Minor Project

"Premier League Table Prediction using Machine Learning"

Submitted in partial fulfillment of the requirements for

the award of the degree of

Bachelor of Technology

In

Information Technology

HMR INSTITUTE OF TECHNOLOGY & MANAGEMENT

HAMIDPUR, DELHI 110 036

Affiliated to

GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY

Sector - 16C Dwarka, Delhi – 110075, India

Submitted To:	Submitted by:
Ms. Priyanka	Ankur Bhardwaj (40813303118)
Professor	Aman Chamola (00113303118)
IT-Department	Bhavya Gaur (35113303118)

DECLARATION

I / We, student(s) of B.Tech in Information Technology hereby declare that the Minor Project entitled "***Premier League Table Prediction using Machine Learning***" which is submitted to Department of Information Technology, HMR Institute of Technology & Management, Hamidpur, Delhi, affiliated to Guru Gobind Singh Indraprastha University, Dwarka (New Delhi) in partial fulfillment of requirement for the award of the degree of Bachelor of Technology in Information Technology, has not been previously formed the basis for the award of any degree, diploma or other similar title or recognition. The list of member(s) involved in the project is listed below:

S.No.	Student's Name	Student's Enrollment Number	Sign
1	Ankur Bhardwaj	40813303118	Ankur
2	Aman Chamola	00113303118	Aman
3	Bhavya Gaur	35113303118	Bhavya

This is to certify that the above statement made by the candidate(s) is correct to the best of my knowledge.

New Delhi

Signature of guide

Date: 23nd December 2021

Head of Department
(Information Technology)
(HMRITM, New Delhi)

ACKNOWLEDGEMENT

I would like to express my special thanks to ***Ms. Priyanka****, our guide whose guidance has been the ones that helped us patch this project and make it a full proof success, his suggestions and instructions have served as a major contributor towards the completion of our project.*

Secondly, I would like to thank you group members with whose help this idea cameup and we were able to complete the project in the limited time frame.

ABSTRACT

Betting is illegal in India but skilled betting apps like dream11 and betway are currently legal because there’s skilled required to win rather than just the chance or luck. Premier League is a football league which is the biggest sporting league in the world. Machine learning is a subset of artificial intelligence (AI) in which algorithms learn by example from historical data to predict outcomes and uncover patterns that are not easily spotted by humans. Machine learning evolved from the study of pattern recognition and explores the notion that algorithms can learn from and make predictions on data. And, as they begin to become more ‘intelligent’, these algorithms can overcome program instructions to make highly accurate, data-driven decisions. Predictive analytics encompasses a variety of statistical techniques (including machine learning, predictive modelling and data mining) and uses statistics (both historical and current) to estimate, or ‘predict’, future outcomes. These outcomes might be behaviors of a customer likely to exhibit or possible changes in the market, for example. Predictive analytics help us to understand possible future occurrences by analyzing the past. In this project it’s intended to combine machine learning algorithm with predictive analytics to predict the premier league football table at the end of the season.

TABLE OF CONTENTS

Chapter 5 - Conclusion and future scope

<u>List of Figures</u>

<u>Chapter-1 Introduction</u>

In this project we are going to predict the table of premier league 2021-22 season based on historical data using various features.

Columns Description –

- **W-** Total wins
- **D –** Total Draws
- **Expected Goals (Xg) –** Expected goals is the most popular stat in football right now. It tells us how many goals were expected by a club in that particular season.
- **Xg conceded -** How many goals were expected to be conceded by a club in that particular season.
- **Gd –** Goal Difference
- **CLS –** Clean Sheets
- **xPTS –** Points expected during the course of the season.

Note – Data was taken and build from websites like fotmob.com, understat.com.

1.1 Problem Statement

Betting is illegal in India but skilled betting apps like dream11 and betway are currently legal because there’s skilled required to win rather than just the chance or luck. Premier League is a football league which is the biggest sporting league in the world.
In this project we’re building a model that will make an accurate **prediction of final PL table at the end of the season** using Machine learning Algorithm

To reach the above objective there have been considered the following steps:

• Extracting and building the dataset from footballing websites like fotmob.com and understat.com.
• Analyzing the data and then manipulating the data according to our needs.
• To explore a model for predicting the total points of a team.
• To create a model that can predict the table of the teams (the final table of the rankings of the teams in the league by evaluating total points of each team).
• Evaluating our model by applying the prediction on already completed 2020-21 season.
• Saving our prediction in the new predicted.csv file.

The desired algorithm is linear regression.

1.2 Motivation –

Machine learning techniques have been around us and has been compared and used for analysis for many kinds of data science applications. The major motivation behind this research-based project was to explore the feature selection methods, data preparation and processing behind the training models in the machine learning. With first hand models and libraries, the challenge we face today is data where beside their abundance, and our cooked models, the accuracy we see during training, testing and actual validation has a higher variance.

1.3 Objective –

Our main objective is to build the best possible model to predict the Premier League table of 2021-22 season using historical data with features including expected goals, wins, losses, expected Points, etc. we’re going to use linear regression in this project to build and train our model.

CHAPTER 2: TECHNOLOGY AND MODULES USED

2.1 Data Science

Data science is an interdisciplinary field focused on extracting knowledge from data sets, which are typically large (see big data), and applying the knowledge and actionable insights from data to solve problems in a wide range of application domains. The field encompasses preparing data for analysis, formulating data science problems, analyzing data, developing data-driven solutions, and presenting findings to inform high-level decisions in a broad range of application domains. As such, it incorporates skills from computer science, statistics, information science, mathematics, information visualization, data integration, graphic design, complex systems, communication and business. Statistician Nathan Yau, drawing on Ben Fry, also links data science to human-computer interaction: users should be able to intuitively control and explore data. In 2015, the American Statistical Association identified database management, statistics and machine learning, and distributed and parallel systems as the three emerging foundational professional communities.

2.2 Data Analysis

Data analysis is a process of inspecting, cleansing, transforming, and modelling data with the goal of discovering useful information, informing conclusions, and supporting decision-making. Data analysis has multiple facets and approaches, encompassing diverse techniques under a variety of names, and is used in different business, science, and social science domains. In today's business world, data analysis plays a role in making decisions more scientific and helping businesses operate more effectively.

Data mining is a particular data analysis technique that focuses on statistical modelling and knowledge discovery for predictive rather than purely descriptive purposes, while business intelligence covers data analysis that relies heavily on aggregation, focusing mainly on business information. In statistical applications, data analysis can be divided into descriptive statistics, exploratory data analysis (EDA), and confirmatory data analysis (CDA). EDA focuses on discovering new features in the data while CDA focuses on confirming or falsifying existing hypotheses. Predictive analytics focuses on the application of statistical models for predictive forecasting or classification, while text analytics applies statistical, linguistic, and structural techniques to extract and classify information from textual sources, a species of unstructured data. All of the above are varieties of data analysis.

2.2.1 Data Preprocessing

Data preprocessing is a data mining technique which is used to transform the raw data in a useful and efficient format. Data preprocessing is most important process. Mostly healthcare related data contains missing vale and other impurities that can cause effective- ness of data. To improve quality and effectiveness obtained after mining process, Data preprocessing is done. To use Machine Learning Techniques on the dataset effectively the process is essential for accurate result and successful prediction.
Missing Values removal - Remove all the instances that have zero (0) as worth. Having zero as worth is not possible. Therefore, this instance is eliminated. Through eliminating irrelevant features/instances we make feature subset and this process is called features subset selection, which reduces dimensionality of data and help to work faster.
Splitting of data - After cleaning the data, data is normalized in training and testing the model. When data is

spitted then we train algorithm on the training data set and keep test data set aside. This training process will produce the training model based on logic and algorithms and values of the feature in training data. Basically, aim of normalization is to bring all the attributes under same scale.

2.3 Machine learning

Machine Learning is the science of getting computers to learn without being explicitly programmed. It is closely related to computational statistics, which focuses on making prediction using computer. In its application across business problems, machine learning is also referred as predictive analysis. Machine Learning is closely related to computational statistics. Machine Learning focuses on the development of computer programs that can access data and use it to learn themselves. The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.

2.3.1 History of Machine Learning

The name machine learning was coined in 1959 by Arthur Samuel. Tom M. Mitchell provided a widely quoted, more formal definition of the algorithms studied in the machine learning field: "
A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E, "This follows Alan Turing's proposal in his paper "Computing Machinery and Intelligence", in which the question" Can machines think?" is replaced with the question "Can machines do what we (as thinking entities) can do?". In Turing's proposal the characteristics that could be possessed by a thinking machine and the various implications in constructing one are exposed.

2.3.2 Types of Machine Learning

The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve. Broadly Machine Learning can be categorized into four categories.

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

Machine learning enables analysis of massive quantities of data. While it generally delivers faster, more accurate results in order to identify profitable opportunities or dangerous risks, it may also require additional time and resources to train it properly.

Supervised Learning:

Supervised Learning is a type of learning in which we are given a data set and we already know what are correct output should look like, having the idea that there is a relationship between the input and output. Basically, it is learning task of learning a function that maps an input to an output based on example input-output pairs. It infers a function from labeled training data consisting of a set of training examples.

Unsupervised Learning:

Unsupervised Learning is a type of learning that allows us to approach problems with little or no idea what our problem should look like. We can derive the structure by clustering the data based on a relationship among the variables in data. With unsupervised learning there is no feedback based on prediction result. Basically, it is a type of self-organized learning that helps in finding previously unknown patterns in data set without pre-existing label.

Reinforcement Learning
Reinforcement learning is a learning method that interacts with its environment by producing actions and discovers errors or rewards. Trial and error search and delayed reward are the most relevant characteristics of reinforcement learning. This method allows machines and software agents to automatically determine the ideal behavior within a specific context in order to maximize its performance. Simple reward feedback is required for the agent to learn which action is best.

2.4 Artificial Intelligence
Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. Specific applications of AI include expert systems, natural language processing, speech recognition and machine vision.

2.4.1 How does AI work?
As the hype around AI has accelerated, vendors have been scrambling to promote how their products and services use AI. Often what they refer to as AI is simply one component of AI, such as machine learning. AI requires a foundation of specialized hardware and software for writing and training machine learning algorithms. No one programming language is synonymous with AI, but a few, including Python, R and Java, are popular.

