National College of Ireland

Configuration Manual

MSc Research Project
Msc in Data Analytics

Akshada Warik
Student ID: x23399155

School of Computing
National College of Ireland

Supervisor: Dr. Abid Yaqoob

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Akshada Warik
Student ID:	x23399155
Programme:	Msc in Data Anakytics
Year:	2025
Module:	MSc Research Project
Supervisor:	Dr. Abid Yaqoob
Submission Due Date:	11/12/2025
Project Title:	Configuration Manual
Word Count:	1196
Page Count:	9

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: Akshada Warik	
Date:	27th January 2026

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

Akshada Warik
x23399155

1 System Configuration

This report gives a detailed breakdown to the configuration and Implementation of the project.

1.1 Hardware

- Processor: 12th Gen Intel(R) Core(TM) i7-12650H (2.30 GHz)
- Installed RAM 16.0 GB (15.6 GB usable)
- System type 64-bit operating system, x64-based processor

1.2 Operating System

- Edition Windows 11 Home Single Language
- Version 25H2
- OS build 26200.7462
- Experience Windows Feature Experience Pack 1000.26100.275.0

2 Software Requirements

Software's required for build and execution:

- Integrated Development Environment : Google Colab
- Scripting Language : Python 3
- Storage : Google Drive
- Other Tools : Notepad ++, Overleaf, Excel

3 Libraries

Various libraries were installed and used in the environment to run the project.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import warnings
warnings.filterwarnings('ignore')

sns.set_style("whitegrid")
plt.rcParams['figure.figsize'] = (12, 6)
```

Figure 1: Libraries Imported

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score, roc_curve
import tensorflow as tf
from tensorflow import keras
from keras.models import Model, Sequential
from keras.layers import LSTM, Dense, Dropout, RepeatVector, TimeDistributed, Input
from keras.callbacks import EarlyStopping, ReduceLROnPlateau
import warnings
warnings.filterwarnings('ignore')
```

Figure 2: Libraries Imported

4 Folder Structure

This is the root folder structure below consisting of all the demo artifacts including code, dataset, report and configure manual.

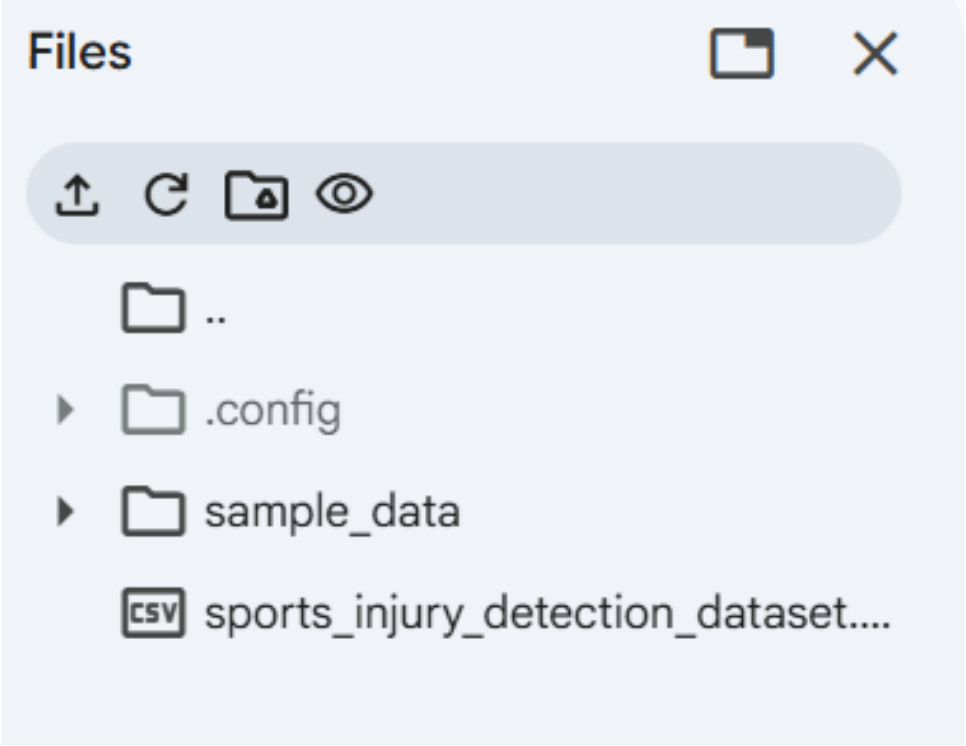

Figure 3: root folder

5 Workflow to replicate the experiment

5.1 Setting Up Your Environment

- Ensure that all the required software and Python libraries are installed.
- Open Google Colab Notebook.
- Set the working directory to the folder containing the dataset files.

