National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Pratik Umesh Shetty
Student ID: x21227578

School of Computing
National College of Ireland

Supervisor: Vladimir Milosavljevic

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Pratik Umesh Shetty
Student ID:	x21227578
Programme:	Data Analytics
Year:	2023
Module:	MSc Research Project
Supervisor:	Vladimir Milosavljevic
Submission Due Date:	14/12/2023
Project Title:	Configuration Manual
Word Count:	729
Page Count:	15

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:	
Date:	30th January 2024

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

Pratik Umesh Shetty
x21227578

1 Introduction

The subsequent manual on configuration presents an elucidation of the prerequisites for the execution of the system that was devised for the purpose of generating a xG Model for Player Analysis through the utilization of ML models, namely Gradient Boosting Classifier and Logistic Regression. Additionally, the manual will comprehensively explicate the stipulated software and hardware requirements that were employed in the triumphant execution of the undertaking.

2 System Configuration

Following are the hardware and software configuration which were used for the implementation of this Project.

The hardware configurations used for implementation are as follows shown in Table 1:

2.1 Hardware Requirement

System	HP Pavilion Gaming Laptop 15-ec1xxx
Operating System	Windows 11 (64 Bits) Home
RAM	8 GB
Hard Disk	1 TB
Graphics Card	NVIDIA geforce GTX 1600 Ti (4 GB)
Processor	AMD Ryzen 5 4600H with Radeon Graphics

