National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Karthik Nousher
Student ID: 22138668

School of Computing
National College of Ireland

Supervisor: Dr. Catherine Mulwa

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Karthik Nousher
Student ID:	22138668
Programme:	Data Analytics **Year:** 2023
Module:	MSc Research Project
Lecturer:	Dr. Catherine Mulwa
Submission Due Date:	14/12/2023
Project Title:	Configuration Manual
Word Count:	654 **Page Count:** 6

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: Karthik Nousher

Date: 14/12/2023

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

Karthik Nousher
Student ID: 22138668

1 Introduction

This is the configuration manual describing guidelines on how to implement the research project which is 'Basketball Performance Prediction Models and Team Efficiency Factors'. The hardware, software requirements and how to run the code are specified in this report.

2 Hardware Configuration

The hardware requirements for the project are given below.

- Processor: Intel(R) Core (TM) i7-7700HQ CPU @ 2.80GHz 2.81 GHz
- RAM: 8GB to 16GB of RAM is usually preferred
- Storage: 128/ 500 GB SSD
- OS: Windows 10 Home Single Language version 20H2

3 Software Configuration

- Jupyter Notebook - Version: 6.4.6
- Microsoft Excel
- Python

4 Environment Setup

4.1 Setup Of Code

- This section will describe the steps to set up the Project. Anaconda is downloaded and installed in the system. A new environment is created, and the latest packages are installed. Jupyter Notebook and CMD prompt for Anaconda is also installed. Installed tools are visible in Figure 2.
- Once the installation is complete, launch Anaconda Navigator from the Start menu. Click on the Jupyter Notebook icon to launch the application.

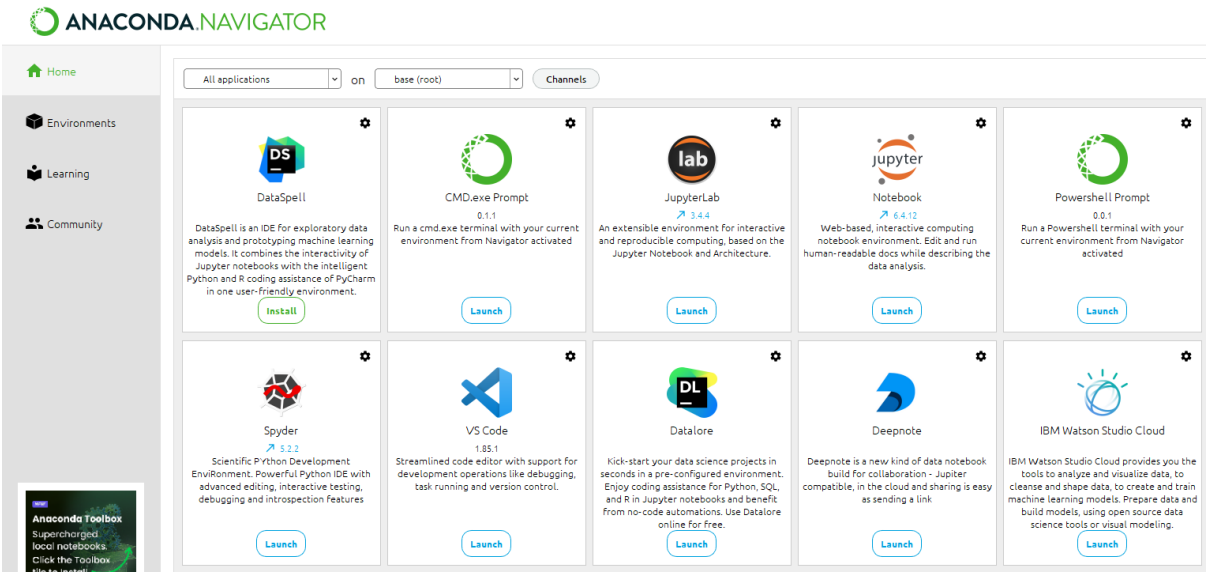

Figure 1 Anaconda Navigator

- Download python from official python website.
- Once the installation is complete, check the version by using the command “python --version”.

```
C:\Users\karth>py --version
Python 3.9.0

C:\Users\karth>
```

Figure 2 Python Version

- Open downloads Folder , find the folder named Basketball_Performance_Prediction
- There are three python files and dataset folder that contains all datasets, Open the “Data_Processing” file and run the commands similarly do it for the rest of the two files.

