Configuration Manual

MSc Research Project
Data Analytics

Forename Surname
Student ID: x23407719

School of Computing
National College of Ireland

Supervisor: Jorge Basilio

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Saleem Pasha Shaik		
Student ID:	23407719		
Programme:	MSc Data Analytics	**Year:**	Jan 2025
Module:	MSc Research Practicum Part		
Lecturer:	Jorge Basilio		
Submission Due Date:	11/12/2025		
Project Title:	Innovative Use of Behavioural Data and Explainable AI for Early Gambling Addiction Detection		
Word Count:	1178	**Page Count:**	11

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: Saleem Pasha Shaik

Date: 11/12/2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

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

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

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

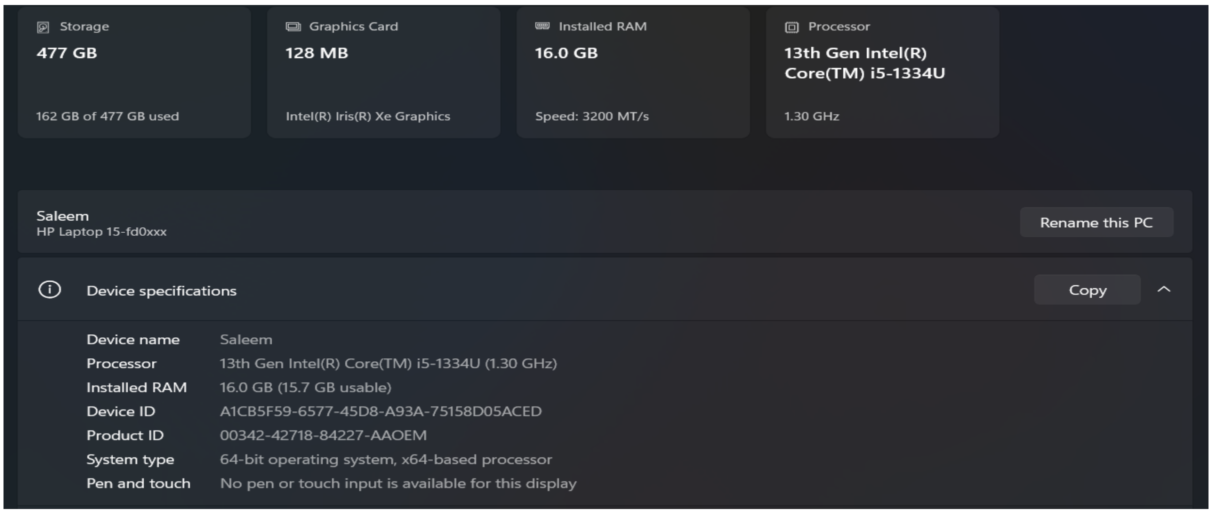
Saleem Pasha Shaik
Student ID: x23407719

1 Introduction

This configuration manual aims to describe the complete system setup used for the implementation and evaluation of the gambling addiction detection framework. The hardware resources, the software environment, the Python packages, and the project structure are defined, allowing other researchers and practitioners to replicate the modelling pipeline correctly. The manual also describes the structure of the project directory, how the dataset is organized, and the flow of execution with respect to every module used in the analysis. Additionally, it provides instructions on how to run the clustering algorithms, train the forecasting models, and explain the AI outputs using LIME, in support of full reproducibility of the results. To maintain accuracy, this manual outlines the exact library versions and dependencies used during experimentation. In this configuration manual, any user will be able to set up the system, execute the complete pipeline, and achieve results that align with those reported in the study.

2 System Configuration

Setting up the environment correctly is crucial for ensuring reproducibility and correct execution of the notebook and model training pipeline. This section details the computing environment, including the hardware and software utilized for running the study and develop. As from the figure, the computational setup consisted of an Intel i5 13th-generation CPU, 16GB RAM, a 64-bit Windows 11 environment, and a 512GB SSD for storage.



Storage	Graphics Card	Installed RAM	Processor
477 GB	128 MB	16.0 GB	13th Gen Intel(R) Core(TM) i5-1334U
162 GB of 477 GB used	Intel(R) Iris(R) Xe Graphics	Speed: 3200 MT/s	1.30 GHz

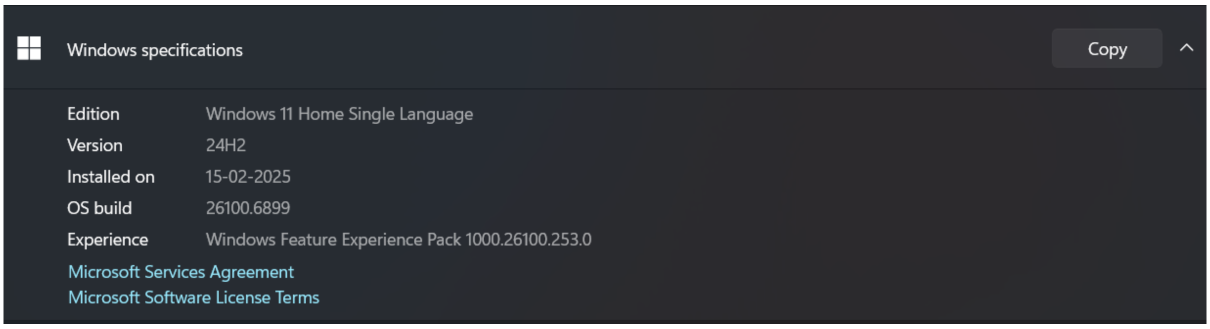
Saleem
HP Laptop 15-fd0xxx

Rename this PC

Device specifications

Copy

Device name	Saleem
Processor	13th Gen Intel(R) Core(TM) i5-1334U (1.30 GHz)
Installed RAM	16.0 GB (15.7 GB usable)
Device ID	A1CB5F59-6577-45D8-A93A-75158D05ACED
Product ID	00342-42718-84227-AAOEM
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display



Edition	Windows 11 Home Single Language
Version	24H2
Installed on	15-02-2025
OS build	26100.6899
Experience	Windows Feature Experience Pack 1000.26100.253.0

Microsoft Services Agreement
Microsoft Software License Terms

2.1 Recommended Hardware

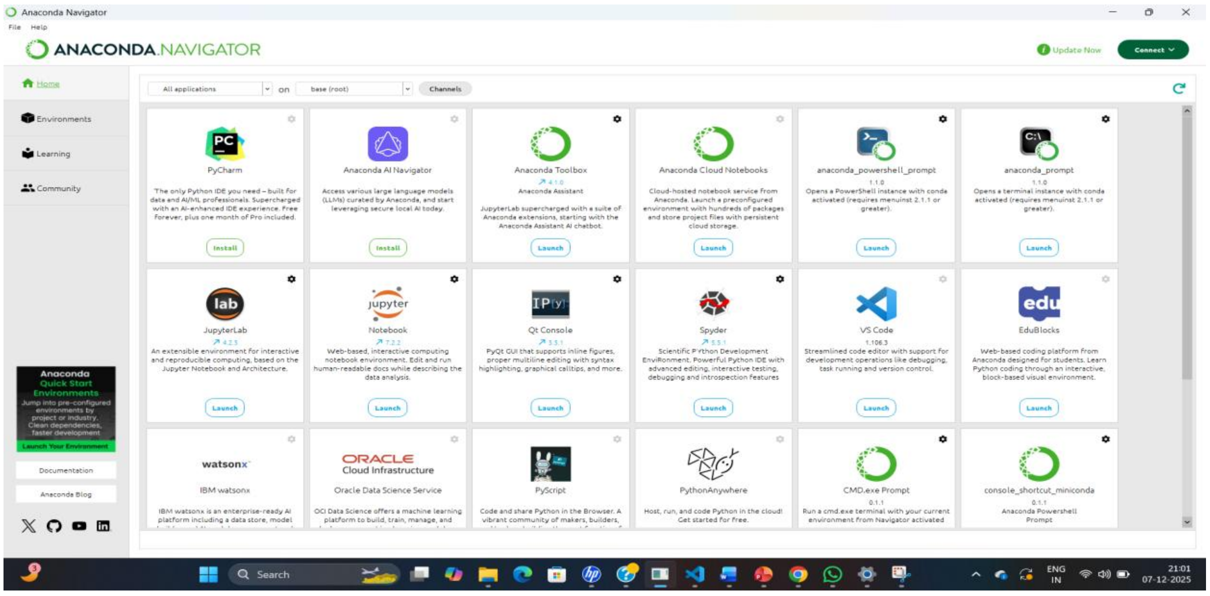
Component	Recommended
GPU	NVIDIA RTX 3060 / T4 / A100 (training benefits from CUDA)
RAM	Minimum 16 GB, recommended 32 GB
CPU	Quad-core or better
Storage	10–20 GB for datasets & models

Deep learning models, especially attention-based ones, perform significantly faster on a dedicated GPU.