Table 1: Hardware Configurations

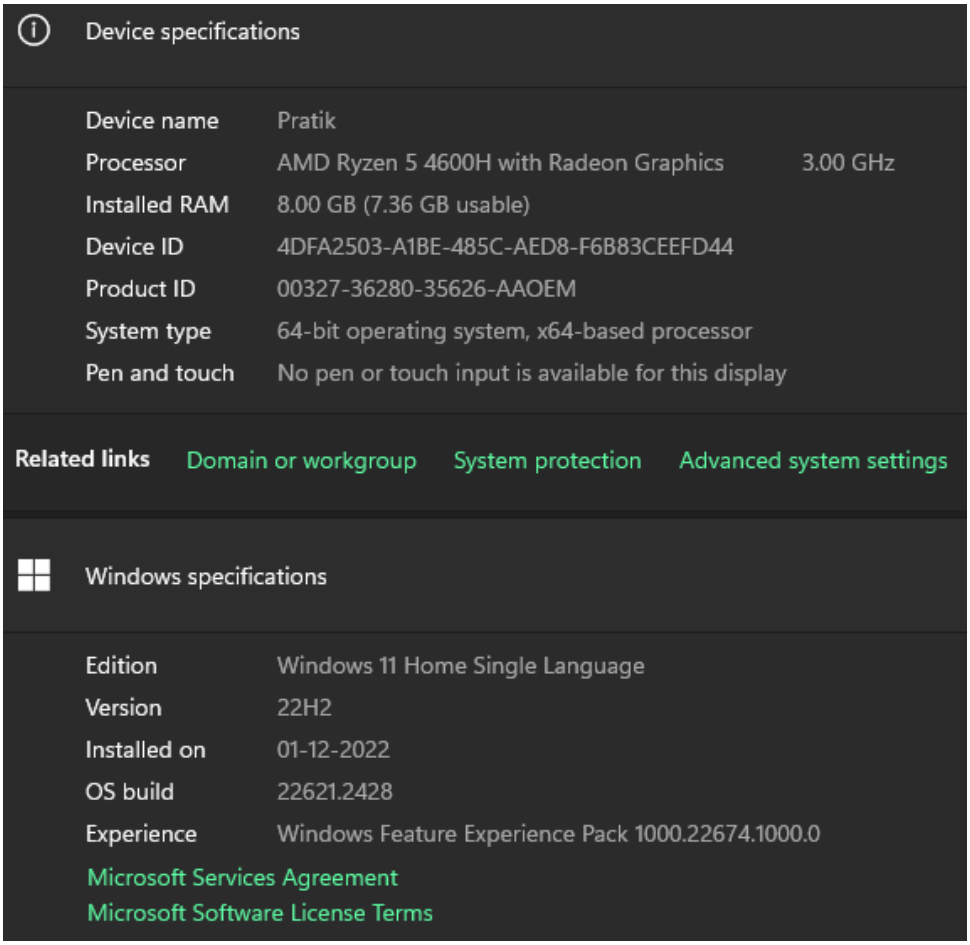


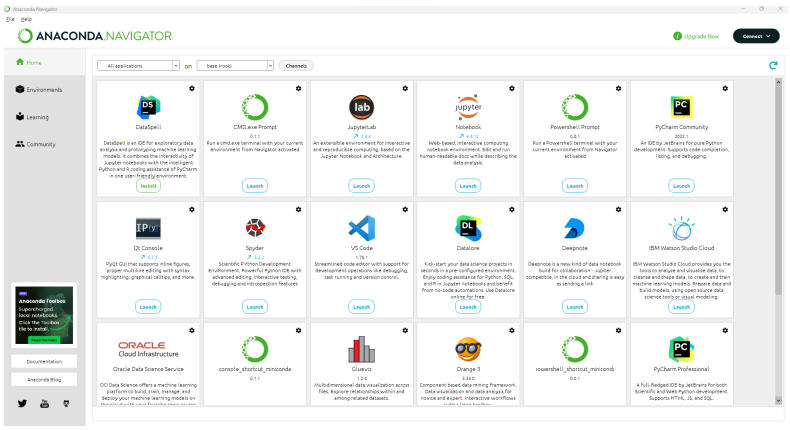
Figure 1: Hardware Configuration

2.2 Software Requirement

The software configurations used for implementation are as follows shown in Table 2:

Software	Version	Architecture
Python	3.8	64 Bits
Jupyter Notebook	6.4.12	64 Bits
Anaconda Navigator	2.3.1	64 Bits

Table 2: Software Versions



(a) Anaconda Navigator

(b) Jupyter

Figure 2: Softwares Used

2.3 Python Libraries Used

- pandas
- numpy
- matplotlib
- scipy
- sklearn
- seaborn
- datetime
- hyperopt

3 Project Implementation

3.1 Dataset Summary

There are three files as shown below, whole dataset is downloaded from kaggle.com

1. events.csv File:
    - Event Types: Identifies eleven event types from textual commentary, revealing diverse in-game occurrences.
    - Player Information: Extracts details about the primary and secondary players involved, crucial for player-centric analysis.
    - Game Details: Scrutinizes game-level data, providing context for events, including league, season, and timestamp.
2. ginf.csv File:
    - Metadata: Investigates game-related information like teams, venue, and outcome.
    - Market Odds: Explores odds from oddsportal.com, offering a quantitative measure of market expectations.
3. dictionary.txt File:
    - Categorical Variables: Decodes integers into meaningful categories for enhanced dataset interpretability.
    - Variable Descriptions: Comprehends textual descriptions to ensure accurate interpretation in subsequent analyses.

3.2 Importing Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy as sp
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import average_precision_score, roc_auc_score, f1_score, precision_score, \
recall_score, cohen_kappa_score, classification_report,confusion_matrix
from sklearn.model_selection import train_test_split
import seaborn as sns
from datetime import datetime
from sklearn.preprocessing import StandardScaler
pd.options.display.max_columns = 999
pd.options.display.max_rows = 50

class color:
   PURPLE = '\033[95m'
   CYAN = '\033[96m'
   DARKCYAN = '\033[36m'
   BLUE = '\033[94m'
   GREEN = '\033[92m'
   YELLOW = '\033[93m'
   RED = '\033[91m'
   BOLD = '\033[1m'
   UNDERLINE = '\033[4m'
   END = '\033[0m'
```

Figure 3: Importing Libraries

3.3 Dataset Loading and Pre-Processing

```
events = pd.read_csv('D:/Project Fin/events.csv')
info = pd.read_csv('D:/Project Fin/ginf.csv')
```

Figure 4: Load the Dataset

We enhance the comprehensiveness of our events dataset by incorporating valuable data extracted from the ginf.csv file. This dataset augmentation includes significant details such as the league and country associated with the events, thereby providing a more holistic understanding of the sporting context. Additionally, we also acquire the precise date on which these events occur, enabling a comprehensive chronology of the recorded information.

```
events = events.merge(info[['id_odsp', 'country', 'date']], on='id_odsp', how='left')
events.head()
```

Figure 5: Mergings the Dataset

```
extract_year = lambda x: datetime.strptime(x, "%Y-%m-%d").year
events['year'] = [extract_year(x) for key, x in enumerate(events['date'])]
```

```
shots = events[events.event_type==1]#Shots will contain everything related to this action of the game and exclude the rest
shots['player'] = shots['player'].str.title()
shots['player2'] = shots['player2'].str.title()
shots['country'] = shots['country'].str.title()
```

Figure 6: Eliminating all other events except 'Shot'

3.4 Exploratory Data Analysis

