National College of Ireland

Configuration Manual

MSc Research Project
Masters of Science in Data Analytics

Aniketh Mahesh Rao
Student ID: X22166343

School of Computing
National College of Ireland

Supervisor: Noel Cosgrave

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Aniketh Mahesh Rao
Student ID:	X22166343
Programme:	Masters of Science in Data Analytics
Year: 2023	2023
Module:	MSc Research Project
Supervisor:	Noel Cosgrave
Submission Due Date:	14/12/2023
Project Title:	Configuration Manual
Word Count:	504
Page Count:	7

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:	Aniketh Mahesh Rao
Date:	14th December 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

Aniketh Mahesh Rao
X22166343

1 Introduction

This configuration manual is designed to accompany the research paper titled "Predictive Model for Pitstop Strategy in Formula 1 using Ensemble Learning." It provides complete details about the software and hardware configurations used in the research project. The manual outlines the necessary libraries and technologies imported using Python, offering a step-by-step guide on how to configure a functional environment. The primary objective of this manual is to allow end users to reproduce the overall research, ensuring that the findings and Machine Learning model can be effectively replicated. By providing clear instructions on the setup of the software, hardware, and technologies used, this manual will help users to recreate the environment and validate or extend the research outcomes.

2 Hardware Specifications

Table 1: Hardware/Software Specification

Hardware/Software	**Configuration**
System Model	HP Pavilion Laptop
Operation System	Windows 10 Home
Processor	8th Gen Intel(R) Core i5
RAM	16.00 GB

3 Working Environment

In the project, the primary programming language employed is Python, and the code execution takes place within a Jupyter Notebook environment. The version of Python utilized is 3.11.5, and you can install it by referring to the provided image [1]. Python serves as the key tool for various stages in the project, including data gathering, cleaning, exploration, transformation, and visualization.

To run the code open the file in the jupyter notebook environment and upload the dataset as highlighted.

[1] `https://www.python.org/downloads/`

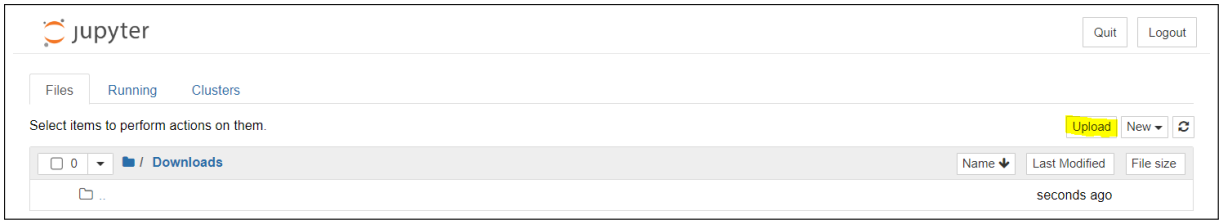

Figure 1: Opening Jupyter Notebook

4 Datasets

Two distinct sources, namely Tracing Insights and Pitwall, have been used to collect three datasets, which are then combined to a singular dataset file. The information extracted from these two open-source websites consists of dataset comprising 25,000 rows and encompassing 40 columns.

5 Importing Libraries and performing Exploratory Data Analysis

All the required libraries which needs to be installed are mentioned in the below Figure 2. All the below libraries are required to run the code and to get the expected results.

```
# Importing all the required Libraries

import numpy as np
import pandas as pd
import warnings
warnings.simplefilter("ignore")
from bs4 import BeautifulSoup
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import StackingClassifier, RandomForestClassifier, GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score
from sklearn.model_selection import train_test_split
```

Figure 2: Importing Libraries

After importing the libraries, dataset can be imported into the jupyter environment and initial few rows have been shown in Figure 3. In which Position and PitFlag are the target variables.

```
# Loading the new dataset file and displaying the first few rows to understand its structure.

# Load the new dataset
new_data = pd.read_excel('Final_Updated.xlsx')

# Display the first few rows of the new dataset
new_data.head()
```

]:

