User Manual

Lower Limb Sports Injury Prediction and Prevention

CA400

Lorcan Dunne - 19311511
Dylan Bagnall - 20413066

Table of Contents

1. Introduction

1.1. Abstract

Lower limb injuries present a significant challenge in high-intensity sports, negatively impacting athletes' performance and well-being. This project seeks to develop a machine learning-based lower limb injury prediction model to aid in proactive injury prevention strategies. The model leverages historical data to identify patterns and risk factors associated with lower limb injuries. We employ multiple machine learning algorithms, including XGBoost, Random Forest, and Decision Trees, combined with k-fold cross-validation and random undersampling techniques to ensure robustness and address potential data imbalances.

The primary goal is to provide athletes and healthcare professionals with valuable insights into an athlete's risk profile for lower limb injuries. Key factors considered include previous playing surfaces, and player workload. This model serves as a

decision support tool, complementing expert medical judgement to facilitate targeted interventions and reduce injury incidence in high-intensity sports.

1.2. Glossary

NFL - National Football League, The major professional American football league in the United States

CSV - Simple text format storing tabular data (rows and columns), with commas separating values.

Decision Tree - Model that makes predictions using a tree-like structure of decision rules. Easy to understand and visualise.

Random Forest - Combines multiple decision trees for improved predictions. Reduces overfitting and handles complex data well.

XGBoost - Optimised gradient boosting algorithm that builds sequential decision trees to correct previous errors. Known for speed and performance.

2. Installation Guide

2.1. **Required Software**

- Visual Studio code

2.2. **Required Libraries**

- Pip
- Imbalanced-learn
- Joblib
- Matplotlib
- Numpy
- Pandas
- PyQt5
- Scikit-learn
- Scipy
- XGBoost

These are the required installations in order to successfully run and evaluate our pre-processing files, models and GUI. These required libraries are referenced in our requirements.txt file in our main directory.

requirements.txt

```
imbalanced-learn==0.12.2
joblib==1.3.2
matplotlib==3.8.2
numpy==1.26.1
pandas==1.2.1
PyQt5==5.15.10
scikit-learn==1.3.2
scipy==1.11.3
xgboost==2.0.3
flake8==3.9.2
pytest==6.2.5
sea-born==0.11.2
```

2.3. **Required Hardware**

- Device with access to a preferred code editor or Integrated Development Environment (IDE)

2.4 **Obtaining the dataset**

- The dataset can be access on Kaggle

3. Getting Started

3.1. Merging the datasets

To start the process of interpreting and organising the data you must merge the two datasets obtained from Kaggle. Injury Record and Play List are the two datasets in the .csv format. Initially the file imports the necessary libraries and reads the data from the csv files and assigns them to variable dataframes. To merge the datasets you must run the following command:

‘Python3 merging.py’

This will run the merging.py file that reads the two data sets and merges them into one with some organisational techniques. This file will then output a new csv file called ‘MergedData_LeftMerge.csv’

3.2. Feature engineering

Now that the new dataset is created containing the 2 datasets obtained from Kaggle, you can begin to run the feature engineering file. This file is used to add new features uniquely created with the aim to improve the overall performance and accuracy of the predictive models. To create these features and add them to the csv file, you can run the command:

‘Python3 feature_engineering.py’

This file adds the unique features to the dataset and saves it to a new dataset called ‘Aggregated_Data2.csv’

3.3. Training

Now that we have our dataset that is merged and contains the features that will improve performance and accuracy, the next step is to split the dataset into training and testing files. This will allow the models to train their predictive accuracies using the training data and then test their performance on new unseen test data. To do this you should run the command:

‘Python3 training.py’

This file will read the ‘Aggregated_Data2.csv’ file, create the binary target variable, ‘Injured’ and split the data into 4 files used for training and testing. These are the expected files that you will see once running this command:

X_train_classification.csv
X_test_classification.csv
y_train_classification.csv
y_test_classification.csv'