```
pie = shots[['shot_outcome', 'id_event']].groupby('shot_outcome').count().reset_index().rename(columns={'id_event': 'count'})
```

```
pie.shot_outcome = pie.shot_outcome.astype(int)
pie.shot_outcome = pie.shot_outcome.replace({1: 'On Target', 2: 'Off Target', 3: 'Blocked', 4: 'Hit the Bar'})
```

```
fig, ax = plt.subplots(figsize=[8,8])
labels = pie['shot_outcome']
colors = ['#ff9999','#66b3ff','#99ff99','#ffcc99']
plt.pie(x=pie['count'], autopct="%.1f%%", labels=labels, explode=[0.06]*4, pctdistance=0.7, colors=colors, shadow=True, \
       textprops=dict(fontsize=16))
plt.title("Shot Outcomes", fontsize=26, fontfamily='serif')
plt.tight_layout()
plt.show()
```

Figure 7: Performing EDA

In our analysis, we ascertain that the ratios of goals and no-goals exhibit a remarkable consistency throughout the course of time. This observation leads us to the compelling inference that, from a statistical perspective, there exists a consistent pattern wherein approximately one out of nine to ten shots result in goals as shown in Figure 12, regardless of the specific location or temporal context under scrutiny. It is noteworthy to mention that this empirical relationship holds true irrespective of the various circumstances surrounding the occurrence of these shots, thereby reinforcing the robustness of our findings.

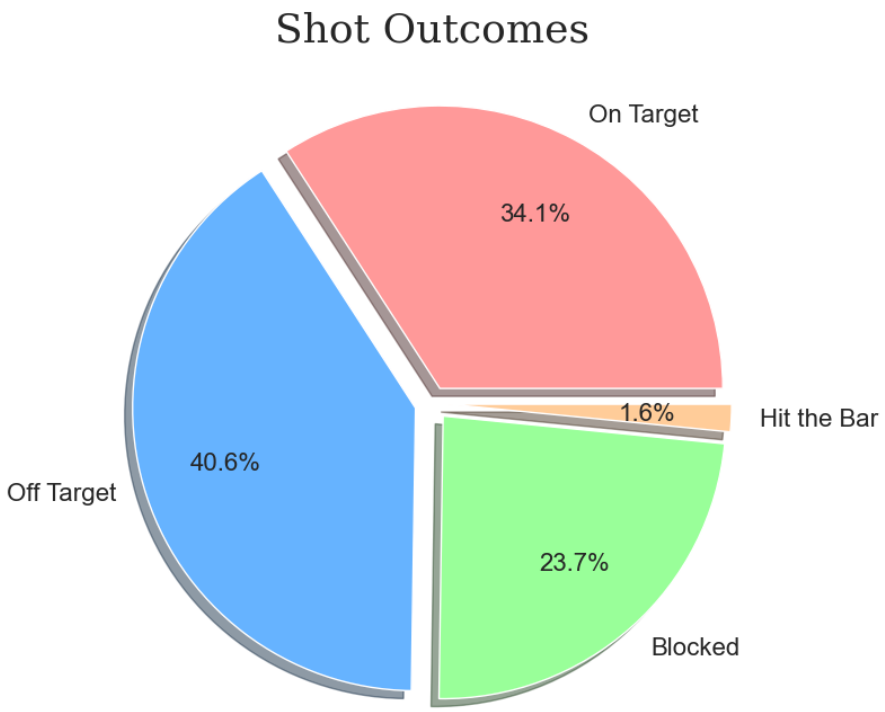


Figure 8: Performing EDA

```
bar = shots[['shot_place', 'id_event']].groupby('shot_place').count().reset_index().rename(columns={'id_event': 'count'})
bar.shot_place = bar.shot_place.astype(int)
bar.shot_place = bar.shot_place.replace({1: 'Bit too high', 2: 'Blocked', 3: 'Bottom left corner', 4: 'Bottom right corner', \
                                         5: 'Centre of the goal', 6: 'High and wide', 7: 'Hits the bar', 8: 'Misses to the left', \
                                         9: 'Misses to the right', 10: 'Too high', 11: 'Top centre of the goal', \
                                         12: 'Top left corner', 13: 'Top right corner'})
```

