

# Create the bar plot
Visa_Case_Status = px.bar(
    visa_case_counts,
    x='Visa Case Status', y='Count',
    color='Visa Case Status', color_continuous_scale='RdBu',
    title='Distribution of Visa Case Status'
)

# Layout and format of the plot
Visa_Case_Status.update_layout(
    xaxis_title='Visa Case Status', yaxis_title='Count',
    title_font_size=16, xaxis_tickangle=-45,
    template='plotly_white').update_yaxes(tickformat=',').update_traces(texttemplate='%{y:,}', textposition='outside')

Visa_Case_Status.show()
```

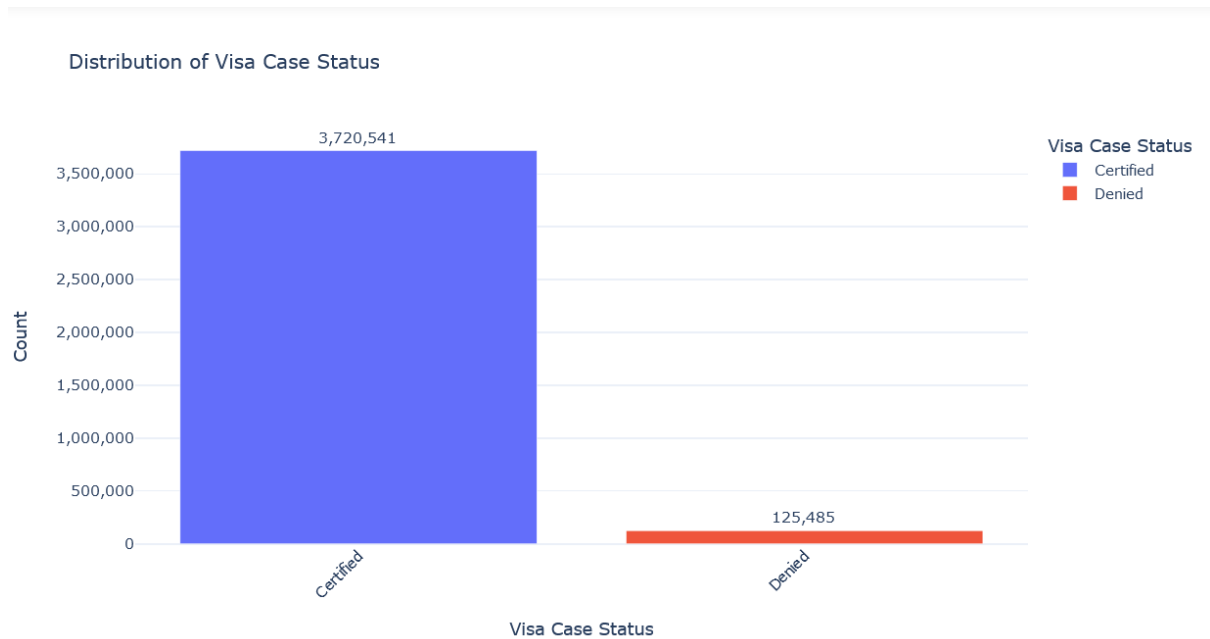


Figure – 8: Python Code to Show visualization

### 3.1 Train - Test Split

```
# Train-test split with stratification
X_train, X_temp, y_train, y_temp = train_test_split(
    X_scaled, y, test_size=0.3, random_state=42, stratify=y)
X_val, X_test, y_val, y_test = train_test_split(
    X_temp, y_temp, test_size=0.5, random_state=42, stratify=y_temp)
```

Figure – 9: Python Code for Train and Test Split

### 3.2 Scaling

StandardScaler was used to normalize the feature values as there was a need to make sure that it had zero mean and unit variance.

