Configuration Manual

MSc Research Project
Msc Data Analytics

Aswin Surendran
21225052

School of Computing
National College of Ireland

Supervisor: Musfira Jilani

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Aswin Surendran

Student ID: 21225052

Programe : MSc in Data Analytics **Year:** 2023

Module: MSc Research Project

Lecturer: Musfira Jilani

Submission Due Date: 14/12/2023

Project Title: Analyzing the Most Influential Factor in Formula One: A Deep Learning Approach for Predicting Driver and Team Ranks

WordCount:1010 **Page Count: 11**

I hereby certify that the information contained in this 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: :Aswin surendran

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

Aswin Surendran
21225052

1 Introduction

The research project, titled "Analyzing the Most Influential Factors in Formula One Racing: A Deep Learning Approach for Predicting Driver and Team Ranks," aims to leverage deep learning techniques to predict Formula One driver and team ranks based on various factors

2 Hardware Configuration

1. Processor: 0.8 GHz Dual-Core Intel Core i5
2. RAM: 4GB to 8GB of RAM is usually preferred.
3. Storage: Minimum 124GB of storage space is required.
4. Operating System: macOS Monterey

3 Software Configuration

1. Jupyter Notebook: Version 6.4.12
2. Microsoft Excel
3. Python

4 Environment Setup

4.1 Python Setup

1. Download Python from the Official Python Website:
    a. Go to the official Python website.
2. Install the file

    a. Run the Installer
3. All required files will be copied during installation.
4. Use the command "python —version" to verify the version after the installation is finished, as shown in Figure 1

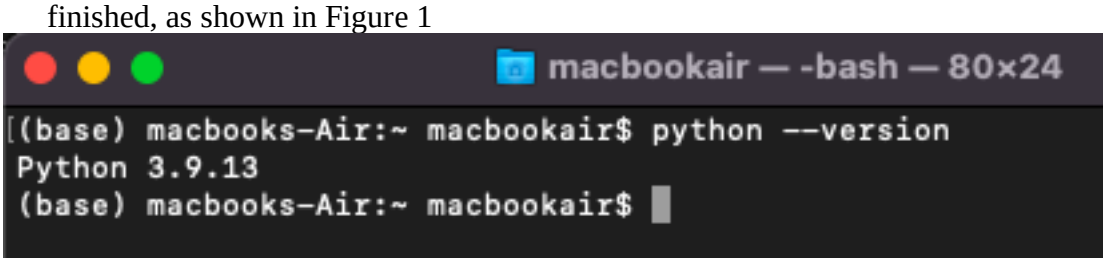

Figure 1 Python Version check

4.2 Setup Of Anaconda

• The procedures for configuring the Jupyter Notebook are covered in this section. The system downloads and installs Anaconda. The most recent packages are installed, and a new environment is created. Also installed are Jupyter Notebook and the Anaconda CMD prompt. In Figure 2, installed tools are visible.

• To set the Anaconda Navigator, simply follow the simple instructions after the download is finished.

•Anaconda Navigator can be launched from the Start menu after the installation is finished. To open the programme, click the Jupyter Notebook icon.