2.2 Software Requirements

Operating System

- Windows 10/11
- Anaconda Navigator
- Jupiter Notebook 7.2.2
- Linux (Ubuntu highly recommended)



PyCharm
The only Python IDE you need – built for data and AI/ML professionals. Supercharged with an AI-enhanced IDE experience. Free forever, plus one month of Pro included.
Install

Anaconda AI Navigator
Access various large language models (LLMs) curated by Anaconda, and start leveraging secure local AI today.
Install

Anaconda Toolbox
4.1.0
Anaconda Assistant
JupyterLab supercharged with a suite of Anaconda extensions, starting with the Anaconda Assistant AI chatbot.
Launch

Anaconda Cloud Notebooks
Cloud-hosted notebook service from Anaconda. Launch a preconfigured environment with hundreds of packages and store project files with persistent cloud storage.
Launch

anaconda_powershell_prompt
1.1.0
Opens a PowerShell instance with conda activated (requires menuinst 2.1.1 or greater).
Launch

anaconda_prompt
1.1.0
Opens a terminal instance with conda activated (requires menuinst 2.1.1 or greater).
Launch

JupyterLab
4.2.5
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
Launch

Notebook
7.2.2
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch

Qt Console
5.5.1
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.
Launch

Spyder
5.5.1
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features
Launch

VS Code
1.106.3
Streamlined code editor with support for development operations like debugging, task running and version control.
Launch

EduBlocks
Web-based coding platform from Anaconda designed for students. Learn Python coding through an interactive, block-based visual environment.
Launch

IBM watsonx
IBM watsonx is an enterprise-ready AI platform including a data store, model

Oracle Data Science Service
OCI Data Science offers a machine learning platform to build, train, manage, and

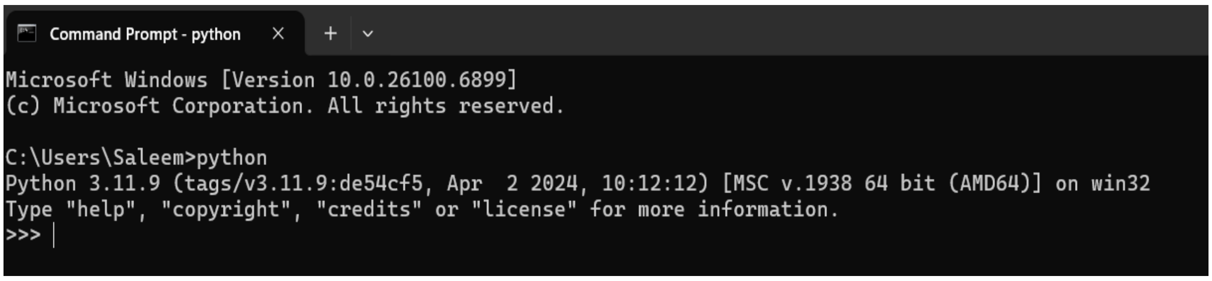
PyScript
Code and share Python in the Browser. A vibrant community of makers, builders,

PythonAnywhere
Host, run, and code Python in the cloud! Get started for free.

CMD.exe Prompt
0.1.1
Run a cmd.exe terminal with your current environment from Navigator activated

console_shortcut_miniconda
0.1.1
Anaconda Powershell Prompt

2.3 Python Version



Command Prompt - python

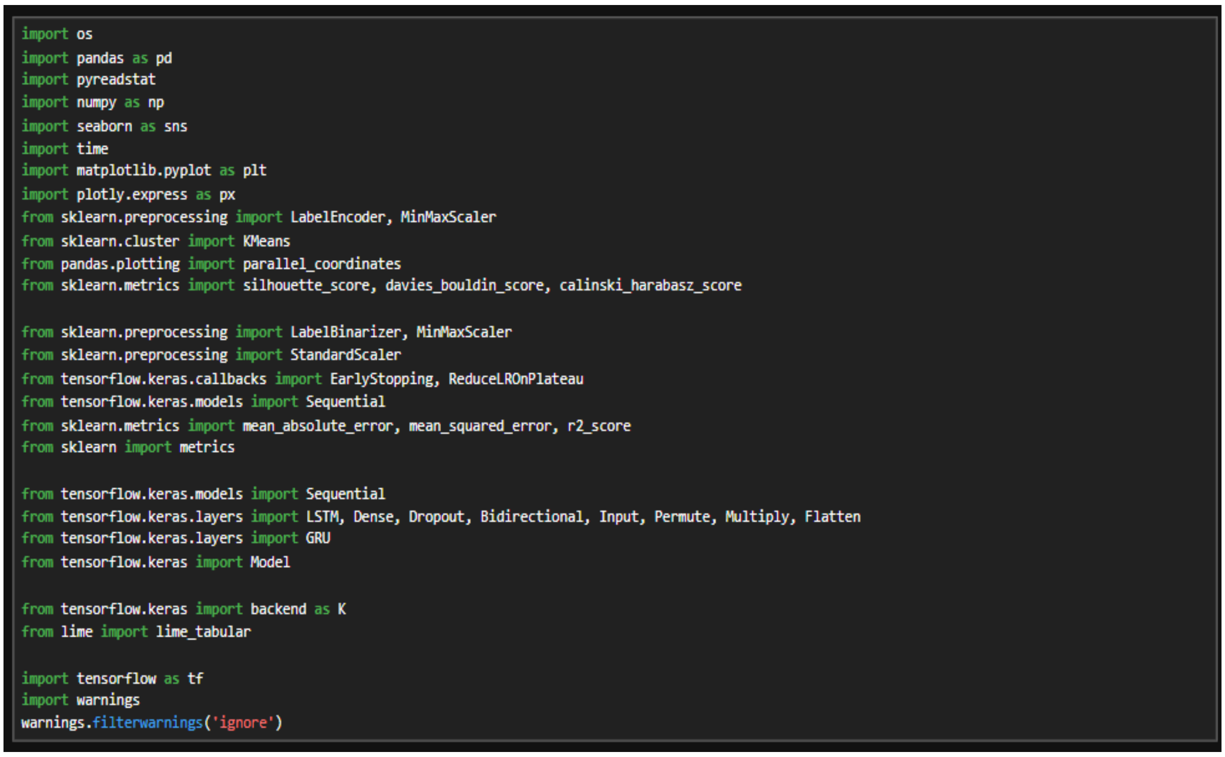
```
Microsoft Windows [Version 10.0.26100.6899]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Saleem>python
Python 3.11.9 (tags/v3.11.9:de54cf5, Apr  2 2024, 10:12:12) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> |
```

From above screenshot, we can see the version of python which is used for this project.Python 3.10 **or** 3.11 recommended

2.1 Required Libraries

The project uses the following major libraries:

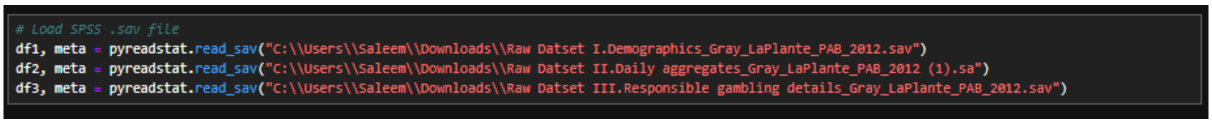
Category	**Libraries Used**
Core Scientific Libraries	numpy, pandas, matplotlib, seaborn, scipy, statsmodels
Machine Learning / Deep Learning	scikit-learn, tensorflow, keras, keras_cv_attention_models, joblib
Explainable AI	lime
System Utilities	os, warnings, datetime, time
Visualization	matplotlib.pyplot, seaborn