	Time	Driver	Position	DriverNumber	PitFlag	LapTime	LapNumber	Stint	Sector1Time	Sector2Time	...	IsAccurate	LapTime_in_seco
0	0 days 01:02:45.730000	VER	1	1	0	0 days 00:01:39.019000	1	1	0.000	42.414	...	False	99.
1	0 days 01:04:45.759000	VER	1	1	0	0 days 00:01:37.974000	2	1	31.342	42.504	...	True	97.
2	0 days 01:06:45.757000	VER	1	1	0	0 days 00:01:38.006000	3	1	31.388	42.469	...	True	98.
3	0 days 01:07:45.756000	VER	1	1	0	0 days 00:01:37.976000	4	1	31.271	42.642	...	True	97.
4	0 days 01:09:45.754000	VER	1	1	0	0 days 00:01:38.035000	5	1	31.244	42.724	...	True	98.

5 rows × 40 columns

Figure 3: First 5 rows of the dataset

Figure 4, shows two line graphs that are plotted to display the relationship between the lap number and two different temperature measurements: air temperature and track temperature.

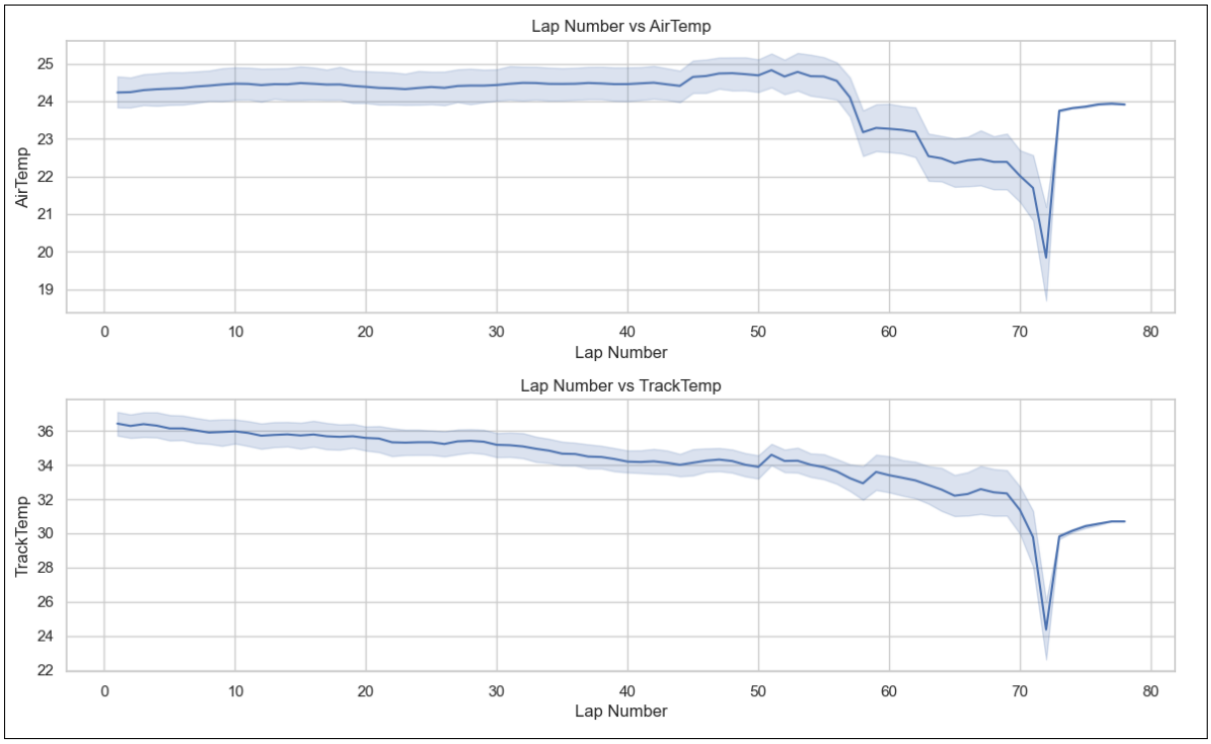

Figure 4: Lap Number Vs Air Temperature

Figure 5 is a clustered bar chart titled "Distribution of Laps with Different Tire Compounds". It shows the usage frequency of different tire compounds over various groups of laps during a racing event.