4. Testing the models

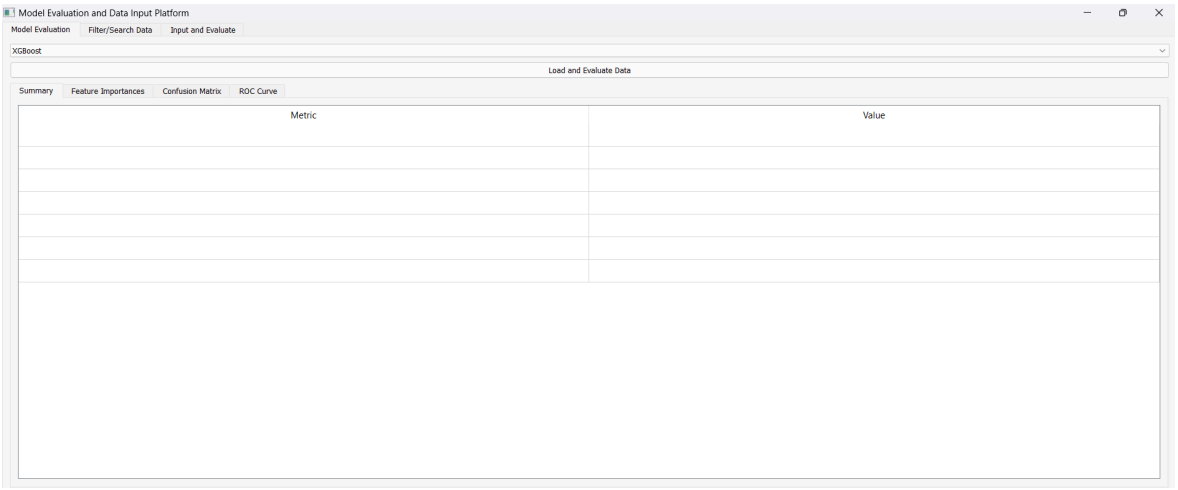
Now that you have everything setup you can begin testing the models and interpreting the data. This is simply done by running the command ‘Python3’ followed by the desired model you wish to run.

‘Python3 Decision-tree.py’
‘Python3 RFkfold.py’
‘Python3 XGBoostkfold.py’

Once these commands are run you will be provided with all the model evaluations metrics along with visualisation graphs.

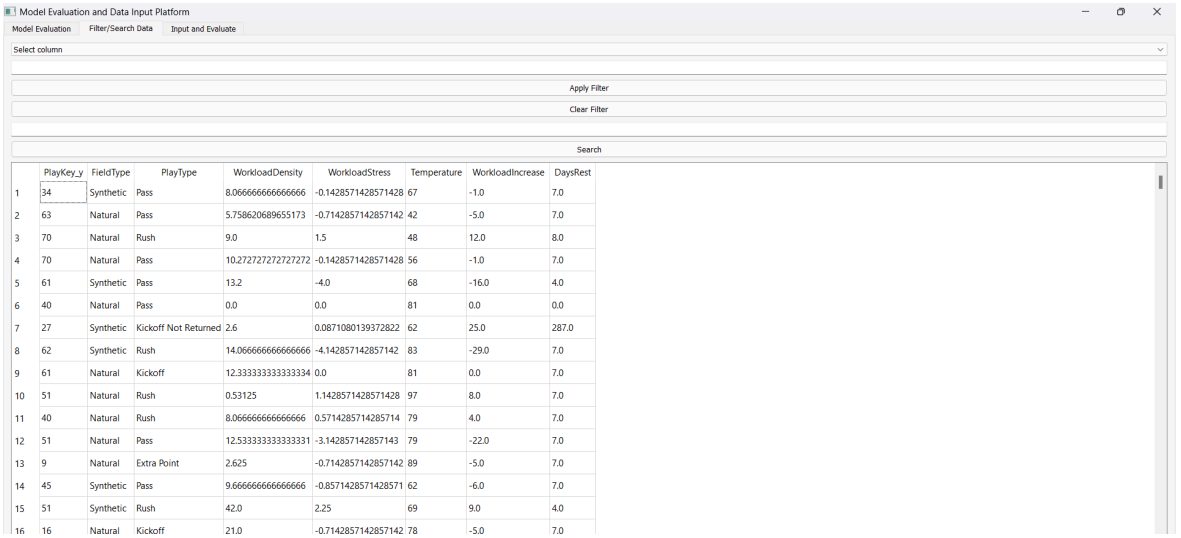
5. Running the GUI

Finally now that all the models have been trained you can run the GUI which will provide a better understanding and visualisation of the models outputs and performance. To run the GUI like the rest of the files you run the command: ‘‘Python3 gui.py’. This will present you with a new window which will have many features. Initially you will be presented with the model evaluation window which will allow you to clearly see how the individual models perform.



As you can see in the image above you can click the model select dropdown feature. This allows you to select the desired model you wish to evaluate. Then you can click evaluate which will portray the model's metrics. You switch through the metrics window which include: ‘Summary’, ‘Feature Importance’, ‘Confusion Matrix’ and ‘ROC Curve’.