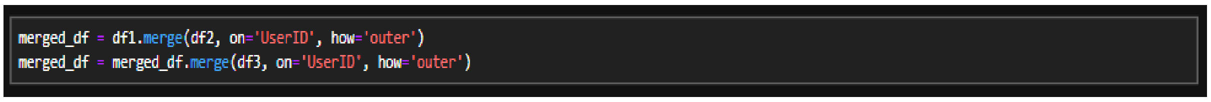
```python
import os
import pandas as pd
import pyreadstat
import numpy as np
import seaborn as sns
import time
import matplotlib.pyplot as plt
import plotly.express as px
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.cluster import KMeans
from pandas.plotting import parallel_coordinates
from sklearn.metrics import silhouette_score, davies_bouldin_score, calinski_harabasz_score

from sklearn.preprocessing import LabelBinarizer, MinMaxScaler
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.models import Sequential
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn import metrics

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout, Bidirectional, Input, Permute, Multiply, Flatten
from tensorflow.keras.layers import GRU
from tensorflow.keras import Model

from tensorflow.keras import backend as K
from lime import lime_tabular

import tensorflow as tf
import warnings
warnings.filterwarnings('ignore')
```

3 Data Sources & Linking

The project uses three datasets: demographics, betting history, and responsible-gambling interventions. The datasets chosen for the project Transparency Project and the title of the data is: Behavioral Characteristics of Internet Gamblers Who Trigger Corporate Responsible Gambling Interventions[2] (1) . All datasets are merged using "**UserID**" to create unified player records. An **outer join** ensures no player is excluded



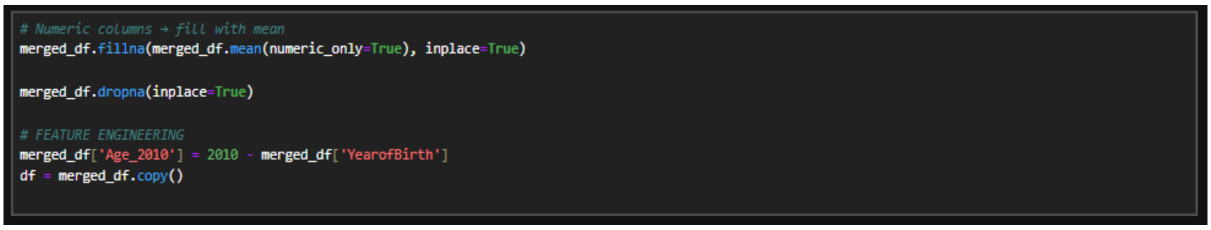
```
# Load SPSS .sav file
df1, meta = pyreadstat.read_sav("C:\\Users\\Saleem\\Downloads\\Raw Datset I.Demographics_Gray_LaPlante_PAB_2012.sav")
df2, meta = pyreadstat.read_sav("C:\\Users\\Saleem\\Downloads\\Raw Datset II.Daily aggregates_Gray_LaPlante_PAB_2012 (1).sa")
df3, meta = pyreadstat.read_sav("C:\\Users\\Saleem\\Downloads\\Raw Datset III.Responsible gambling details_Gray_LaPlante_PAB_2012.sav")
```



```
merged_df = df1.merge(df2, on='UserID', how='outer')
merged_df = merged_df.merge(df3, on='UserID', how='outer')
```

4 Data Cleaning & Feature Engineering

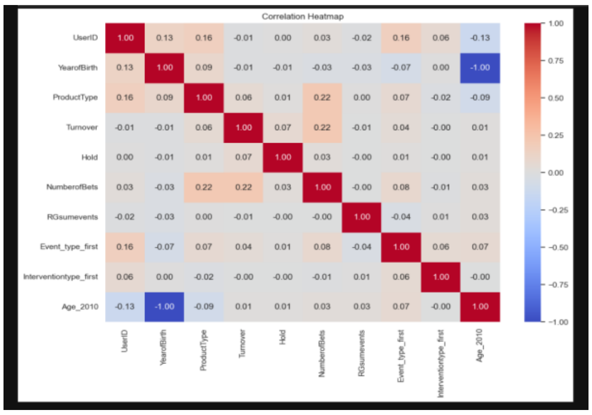
Missing values, inconsistent timestamps, and invalid numeric fields are corrected. Additional features like turnover mean, volatility, intervention counts, and recency are engineered. These enriched features support meaningful clustering.



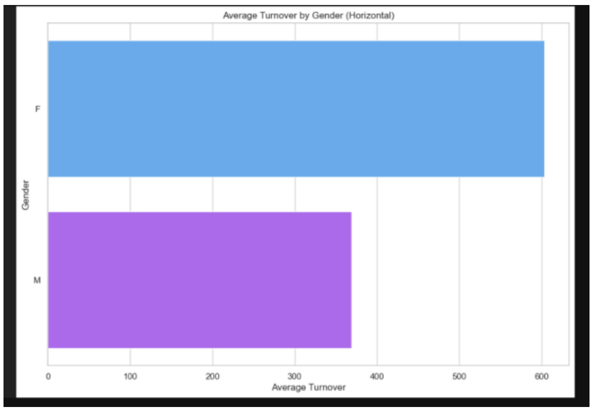
```
# Numeric columns → fill with mean
merged_df.fillna(merged_df.mean(numeric_only=True), inplace=True)

merged_df.dropna(inplace=True)

# FEATURE ENGINEERING
merged_df['Age_2010'] = 2010 - merged_df['YearofBirth']
df = merged_df.copy()
```

5 Data Visualization

Basic visual analysis is performed to understand behavioural patterns. Plots include distributions, correlations, time-series trends, and intervention frequencies. These visuals reveal anomalies and support early insights.

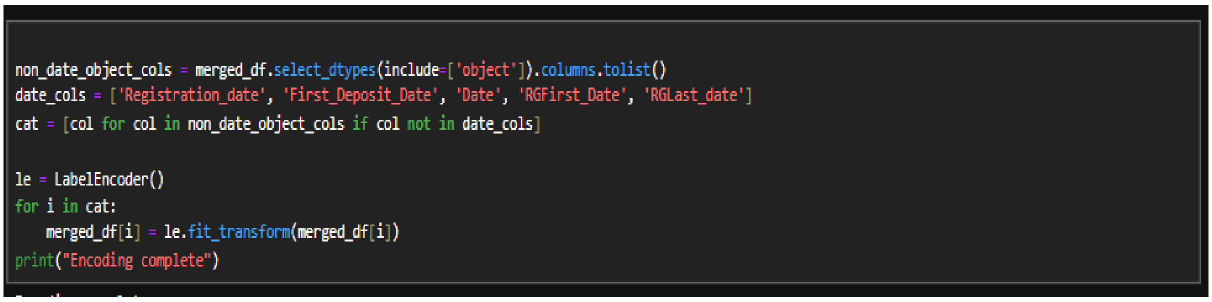


	UserID	YearofBirth	ProductType	Turnover	Hold	NumberofBets	RGsumevents	Event_type_first	Interventiontype_first	Age_2010
UserID	1.00	0.13	0.16	-0.01	0.00	0.03	-0.02	0.16	0.06	-0.13
YearofBirth	0.13	1.00	0.09	-0.01	-0.01	-0.03	-0.03	-0.07	0.00	-1.00
ProductType	0.16	0.09	1.00	0.06	0.01	0.22	0.00	0.07	-0.02	-0.09
Turnover	-0.01	-0.01	0.06	1.00	0.07	0.22	-0.01	0.04	-0.00	0.01
Hold	0.00	-0.01	0.01	0.07	1.00	0.03	-0.00	0.01	-0.00	0.01
NumberofBets	0.03	-0.03	0.22	0.22	0.03	1.00	-0.00	0.08	-0.01	0.03
RGsumevents	-0.02	-0.03	0.00	-0.01	-0.00	-0.00	1.00	-0.04	0.01	0.03
Event_type_first	0.16	-0.07	0.07	0.04	0.01	0.08	-0.04	1.00	0.06	0.07
Interventiontype_first	0.06	0.00	-0.02	-0.00	-0.00	-0.01	0.01	0.06	1.00	-0.00
Age_2010	-0.13	-1.00	-0.09	0.01	0.01	0.03	0.03	0.07	-0.00	1.00