```
: # scale features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Figure -10: Python code for scaling

### 3.3 UnderSampling

```
# Function to balance the data using undersampling
def balance_data_with_undersampling(X, y):
    undersampler = RandomUnderSampler(random_state=42)
    X_balanced, y_balanced = undersampler.fit_resample(X, y)
    return X_balanced, y_balanced

# Apply undersampling
X_train_balanced, y_train_balanced = balance_data_with_undersampling(X_train, y_train)
```

Figure -11: Python code for UnderSampling

### 3.4 Bi-LSTM model

Bi-LSTM Layers: Two bidirectional LSTM layers were added to capture long-range needs in both forward and backward directions, The first Bi-LSTM layer has a tunable number of units (lstm\_units1) with a dropout rate (dropout\_rate1). The second Bi-LSTM layer also has tunable units (lstm\_units2) and dropout (dropout\_rate2).



```

# Build BiLSTM model with hyperparameters to tune
def build_bilstm_model(hp, input_dim):
    model = Sequential()

    # Define Input Layer
    model.add(Input(shape=(input_dim, 1))) # Specify input shape here

    # BiLSTM Layer 1
    model.add(Bidirectional(LSTM(hp.Int('lstm_units1', min_value=32, max_value=128, step=32), return_sequences=True)))
    model.add(Dropout(hp.Float('dropout_rate1', min_value=0.2, max_value=0.5, step=0.1)))

    # BiLSTM Layer 2
    model.add(Bidirectional(LSTM(hp.Int('lstm_units2', min_value=16, max_value=64, step=16))))
    model.add(Dropout(hp.Float('dropout_rate2', min_value=0.2, max_value=0.5, step=0.1)))

    # Dense Layer
    model.add(Dense(hp.Int('dense_units', min_value=32, max_value=128, step=32), activation='relu'))
    model.add(Dropout(hp.Float('dropout_rate3', min_value=0.2, max_value=0.5, step=0.1)))

    # Output Layer
    model.add(Dense(1, activation='sigmoid'))

    # Compile the model
    model.compile(
        optimizer=tf.keras.optimizers.Adam(
            hp.Choice('learning_rate', values=[1e-3, 1e-4, 1e-5])),
        loss='binary_crossentropy',
        metrics=['accuracy']
    )

    return model

```

Figure – 12: Bi-LSTM Model code

### 3.5 Hyperparameter Tuning

We used Keras Tuner with a RandomSearch strategy to optimize the following hyperparameters, Number of units in the first and second Bi-LSTM layers (lstm\_units1, lstm\_units2). Number of units in the dense layer coined as dense\_units. Learning\_rate for of Adam optimizer. In total 8 trials were run with 15 epochs each though early stopping with validation loss.

```

# Keras Tuner for hyperparameter tuning
tuner = kt.RandomSearch(
    lambda hp: build_bilstm_model(hp, input_dim),
    objective='val_accuracy',
    max_trials=8,
    executions_per_trial=1,
    directory='bilstm_tuning',
    project_name='bilstm_hyperparameter_tuning',
    overwrite=True
)

# Early stopping to avoid overfitting during tuning
early_stop_tuning = EarlyStopping(monitor='val_loss', patience=3)

# Perform hyperparameter search
tuner.search(
    X_train_lstm, y_train_balanced,
    validation_data=(X_val_lstm, y_val),
    epochs=15,
    batch_size=64,
    callbacks=[early_stop_tuning]
)

# Get the best hyperparameters
best_hps = tuner.get_best_hyperparameters(num_trials=1)[0]
print("Best Hyperparameters:", best_hps.values)