```
sns.set_style("whitegrid")
fig, ax = plt.subplots(figsize=[13,6])
labels = bar['shot_place']
colors = ['#ff9999','#66b3ff','#99ff99','#ffcc99']
ax = sns.barplot(x=bar['count'], y=labels)
ax.set_yticklabels(labels, size = 13)
ax.set_xticks(np.arange(1000,57000,4000))
ax.set_ylabel(ylabel='')
plt.title("Shot Placement", fontsize=25, fontfamily='serif')
plt.tight_layout()
ax.grid(color='black', linestyle='-', linewidth=0.1, axis='x')
plt.show()
```

Figure 9: Performing EDA

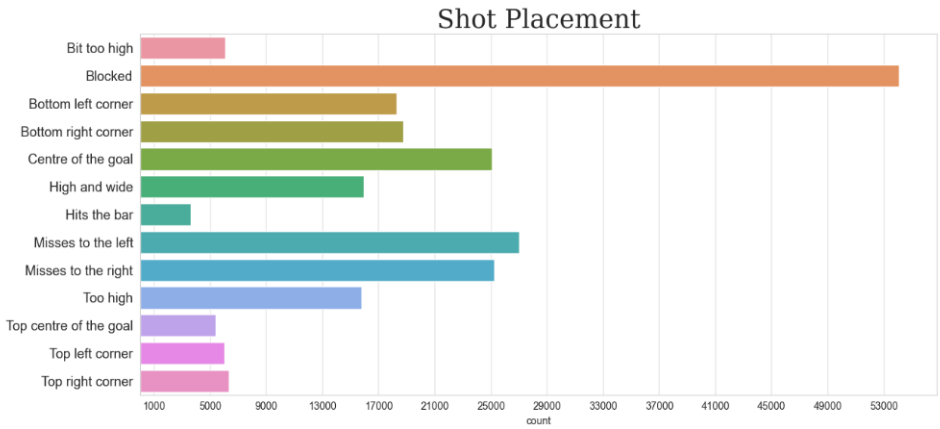


Figure 10: Performing EDA

```
goals  = shots[['is_goal', 'id_event', 'country']].groupby(['is_goal', 'country']).count().reset_index().rename(columns={'id_ever
goals.is_goal = goals.is_goal.replace({1: 'Goal', 0: 'No Goal'})
```

```
goals['percentage']=0
for i in range(len(goals)):
    for country in goals.country.unique():
        if goals.iloc[i,goals.columns.get_loc("country")]==country:
            goals.iloc[i,goals.columns.get_loc("percentage")]=goals.iloc[i,goals.columns.get_loc("count")] / \
                                                              goals[goals.country==country]['count'].sum()
goals['percentage']=round(goals['percentage']*100,2)
```

```
def show_values_on_bars(axs):
    def _show_on_single_plot(ax):
        for p in ax.patches:
            _x = p.get_x() + p.get_width() / 2
            _y = p.get_y() + p.get_height()
            value = '{:.2f}%'.format(p.get_height())
            ax.text(_x, _y+2, value, ha="center", fontsize=14)

    if isinstance(axs, np.ndarray):
        for idx, ax in np.ndenumerate(axs):
            _show_on_single_plot(ax)
    else:
        _show_on_single_plot(axs)
```

```
sns.set_style("whitegrid")
fig, ax = plt.subplots(figsize=[14,6])
colors = ['#ff9999','#66b3ff','#99ff99','#ffcc99']
ax = sns.barplot(data=goals, y='percentage', hue='is_goal', x='country')
ax.set_yticks(np.arange(0,110,10))
ax.set_ylabel(ylabel='Percentage %', fontsize=15, fontfamily='serif')
ax.set_xlabel(xlabel='League', fontsize=15, fontfamily='serif')
ax.set_xticklabels(labels=ax.get_xticklabels(), fontsize=16, fontfamily='serif')
plt.title("Goal/No-Goal per Country", fontsize=24, fontfamily='serif')
plt.tight_layout()
ax.grid(color='black', linestyle='-', linewidth=0.1, axis='y')
plt.legend(fontsize=12)
show_values_on_bars(ax)
plt.show()
```

Figure 11: Performing EDA

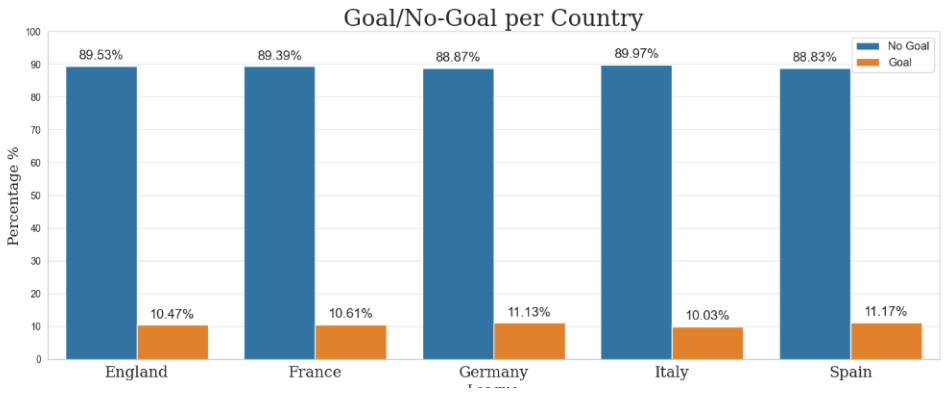


Figure 12: Performing EDA

4 xG Model