5.2 Data Collection

- The dataset contains 1,000 training sessions representing real-time physiological and biomechanical signals (1).
- Data is loaded from CSV format into Python using pandas.
- Each record includes features such as heart rate, respiratory rate, impact force, SpO2, fatigue index, and injury labels.

6 Pre-Processing & EDA

6.1 Data Preparation

- Missing values are checked and addressed.
- Continuous features are normalized using `MinMaxScaler` for deep learning models.
- Injury labels are encoded as:
  - 1 = Injury Occurred
  - 0 = No Injury
- The dataset is split into training (70%), validation (15%), and test (15%) sets.

Tasks

- Load dataset using `pandas`.
- Perform summary statistics.
- Check for missing values.
- Apply normalization:
  - `StandardScaler` for classical ML models.
  - `MinMaxScaler` (0–1 scaling) for LSTM and other deep learning models.
- Generate sliding windows of $T = 30$ time steps for temporal models.

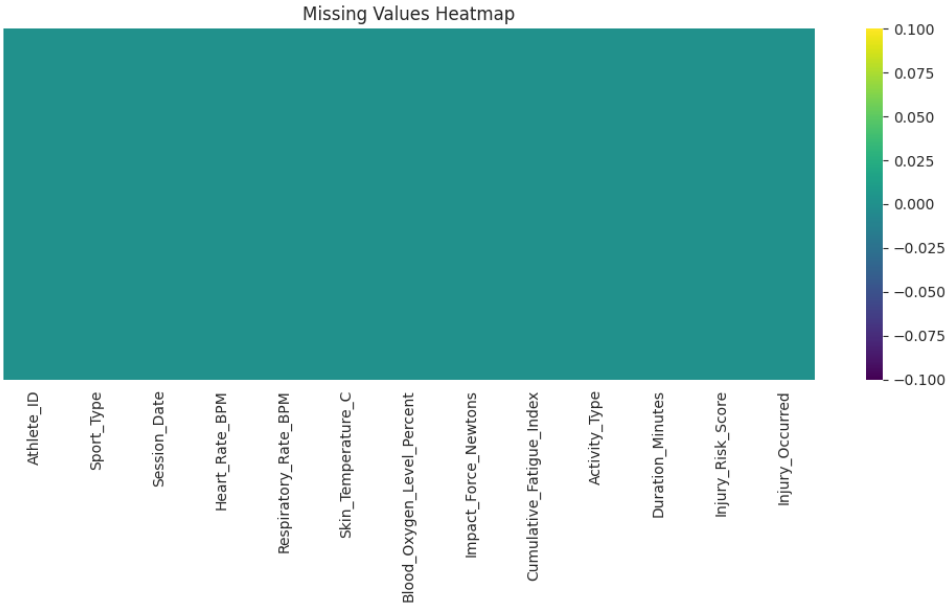

Figure 4: Missing values Heatmap

6.2 Exploratory Data Analysis (EDA)

Key EDA Tasks

- Visualizing numeric feature distributions .
- Assessing class imbalance : 61.7% non-injury vs. 38.3% injury.
- Generating boxplots comparing injured vs. non-injured groups .
- Creating a correlation heatmap highlighting strongest associations (Fatigue Index and Impact Force) .
- Producing pair plots to inspect joint feature behavior.

Statistical Significance Testing

All statistical significance tests (independent t-tests) demonstrated that:

- **Fatigue Index**,
- **Impact Force**, and
- **Injury Risk Score**

were highly significant predictors of injury, with all showing $p < 0.001$.