6 Data Preprocessing

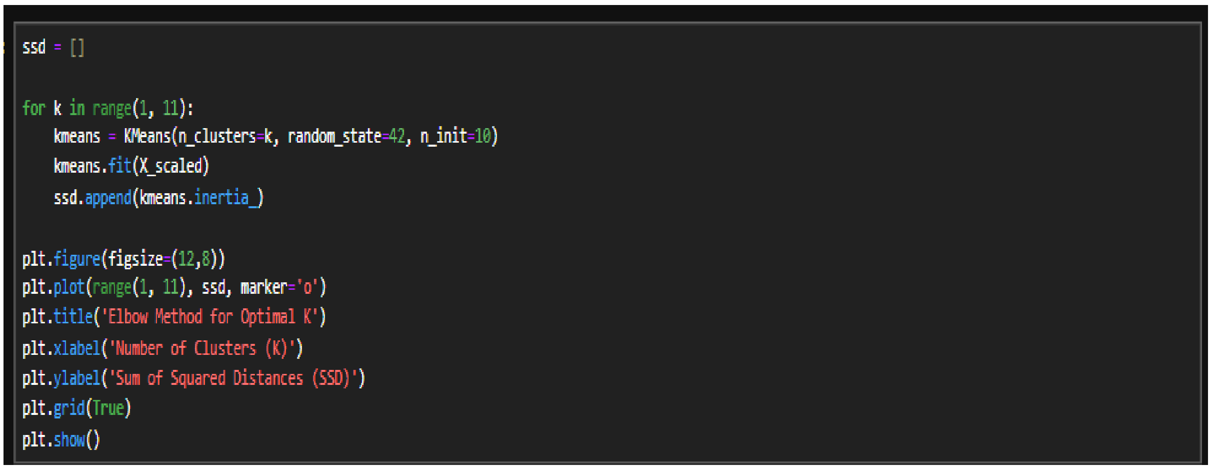
Categorical variables are label encoded while continuous behavioural features are scaled. Invalid or incomplete records are removed to ensure consistent clustering inputs. The final dataset contains clean, numerical features.



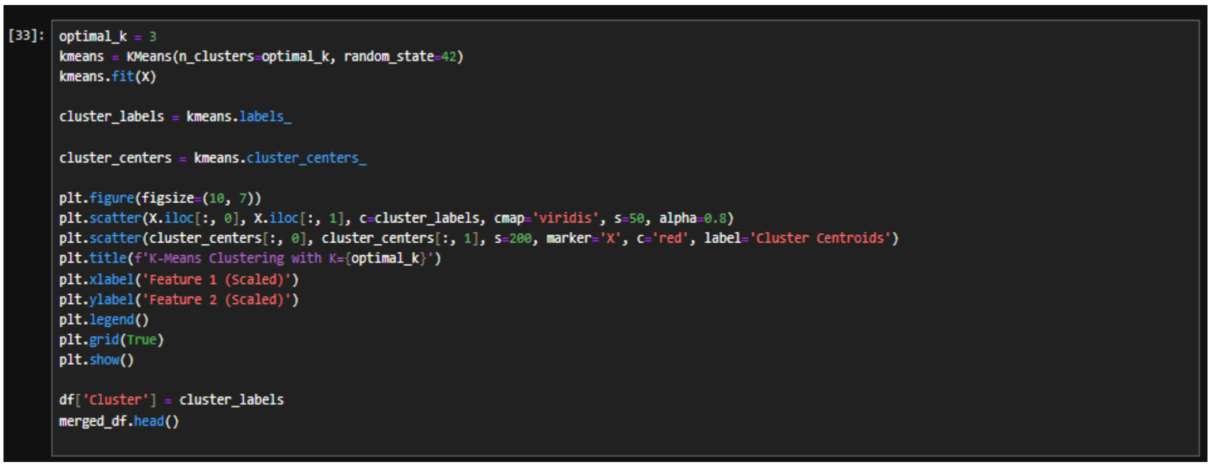
```
non_date_object_cols = merged_df.select_dtypes(include=['object']).columns.tolist()
date_cols = ['Registration_date', 'First_Deposit_Date', 'Date', 'RGFirst_Date', 'RGLast_date']
cat = [col for col in non_date_object_cols if col not in date_cols]

le = LabelEncoder()
for i in cat:
    merged_df[i] = le.fit_transform(merged_df[i])
print("Encoding complete")
```

7 Clustering Methodology (K-Means)

K-Means (k=3) segments players into casual, moderate-risk, and high-risk groups. Inputs include turnover statistics, volatility, interventions, and recency. Each user receives a cluster label for downstream analysis.



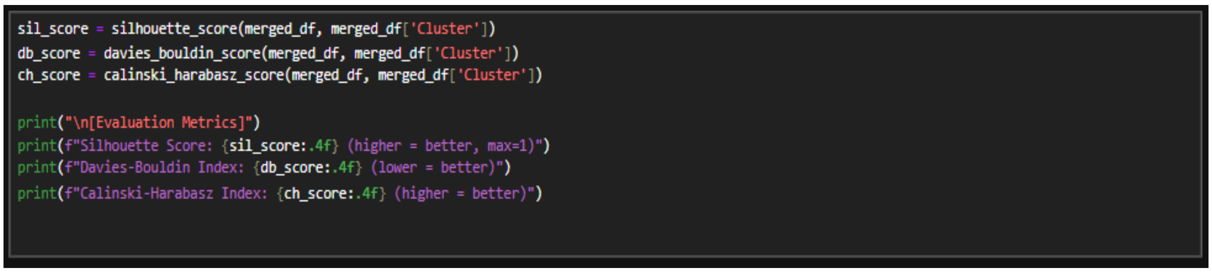
```
ssd = []

for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(X_scaled)
    ssd.append(kmeans.inertia_)

plt.figure(figsize=(12,8))
plt.plot(range(1, 11), ssd, marker='o')
plt.title('Elbow Method for Optimal K')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Sum of Squared Distances (SSD)')
plt.grid(True)
plt.show()
```



```
[33]: optimal_k = 3
      kmeans = KMeans(n_clusters=optimal_k, random_state=42)
      kmeans.fit(X)

      cluster_labels = kmeans.labels_

      cluster_centers = kmeans.cluster_centers_

      plt.figure(figsize=(10, 7))
      plt.scatter(X.iloc[:, 0], X.iloc[:, 1], c=cluster_labels, cmap='viridis', s=50, alpha=0.8)
      plt.scatter(cluster_centers[:, 0], cluster_centers[:, 1], s=200, marker='X', c='red', label='Cluster Centroids')
      plt.title(f'K-Means Clustering with K={optimal_k}')
      plt.xlabel('Feature 1 (Scaled)')
      plt.ylabel('Feature 2 (Scaled)')
      plt.legend()
      plt.grid(True)
      plt.show()

      df['Cluster'] = cluster_labels
      merged_df.head()
```

8 Cluster Evaluation Metrics

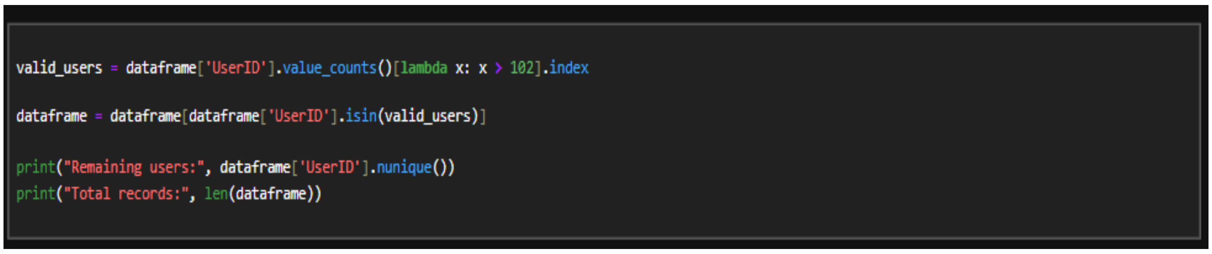
Silhouette Score, DBI, CH Index, and the Elbow Method are used to validate clustering quality. These metrics help verify separation between risk-based player groups. Higher silhouette and CH, lower DBI indicate good clustering.



```python
sil_score = silhouette_score(merged_df, merged_df['Cluster'])
db_score = davies_bouldin_score(merged_df, merged_df['Cluster'])
ch_score = calinski_harabasz_score(merged_df, merged_df['Cluster'])

print("\n[Evaluation Metrics]")
print(f"Silhouette Score: {sil_score:.4f} (higher = better, max=1)")
print(f"Davies-Bouldin Index: {db_score:.4f} (lower = better)")
print(f"Calinski-Harabasz Index: {ch_score:.4f} (higher = better)")
```