```
data = pd.get_dummies(shots.iloc[:,-8:-3], columns=['location', 'bodypart','assist_method', 'situation'])
data.columns = ['fast_break', 'loc_centre_box', 'loc_diff_angle_lr', 'diff_angle_left', 'diff_angle_right',
                'left_side_box', 'left_side_6ybox', 'right_side_box', 'right_side_6ybox', 'close_range',
                'penalty', 'outside_box', 'long_range', 'more_35y', 'more_40y', 'not_recorded', 'right_foot',
                'left_foot', 'header', 'no_assist', 'assist_pass', 'assist_cross', 'assist_header',
                'assist_through_ball', 'open_play', 'set_piece', 'corner', 'free_kick']
data['is_goal'] = shots['is_goal']
```

```
print(len(data))
print(data.is_goal.sum())
print(len(data.columns)-1)
```

```
229135
24441
28
```

```
data.head()
```

	fast_break	loc_centre_box	loc_diff_angle_lr	diff_angle_left	diff_angle_right	left_side_box	left_side_6ybox	right_side_box	right_side_6ybox	close_range
0	0	0	0	0	0	1	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	1	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0

```
#Lets split the data in 65-35 for training and testing of model
X = data.iloc[:,:-1]
y = data.iloc[:,-1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.35, random_state=1)
```

Figure 13: xG Model Preparation

4.1 Gradient Boosting Classifier

As there is no discernible variation observed when attempting various numerical inputs for the parameter, it can be inferred that there is an absence of any indications that would suggest the presence of overfitting as shown in Figure 14.

```python
from hyperopt import fmin, tpe, hp, STATUS_OK, Trials

def evaluate_model(params):
    model = GradientBoostingClassifier(
                        learning_rate=params['learning_rate'],
                        min_samples_leaf=params['min_samples_leaf'],
                        max_depth = params['max_depth'],
                        max_features = params['max_features']
                        )

    model.fit(X_train, y_train)
    return {
        'learning_rate': params['learning_rate'],
        'min_samples_leaf': params['min_samples_leaf'],
        'max_depth': params['max_depth'],
        'max_features': params['max_features'],
        'train_ROCAUC': roc_auc_score(y_train, model.predict_proba(X_train)[:, 1]),
        'test_ROCAUC': roc_auc_score(y_test, model.predict_proba(X_test)[:, 1]),
        'recall': recall_score(y_test, model.predict(X_test)),
        'precision': precision_score(y_test, model.predict(X_test)),
        'f1_score': f1_score(y_test, model.predict(X_test)),
        'train_accuracy': model.score(X_train, y_train),
        'test_accuracy': model.score(X_test, y_test),
    }

def objective(params):
    res = evaluate_model(params)

    res['loss'] = - res['test_ROCAUC'] # Esta loss es la que hyperopt intenta minimizar
    res['status'] = STATUS_OK # Asi le decimos a hyperopt que el experimento salio bien
    return res

hyperparameter_space = {
        'learning_rate': hp.uniform('learning_rate', 0.05, 0.3),
        'min_samples_leaf': hp.choice('min_samples_leaf', range(15, 200)),
        'max_depth': hp.choice('max_depth', range(2, 20)),
        'max_features': hp.choice('max_features', range(3, 27))
}
```

```python
trials = Trials()
fmin(
    objective,
    space=hyperparameter_space,
    algo=tpe.suggest,
    max_evals=50,
    trials=trials
);
```

```
100%|██████████| 50/50 [14:25<00:00, 17.31s/trial, best loss: -0.8194061214247517]
```

Figure 14: xG Model using GBC

```python
model = GradientBoostingClassifier(
                        learning_rate=0.285508,
                        min_samples_leaf=99,
                        max_depth = 19,
                        max_features = 7
                        )
model.fit(X_train, y_train)
```

```
GradientBoostingClassifier(learning_rate=0.285508, max_depth=19, max_features=7,
                           min_samples_leaf=99)
```