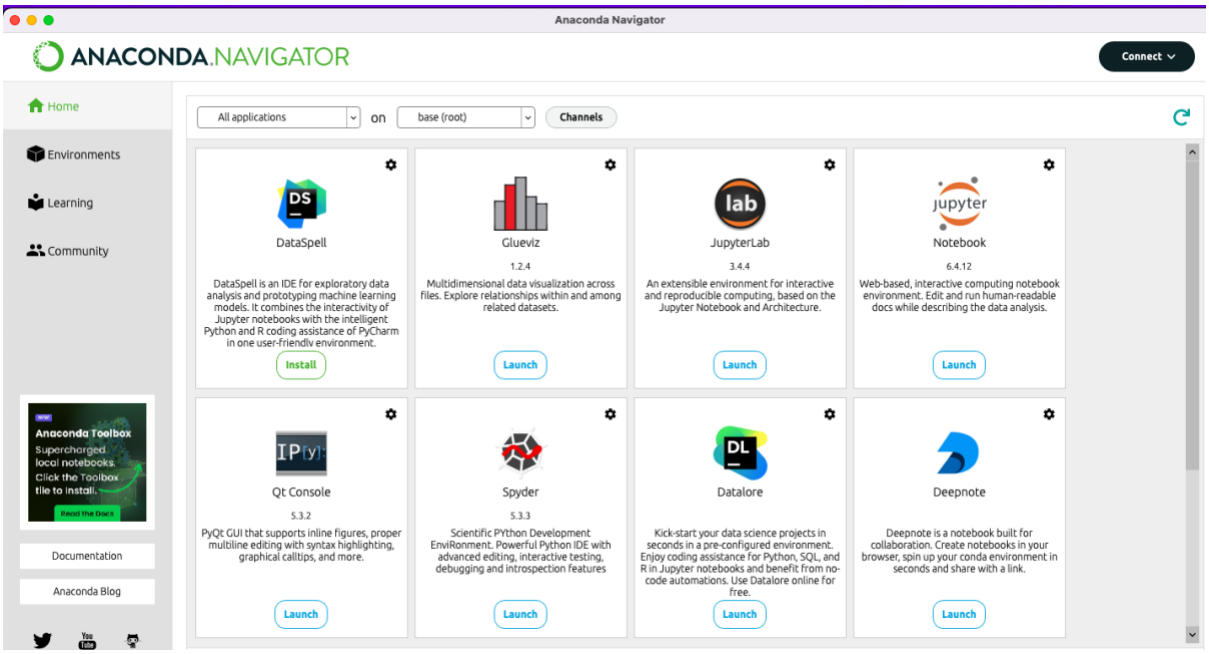

Figure 2 Anaconda Navigator

5 Data Selection

Data for this research topic was gathered from the official Formula One website, specifically extracting race results spanning the years 2019, 2020, 2021, and 2022. The dataset encompasses various attributes, including Track, Car Number (No), Driver, Team, Starting Grid, Laps, Fastest Lap, and the respective year of each race.

The data collection utilized the following source links for each respective year:
- 2019: https://www.formula1.com/en/results.html/2019/races.html
- 2020 :https://www.formula1.com/en/results.html/2020/races.html
- 2021: https://www.formula1.com/en/results.html/2021/races.html
- 2022: https://www.formula1.com/en/results.html/2022/races.html

These datasets serve as the foundation for the research, providing valuable information for the analysis and prediction of Formula One driver and team ranks.

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 3, these data frames are then combined and then stored in a dictionary called "team_data" as shown in Figure 4

Data Loading

```
# reading data
df = pd.read_csv("merged_data.csv")
```

```
df.head(10)
```

	Track	Position	No	Driver	Team	Starting Grid	Laps	Points	Fastest Lap	year
0	Australia	1	77	Valtteri Bottas	Mercedes	2.0	58	26.0	Yes	2019
1	Australia	2	44	Lewis Hamilton	Mercedes	1.0	58	18.0	No	2019
2	Australia	3	33	Max Verstappen	Red Bull Racing Honda	4.0	58	15.0	No	2019
3	Australia	4	5	Sebastian Vettel	Ferrari	3.0	58	12.0	No	2019
4	Australia	5	16	Charles Leclerc	Ferrari	5.0	58	10.0	No	2019
5	Australia	6	20	Kevin Magnussen	Haas Ferrari	7.0	58	8.0	No	2019
6	Australia	7	27	Nico Hulkenberg	Renault	11.0	57	6.0	No	2019
7	Australia	8	7	Kimi Raikkönen	Alfa Romeo Racing Ferrari	9.0	57	4.0	No	2019
8	Australia	9	18	Lance Stroll	Racing Point BWT Mercedes	16.0	57	2.0	No	2019
9	Australia	10	26	Daniil Kvyat	Scuderia Toro Rosso Honda	15.0	57	1.0	No	2019