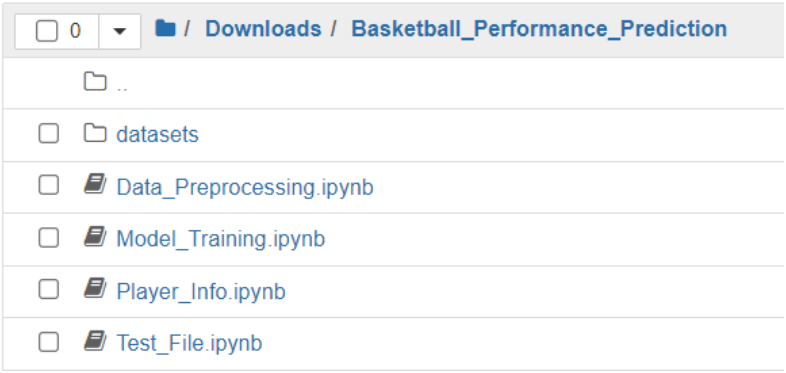

Figure 3 Folder Structure

5 Data Selection

The dataset for this research topic was obtained from Kaggle , Game Data The dataset includes game data, teams involved, scores, game location etc., Secondly, Player data includes player information, position, weight, performance metrics.

6 Implementation

6.1 Data pre-processing

- The three datasets are loaded and saved and are then converted to data frames using the panda's library as shown in Figure 4, these data frames are then combined and then stored in a dictionary called "team_data" as shown in Figure 5

```
In [2]: games = pd.read_csv('input/games.csv')
        details = pd.read_csv('input/games_details.csv')
        teams = pd.read_csv('input/teams.csv')
        ranking = pd.read_csv('input/ranking.csv')
```

Figure 2 Load Data

```
teams_ = ranking['TEAM'].values
games_played = ranking['G'].values
wins = ranking['W'].values
losses = ranking['L'].values

team_data = {}
for team, played, win, loss in zip(teams_, games_played, wins, losses):
    if team not in team_data:
        team_data[team] = {'games_played': 0, 'wins': 0, 'losses': 0}

    team_data[team]['games_played'] += played
    team_data[team]['wins'] += win
    team_data[team]['losses'] += loss
unique_teams = list(team_data.keys())
games_played_unique = [team_data[team]['games_played'] for team in unique_teams]
wins_unique = [team_data[team]['wins'] for team in unique_teams]
losses_unique = [team_data[team]['losses'] for team in unique_teams]
```

Figure 3 Combine Data

```
data_scaled.head()
```

	TEAM_ID	NEXT_SEASON	FGM	FGA	FG_PCT	FG3M	FG3A	FG3_PCT	FTM	FTA	FT_PCT	OREB	DREB	F
0	1610612737	2004	-1.477751	-1.321883	-1.565368	-1.192284	-1.150786	-1.789278	-0.433451	-0.563153	-0.871429	0.125248	-1.034993	-0.841
1	1610612737	2005	-1.129855	-1.039164	-0.842839	-1.612394	-1.509828	-1.585357	-0.808677	-0.406095	-0.708392	0.959516	-1.462238	-0.929
2	1610612737	2006	-0.635347	-0.804455	-0.826860	-1.088186	-1.221486	-1.087927	0.510300	0.579093	0.210326	1.086777	-1.352688	-0.794
3	1610612737	2007	-0.861479	-0.819124	-0.619769	-1.329842	-1.282481	-1.322163	0.741500	0.676184	0.198713	0.563592	-1.076074	-0.731
4	1610612737	2008	0.013228	0.090376	-0.354119	-1.140235	-1.159104	-1.408916	1.742103	1.589981	1.071231	1.426140	-0.098340	0.387

Figure 6 Pre processed Data

- Unwanted cells are removed and cells with null values are removed, as shown in Figure 5 and pre combined data is shown in Figure 6

```
ranking = ranking.drop(labels=['season', 'year'], axis=1)

ra = ranking.copy(deep=False)
ra = ra.groupby(['TEAM_ID','SEASON_ID'])['G','W'].max()
ra = pd.DataFrame(ra)
ra.reset_index(inplace=True)

data.drop(columns=['SEASON_ID','SEASON'], inplace=True)
data.dropna(inplace=True)

data = swap_columns(data, 'W', 'NEXT_SEASON')
data = data.astype({'NEXT_SEASON': 'int64'})
data.rename(columns={'W' : 'NEXT_W'}, inplace=True)
data.reset_index(inplace=True)
data.drop(columns=['index'], inplace=True)
```

Figure 4 Data Cleaning

- The three datasets are then combined to form a single dataset as shown in Figure 7

```
teams_ = ranking['TEAM'].values
games_played = ranking['G'].values
wins = ranking['W'].values
losses = ranking['L'].values

team_data = {}
for team, played, win, loss in zip(teams_, games_played, wins, losses):
    if team not in team_data:
        team_data[team] = {'games_played': 0, 'wins': 0, 'losses': 0}

    team_data[team]['games_played'] += played
    team_data[team]['wins'] += win
    team_data[team]['losses'] += loss
```

Figure 5 Combine Data

- The cleaned data is then stored to a different file named "preprocessed_data.csv".
- Open the file named "ModlTraining.py", here "preprocessed_data.csv" is opened and all the model training is done here.