In general, AI systems work by ingesting large amounts of labeled training data, analyzing the data for correlations and patterns, and using these patterns to make predictions about future states. In this way, a chatbot that is fed examples of text chats can learn to produce lifelike exchanges with people, or an image recognition tool can learn to identify and describe objects in images by reviewing millions of examples.

AI programming focuses on three cognitive skills: learning, reasoning and self-correction.

Learning processes- This aspect of AI programming focuses on acquiring data and creating rules for how to turn the data into actionable information. The rules, which are called algorithms, provide computing devices with step-by-step instructions for how to complete a specific task.
Reasoning processes- This aspect of AI programming focuses on choosing the right algorithm to reach a desired outcome.
Self-correction processes- This aspect of AI programming is designed to continually fine-tune algorithms and ensure they provide the most accurate results possible.

2.4.2 Why is artificial intelligence important?
AI is important because it can give enterprises insights into their operations that they may not have been aware of previously and because, in some cases, AI can perform tasks better than humans. Particularly when it comes to repetitive, detail-oriented tasks like analyzing large numbers of legal documents to ensure relevant fields are filled in properly, AI tools often complete jobs quickly and with relatively few errors.

This has helped fuel an explosion in efficiency and opened the door to entirely new business opportunities for some larger enterprises. Prior to the current wave of AI, it would have been hard to imagine using computer software to connect riders to taxis, but today Uber has become one of the largest companies in the world by doing just that. It utilizes sophisticated machine learning algorithms to predict when people are likely to need rides in certain areas, which helps proactively get drivers on the road before they're needed. As another example, Google has become one of the largest players for a range of online services by using machine learning to understand how people use their services and then improving them. In 2017, the company's CEO, Sundar Pichai, pronounced that Google would operate as an "AI first" company.

Today's largest and most successful enterprises have used AI to improve their operations and gain advantage on their competitors.

2.4.3 What are the advantages and disadvantages of artificial intelligence?

Artificial neural networks and deep learning artificial intelligence technologies are quickly evolving, primarily because AI processes large amounts of data much faster and makes predictions more accurately than humanly possible.

While the huge volume of data being created on a daily basis would bury a human researcher, AI applications that use machine learning can take that data and quickly turn it into actionable information. As of this writing, the primary disadvantage of using AI is that it is expensive to process the large amounts of data that AI programming requires.

Advantages:

- Good at detail-oriented jobs;
- Reduced time for data-heavy tasks;
- Delivers consistent results; and
- AI-powered virtual agents are always available.

Disadvantages:

- Expensive;
- Requires deep technical expertise;
- Limited supply of qualified workers to build AI tools;
- Only knows what it's been shown; and
- Lack of ability to generalize from one task to another.

2.4.4 Types of technology used in A.I.

AI is incorporated into a variety of different types of technology. Here are six examples:

Automation- When paired with AI technologies, automation tools can expand the volume and types of tasks performed. An example is robotic process automation (RPA), a type of software that automates repetitive, rules-based data processing tasks traditionally done by humans. When combined with machine learning and emerging AI tools, RPA can automate bigger portions of enterprise jobs, enabling RPA's tactical bots to pass along intelligence from AI and respond to process changes.

Machine learning- This is the science of getting a computer to act without programming. Deep learning is a subset of machine learning that, in very simple terms, can be thought of as the automation of predictive analytics. There are three types of machine learning algorithms:

Supervised learning- Data sets are labeled so that patterns can be detected and used to label new data sets.

Unsupervised learning- Data sets aren't labeled and are sorted according to similarities or differences.

Reinforcement learning- Data sets aren't labeled but, after performing an action or several actions, the AI system is given feedback.

Machine vision- This technology gives a machine the ability to see. Machine vision captures and analyzes visual information using a camera, analog-to-digital conversion and digital signal processing. It is often compared to human eyesight, but machine vision isn't bound by biology and can be programmed to see through walls, for example. It is used in a range of applications from signature identification to medical image analysis. Computer vision, which is focused on machine-based image processing, is often conflated with machine vision.

Natural language processing (NLP)- This is the processing of human language by a computer program. One of the older and best-known examples of NLP is spam detection, which looks at the subject line and text of an email and decides if it's junk. Current approaches to NLP are based on machine learning. NLP tasks include text translation, sentiment analysis and speech recognition.

Robotics- This field of engineering focuses on the design and manufacturing of robots. Robots are often used to perform tasks that are difficult for humans to perform or perform consistently. For example, robots are used in assembly lines for car production or by NASA to move large objects in space. Researchers are also using machine learning to build robots that can interact in social settings.

Self-driving cars- Autonomous vehicles use a combination of computer vision, image recognition and deep learning to build automated skill at piloting a vehicle while staying in a given lane and avoiding unexpected obstructions, such as pedestrians.

2.5 Python

2.5.1 Python Language Introduction

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- Python is Object-Oriented − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

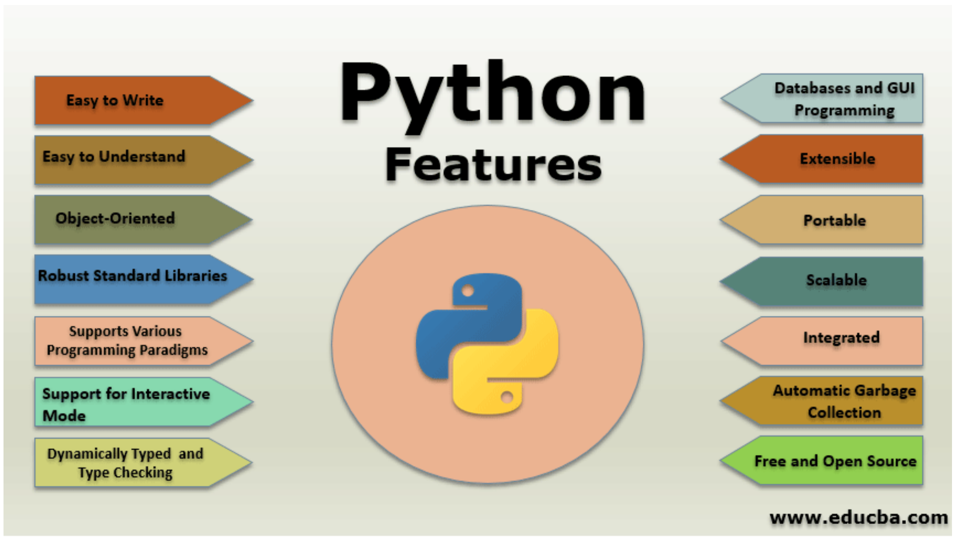

Python
Features
Easy to Write
Easy to Understand
Object-Oriented
Robust Standard Libraries
Supports Various Programming Paradigms
Support for Interactive Mode
Dynamically Typed and Type Checking
Databases and GUI Programming
Extensible
Portable
Scalable
Integrated
Automatic Garbage Collection
Free and Open Source
www.educba.com

Figure 2.5.1 Features of Python

2.5.2 History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Smalltalk, and UNIX shell and other scripting languages.

Python Features

- Easy-to-learn − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

- Easy-to-read − Python code is more clearly defined and visible to the eyes.
- Easy-to-maintain − Python's source code is fairly easy-to-maintain.
- A broad standard library − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh
- Interactive Mode − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- Portable − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

Python graphical user interfaces (GUIs)

- Tkinter − Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. We would look this option in this chapter
- wxPython –**wxPython** is essentially a Python extension module that acts as a wrapper for the wxWidgets API. wxPython **allows Python developers to create native user interfaces** that add zero additional overhead to the application.
- JPython − JPython is a Python port for Java which gives Python scripts seamless access to Java class libraries on the local machine. http://www.jython.org.

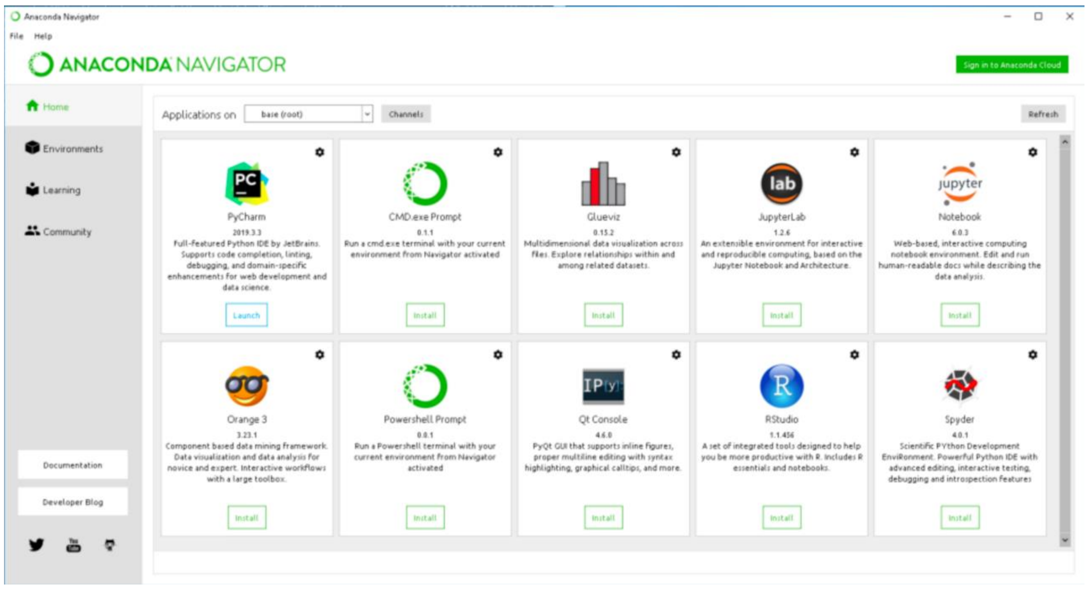
There are many other interfaces available, which you can find them on the internet.

2.6 Anaconda Navigator

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository. It is available for Windows, macOS, and Linux.

2.6.1 Conda Navigator uses

In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages and use multiple environments to separate these different versions.
The command-line program conda is both a package manager and an environment manager. This helps data scientists ensure that each version of each package has all the dependencies it requires and works correctly.
Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages, and update them – all inside Navigator.



PyCharm
2019.3.3
Full-featured Python IDE by JetBrains. Supports code completion, linting, debugging, and domain-specific enhancements for web development and data science.

CMD.exe Prompt
0.1.1
Run a cmd.exe terminal with your current environment from Navigator activated

Glueviz
0.15.2
Multidimensional data visualization across files. Explore relationships within and among related datasets.