Figure 3 Load Data

```
# Dropping specific columns from both 2021 and 2022 DataFrames
df_2021 = df_2021.drop(labels=['Time/Retired', 'Fastest Lap'], axis=1)
df_2022 = df_2022.drop(labels=['Time/Retired', 'Fastest Lap'], axis=1)

df2 = pd.concat(objs=[df_2021, df_2022], ignore_index=True)
df2.head()
```

Figure 4 Combine Data

- Unwanted cells are removed and cells with null values are removed, as shown in Figure 5

```
df.isnull().sum()
```

```
Track           0
Position        0
No              0
Driver          0
Team            0
Starting Grid   1
Laps            0
Points          0
Fastest Lap     0
year            0
dtype: int64
```

```
# Handling missing values in 'Starting Grid' by filling with the median
df['Starting Grid'].fillna(value=df['Starting Grid'].median(), inplace=True)

#Checking for remaining missing values and
df.isnull().sum()
```

```
Track           0
Position        0
No              0
Driver          0
Team            0
Starting Grid   0
Laps            0
Points          0
Fastest Lap     0
year            0
dtype: int64
```

Figure 5 Data Cleaning

- The cleaned data is then stored to a different file named "preprocessed_data.csv".

6.2 Code Execuation

- Open "preprocessed_data.csv"
- Dataset is the deployed for taining and testing purpose as shown in Figure 6 and 7

```
In [18]: # Create training feature set (X_train) and target variable (y_train)
         X_train = df_train.drop(labels='Position', axis=1)
         X_train.head()
```

Out[18]:

	Track	No	Driver	Starting Grid	Laps	Points	Fastest Lap
0	0	77	0	2.0	58	26.0	0
1	0	44	1	1.0	58	18.0	1
2	0	33	2	4.0	58	15.0	1
3	0	5	3	3.0	58	12.0	1
4	0	16	4	5.0	58	10.0	1

Figure 6 :Training

```
In [20]: # Create testing feature set (X_test) and target variable (y_test)
         X_test = df_test.drop(labels='Position', axis=1)
         X_test.head()
```

Out[20]:

	Track	No	Driver	Starting Grid	Laps	Points	Fastest Lap
1200	1	16	4	1.0	57	26.0	0
1201	1	55	19	3.0	57	18.0	1
1202	1	44	1	5.0	57	15.0	1
1203	1	63	15	9.0	57	12.0	1
1204	1	20	5	7.0	57	10.0	1

Figure 7 : Testing

Time Series Prediction with LSTM and GRU For Individual Rank (Driver)):Train and Evaluation

Model Training Parameters:LSTU

In the model training configuration, we set the number of training epochs to 50, the batch size to 32, and provide paths to the testing input and output data for validation, as showin in figure

```
In [26]: # model training
         history = model.fit(x_train, y_train, epochs=50, batch_size=32, validation_data=(x_test, y_test), shuffle=False)
33/33 [==============================] - 10s 301ms/step - loss: 2.5113 - val_loss: 2.3555
Epoch 42/50
33/33 [==============================] - 13s 381ms/step - loss: 2.4898 - val_loss: 2.4310
Epoch 43/50
33/33 [==============================] - 12s 355ms/step - loss: 2.4365 - val_loss: 2.4661
Epoch 44/50
33/33 [==============================] - 12s 374ms/step - loss: 2.6091 - val_loss: 2.4702
Epoch 45/50
33/33 [==============================] - 11s 349ms/step - loss: 2.4150 - val_loss: 2.2893
Epoch 46/50
33/33 [==============================] - 11s 325ms/step - loss: 2.3712 - val_loss: 2.3840
Epoch 47/50
33/33 [==============================] - 10s 296ms/step - loss: 2.4280 - val_loss: 2.4634
Epoch 48/50
33/33 [==============================] - 10s 311ms/step - loss: 2.3501 - val_loss: 2.3898
Epoch 49/50
33/33 [==============================] - 10s 298ms/step - loss: 2.3179 - val_loss: 2.5350
Epoch 50/50
33/33 [==============================] - 10s 304ms/step - loss: 2.4809 - val_loss: 2.5273
```

Figure 8:LSTM :Training Code Configuration

Evaluation and Model Saving

In this part of the analysis, we compute key metrics, namely Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE), to assess the performance of the Long Short-Term Memory (LSTM) model.