```python
print('The test set contains {} examples (shots) of which {} are positive (goals).'.format(len(y_test), y_test.sum()))
print('The accuracy of classifying whether a shot is goal or not is {}%.'.format(round(model.score(X_test, y_test)*100),2))
print('Our classifier obtains an ROC-AUC of {}%'.format(round(roc_auc_score(y_test, model.predict_proba(X_test)[:, 1])*100),2))
```

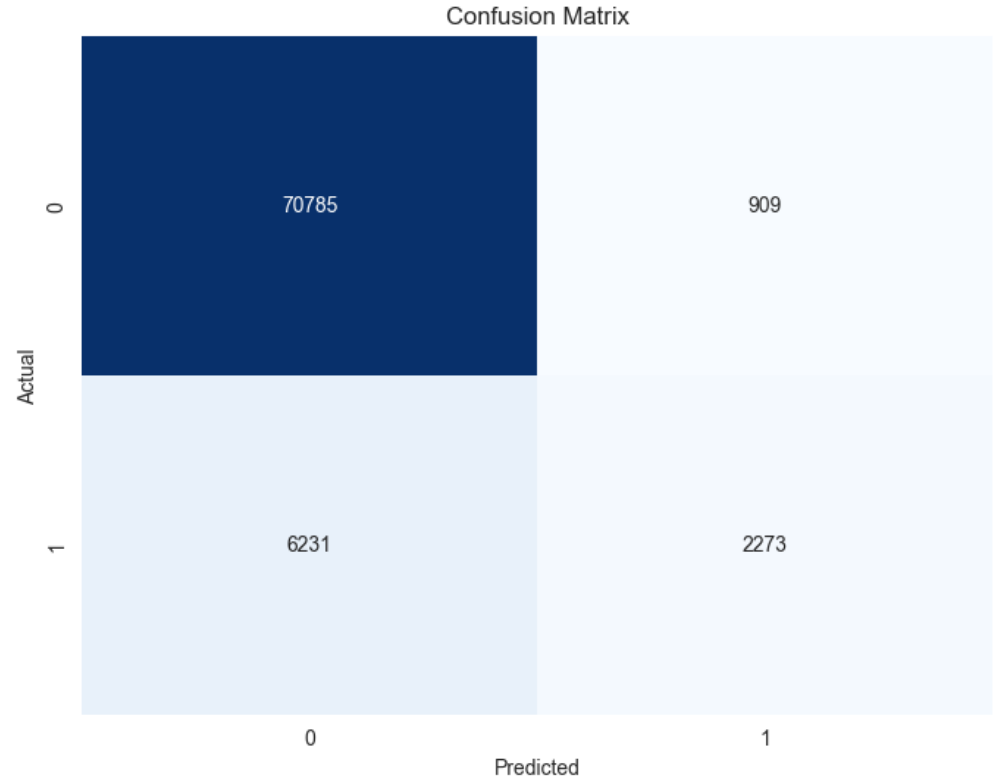
```
The test set contains 80198 examples (shots) of which 8504 are positive (goals).
The accuracy of classifying whether a shot is goal or not is 91%.
Our classifier obtains an ROC-AUC of 82%
```

```python
print('The baseline performance for PR-AUC is {}%. This is the PR-AUC that what we would get by random guessing.'.format(round(y_
print('Our model obtains an PR-AUC of {}%.'.format(round(average_precision_score(y_test, model.predict_proba(X_test)[:, 1])*100,2
print('Our classifier obtains a Cohen Kappa of {}.'.format(round(cohen_kappa_score(y_test,model.predict(X_test)),2)))
```

```
The baseline performance for PR-AUC is 0.11%. This is the PR-AUC that what we would get by random guessing.
Our model obtains an PR-AUC of 47.33%.
Our classifier obtains a Cohen Kappa of 0.35.
```

```python
print(color.BOLD + color.YELLOW + 'Confusion Matrix:\n' + color.END)
print(confusion_matrix(y_test,model.predict(X_test)))
print(color.BOLD +  color.YELLOW + '\n Report:' + color.END)
print(classification_report(y_test,model.predict(X_test)))
```

Figure 15: GBC Performance Metrics Code



Classification Report:

	precision	recall	f1-score	support
0	0.92	0.99	0.95	71694
1	0.71	0.27	0.39	8504
accuracy			0.91	80198
macro avg	0.82	0.63	0.67	80198
weighted avg	0.90	0.91	0.89	80198

Figure 16: GBC Confusion Matrix

4.2 Logistic Regression