JupyterLab
1.2.6
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook
6.0.3
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Orange 3
3.23.1
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

Powershell Prompt
0.0.1
Run a Powershell terminal with your current environment from Navigator activated

Qt Console
4.6.0
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

Spyder
4.0.1
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

Figure 2.6.1 Anaconda Navigator

2.6.2 Applications that can be accessed using Conda Navigator -

- JupyterLab
- Jupyter Notebook
- Spyder
- PyCharm
- VSCode
- Glueviz
- Orange 3 App
- RStudio
- Anaconda Prompt (Windows only)
- Anaconda PowerShell (Windows only)

2.7 Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, machine learning and much more.

The Jupyter Notebook is a living online notebook, letting faculty and students weave together computational information (code, data, statistics) with narrative, multimedia, and graphs. Faculty can use it to set up interactive textbooks, full of explanations and examples which students can test out right from their browsers. Students can use it to explain their reasoning, show their work, and draw connections between their classwork and the world outside. Scientists, journalists, and researchers can use it to open up their data, share the stories behind their computations, and enable future collaboration and innovation

Figure 2.7 Jupyter Notebook Logo

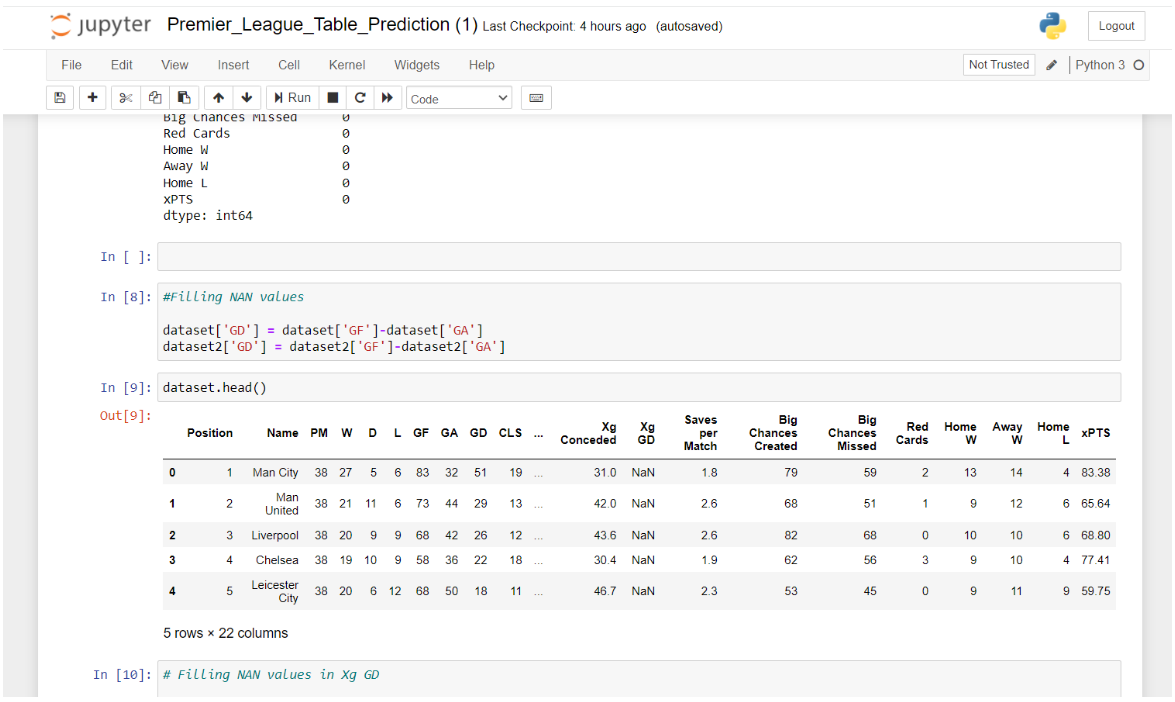
2.7.1 Features of Jupyter Notebook

A Jupyter Notebook is fundamentally a JSON file with a number of annotations. There are three main parts of the Notebook as follows.

• Metadata: a data dictionary of definitions used to set-up and display the notebook.

• Notebook format: version numbers of the software used to create the notebook. The version number is used for backward compatibility.

• List of cells: there are three different types of cells — markdown (display), code (to excite), and output.



```
Big Chances Missed      0
Red Cards               0
Home W                  0
Away W                  0
Home L                  0
xPTS                    0
dtype: int64
```

In [ ]:

In [8]:
```
#Filling NAN values

dataset['GD'] = dataset['GF']-dataset['GA']
dataset2['GD'] = dataset2['GF']-dataset2['GA']
```

In [9]:
```
dataset.head()
```

Out[9]:

	Position	Name	PM	W	D	L	GF	GA	GD	CLS	...	Xg Conceded	Xg GD	Saves per Match	Big Chances Created	Big Chances Missed	Red Cards	Home W	Away W	Home L	xPTS
0	1	Man City	38	27	5	6	83	32	51	19	...	31.0	NaN	1.8	79	59	2	13	14	4	83.38
1	2	Man United	38	21	11	6	73	44	29	13	...	42.0	NaN	2.6	68	51	1	9	12	6	65.64
2	3	Liverpool	38	20	9	9	68	42	26	12	...	43.6	NaN	2.6	82	68	0	10	10	6	68.80
3	4	Chelsea	38	19	10	9	58	36	22	18	...	30.4	NaN	1.9	62	56	3	9	10	4	77.41
4	5	Leicester City	38	20	6	12	68	50	18	11	...	46.7	NaN	2.3	53	45	0	9	11	9	59.75

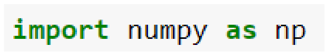
5 rows × 22 columns

In [10]:
```
# Filling NAN values in Xg GD
```

Figure 2.7.1 Jupyter Notebook

2.8 Modules Used in this Project

2.8.1 Numpy



```
import numpy as np
```

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python.
Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data.

Features

These are the important features of NumPy:

1. High-performance N-dimensional array object

This is the most important feature of the NumPy library. It is the homogeneous array object. We perform all the operations on the array elements. The arrays in NumPy can be one dimensional or multidimensional.

a. One dimensional array

The one-dimensional array is an array consisting of a single row or column. The elements of the array are of homogeneous nature.

b. Multidimensional array

In this case, we have various rows and columns. We consider each column as a dimension. The structure is similar to an excel sheet. The elements are homogenous.

2. It contains tools for integrating code from C/C++ and Fortran

We can use the functions in NumPy to work with code written in other languages. We can hence integrate the functionalities available in various programming languages. This helps implement inter-platform functions.

3. It contains a multidimensional container for generic data

Here generic data refers to the parameterized data type of arrays. It can perform functions on the generic data types. The arrays in NumPy are of homogenous nature. These array elements are assigned parameters. The parameters help increase the diversity of the arrays.

4. Additional linear algebra, Fourier transform, and random number capabilities

It has the capability to perform complex operations of the elements like linear algebra, Fourier transform, etc. We have separate modules for each of the complex functions. We have the linalg module for linear algebra functions.

Similarly, we have fft functions for Fourier Transform in NumPy. We have a matrix module for applying functions on matrices. We also have special functions for plotting graphs in the matplotlib module of NumPy. Hence, it is a very diverse library to work with arrays.

5. It consists of broadcasting functions

The broadcasting of array is a very useful concept when we work with arrays of uneven shapes. It broadcasts the shape of smaller arrays according to the larger ones. The broadcasting of arrays has some rules and limitations in its implementation.

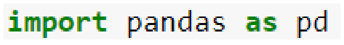
For broadcasting one of the arrays needs to be onedimensional or both the arrays are supposed to be of the same shape. There are also a few other limitations on the shape of the arrays.

6. It had data type definition capability to work with varied databases

We can work with arrays of different data types. We can use the dtype function to determine the data type and hence get a clear idea about the available data set.

With the array definition, we have an additional dtype argument to perform array functions. The knowledge of the data type of array is very important due to the restrictions on NumPy operations.

2.8.2 Pandas



```
import pandas as pd
```

Pandas are turning up to be the most popular Python library that is used for data analysis with support for fast, flexible, and expressive data structures designed to work on both "relational"
or "labeled" data. Pandas today is an inevitable library for solving practical, real-world data analysis in Python. Pandas is highly stable, providing highly optimized performance. The backend code is purely written in C or Python.
Pandas support and perform well with different kinds of data including the below:

- Tabular data with columns of heterogeneous data. For instance, consider the data coming from the SQL table or Excel spreadsheet.
- Ordered and unordered time series data. The frequency of time series need not be fixed, unlike other libraries and tools. Pandas is exceptionally robust in handling uneven time-series data.
- Arbitrary matrix data with the homogeneous or heterogeneous type of data in the rows and columns.
- Any other form of statistical or observational data sets. The data need not be labeled at all. Pandas data structure can process it even without labelling.

2.8.3 Matplotlib

Matplotlib is a data visualization library that is used for 2D plotting to produce publication-quality image plots and figures in a variety of formats. The library helps to generate histograms, plots, error charts, scatter plots, bar charts with just a few lines of code.
It provides a MATLAB-like interface and is exceptionally user-friendly. It works by using standard GUI toolkits like GTK+, wxPython, Tkinter, or Qt to provide an object-oriented API that helps programmers to embed graphs and plots into their application.