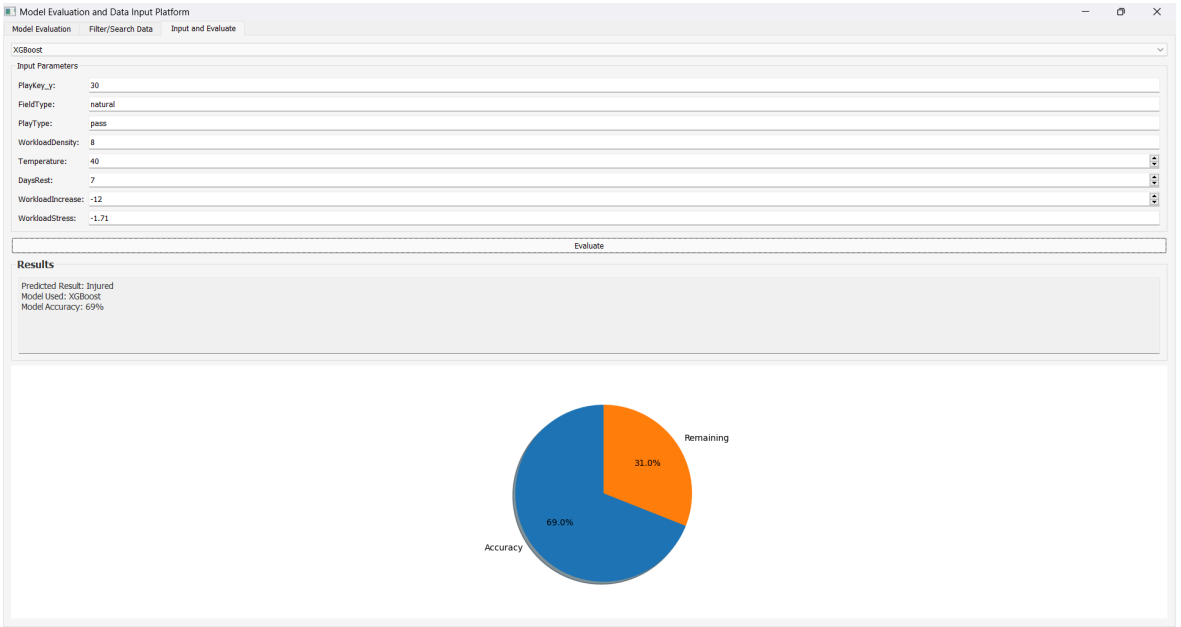
Next you can click on the ‘Filter/Search Data’ window. This will allow you to filter through the data set, see how it is formatted cleary and have a better understanding of the overall usage of the dataset.



	PlayKey_y	FieldType	PlayType	WorkloadDensity	WorkloadStress	Temperature	WorkloadIncrease	DaysRest
1	34	Synthetic	Pass	8.066666666666666	-0.1428571428571428	67	-1.0	7.0
2	63	Natural	Pass	5.758620689655173	-0.7142857142857142	42	-5.0	7.0
3	70	Natural	Rush	9.0	1.5	48	12.0	8.0
4	70	Natural	Pass	10.272727272727272	-0.1428571428571428	56	-1.0	7.0
5	61	Synthetic	Pass	13.2	-4.0	68	-16.0	4.0
6	40	Natural	Pass	0.0	0.0	81	0.0	0.0
7	27	Synthetic	Kickoff Not Returned	2.6	0.0871080139372822	62	25.0	287.0
8	62	Synthetic	Rush	14.066666666666666	-4.142857142857142	83	-29.0	7.0
9	61	Natural	Kickoff	12.333333333333334	0.0	81	0.0	7.0
10	51	Natural	Rush	0.53125	1.1428571428571428	97	8.0	7.0
11	40	Natural	Rush	8.066666666666666	0.5714285714285714	79	4.0	7.0
12	51	Natural	Pass	12.533333333333331	-3.142857142857143	79	-22.0	7.0
13	9	Natural	Extra Point	2.625	-0.7142857142857142	89	-5.0	7.0
14	45	Synthetic	Pass	9.666666666666666	-0.8571428571428571	62	-6.0	7.0
15	51	Synthetic	Rush	42.0	2.25	69	9.0	4.0
16	16	Natural	Kickoff	21.0	-0.7142857142857142	78	-5.0	7.0

As you can see from the image above you can select the feature you want to filter and search for it in the dataset. When you are finished with that feature you can click the clear filter which will clear the feature you searched for and show the dataset without any filters. Alternatively you can manually search for something in the dataset using the search bar.

And finally you can click onto the ‘Input and Evaluation’ window. This window will allow you to input your own unique data into the feature boxes and evaluate using your desired model.



As you can see you can input into the text boxes provided, change the model and click evaluate, allowing you to be able to predict an injury based on your own needs.