7 Feature Engineering

Basic feature engineering was performed to organize and prepare the pre-processed dataset for modeling. Additional derived metrics were added where needed, and all features were cleaned and formatted. The final feature set was then split into an 80–20 ratio for training and testing.

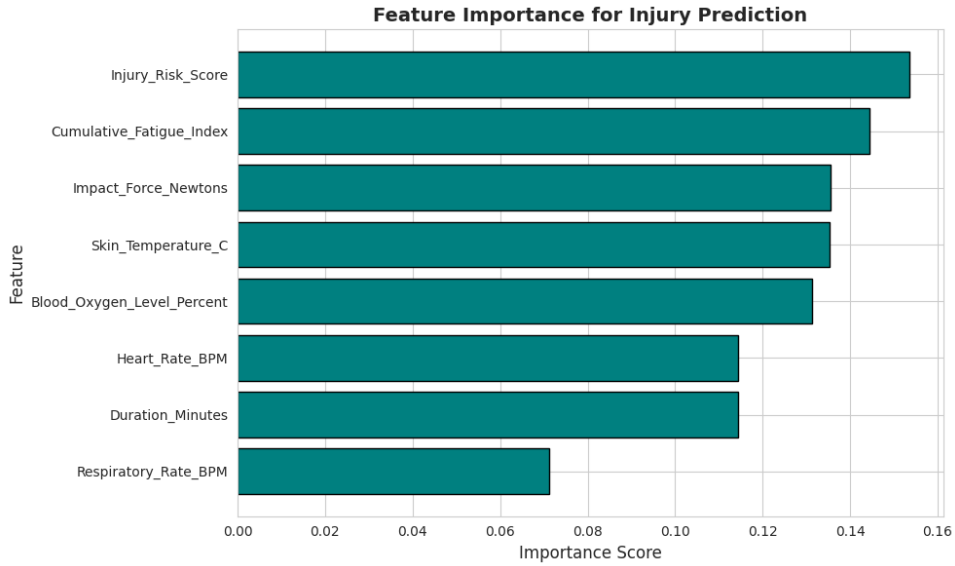

Figure 5: Feature Importance for Injury Prediction

```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

Figure 6: Train/test split (80/20)

8 Model Training

Classical Machine Learning Models

Classical machine learning models were trained using the `scikit-learn` library. The models evaluated include:

- Logistic Regression
- Random Forest
- Support Vector Machine (SVM) with an RBF kernel

Deep Learning Architectures

Two deep learning models were developed to capture temporal structure in the wearable sensor data.

1) Baseline LSTM Classifier

- Sequence-to-label architecture.
- Sliding window of 30 time steps.

- Designed to capture short-term fatigue and biomechanical variations.
- Training curves are shown in Figure 8

2) LSTM Autoencoder

- Trained exclusively on non-injury sequences to learn "normal" biomechanical patterns.
- Reconstruction error threshold used for anomaly-based injury classification.
- Achieved perfect Recall (1.00), indicating all injury cases were detected.
- Training curves are shown in Figure 8.

9 Model Evaluation

Evaluation Metrics

The following performance metrics were used to evaluate all classical and deep learning models:

- Accuracy
- Precision
- Recall
- F1 Score
- ROC-AUC

Key Results

- **Best Recall:** LSTM Autoencoder (1.0000).
- **Best classical model:** SVM (RBF) with Recall = 0.9187.
- **Highest overall F1-Score:** LSTM Autoencoder (0.7616).

Model	Accuracy	Precision	Recall	F1	ROC-AUC
Logistic Regression	0.595	0.6250	0.8537	0.7216	0.5657
Random Forest	0.630	0.6450	0.8862	0.7466	0.5611
SVM (RBF)	0.620	0.6313	0.9187	0.7483	0.6045
Baseline LSTM	0.600	0.6091	0.9756	0.7500	0.5942
LSTM Autoencoder	0.615	0.6150	1.0000	0.7616	0.5319

Figure 7: Unified Comparison of Classical and Deep Learning Models

Unified Model Comparison

To directly compare all five models (three classical and two deep learning), their evaluation metrics are consolidated in Table.

Key Insights from the Comparison

- **Highest Recall:** LSTM Autoencoder (1.0000), followed by:
  - Baseline LSTM (0.9756)
  - SVM (0.9187)
- **Best ROC-AUC among classical models:** SVM (0.6045)

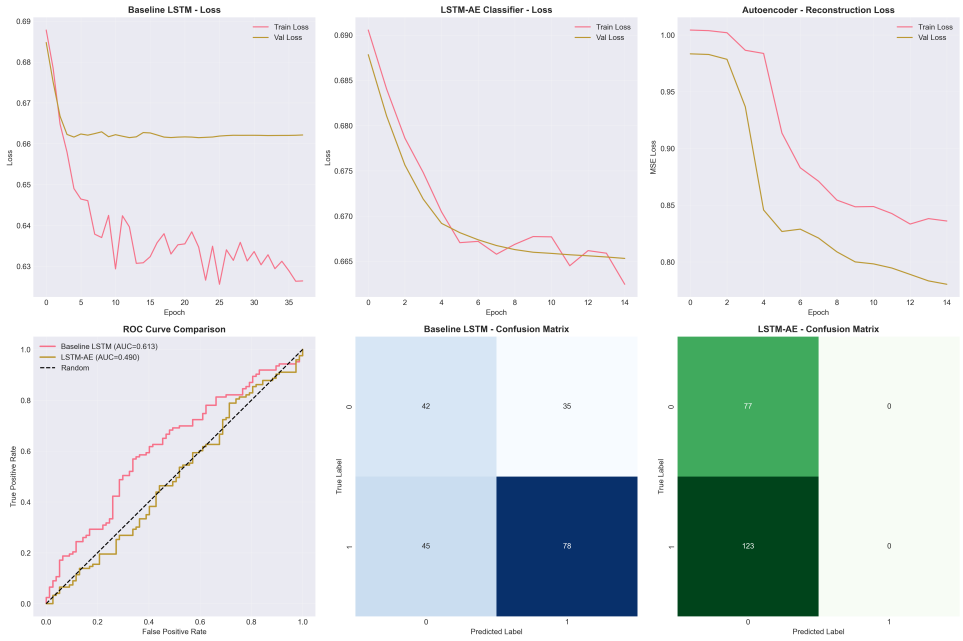