2.8.3.1 Matplotlib examples

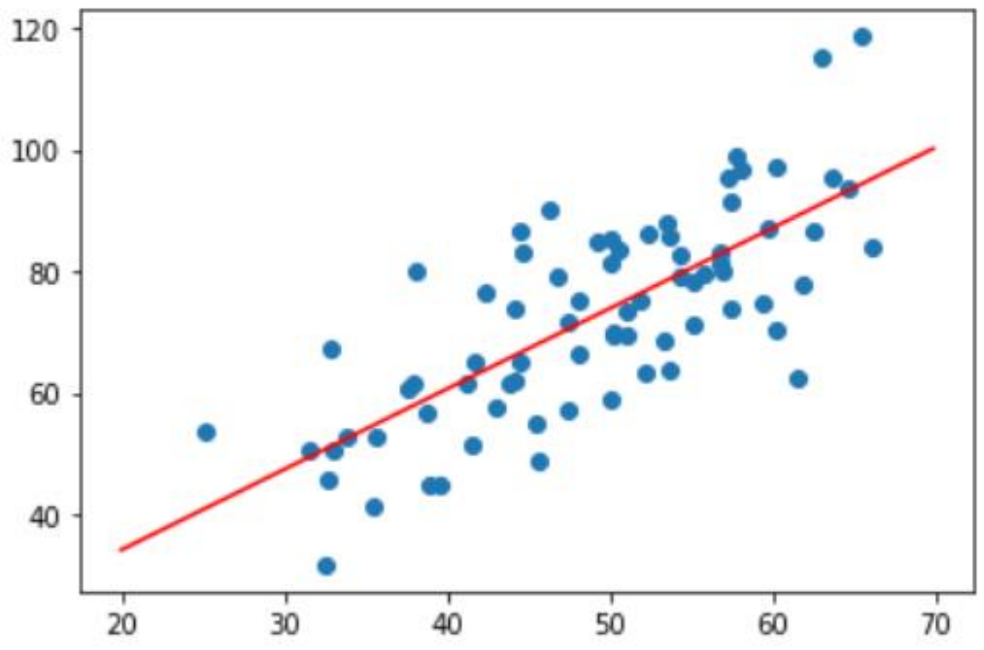


Figure 2.8.3 matplotlib

- Line chart
- Histogram
- Bar chart
- Pie chart
- Legend
- Matplotlib save figure to image
- Matplotlib update plot
- Plot time with Matplot
- Generate heatmap in matplotlib
- Scatterplot

- 3d scatterplot
- Subplot
- Matrix correlation

2.8.4 Seaborn

Seaborn is a library for making statistical graphics in Python. It is built on top of matplotlib and closely integrated with pandas data structures.
Here is some of the functionality that seaborn offers:

- A dataset-oriented API for examining relationships between multiple variables
- Convenient views onto the overall structure of complex datasets
- Specialized support for using categorical variables to show observations or aggregate statistics
- Options for visualizing univariate or bivariate distributions and for comparing them between subsets of data
- Automatic estimation and plotting of linear regression models for different kinds of dependent variables
- High-level abstractions for structuring multi-plot grids that let you easily build complex visualizations
- Concise control over matplotlib figure styling with several built-in themes
- Tools for choosing color palettes that faithfully reveal patterns in your data
- Seaborn aims to make visualization a central part of exploring and understanding data. Its dataset-oriented plotting functions operate on data frames and arrays containing whole datasets and internally perform the necessary semantic mappings and statistical aggregations to produce informative plots.

2.8.5 Scikit-learn / Sklearn

Scikit-learn is probably the most useful library for machine learning in Python. The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction. Please note that sklearn is used to build machine learning models. It should not be used for reading the data, manipulating and summarizing it. There are better libraries for that (e.g. NumPy, Pandas etc.)

Components of scikit-learn:

Scikit-learn comes loaded with a lot of features. Here are a few of them to help you understand the spread:

- **Supervised learning algorithms:** Think of any supervised machine learning algorithm you might have heard about and there is a very high chance that it is part of scikit-learn. Starting from Generalized linear models (e.g Linear Regression), Support Vector Machines (SVM), Decision Trees to Bayesian methods – all of them are part of scikit-learn toolbox. The spread of machine learning algorithms is one of the big reasons for the high usage of scikit-learn. I started using scikit to solve supervised learning problems and would recommend that to people new to scikit / machine learning as well.

- **Cross-validation:** There are various methods to check the accuracy of supervised models on unseen data using sklearn.
- **Unsupervised learning algorithms:** Again, there is a large spread of machine learning algorithms in the offering – starting from clustering, factor analysis, principal component analysis to unsupervised neural networks.
- **Various toy datasets:** This came in handy while learning scikit-learn. I had learned SAS using various academic datasets (e.g. IRIS dataset, Boston House prices dataset). Having them handy while learning a new library helped a lot.
- **Feature extraction:** Scikit-learn for extracting features from images and text (e.g. Bag of words)

2.8.6 Regression Analysis

Regression is a supervised learning technique which helps in finding the correlation between variables and enables us to predict the continuous output variable based on the one or more predictor variables. It is mainly used for prediction, forecasting, time series modeling, and determining the causal-effect relationship between variables**.**

In Regression, we plot a graph between the variables which best fits the given datapoints, using this plot, the machine learning model can make predictions about the data. In simple words, "Regression shows a line or curve that passes through all the datapoints on target-predictor graph in such a way that the vertical distance between the datapoints and the regression line is minimum." The distance between datapoints and line tells whether a model has captured a strong relationship or not.

Some examples of regression can be as:

- Prediction of rain using temperature and other factors
- Determining Market trends
- Prediction of road accidents due to rash driving.

2.8.7 Linear Regression

Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as sales, salary, age, product price, etc.

Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

The linear regression model provides a sloped straight line representing the relationship between the variables.

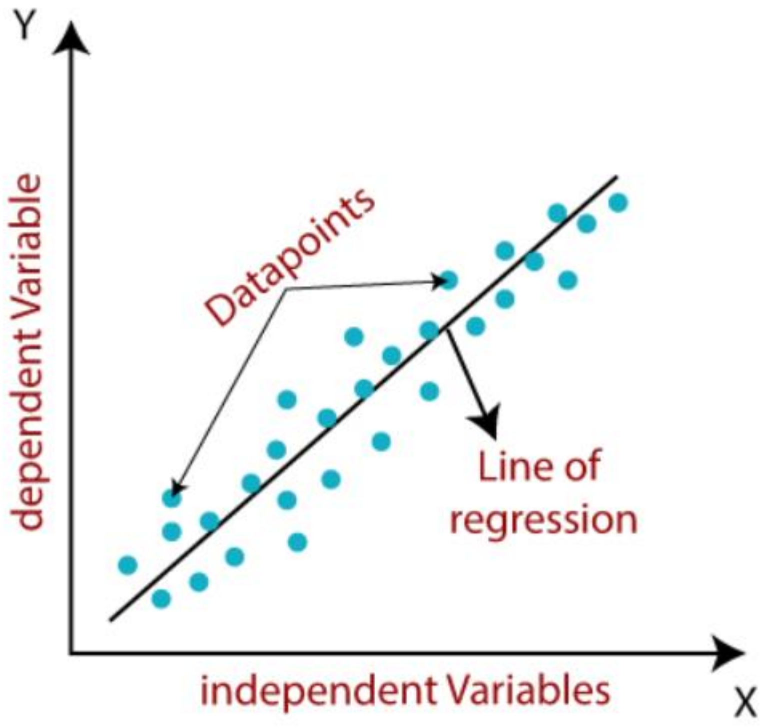


Figure 2.8.7

CHAPTER 3: REQUIREMENT ANALYSIS

3.1 Software Requirements

The software requirements in this project include:

- Python 3.7
- Jupyter Notebook
- Anaconda navigator

Python:
Python is used for creating backbone structure. Python is intended to be a highly readable language. It is designed to have an uncluttered visual layout, it uses whitespace indentation, rather than curly braces or keywords. Python has a large standard library, commonly cited as one of Python's greatest strengths.
Python Features

Python's features include −

- Easy-to-learn − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- Easy-to-read − Python code is more clearly defined and visible to the eyes.
- Easy-to-maintain − Python's source code is fairly easy-to-maintain.
- A broad standard library − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh
- Interactive Mode − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- Portable − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

Python graphical user interfaces (GUIs)

- Tkinter − Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. We would look this option in this chapter
- wxPython –**wxPython** is essentially a Python extension module that acts as a wrapper for the wxWidgets API. wxPython **allows Python developers to create native user interfaces** that add zero additional overhead to the application.
- JPython − JPython is a Python port for Java which gives Python scripts seamless access to Java class libraries on the local machine. http://www.jython.org.

There are many other interfaces available, which you can find them on the internet.

Jupyter Notebook:
The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, machine learning and much more.
Features of Jupyter Notebook
A Jupyter Notebook is fundamentally a JSON file with a number of annotations. There are three main parts of the Notebook as follows.
• Metadata: a data dictionary of definitions used to set-up and display the notebook.
• Notebook format: version numbers of the software used to create the notebook. The version number is used for backward compatibility.
• List of cells: there are three different types of cells — markdown (display), code (to excite), and output.
Anaconda navigator:
Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository. It is available for Windows, macOS, and Linux.

3.1.1 Modules Required :-

- Numpy 1.21.1
- Matplotlib
- Pandas 1.1.3
- Logistic Regression
- Decision Trees
- Random Forrest
- Seaborn 0.11.0
- Sklearn

3.2 Hardware Requirements
Linux: GNOME or KDE desktop GNU C Library (glibc) 2.15 or later, 2 GB RAM minimum, 4 GB RAM recommended, 1280 x 800 minimum screen resolution.
Windows: Microsoft R Windows R 8/7/Vista (32 or 64-bit) 2 GB RAM minimum, 4 GB RAM recommended, 1280 x 800 minimum screen resolution, Intel R processor with support for Intel R VT-x, Intel R EM64T (Intel R 64) Execute Disable (XD) Bit functionality.

3.3 Supportive Operating Systems :
The supported Operating Systems for client include:

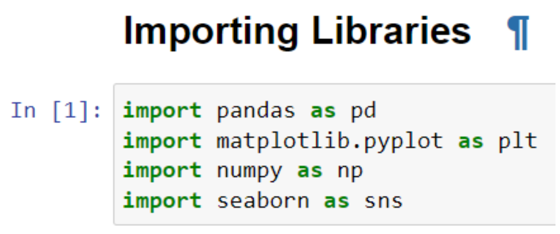
- Windows 8/10/11
- Linux any flavor.
- Ubuntu 20.04/18.04

CHAPTER 4: WORKING AND SNAPSHOTS

In this section, we will see the step-wise implementation of our project using machine learning models with python.

4.1 Import the Libraries

Collecting training data is a very crucial step while building any machine learning model. It may sound like an incredibly painful process. In this project, we have used about 2000 plus data to train our model.