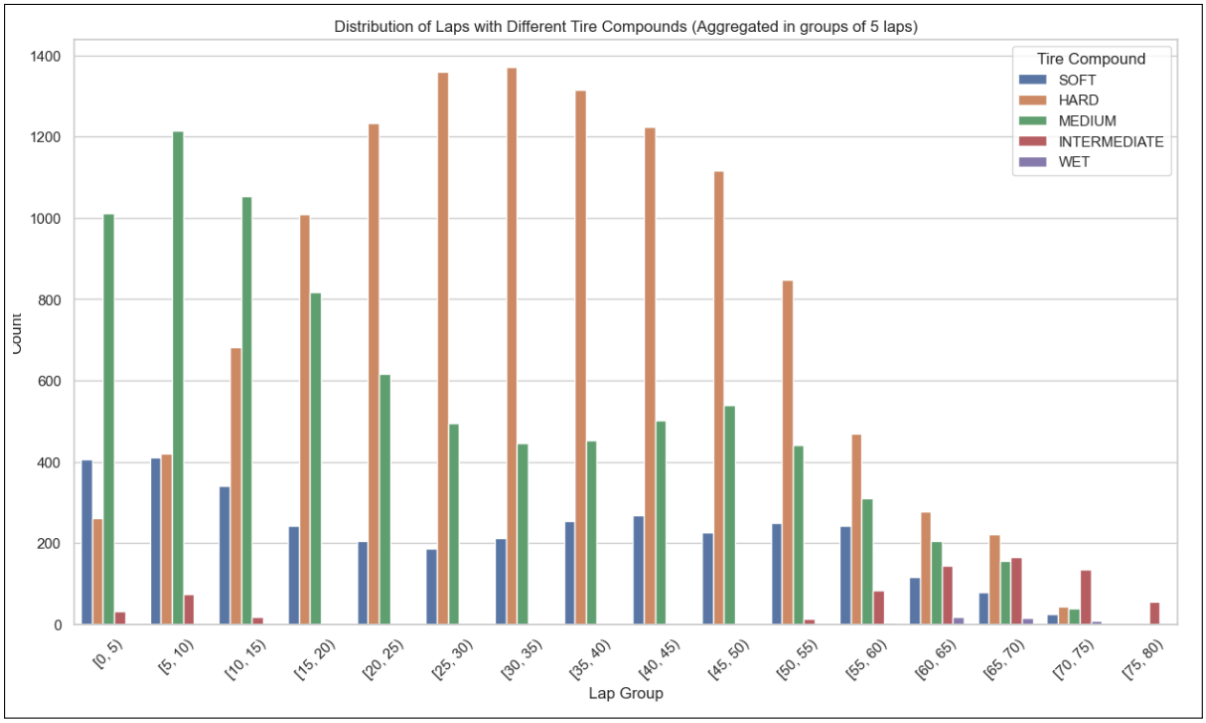

Figure 5: Lap Number Vs Track Temperature

6 Machine Learning Model

```
# Splitting the dataset for 'Position' prediction
X = data.drop(['Position'], axis=1)  # Features
y_position = data['Position']  # Target for 'Position'

# Splitting data into training and testing sets for 'Position'
X_train_pos, X_test_pos, y_train_pos, y_test_pos = train_test_split(X, y_position, test_size=0.2, random_state=42)

X1 = data.drop(['Position', 'PitFlag'], axis=1)  # Features
# Repeat the process for 'PitFlag' prediction
y_pitflag = data['PitFlag']  # Target for 'PitFlag'

# Splitting data for 'PitFlag'
X_train_pf, X_test_pf, y_train_pf, y_test_pf = train_test_split(X1, y_pitflag, test_size=0.2, random_state=42)
```