9 Filtering High-Risk Players

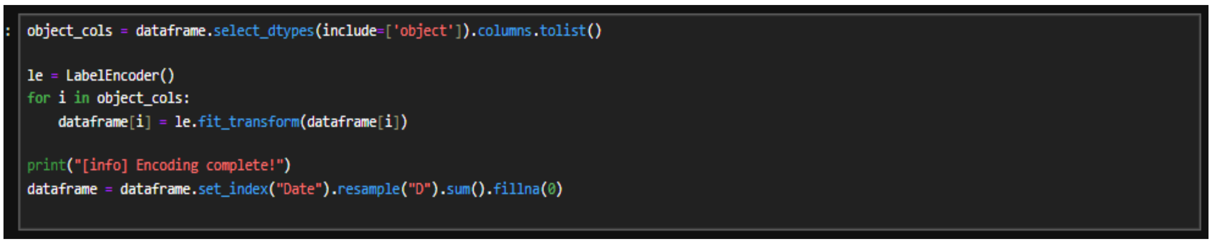
Only the cluster tagged as **High-Risk** is retained for detailed modelling. A single high-risk user is selected for time-series prediction. Their full behavioural timeline is extracted for analysis.



```python
valid_users = dataframe['UserID'].value_counts()[lambda x: x > 102].index

dataframe = dataframe[dataframe['UserID'].isin(valid_users)]

print("Remaining users:", dataframe['UserID'].nunique())
print("Total records:", len(dataframe))
```

10 Time-Series Data Preprocessing

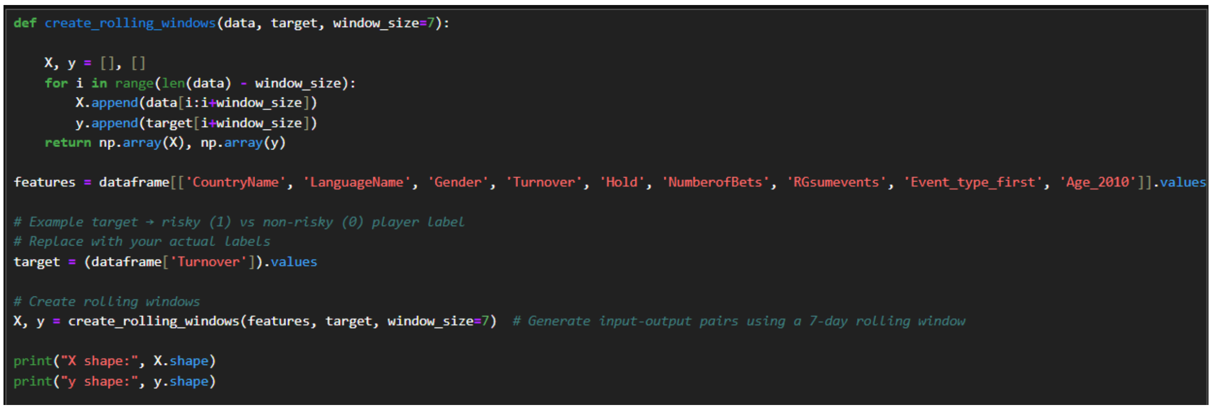
Time gaps, missing turnover values, and irregular timestamps are corrected. Invalid entries are removed to maintain chronological consistency. This ensures clean input for sequence modelling.



```python
object_cols = dataframe.select_dtypes(include=['object']).columns.tolist()

le = LabelEncoder()
for i in object_cols:
    dataframe[i] = le.fit_transform(dataframe[i])

print("[info] Encoding complete!")
dataframe = dataframe.set_index("Date").resample("D").sum().fillna(0)
```

11 Window Rolling (7)

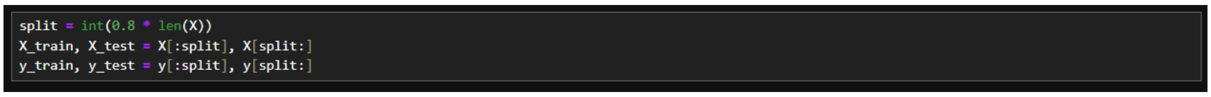
A 7-day sliding window converts the sequence into supervised learning format. Inputs (X) represent past 7 days, while outputs (y) predict the next day. Data is reshaped into (samples, 7, features) format for DL models.



```python
def create_rolling_windows(data, target, window_size=7):

    X, y = [], []
    for i in range(len(data) - window_size):
        X.append(data[i:i+window_size])
        y.append(target[i+window_size])
    return np.array(X), np.array(y)

features = dataframe[['CountryName', 'LanguageName', 'Gender', 'Turnover', 'Hold', 'NumberofBets', 'RGsumevents', 'Event_type_first', 'Age_2010']].values

# Example target → risky (1) vs non-risky (0) player label
# Replace with your actual labels
target = (dataframe['Turnover']).values

# Create rolling windows
X, y = create_rolling_windows(features, target, window_size=7)  # Generate input-output pairs using a 7-day rolling window

print("X shape:", X.shape)
print("y shape:", y.shape)
```

12 Train-Test

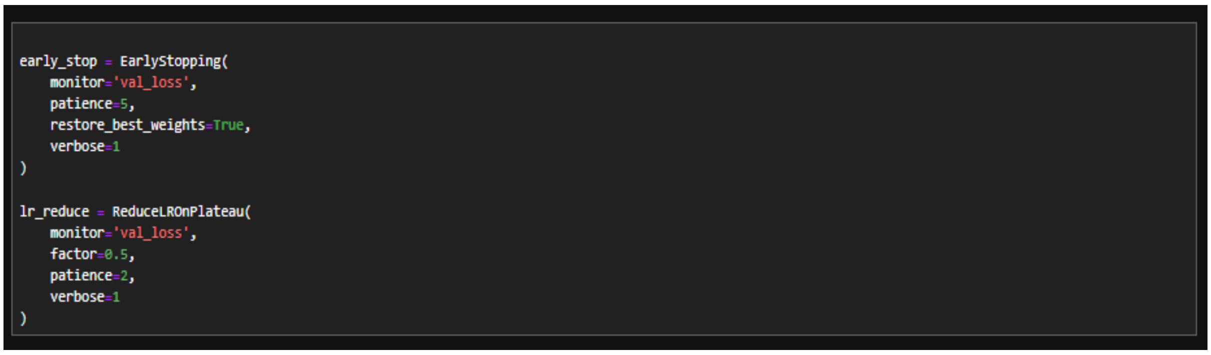
The processed sequences are split into **80% train** and **20% test**. This ensures models learn historical trends and generalize to unseen data. Splitting maintains chronological order.



```python
split = int(0.8 * len(X))
X_train, X_test = X[:split], X[split:]
y_train, y_test = y[:split], y[split:]
```

13 Early Stop & LR Scheduler

Early stopping was applied to prevent overfitting by monitoring the model's validation loss during training. If the validation loss did not improve for five consecutive epochs, training was halted automatically. The best-performing model weights were restored to ensure optimal generalization. A learning-rate reduction mechanism was also used to lower the learning rate when progress plateaued, allowing the model to refine learning more effectively.