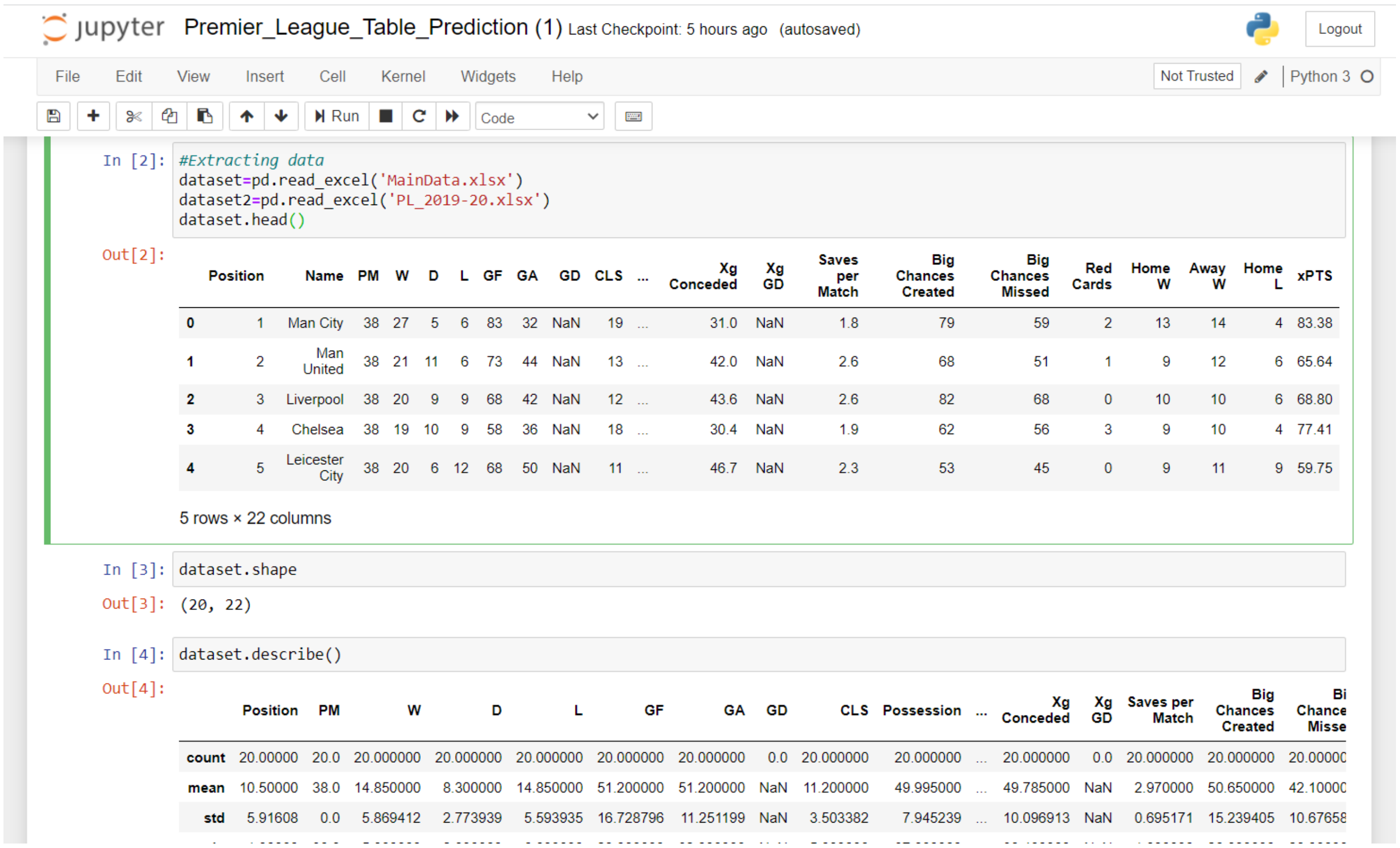
Importing Libraries ¶

```
In [1]: import pandas as pd
        import matplotlib.pyplot as plt
        import numpy as np
        import seaborn as sns
```

Figure 4.1(a). Importing Modules

4.2 Data collection

In data collection we will take a look on the data we have



jupyter Premier_League_Table_Prediction (1) Last Checkpoint: 5 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

```
In [2]: #Extracting data
        dataset=pd.read_excel('MainData.xlsx')
        dataset2=pd.read_excel('PL_2019-20.xlsx')
        dataset.head()
Out[2]:
```

	Position	Name	PM	W	D	L	GF	GA	GD	CLS	...	Xg Conceded	Xg GD	Saves per Match	Big Chances Created	Big Chances Missed	Red Cards	Home W	Away W	Home L	xPTS
0	1	Man City	38	27	5	6	83	32	NaN	19	...	31.0	NaN	1.8	79	59	2	13	14	4	83.38
1	2	Man United	38	21	11	6	73	44	NaN	13	...	42.0	NaN	2.6	68	51	1	9	12	6	65.64
2	3	Liverpool	38	20	9	9	68	42	NaN	12	...	43.6	NaN	2.6	82	68	0	10	10	6	68.80
3	4	Chelsea	38	19	10	9	58	36	NaN	18	...	30.4	NaN	1.9	62	56	3	9	10	4	77.41
4	5	Leicester City	38	20	6	12	68	50	NaN	11	...	46.7	NaN	2.3	53	45	0	9	11	9	59.75

5 rows × 22 columns

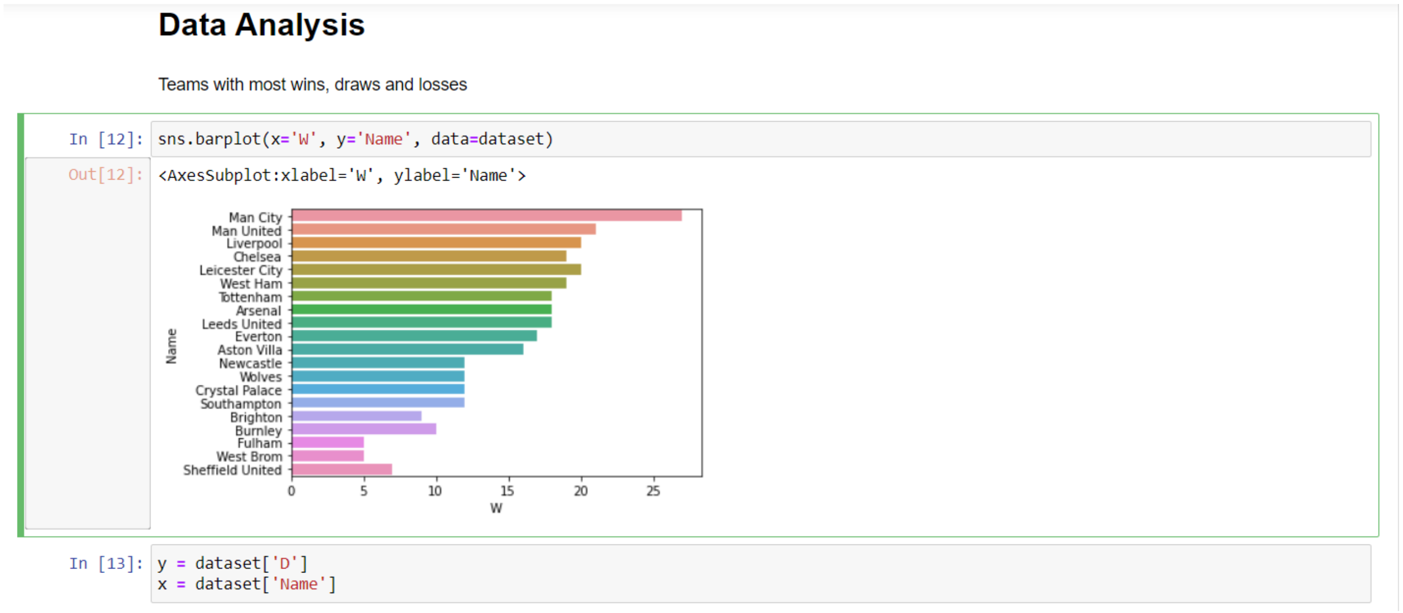
```
In [3]: dataset.shape
Out[3]: (20, 22)

In [4]: dataset.describe()
Out[4]:
```

	Position	PM	W	D	L	GF	GA	GD	CLS	Possession	...	Xg Conceded	Xg GD	Saves per Match	Big Chances Created	Bi Chance Misse
count	20.00000	20.0	20.000000	20.000000	20.000000	20.000000	20.000000	0.0	20.000000	20.000000	...	20.000000	0.0	20.000000	20.000000	20.00000
mean	10.50000	38.0	14.850000	8.300000	14.850000	51.200000	51.200000	NaN	11.200000	49.995000	...	49.785000	NaN	2.970000	50.650000	42.10000
std	5.91608	0.0	5.869412	2.773939	5.593935	16.728796	11.251199	NaN	3.503382	7.945239	...	10.096913	NaN	0.695171	15.239405	10.67658

Figure 4.2(a) showing rows and columns of the data frame

4.3 Data Analysis

In this we will analyze the PL 2020-21 season's data and will show the distribution of the data



Data Analysis

Teams with most wins, draws and losses

```
In [12]: sns.barplot(x='W', y='Name', data=dataset)
Out[12]: <AxesSubplot:xlabel='W', ylabel='Name'>
```

```
In [13]: y = dataset['D']
         x = dataset['Name']
```