Figure 6: Training and Testing Set

Before executing the model, we split the data into training and testing data, as seen in Figure 6, testing data is 20 percent and remaining is the training set for both the target variables.

```python
# Defining base learners
base_learners = [
    ('rf', RandomForestClassifier(n_estimators=100, random_state=42)),
    ('svc', SVC(probability=True, random_state=42)),
    ('gb', GradientBoostingClassifier(n_estimators=100, random_state=42))
]

# Defining the meta-learner
meta_learner = LogisticRegression()

# Building the stacking ensemble
stacked_model = StackingClassifier(estimators=base_learners, final_estimator=meta_learner, cv=5)

# Training the stacked model
stacked_model.fit(X_train_pf, y_train_pf)

# Making predictions and evaluating the model
y_pred = stacked_model.predict(X_test_pf)
print("Accuracy:", accuracy_score(y_test_pf, y_pred))
print("\nClassification Report:\n", classification_report(y_test_pf, y_pred))
```

Figure 7: Meta-Learner for Pitstop

In Figure 7, Ensemble Learning has been used for predicting PitFLag through a stacking approach, which is a form of meta-learning since it involves learning how to best combine the predictions of multiple models.

```python
sample_data = X1.iloc[0:57]
actual_label = y_pitflag.iloc[0:56]

# Making predictions with each model on the sample
pred = stacked_model.predict(sample_data)

Pit_lap = np.where(pred == 1)[0]
print(f"The driver should take pit stops at laps {Pit_lap}")
```

Figure 8: Pitstop Prediction

Prediction output is displayed, the lap numbers on which the driver should make a pitstop in Figure 8.

```
# Defining base learners
base_learners = [
    ('rf', RandomForestClassifier(n_estimators=100, random_state=42)),
    ('svc', SVC(probability=True, random_state=42)),
    ('gb', GradientBoostingClassifier(n_estimators=100, random_state=42))
]

# Defining the meta-learner
meta_learner = LogisticRegression()

# Building the stacking ensemble
stacked_model1 = StackingClassifier(estimators=base_learners, final_estimator=meta_learner, cv=5)

# Training the stacked model
stacked_model1.fit(X_train_pos, y_train_pos)

# Making predictions and evaluating the model
y_pred = stacked_model1.predict(X_test_pos)
print("Accuracy:", accuracy_score(y_test_pos, y_pred))
print("\nClassification Report:\n", classification_report(y_test_pos, y_pred))
```

Figure 9: Meta-Learner for Position

In Figure 9, position is the target variable and model will predict the position of the driver based on sample data.

```
sample_data = X.iloc[0:57]
actual_label = y_position[56]

# Making predictions with each model on the sample
pred = stacked_model1.predict(sample_data)

print(f"Predicted Position for Driver is {pred[-1]}")
print(f"Actual Position for Driver is {actual_label}")
```

Figure 10: Driver Position Prediction

In Figure 10, position of the drivers is displayed, if they make the pitstop in the predicted laps shown in the output of Figure 9.

```
# Reducing the index size for the plot to focus on a smaller subset of data for clearer visualization

# Selecting a subset of data for clearer visualization
subset_data = plot_data.sample(n=100, random_state=42)  # Adjust the sample size as needed

# Plotting Actual vs Predicted results on the subset
plt.figure(figsize=(12, 6))
plt.scatter(subset_data.index, subset_data['Actual'], color='blue', label='Actual', alpha=0.5)
plt.scatter(subset_data.index, subset_data['Predicted'], color='red', label='Predicted', alpha=0.5)
plt.title('Actual vs Predicted Positions')
plt.xlabel('Index (Subset)')
plt.ylabel('Position')
plt.xticks([])
plt.legend()
plt.show()
```

Figure 11: Driver Position Prediction

In Figure 11 and 12, there are two sets of data points are depicted: blue points represent actual positions, and orange points represent predicted positions. The points are spread across the plot, showing the variability and relationship between the actual and predicted values. Some points are overlapping which are highlighted in red, indicating instances where the predictions match the actual positions closely, while others are separate, highlighting discrepancies between prediction and reality

Figure 12: Driver Position Prediction

7 Conclusion

In conclusion, this manual provides a thorough overview of the project's key technologies and provides in-depth information on how to configure and use them. Its main goal is to enable end users to set up their workspaces such that the supplied code executes successfully. The manual guarantees consistency and accessibility by clarifying the primary technologies and how they are configured, which improves the use of the project's codebase.