```
model = LogisticRegression(max_iter=400)
model.fit(X_train, y_train)
```

```
LogisticRegression(max_iter=400)
```

```
print('The test set contains {} examples (shots) of which {} are positive (goals).'.format(len(y_test), y_test.sum()))
print('The accuracy of classifying whether a shot is goal or not is {}%.'.format(round(model.score(X_test, y_test)*100),2))
print('Our classifier obtains an ROC-AUC of {}%'.format(round(roc_auc_score(y_test, model.predict_proba(X_test)[:, 1])*100),2))
```

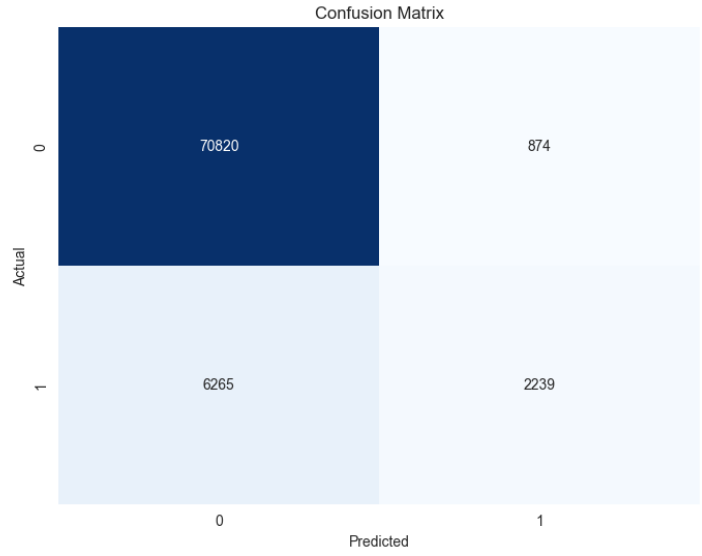
```
The test set contains 80198 examples (shots) of which 8504 are positive (goals).
The accuracy of classifying whether a shot is goal or not is 91%.
Our classifier obtains an ROC-AUC of 82%
```

```
print('The baseline performance for PR-AUC is {}%. This is the PR-AUC that what we would get by random guessing.'.format(round(y_
print('Our model obtains an PR-AUC of {}%.'.format(round(average_precision_score(y_test, model.predict_proba(X_test)[:, 1])*100,2
print('Our classifier obtains a Cohen Kappa of {}.'.format(round(cohen_kappa_score(y_test,model.predict(X_test)),2)))
```

```
The baseline performance for PR-AUC is 0.11%. This is the PR-AUC that what we would get by random guessing.
Our model obtains an PR-AUC of 47.08%.
Our classifier obtains a Cohen Kappa of 0.35.
```

```
print(color.BOLD + color.YELLOW + color.UNDERLINE + 'Confusion Matrix:\n' + color.END)
print(confusion_matrix(y_test,model.predict(X_test)))
print(color.BOLD +  color.YELLOW + color.UNDERLINE + '\n Report:' + color.END)
print(classification_report(y_test,model.predict(X_test)))
```

Figure 17: LR Performance Metrics Code




```
Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.99      0.95     71694
           1       0.72      0.26      0.39      8504

    accuracy                           0.91     80198
   macro avg       0.82      0.63      0.67     80198
weighted avg       0.90      0.91      0.89     80198
```

Figure 18: LR Confusion Matrix

4.3 Discussion

Almost precisely identical outcomes as those obtained from employing the technique of Gradient Boosting. In circumstances where this particular scenario arises, it is generally advisable to prioritize the utilization of the more simplistic model, specifically in this instance, the Logistic Regression method. Nevertheless, it should be noted that there

exists a total of 39 objectives that were accurately recognized as such through the application of Gradient Boosting, but regrettably, were not successfully captured by the Logistic Regression approach. Although this disparity may not be extensively significant, I shall ultimately opt for employing the Gradient Boosting technique due to this particular reason.

4.4 Player Analysis using xG Model

```
shots['prediction'] = model.predict_proba(X)[:, 1]
shots['difference'] = shots['prediction'] - shots['is_goal']
```

Figure 19: Player Analysis

Which players are the best finishers?