Figure 4.3(a) Total wins by each club

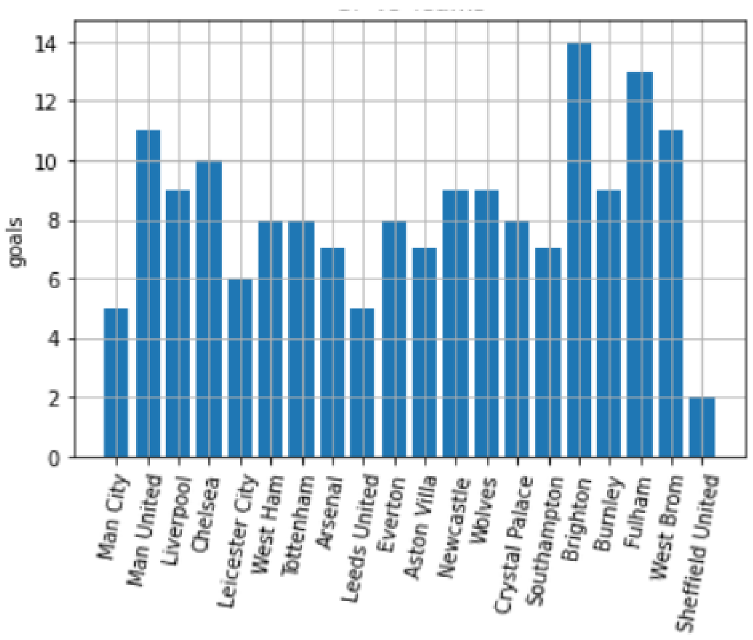
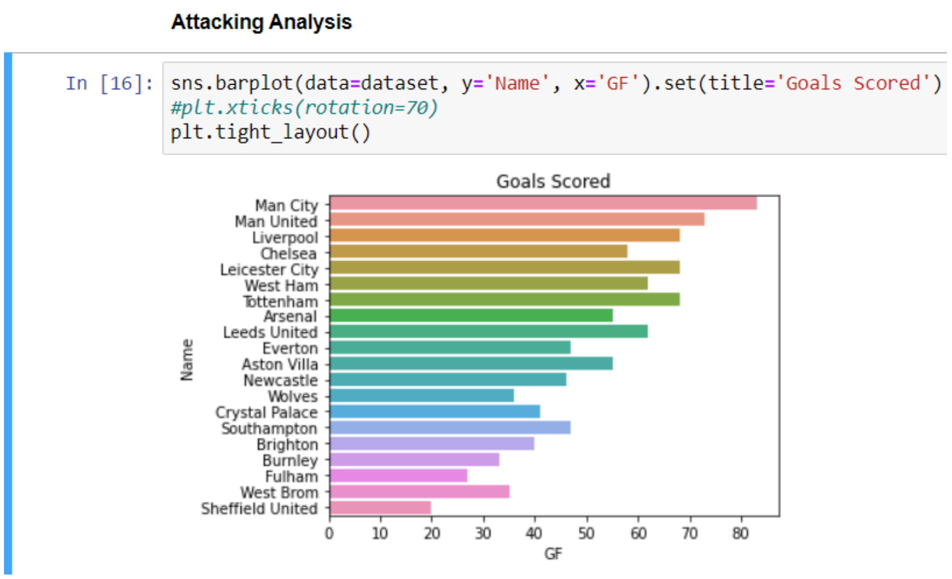


Figure 4.3(b) Total Draws by each team



Attacking Analysis

```
In [16]: sns.barplot(data=dataset, y='Name', x='GF').set(title='Goals Scored')
         #plt.xticks(rotation=70)
         plt.tight_layout()
```

Figure 4.3(c) Goals Scored by each team

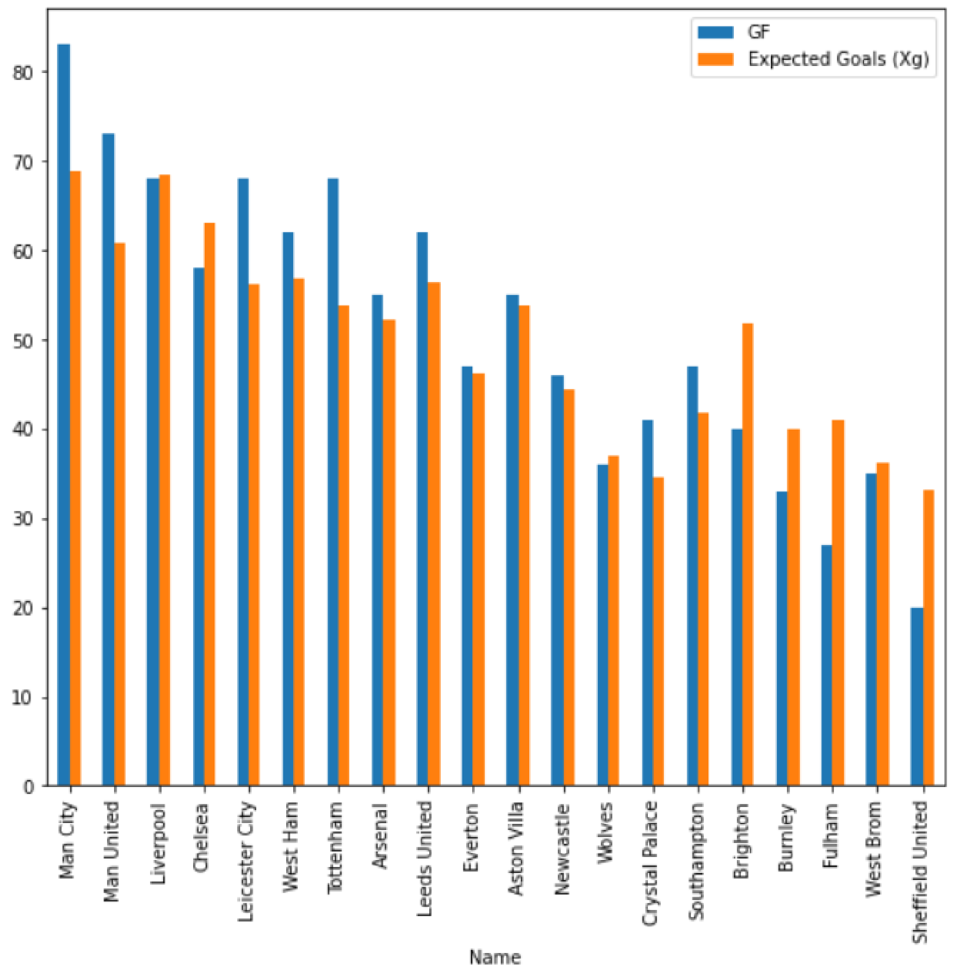


Figure 4.3(d) Goals scored vs Xg

Defending Analysis

```
In [20]: sns.barplot(data=dataset, x='Name', y='GA').set(title='Goals Conceded')
         plt.xticks(rotation=70)
         plt.tight_layout()
```

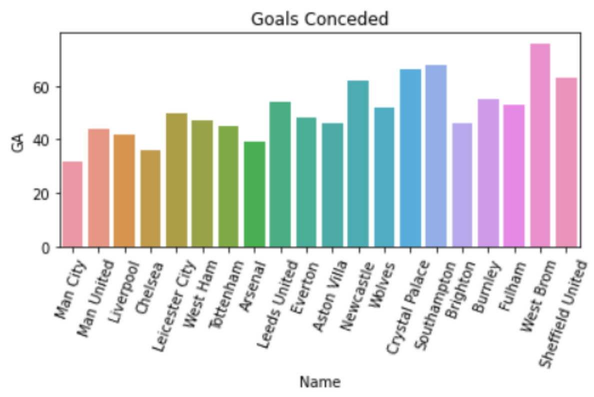


Figure 4.3(e) Goals conceded

```
In [23]: dataset.plot(x="Name", y=["GA","Xg Conceded"], kind="bar",figsize=(9,8))
Out[23]: <AxesSubplot:xlabel='Name'>
```

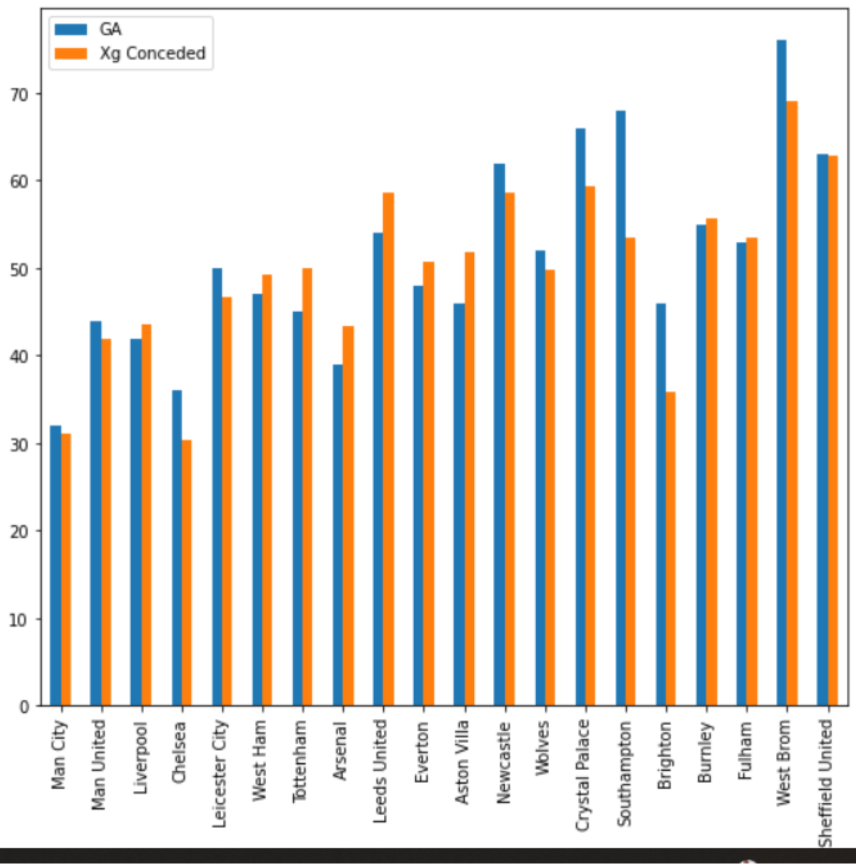


Figure 4.3(f) Goals conceded vs Xg conceded

Chances created vs Chances missed

```
In [26]: dataset.plot(x="Name", y=["Big Chances Created","Big Chances Missed"], kind="bar",figsize=(9,8))

Out[26]: <AxesSubplot:xlabel='Name'>
```

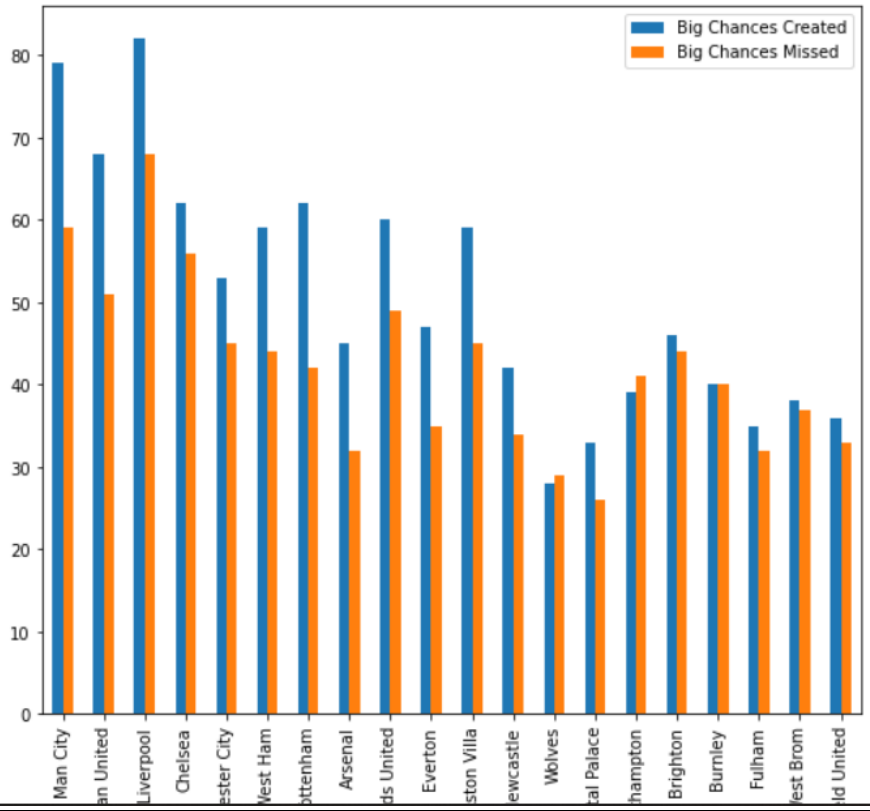


Figure 4.3(g**)** chances created vs chances missed

Goal Difference vs Expected Goal Difference

Expected Goal Difference Xg GD = Xg – Xa

```
In [27]: dataset.plot(x="Name", y=["GD","Xg GD"], kind="bar",figsize=(9,8))
```

```
Out[27]: <AxesSubplot:xlabel='Name'>
```