The trained LSTM model is saved to a file named "LongShortTermMemory_model_individual.h5," as showin in figure 9 .This file is stored in the "models" directory

```
In [29]: # collecting mse, rmse and mae values for LSTM model
         lstm_mse = mean_squared_error(y_true=y_test.values.ravel(), y_pred=[int(i[0]) for i in model_prediction])
         lstm_rmse = np.sqrt(mean_squared_error(y_true=y_test.values.ravel(), y_pred=[int(i[0]) for i in model_prediction]))
         lstm_mae = mean_absolute_error(y_true=y_test.values.ravel(), y_pred=[int(i[0]) for i in model_prediction])

In [30]: # model saving
         model.save("models/LongShortTermMemory_model_individual.h5")
```

Figure 9:LSTM Model saving

Model Training Parameters:GRU

The same procedure done in LSTM is followed in this , below figure are code.

```
In [35]: # GRU model training
         history = model.fit(x_train, y_train, epochs=50, batch_size=32, validation_data=(x_test, y_test), shuffle=False)
```

```
33/33 [==============================] - 9s 283ms/step - loss: 2.4142 - val_loss: 2.3310
Epoch 42/50
33/33 [==============================] - 10s 289ms/step - loss: 2.3741 - val_loss: 2.2338
Epoch 43/50
33/33 [==============================] - 9s 268ms/step - loss: 2.3623 - val_loss: 2.2410
Epoch 44/50
33/33 [==============================] - 10s 299ms/step - loss: 2.3522 - val_loss: 2.2907
Epoch 45/50
33/33 [==============================] - 10s 305ms/step - loss: 2.3426 - val_loss: 2.3832
Epoch 46/50
33/33 [==============================] - 11s 326ms/step - loss: 2.3013 - val_loss: 2.3380
Epoch 47/50
33/33 [==============================] - 8s 244ms/step - loss: 2.2750 - val_loss: 2.3959
Epoch 48/50
33/33 [==============================] - 9s 274ms/step - loss: 2.3011 - val_loss: 2.3279
Epoch 49/50
33/33 [==============================] - 14s 439ms/step - loss: 2.2478 - val_loss: 2.3881
Epoch 50/50
33/33 [==============================] - 12s 351ms/step - loss: 2.2996 - val_loss: 2.4489
```

Figure 9: GRU :Training Code Configuration

Evaluation and Model Saving

The trained GRU model is saved to a file named " GatedRecurrentUnit_model_individual.h5," as showin in figure 10 .This file is also stored in the "models" directory

```
In [38]: # collecting mse, rmse and mae values for GRU model
         gru_mse = mean_squared_error(y_true=y_test.values.ravel(), y_pred=[int(i[0]) for i in model_prediction])
         gru_rmse = np.sqrt(mean_squared_error(y_true=y_test.values.ravel(), y_pred=[int(i[0]) for i in model_prediction]))
         gru_mae = mean_absolute_error(y_true=y_test.values.ravel(), y_pred=[int(i[0]) for i in model_prediction])

In [39]: # saving the trained GRU model
         model.save("models/GatedRecurrentUnit_model_individual.h5")
```