Figure 8: Model Comparison

10 Fairness & Bias

Fairness considerations were incorporated to ensure that the predictive models operated consistently across all athlete sessions and did not produce outputs influenced by unintended data imbalances. Since the dataset contains variations in physiological and biomechanical measurements across different sports and training conditions, the modelling pipeline included checks to verify that predictions were based on relevant performance-related indicators rather than disproportionately favouring specific groups of observations. These assessments helped confirm that the models behaved reliably and aligned with expected physiological patterns associated with fatigue, workload, and injury risk.

11 Possible Conflicts & Troubleshooting

CPU Limitation

Certain components of the experimental workflow—particularly the temporal deep learning models—require substantial computational resources. On lower-specification hardware, training may become slow, unstable, or fail due to insufficient processing power. To mitigate these issues, users may:

- reduce batch size,
- decrease hidden layer dimensions,
- shorten sequence length, or
- run the notebooks in a cloud-based environment that provides GPU acceleration.

Library Conflicts

Inconsistent or incompatible library versions may lead to unexpected behaviour during preprocessing or model training. To maintain reproducibility, users are advised to install stable and current versions of the core scientific libraries, including:

- `numpy`,
- `pandas`,
- `scikit-learn`,
- `TensorFlow` or `PyTorch` (depending on framework used).

Ensuring library version alignment across environments helps avoid execution errors and supports smooth operation of the entire configuration pipeline.

Notebook Crashes: Environment Preflight Test

Before training high-complexity models, it is recommended to perform an environment preflight check to confirm that the hardware can support the configured architecture. This includes:

- verifying available memory capacity,
- confirming that sequence lengths are appropriate for the device,
- checking that model parameters do not exceed system resource limits.

Conducting this validation step helps reduce the risk of runtime crashes and ensures stable model training performance.

References

[1] Ziya07, "College sports injury detection dataset," https://www.kaggle.com/datasets/ziya07/college-sports-injury-detection, 2023, accessed: 2025-01-10.