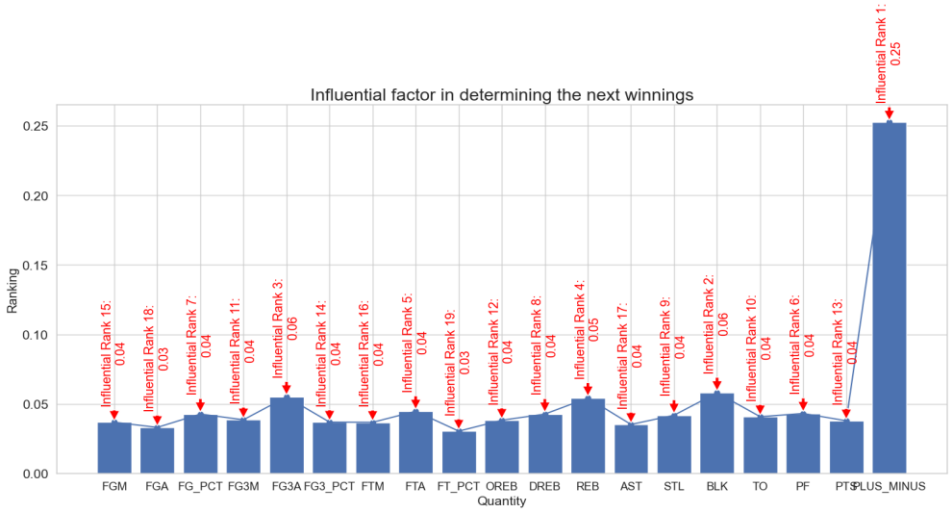

Figure 8 Influential Factors.

- Line number 79 of data_Preprocessing gives the visualization as shown in Figure 8

6.2 Hyper Parameterised Tuning

- Hyperparameter tuning is done using the python library GridSearchCV, it trains the dataset on another data and evaluates its performance and then applied to the subset. This process is done to improve the performance of the model as shown in Figure 6.

```
from sklearn.linear_model import LinearRegression

lr_model = LinearRegression()

param_grid = {
    'fit_intercept': [False, True]
}

grid_search = GridSearchCV(estimator=lr_model, param_grid=param_grid,
                           scoring='neg_mean_squared_error', cv=5)

grid_search.fit(X_train.values, y_train['NEXT_W'].values)
```

Figure 6 Hyper Parameter for Linear Regression

- Similarly, it is done for decision tree and random forest.
- Model Comparison metrics is done, three different comparison metrics are used here mae, rmse and mse. Basically, it is done to understand the best model that performs which can be found out from these metrics.

```
chart_data = [rf_mse, lr_mse, dt_mse]
chart_labels = ['RandomForestRegressor', 'LinearRegression', 'DecisionTreeRegressor']
min_index = chart_data.index(min(chart_data))

with plt.style.context(style='bmh'):
    plt.figure(figsize=(6,5))
    plt.rcParams['font.size'] = 10
    plt.bar(x=chart_labels, height=chart_data)
    plt.annotate('Lowest', xy=(min_index, min(chart_data)),
                 xytext=(min_index, min(chart_data) + 0.1),
                 arrowprops=dict(facecolor='red', shrink=0.05),
                 color='red', fontsize=15)

    plt.title(label="Comparison of 'mean squared error' loss among the models")
    plt.xlabel(xlabel='Models')
    plt.ylabel(ylabel='Mean Squared Error Loss')
    plt.xticks(rotation=90)
    plt.show()
```

Figure 7 Performance Comparison (using mse, mae, rmse)

- After the comparison is done, it is then saved as a pickle file under the models folder.

```
with open(file="models/LinearRegressor_model.pkl", mode='wb') as file:
    pickle.dump(obj=lr_model, file=file)
```

6.3 User Input

- From the previous step, we get linear regression model as the best performing.
- Taking user input of specific year and team to do model evaluation onto the specified data which is most relevant. Which is shown in Figure 8.

```
user_input_year = input(f"Enter appropriate year {df['NEXT_SEASON'].unique().tolist()}")
user_input_year = int(user_input_year)
```

```
Enter appropriate year [2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 20
0, 2021, 2022]2004
```

```
user_df = df.loc[df['NEXT_SEASON']==user_input_year]
user_df = user_df.reset_index(drop=True)
user_df.head()
```

Figure 8 User Input

- The true and predicted efficiency of the team which is chosen above using the user input is then calculated based on the number of wins, total games played, and model predictions. The following result is then stored as data frame named **result** as shown in Figure 9.

```
result.head()
```

	Team	True Wins	Predicted Wins	True Efficiency (%)	Predicted Efficiency (%)
0	Hawks	13	35	15.853659	42.682927

Figure 9 Result