```

Figure – 13: Code for Hyperparameter Tuning

### 3.6 Evaluation

The Bi-LSTM model was evaluated using the test set, with key metrics including accuracy, recall, F1-score, and a confusion matrix. Validation accuracy during tuning: 66.65%. The learning curves indicate overfitting after the second epoch, as validation loss increased while training accuracy improved.

```

Trial 8 Complete [00h 12m 35s]
val_accuracy: 0.6261648535728455

Best val_accuracy So Far: 0.6665840148925781
Total elapsed time: 02h 50m 00s
Best Hyperparameters: {'lstm_units1': 128, 'dropout_rate1': 0.2, 'lstm_units2': 32, 'dropout_rate2': 0.2, 'dense_units': 64, 'dropout_rate3': 0.2, 'learning_rate': 0.001}
Epoch 1/20
2745/2745 ————— 240s 84ms/step - accuracy: 0.5720 - loss: 0.6790 - val_accuracy: 0.6101 - val_loss: 0.6764
Epoch 2/20
2745/2745 ————— 233s 85ms/step - accuracy: 0.5773 - loss: 0.6761 - val_accuracy: 0.5947 - val_loss: 0.6530
Epoch 3/20
2745/2745 ————— 262s 85ms/step - accuracy: 0.5850 - loss: 0.6728 - val_accuracy: 0.6588 - val_loss: 0.6364
Epoch 4/20
2745/2745 ————— 242s 88ms/step - accuracy: 0.5873 - loss: 0.6690 - val_accuracy: 0.5773 - val_loss: 0.6465
Epoch 5/20
2745/2745 ————— 256s 86ms/step - accuracy: 0.5903 - loss: 0.6674 - val_accuracy: 0.5445 - val_loss: 0.6686
Epoch 6/20
2745/2745 ————— 232s 84ms/step - accuracy: 0.5938 - loss: 0.6644 - val_accuracy: 0.5233 - val_loss: 0.6975
18029/18029 ————— 172s 10ms/step - accuracy: 0.6579 - loss: 0.6367
18029/18029 ————— 169s 9ms/step
Confusion Matrix:
[[369986 188095]
 [ 9272  9551]]
Accuracy: 0.6578858874266775
Weighted F1 Score: 0.7665607061887559
Recall: 0.5074111459384795

```

Figure – 14: Evaluation result for Bi-LSTM

### 3.7 Python Code to build the XGBOOST Model:

The model was trained on the balanced training set and on the validation set during the training time. Early stopping was used using validation loss to avoid overfitting Training and validation log loss values were saved at every epoch.

```
# XGBoost model with eval_metric included in initialization
xgb_model = xgb.XGBClassifier(
    objective='binary:logistic',
    use_label_encoder=False,
    random_state=42,
    n_estimators=100,
    max_depth=5,
    learning_rate=0.1,
    subsample=0.8,
    colsample_bytree=0.8,
    eval_metric='logloss')

# Train the XGBoost model
xgb_model.fit(
    X_train_balanced, y_train_balanced,
    eval_set=[(X_train_balanced, y_train_balanced), (X_val, y_val)],
    verbose=True)

# Evaluate the model on the test set
y_test_pred = xgb_model.predict(X_test)
test_accuracy = accuracy_score(y_test, y_test_pred)
conf_matrix = confusion_matrix(y_test, y_test_pred)

print("Test Metrics:")
print("Confusion Matrix:\n", conf_matrix)
print(f"Accuracy: {test_accuracy}")

# Extract evaluation metrics from the training process
results = xgb_model.evals_result()
```

Figure – 15: XGBoost Model

### 3.8 Evaluation

Both log loss for training and validation reduced from epoch to epoch, demonstrating effective learning over epochs. Training log loss: 0.63667 at the final epoch. Validation log loss: 0.64276 at the final

epoch. Training vs validation accuracy curves show good convergence, with no evidence of overfitting. The XGBoost model was evaluated on the test set, yielding the following results of Accuracy: 60.39%

```
Test Metrics:  
Confusion Matrix:  
[[336381 221700]  
 [ 6819 12004]]  
Accuracy: 0.6038873018734486
```

Figure – 16: Result

### 3.9 Comparison on both models

The results obtained from the XGBoost model are slightly lower than that of the Bi-LSTM model, accuracy which is 60.39% and 65.79% respectively. However, the training of the XGBoost model was faster and computationally less complex because of the basic design structure. Both models Bi-LSTM and XGBoost faced challenges in correctly identifying the minority class, indicating the need for advanced balancing techniques.