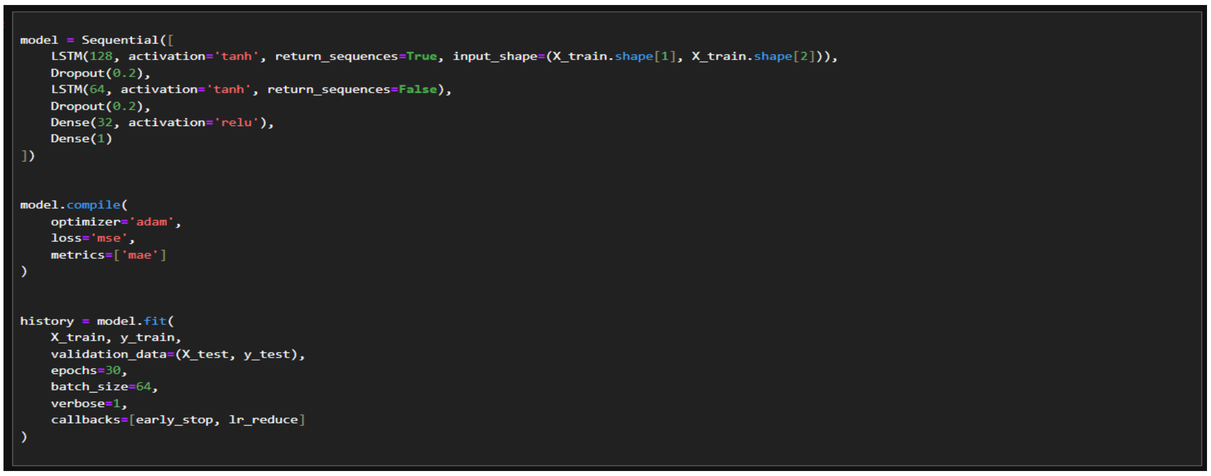


```python
early_stop = EarlyStopping(
    monitor='val_loss',
    patience=5,
    restore_best_weights=True,
    verbose=1
)

lr_reduce = ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.5,
    patience=2,
    verbose=1
)
```

14 Time-Series Model

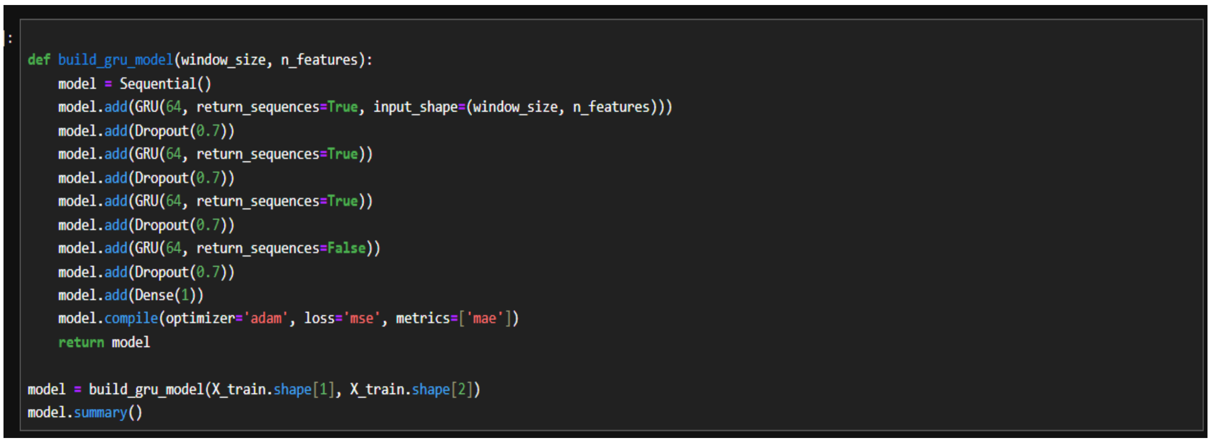
Four models are trained: LSTM, GRU, BiLSTM, and BiLSTM with Cross Attention. Early stopping and LR scheduling improve stability and reduce overfitting. The best-performing architecture is selected for final evaluation.

LSTM:



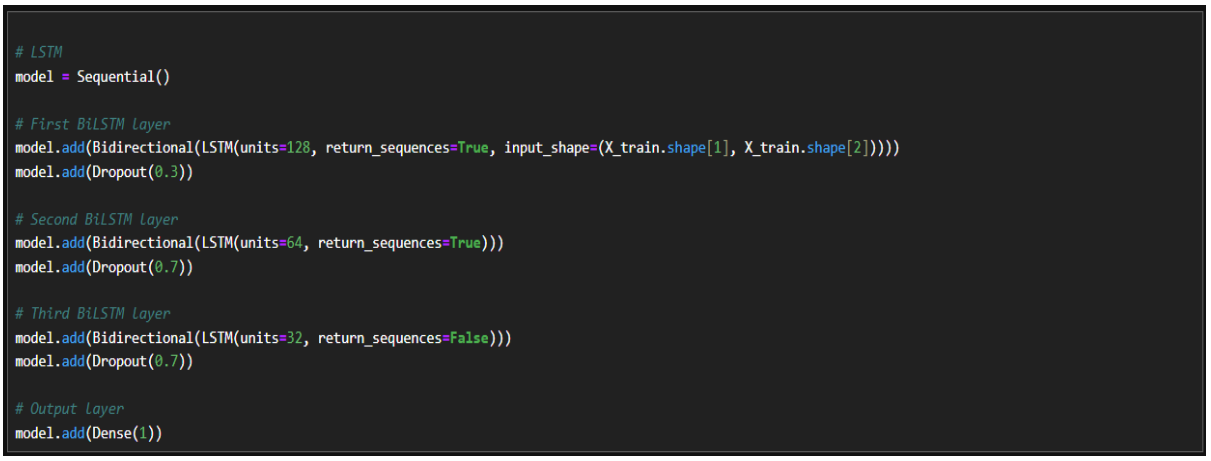
```
model = Sequential([
    LSTM(128, activation='tanh', return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2])),
    Dropout(0.2),
    LSTM(64, activation='tanh', return_sequences=False),
    Dropout(0.2),
    Dense(32, activation='relu'),
    Dense(1)
])


model.compile(
    optimizer='adam',
    loss='mse',
    metrics=['mae']
)


history = model.fit(
    X_train, y_train,
    validation_data=(X_test, y_test),
    epochs=30,
    batch_size=64,
    verbose=1,
    callbacks=[early_stop, lr_reduce]
)
```

GRU:



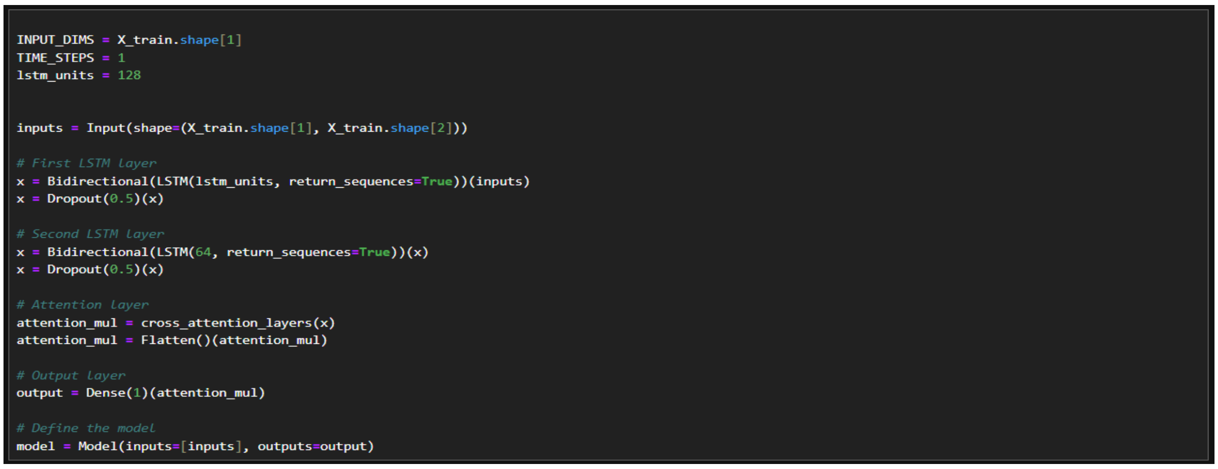
```
def build_gru_model(window_size, n_features):
    model = Sequential()
    model.add(GRU(64, return_sequences=True, input_shape=(window_size, n_features)))
    model.add(Dropout(0.7))
    model.add(GRU(64, return_sequences=True))
    model.add(Dropout(0.7))
    model.add(GRU(64, return_sequences=True))
    model.add(Dropout(0.7))
    model.add(GRU(64, return_sequences=False))
    model.add(Dropout(0.7))
    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mse', metrics=['mae'])
    return model

model = build_gru_model(X_train.shape[1], X_train.shape[2])
model.summary()
```