Figure 10: GRU Model saving

Comparing MSE, RMSE, and MAE: LSTM vs GRU

In this section ,Evaluation metrics of LSTM values and GRU valuesa are used for model comparisons using Three distinct comparison metrics such as MAE, RMSE, and MSE are employed , to understand the best model

```
In [43]: # Comparing the mse, rmse and mae values between LSTM and GRU model

         bar_width = 0.35
         r1 = np.arange(len(labels))
         r2 = [x + bar_width for x in r1]

         with plt.style.context(style="fivethirtyeight"):
             plt.figure(figsize=(18,8))
             plt.rcParams['font.size']=15
             plt.bar(r1, lstm_values, color='blue', width=bar_width, edgecolor='grey', label='LSTM')
             plt.bar(r2, gru_values, color='green', width=bar_width, edgecolor='grey', label='GRU')
             for i, value in enumerate(lstm_values):
                 plt.text(i, value, f'{value:.2f}', ha='center', va='bottom', color='black')
             for i, value in enumerate(gru_values):
                 plt.text(i + bar_width, value, f'{value:.2f}', ha='center', va='bottom', color='black')
             plt.xlabel('Metrics')
             plt.ylabel('Values')
             plt.title('LSTM vs GRU Model Evaluation Metrics')
             plt.xticks([r + bar_width/2 for r in range(len(labels))], labels)
             plt.legend()
             plt.show()
```

Figure 10: Performance Metrics Comparison: LSTM vs GRU Models

Time Series Prediction with LSTM and GRU For Team Rank predicition :Train and Evaluation

Model Training Parameters:LSTU

```
[26]: # model training
history = model.fit(x_train, y_train, epochs=50, batch_size=32, validation_data=(x_test, y_test), shuffle=False)
```

```
33/33 [==============================] - 14s 414ms/step - loss: 3.0305 - val_loss: 2.5044
Epoch 42/50
33/33 [==============================] - 10s 308ms/step - loss: 3.1814 - val_loss: 2.4643
Epoch 43/50
33/33 [==============================] - 11s 338ms/step - loss: 2.9987 - val_loss: 2.4335
Epoch 44/50
33/33 [==============================] - 13s 397ms/step - loss: 2.9690 - val_loss: 2.3770
Epoch 45/50
33/33 [==============================] - 9s 271ms/step - loss: 3.0121 - val_loss: 2.4319
Epoch 46/50
33/33 [==============================] - 10s 308ms/step - loss: 2.9957 - val_loss: 2.4095
Epoch 47/50
33/33 [==============================] - 9s 271ms/step - loss: 3.0631 - val_loss: 2.4872
Epoch 48/50
33/33 [==============================] - 10s 298ms/step - loss: 2.9298 - val_loss: 2.3519
Epoch 49/50
33/33 [==============================] - 15s 461ms/step - loss: 2.9139 - val_loss: 2.4218
Epoch 50/50
33/33 [==============================] - 13s 385ms/step - loss: 2.9181 - val_loss: 2.4286
```

Figure 11

Evaluation and Model Saving

```
[29]: # collecting mse, rmse and mae values for LSTM model
lstm_mse = mean_squared_error(y_true=y_test.values.ravel(), y_pred=[int(i[0]) for i in model_prediction])
lstm_rmse = np.sqrt(mean_squared_error(y_true=y_test.values.ravel(), y_pred=[int(i[0]) for i in model_prediction]))
lstm_mae = mean_absolute_error(y_true=y_test.values.ravel(), y_pred=[int(i[0]) for i in model_prediction])
```

```
[30]: # model saving
model.save("models/LongShortTermMemory_model_team.h5")
```

Figure 12

Model Training Parameters:GRU

```
In [35]: # GRU model training
history = model.fit(x_train, y_train, epochs=50, batch_size=32, validation_data=(x_test, y_test), shuffle=False)
```

```
33/33 [==============================] - 8s 230ms/step - loss: 2.6829 - val_loss: 2.7202
Epoch 42/50
33/33 [==============================] - 7s 202ms/step - loss: 2.6829 - val_loss: 2.7329
Epoch 43/50
33/33 [==============================] - 9s 271ms/step - loss: 2.6860 - val_loss: 2.8361
Epoch 44/50
33/33 [==============================] - 8s 233ms/step - loss: 2.6296 - val_loss: 3.5820
Epoch 45/50
33/33 [==============================] - 9s 268ms/step - loss: 2.6865 - val_loss: 2.6597
Epoch 46/50
33/33 [==============================] - 10s 294ms/step - loss: 2.6543 - val_loss: 2.9579
Epoch 47/50
33/33 [==============================] - 8s 255ms/step - loss: 2.7301 - val_loss: 2.6077
Epoch 48/50
33/33 [==============================] - 7s 205ms/step - loss: 2.7816 - val_loss: 2.7598
Epoch 49/50
33/33 [==============================] - 8s 251ms/step - loss: 2.6320 - val_loss: 2.8060
Epoch 50/50
33/33 [==============================] - 8s 257ms/step - loss: 2.5870 - val_loss: 3.7786
```