```
players = shots.groupby('player').sum().reset_index()
players.rename(columns={'is_goal': 'trueGoals', 'prediction': 'expectedGoals'}, inplace=True)
players.expectedGoals = round(players.expectedGoals,2)
players.difference = round(players.difference,2)
players['ratio'] = players['trueGoals'] / players['expectedGoals']
```

```
print(round(players.expectedGoals.corr(players.trueGoals),3))
```

```
0.977
```

Best Finishers

```
show = players.sort_values(['difference', 'trueGoals']).reset_index(drop=True)
show['rank'] = show.index+1
show = show[['rank', 'player', 'difference', 'trueGoals', 'expectedGoals']].head(10)
show.head(5)
```

	rank	player	difference	trueGoals	expectedGoals
0	1	Lionel Messi	-58.80	205	146.20
1	2	Zlatan Ibrahimovic	-33.67	153	119.33
2	3	Cristiano Ronaldo	-32.37	198	165.63
3	4	Luis Suarez	-31.74	96	64.26
4	5	Gonzalo Higuain	-31.72	118	86.28

```
sns.set_style("dark")
fig, ax = plt.subplots(figsize=[12,5])
ax = sns.barplot(x=abs(show['difference']), y=show['player'], palette='viridis', alpha=0.9)
ax.set_xticks(np.arange(0,65,5))
ax.set_xlabel(xlabel='Diff. between Goals Scored and Goals Expected', fontsize=12)
ax.set_ylabel(ylabel='')
ax.set_yticklabels(labels=ax.get_yticklabels(), fontsize=12)
plt.title("Best Finishers: most goals on top of expected", fontsize=20, fontfamily='serif')
ax.grid(color='black', linestyle='-', linewidth=0.1, alpha=0.8, axis='x')
plt.show()
```

Figure 20: Best Finisher

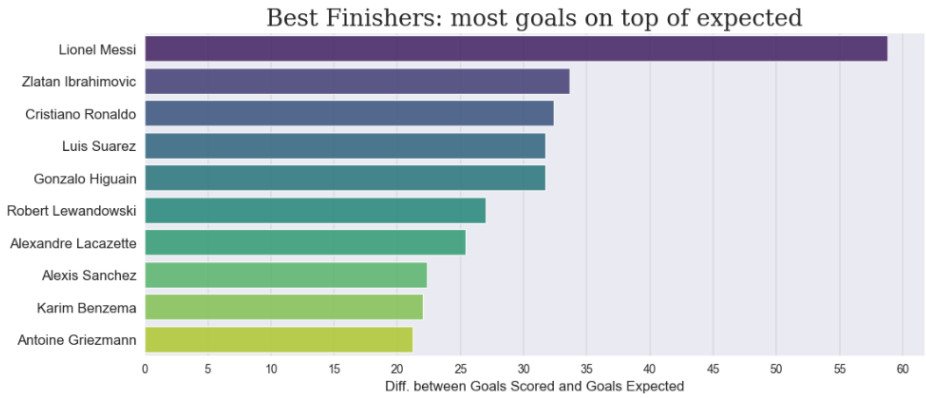


Figure 21: Best Finisher

5 Future Work

Exploring advanced defensive metrics, such as the count of defenders and defensive pressure, holds significant promise for enhancing predictive capabilities. The augmentation of spatial granularity through the incorporation of precise shot coordinates has the potential to further elevate model accuracy. Additionally, delving into Deep Learning methodologies, particularly utilizing structures like Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs), offers a valuable avenue for capturing intricate spatial-temporal dependencies within the data, providing a more nuanced understanding of defensive dynamics in sports analytics.

References

Baumann, A. (2022). *A multi-stage clustering algorithm to re-evaluate basketball positions and performance analysis*, Master's thesis, Dublin, National College of Ireland. Submitted.
URL: *https://norma.ncirl.ie/6082/*

Chavan, A. (2019). *Recruitment of suitable football player by using machine learning techniques*, Master's thesis, Dublin, National College of Ireland. Submitted.
URL: *https://norma.ncirl.ie/4307/*

Gibney, R. (2022). *Using supervised learning techniques to predict kicking outcomes in the nfl*, Master's thesis, Dublin, National College of Ireland. Submitted.
URL: *https://norma.ncirl.ie/6589/*

Gorman, D. (2017). *Sports Analytics: Analysis of the National Football League*, PhD thesis, Dublin, National College of Ireland. Submitted.
URL: *https://norma.ncirl.ie/2661/*

Kumar, S. (2020). *Artificial neural network for betting rate in football*, Master's thesis, Dublin, National College of Ireland. Submitted.
URL: *https://norma.ncirl.ie/4404/*

Selvaraj, S. (2016). *Analysis of player ratings based on intrinsic factors to support team selection*, Master's thesis, Dublin, National College of Ireland. Submitted.
URL: *https://norma.ncirl.ie/2498/*