BiLSTM:



```
# LSTM
model = Sequential()

# First BiLSTM layer
model.add(Bidirectional(LSTM(units=128, return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2]))))
model.add(Dropout(0.3))

# Second BiLSTM layer
model.add(Bidirectional(LSTM(units=64, return_sequences=True)))
model.add(Dropout(0.7))

# Third BiLSTM layer
model.add(Bidirectional(LSTM(units=32, return_sequences=False)))
model.add(Dropout(0.7))

# Output layer
model.add(Dense(1))
```

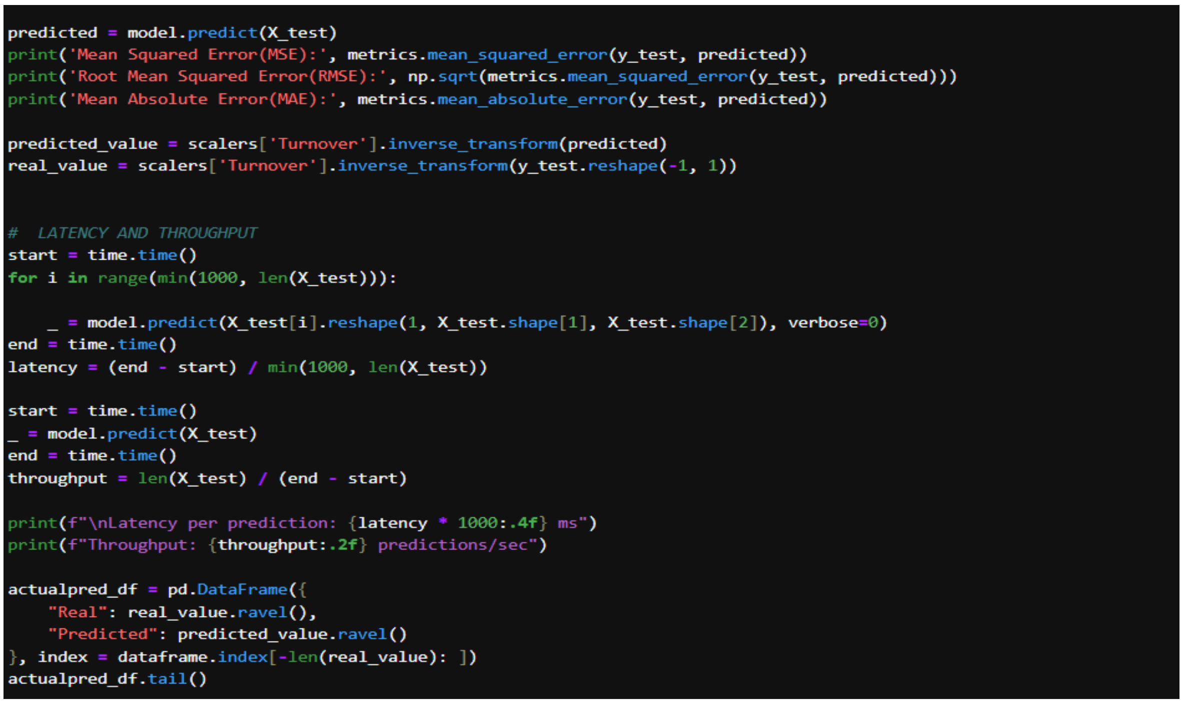
BiLSTM Cross Attention:



```
INPUT_DIMS = X_train.shape[1]
TIME_STEPS = 1
lstm_units = 128


inputs = Input(shape=(X_train.shape[1], X_train.shape[2]))

# First LSTM layer
x = Bidirectional(LSTM(lstm_units, return_sequences=True))(inputs)
x = Dropout(0.5)(x)

# Second LSTM layer
x = Bidirectional(LSTM(64, return_sequences=True))(x)
x = Dropout(0.5)(x)

# Attention layer
attention_mul = cross_attention_layers(x)
attention_mul = Flatten()(attention_mul)

# Output layer
output = Dense(1)(attention_mul)

# Define the model
model = Model(inputs=[inputs], outputs=output)
```

15 Evaluation Metrics

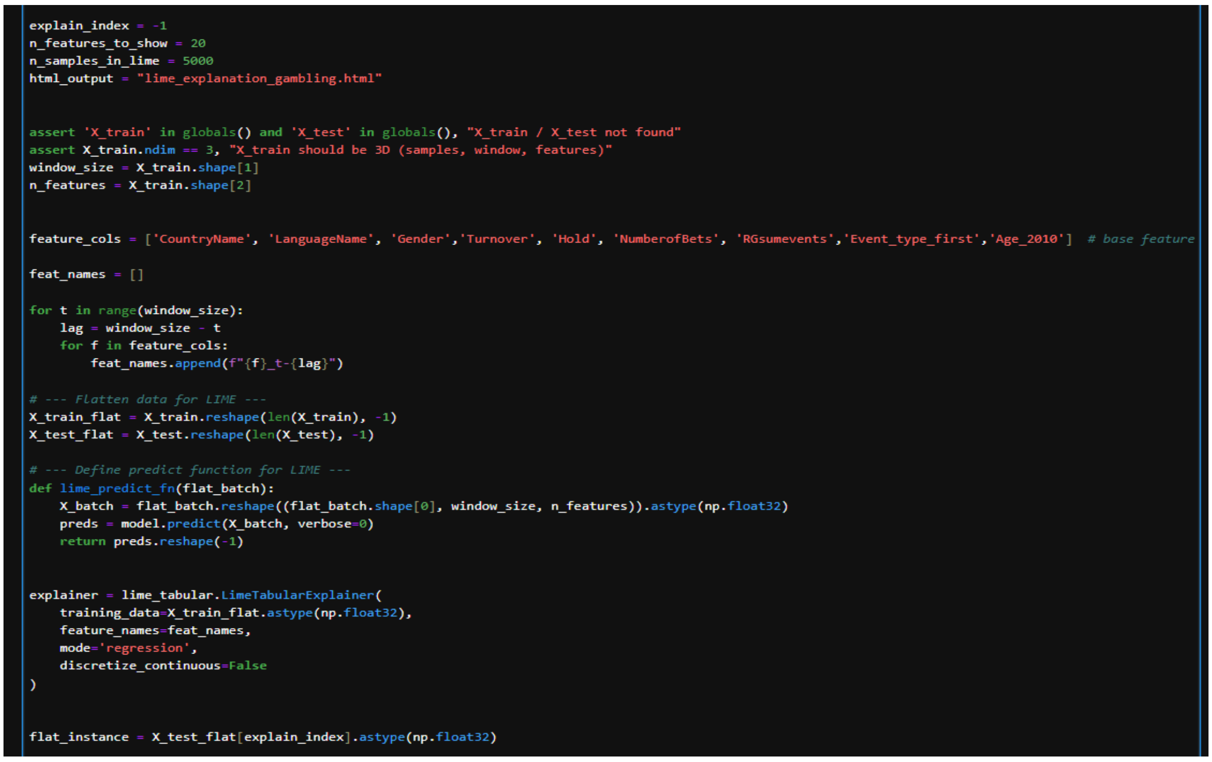
Performance is assessed using RMSE, MAE, and MSE. Latency and throughput measure system efficiency during inference. Together, they reflect both predictive accuracy and operational performance.