Figure 13

Evaluation and Model Saving

```
[38]: # collecting mse, rmse and mae values for GRU model
gru_mse = mean_squared_error(y_true=y_test.values.ravel(), y_pred=[int(i[0]) for i in model_prediction])
gru_rmse = np.sqrt(mean_squared_error(y_true=y_test.values.ravel(), y_pred=[int(i[0]) for i in model_prediction]))
gru_mae = mean_absolute_error(y_true=y_test.values.ravel(), y_pred=[int(i[0]) for i in model_prediction])

[39]: # saving the trained GRU model
model.save("models/GatedRecurrentUnit_model_team.h5")
```

Figure 14

Comparing MSE, RMSE, and MAE: LSTM vs GRU

```
In [40]: # Comparing the mse, rmse and mae values between LSTM and GRU model

labels = ['MSE', 'RMSE', 'MAE']
lstm_values = [lstm_mse, lstm_rmse, lstm_mae]
gru_values = [gru_mse, gru_rmse, gru_mae]

bar_width = 0.35
r1 = np.arange(len(labels))
r2 = [x + bar_width for x in r1]

with plt.style.context(style="fivethirtyeight"):
    plt.figure(figsize=(18,8))
    plt.rcParams['font.size']=15
    plt.bar(r1, lstm_values, color='blue', width=bar_width, edgecolor='grey', label='LSTM')
    plt.bar(r2, gru_values, color='brown', width=bar_width, edgecolor='grey', label='GRU')
    for i, value in enumerate(lstm_values):
        plt.text(i, value, f'{value:.2f}', ha='center', va='bottom', color='black')
    for i, value in enumerate(gru_values):
        plt.text(i + bar_width, value, f'{value:.2f}', ha='center', va='bottom', color='black')
    plt.xlabel('Metrics')
    plt.ylabel('Values')
    plt.title('LSTM vs GRU Model Evaluation Metrics')
    plt.xticks([r + bar_width/2 for r in range(len(labels))], labels)
    plt.legend()
    plt.show()
```

Figure 15

The chosen approach involves utilizing the LSTM model for predicting both driver and team rankings, as determined through a thorough examination of the comparison matrix

6.3 User interface

1. Driver Rank Prediction User Interface

Then the pre trained model is loaded from model directory for individual and team predictions from the specified file paths.

```
[2]: # Load the best pre-trained individual model from the specified file path
individual_model = load_model("models/LongShortTermMemory_model_individual.h5")

# Load the best pre-trained team model from the specified file path
team_model = load_model("models/LongShortTermMemory_model_team.h5")
```