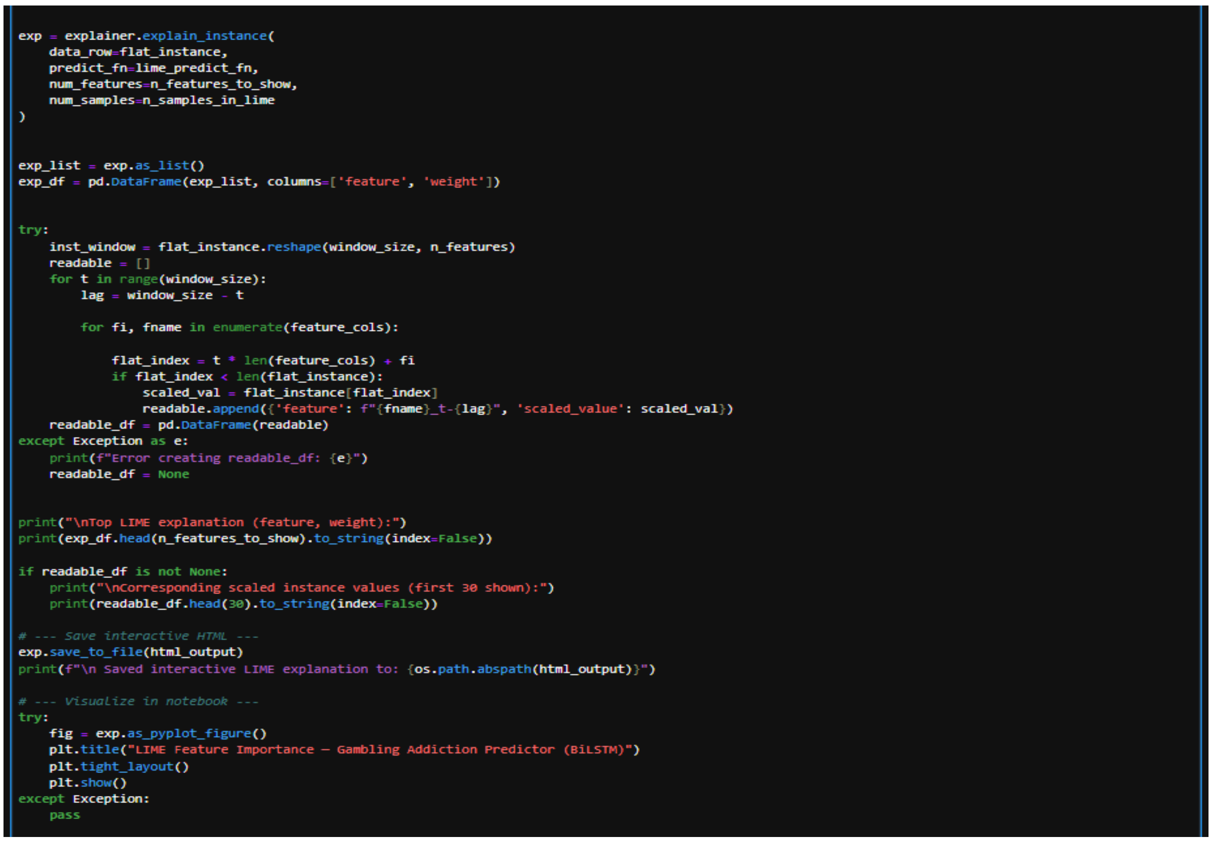
```
predicted = model.predict(X_test)
print('Mean Squared Error(MSE):', metrics.mean_squared_error(y_test, predicted))
print('Root Mean Squared Error(RMSE):', np.sqrt(metrics.mean_squared_error(y_test, predicted)))
print('Mean Absolute Error(MAE):', metrics.mean_absolute_error(y_test, predicted))

predicted_value = scalers['Turnover'].inverse_transform(predicted)
real_value = scalers['Turnover'].inverse_transform(y_test.reshape(-1, 1))


#  LATENCY AND THROUGHPUT
start = time.time()
for i in range(min(1000, len(X_test))):

    _ = model.predict(X_test[i].reshape(1, X_test.shape[1], X_test.shape[2]), verbose=0)
end = time.time()
latency = (end - start) / min(1000, len(X_test))

start = time.time()
_ = model.predict(X_test)
end = time.time()
throughput = len(X_test) / (end - start)

print(f"\nLatency per prediction: {latency * 1000:.4f} ms")
print(f"Throughput: {throughput:.2f} predictions/sec")

actualpred_df = pd.DataFrame({
    "Real": real_value.ravel(),
    "Predicted": predicted_value.ravel()
}, index = dataframe.index[-len(real_value): ])
actualpred_df.tail()
```

16 Explainability AI

LIME identifies which features and time steps drive the model's predictions. It highlights behavioural triggers contributing to risk increases. This ensures transparency in deep-learning–based addiction detection.



```python
explain_index = -1
n_features_to_show = 20
n_samples_in_lime = 5000
html_output = "lime_explanation_gambling.html"


assert 'X_train' in globals() and 'X_test' in globals(), "X_train / X_test not found"
assert X_train.ndim == 3, "X_train should be 3D (samples, window, features)"
window_size = X_train.shape[1]
n_features = X_train.shape[2]


feature_cols = ['CountryName', 'LanguageName', 'Gender','Turnover', 'Hold', 'NumberofBets', 'RGsumevents','Event_type_first','Age_2010']  # base feature

feat_names = []

for t in range(window_size):
    lag = window_size - t
    for f in feature_cols:
        feat_names.append(f"{f}_t-{lag}")

# --- Flatten data for LIME ---
X_train_flat = X_train.reshape(len(X_train), -1)
X_test_flat = X_test.reshape(len(X_test), -1)

# --- Define predict function for LIME ---
def lime_predict_fn(flat_batch):
    X_batch = flat_batch.reshape((flat_batch.shape[0], window_size, n_features)).astype(np.float32)
    preds = model.predict(X_batch, verbose=0)
    return preds.reshape(-1)


explainer = lime_tabular.LimeTabularExplainer(
    training_data=X_train_flat.astype(np.float32),
    feature_names=feat_names,
    mode='regression',
    discretize_continuous=False
)


flat_instance = X_test_flat[explain_index].astype(np.float32)
```



```python
exp = explainer.explain_instance(
    data_row=flat_instance,
    predict_fn=lime_predict_fn,
    num_features=n_features_to_show,
    num_samples=n_samples_in_lime
)


exp_list = exp.as_list()
exp_df = pd.DataFrame(exp_list, columns=['feature', 'weight'])


try:
    inst_window = flat_instance.reshape(window_size, n_features)
    readable = []
    for t in range(window_size):
        lag = window_size - t

        for fi, fname in enumerate(feature_cols):

            flat_index = t * len(feature_cols) + fi
            if flat_index < len(flat_instance):
                scaled_val = flat_instance[flat_index]
                readable.append({'feature': f"{fname}_t-{lag}", 'scaled_value': scaled_val})
    readable_df = pd.DataFrame(readable)
except Exception as e:
    print(f"Error creating readable_df: {e}")
    readable_df = None


print("\nTop LIME explanation (feature, weight):")
print(exp_df.head(n_features_to_show).to_string(index=False))

if readable_df is not None:
    print("\nCorresponding scaled instance values (first 30 shown):")
    print(readable_df.head(30).to_string(index=False))

# --- Save interactive HTML ---
exp.save_to_file(html_output)
print(f"\n Saved interactive LIME explanation to: {os.path.abspath(html_output)}")

# --- Visualize in notebook ---
try:
    fig = exp.as_pyplot_figure()
    plt.title("LIME Feature Importance — Gambling Addiction Predictor (BiLSTM)")
    plt.tight_layout()
    plt.show()
except Exception:
    pass
```

Repositories:

GitHub: https://github.com/saleempasha6969-cyber/Research-Practicum

1) **Data Source: http://www.thetransparencyproject.org/download_index.php**

Dataset Links:

1. **http://www.thetransparencyproject.org/datasets/Raw%20Datset%20I.Demographics_Gray_LaPlante_PAB_2012.sav**
2. **http://www.thetransparencyproject.org/datasets/Raw%20Datset%20II.Daily%20aggregates_Gray_LaPlante_PAB_2012.sav**
3. **http://www.thetransparencyproject.org/datasets/Raw%20Datset%20III.Responsible%20gambling%20details_Gray_LaPlante_PAB_2012.sav**

References

Python: https://www.python.org

Tensorflow : https://www.tensorflow.org/api_docs/python/tf/keras