Figure 16

As shown in figure 17, in the given scenario the user inputs details for a race in Bahrain, where driver (car number) starts from the nd position, completes laps, earns points, and sets the fastest lap. The machine learning model processes this information and predicts that will finish in position

```
Enter track : ['Australia', 'Bahrain', 'China', 'Azerbaijan', 'Spain', 'Monaco', 'Canada', 'France', 'Austria', 'G
eat Britain', 'Germany', 'Hungary', 'Belgium', 'Italy', 'Singapore', 'Russia', 'Japan', 'Mexico', 'United States',
'Brazil', 'Abu Dhabi', 'Styria', 'Tuscany', 'Eifel', 'Portugal', 'Emilia Romagna', 'Turkey', 'Sakhir', 'Netherland
s', 'Qatar', 'Saudi Arabia', 'Miami']Abu Dhabi
Enter NO:16
Enter driver: ['Valtteri Bottas', 'Lewis Hamilton', 'Max Verstappen', 'Sebastian Vettel', 'Charles Leclerc', 'Kevi
Magnussen', 'Nico Hulkenberg', 'Kimi Raikkönen', 'Lance Stroll', 'Daniil Kvyat', 'Pierre Gasly', 'Lando Norris', '
ergio Perez', 'Alexander Albon', 'Antonio Giovinazzi', 'George Russell', 'Robert Kubica', 'Romain Grosjean', 'Dani
l Ricciardo', 'Carlos Sainz', 'Esteban Ocon', 'Nicholas Latifi', 'Jack Aitken', 'Pietro Fittipaldi', 'Yuki Tsunod
a', 'Mick Schumacher', 'Fernando Alonso', 'Nikita Mazepin', 'Guanyu Zhou', 'Nyck De Vries']Charles Leclerc
Enter starting grid:3
Enter laps:55
Enter point:15
Enter fastest lap: ['Yes', 'No']No
```

Figure 17

The Figure 18 shows the output of the predicted output:

```
1/1 [==============================] - 3s 3s/step
Predicted Position: 3
```

Figure 18

2. Team Rank Prediction User Interface:

As shown in figure 19, in the given scenario the user inputs details .The machine learning model processes this information and predicts the position of team.

```
Enter track : ['Australia', 'Bahrain', 'China', 'Azerbaijan', 'Spain', 'Monaco', 'Canada', 'France', 'Austria', 'Gr
eat Britain', 'Germany', 'Hungary', 'Belgium', 'Italy', 'Singapore', 'Russia', 'Japan', 'Mexico', 'United States',
'Brazil', 'Abu Dhabi', 'Styria', 'Tuscany', 'Eifel', 'Portugal', 'Emilia Romagna', 'Turkey', 'Sakhir', 'Netherland
s', 'Qatar', 'Saudi Arabia', 'Miami']Bahrain
Enter team: ['Mercedes', 'Red Bull Racing Honda', 'Ferrari', 'Haas Ferrari', 'Renault', 'Alfa Romeo Racing Ferrar
i', 'Racing Point BWT Mercedes', 'Scuderia Toro Rosso Honda', 'McLaren Renault', 'Williams Mercedes', 'AlphaTauri H
onda', 'McLaren Mercedes', 'Aston Martin Mercedes', 'Alpine Renault', 'Alfa Romeo Ferrari', 'AlphaTauri RBPT', 'Ast
on Martin Aramco Mercedes', 'Red Bull Racing RBPT']Red Bull Racing RBPT
Enter starting grid:2
Enter laps:54
Enter point:0
Enter fastest lap: ['Yes', 'No']No
```

Figure 19

Figure 20, predicts the rank of team from the input detail after processing

```
1/1 [==============================] - 3s 3s/step
Predicted Position: 13
```

Figure 20

References

Peng, B., Li, J., Akkas, S., Araki, T., Yoshiyuki, O. and Qiu, J. (2021). Rank position forecasting in car racing, 2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 724–733

Henderson, D. A. and Kirrane, L. J. (2018). A comparison of truncated and time-weighted plackett–luce models for probabilistic forecasting of formula one results.

Allender, M. (2009). The role of driver experience in predicting the outcome of nascar races: an empirical analysis, The Sport Journal 12(2)

Feng, G. and Buyya, R. (2016). Maximum revenue-oriented resource allocation in cloud, *IJGUC* 7(1): 12–21.

Gomes, D. G., Calheiros, R. N. and Tolosana-Calasanz, R. (2015). Introduction to the special issue on cloud computing: Recent developments and challenging issues, *Computers & Electrical Engineering* 42: 31–32.

Kune, R., Konugurthi, P., Agarwal, A., Rao, C. R. and Buyya, R. (2016). The anatomy of big data computing, *Softw., Pract. Exper.* 46(1): 79–105.