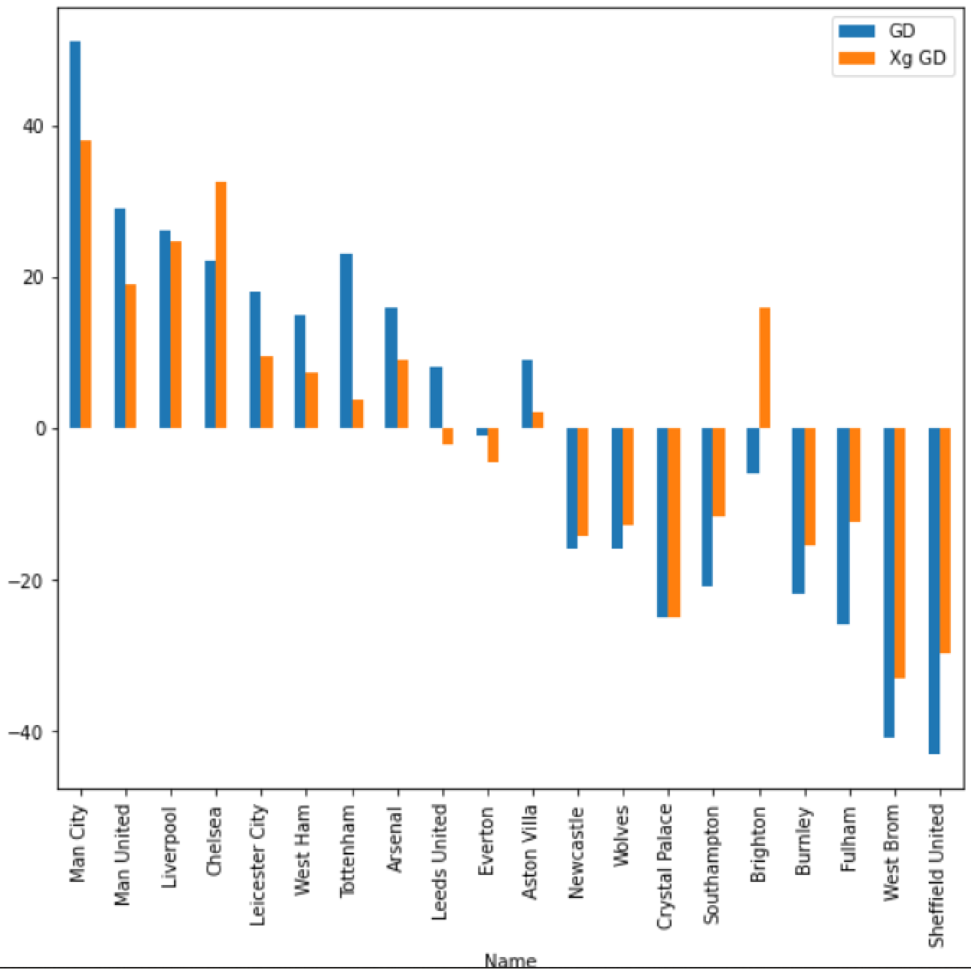


Figure 4.3(h) GD vs Xg GD

Clean Sheets vs Saves Per Match Scatter plot

```
In [29]: sns.relplot(x="Saves per Match", y="CLS", hue = 'Name',kind = 'scatter',data=dataset)
         plt.show()
```

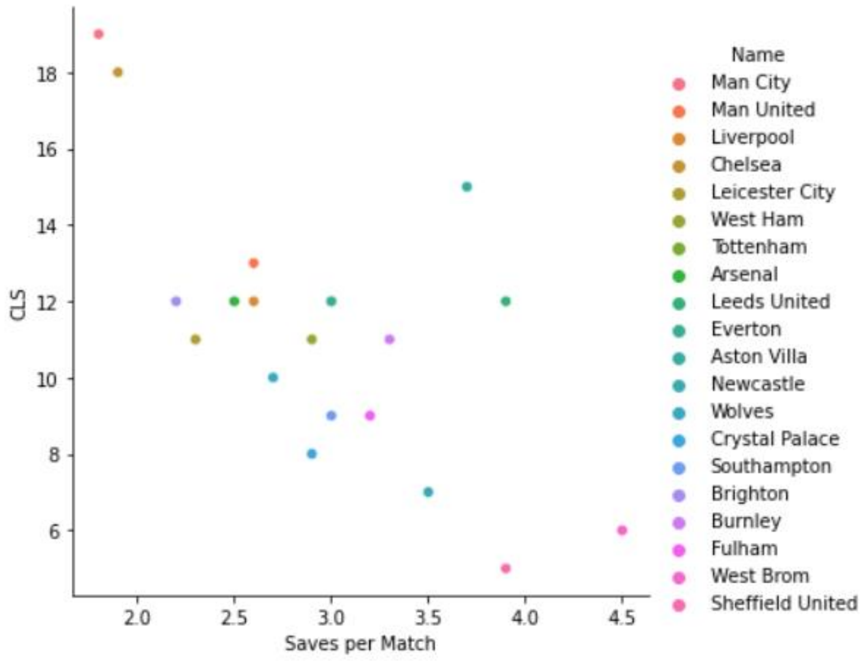


Figure 4.3(i) Saves per match vs Clean sheets

```
In [30]: sns.barplot(x='Red Cards', y='Name', data=dataset)
Out[30]: <AxesSubplot:xlabel='Red Cards', ylabel='Name'>
```

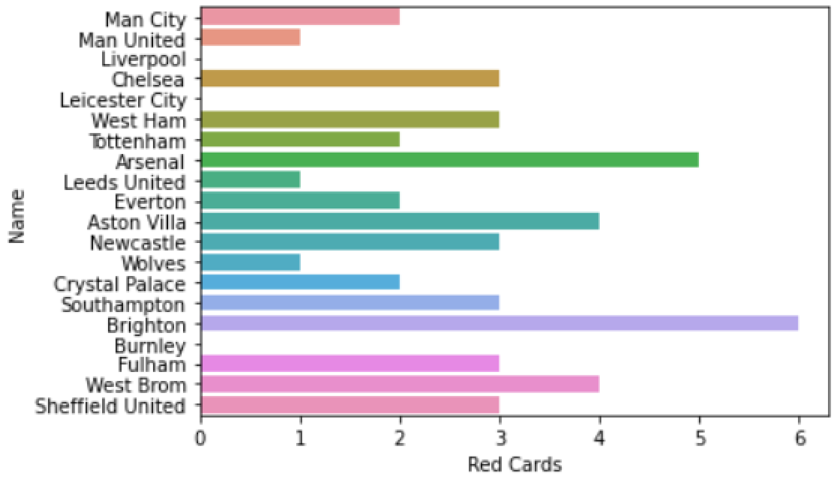


Figure 4.3(j) Red cards

Finding Correlation between Pairs and Variables using Correlation Matrix function

```
In [41]: ### Correlation matrix implementation

         corr_mat=newdf2.corr()
         sns.heatmap(corr_mat, annot=True)

Out[41]: <AxesSubplot:>
```

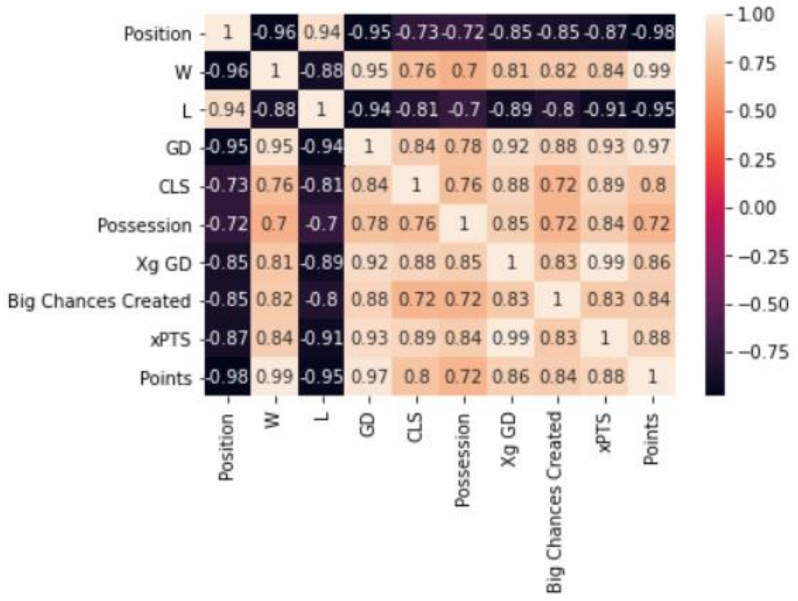


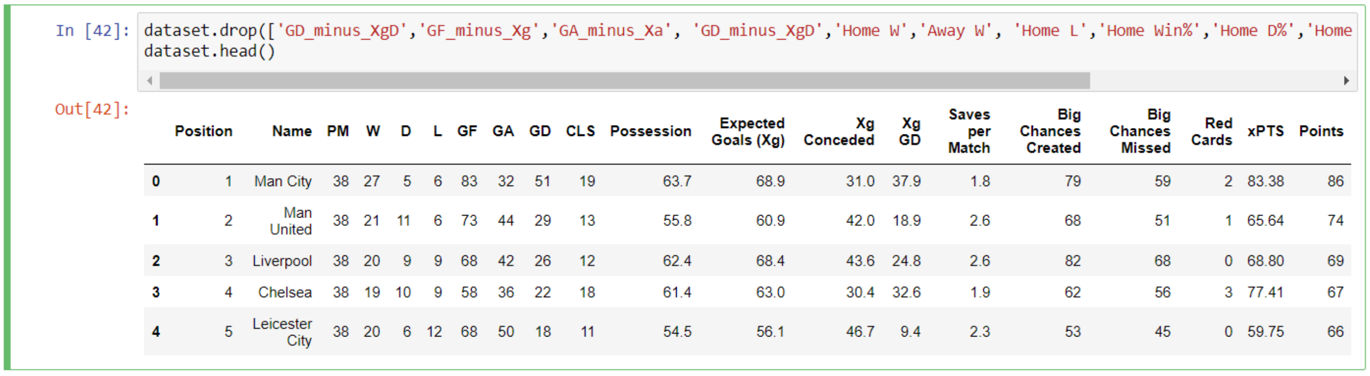
Figure 4.3(k) Red cards

4.4 Data Preprocessing

Column Description:

W: Win
D: Draw
L: Loss
GF: Goals For (Goals Scored)
GA: Goals Against (Goals Conceded)
GD: Goal Difference
Xg: Expected Goals
Xg GD: Expected Goal Difference
xPTS: Expected Points
CLS: Clean Sheets

Bottom 3 Teams Got Relegated so we won't be using their Data as they won't be in the league this season

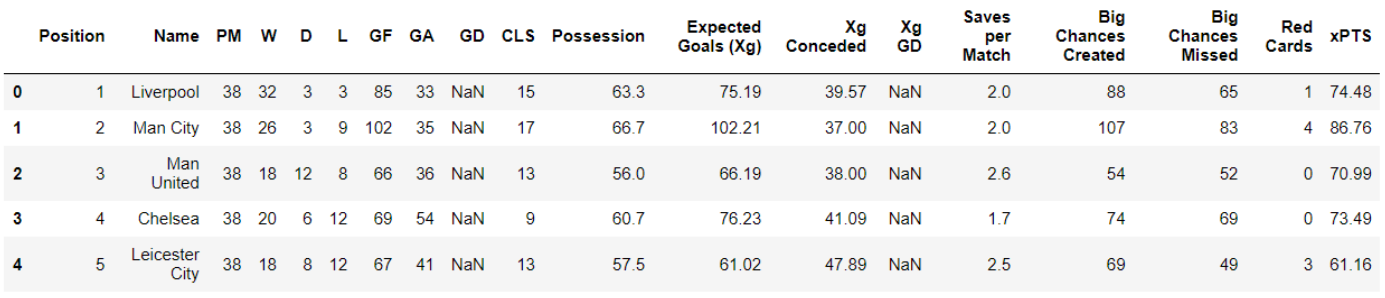


```
In [42]: dataset.drop(['GD_minus_XgD','GF_minus_Xg','GA_minus_Xa', 'GD_minus_XgD','Home W','Away W', 'Home L','Home Win%','Home D%','Home
         dataset.head()
```

Out[42]:

	Position	Name	PM	W	D	L	GF	GA	GD	CLS	Possession	Expected Goals (Xg)	Xg Conceded	Xg GD	Saves per Match	Big Chances Created	Big Chances Missed	Red Cards	xPTS	Points
0	1	Man City	38	27	5	6	83	32	51	19	63.7	68.9	31.0	37.9	1.8	79	59	2	83.38	86
1	2	Man United	38	21	11	6	73	44	29	13	55.8	60.9	42.0	18.9	2.6	68	51	1	65.64	74
2	3	Liverpool	38	20	9	9	68	42	26	12	62.4	68.4	43.6	24.8	2.6	82	68	0	68.80	69
3	4	Chelsea	38	19	10	9	58	36	22	18	61.4	63.0	30.4	32.6	1.9	62	56	3	77.41	67
4	5	Leicester City	38	20	6	12	68	50	18	11	54.5	56.1	46.7	9.4	2.3	53	45	0	59.75	66

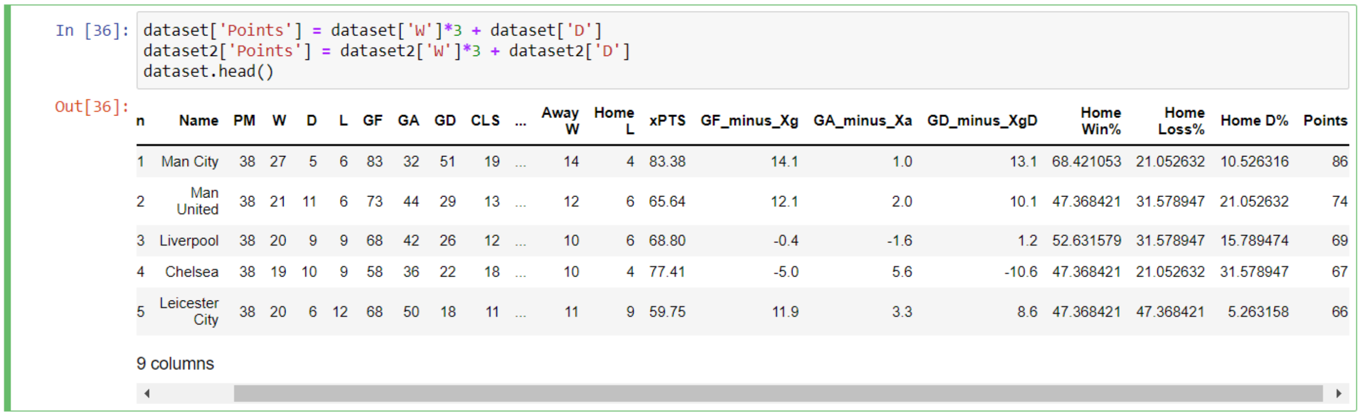
Figure 4.4(a) Dataset of 2020-21 season



	Position	Name	PM	W	D	L	GF	GA	GD	CLS	Possession	Expected Goals (Xg)	Xg Conceded	Xg GD	Saves per Match	Big Chances Created	Big Chances Missed	Red Cards	xPTS
0	1	Liverpool	38	32	3	3	85	33	NaN	15	63.3	75.19	39.57	NaN	2.0	88	65	1	74.48
1	2	Man City	38	26	3	9	102	35	NaN	17	66.7	102.21	37.00	NaN	2.0	107	83	4	86.76
2	3	Man United	38	18	12	8	66	36	NaN	13	56.0	66.19	38.00	NaN	2.6	54	52	0	70.99
3	4	Chelsea	38	20	6	12	69	54	NaN	9	60.7	76.23	41.09	NaN	1.7	74	69	0	73.49
4	5	Leicester City	38	18	8	12	67	41	NaN	13	57.5	61.02	47.89	NaN	2.5	69	49	3	61.16

Figure 4.4(b) Dataset of 2019-20 season

Adding Total Points in this Dataset



```
In [36]: dataset['Points'] = dataset['W']*3 + dataset['D']
         dataset2['Points'] = dataset2['W']*3 + dataset2['D']
         dataset.head()
```

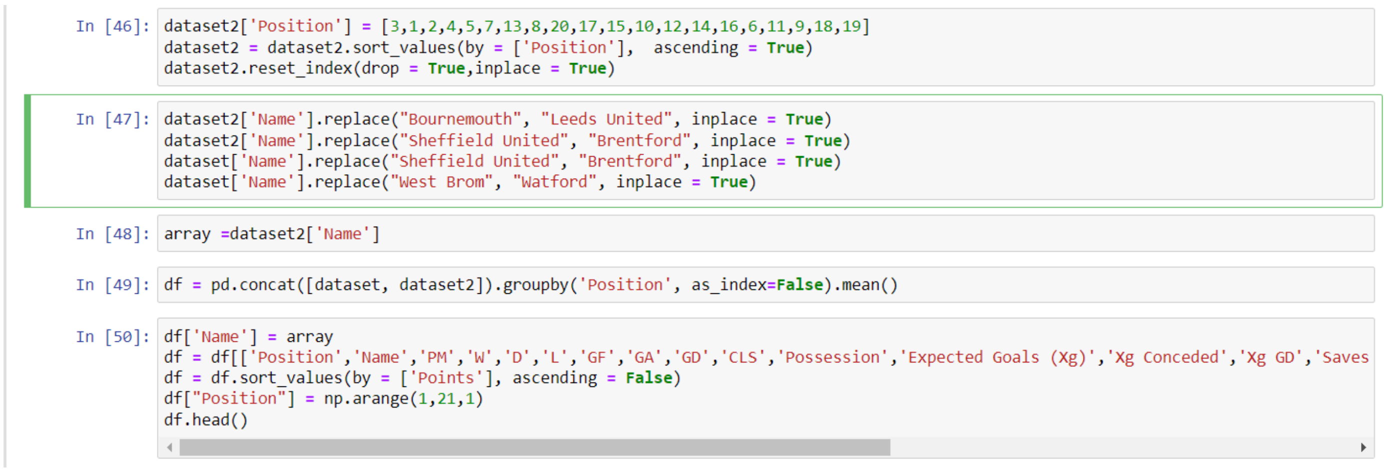
Out[36]:

n	Name	PM	W	D	L	GF	GA	GD	CLS	...	Away W	Home L	xPTS	GF_minus_Xg	GA_minus_Xa	GD_minus_XgD	Home Win%	Home Loss%	Home D%	Points
1	Man City	38	27	5	6	83	32	51	19	...	14	4	83.38	14.1	1.0	13.1	68.421053	21.052632	10.526316	86
2	Man United	38	21	11	6	73	44	29	13	...	12	6	65.64	12.1	2.0	10.1	47.368421	31.578947	21.052632	74
3	Liverpool	38	20	9	9	68	42	26	12	...	10	6	68.80	-0.4	-1.6	1.2	52.631579	31.578947	15.789474	69
4	Chelsea	38	19	10	9	58	36	22	18	...	10	4	77.41	-5.0	5.6	-10.6	47.368421	21.052632	31.578947	67
5	Leicester City	38	20	6	12	68	50	18	11	...	11	9	59.75	11.9	3.3	8.6	47.368421	47.368421	5.263158	66

9 columns

Figure 4.5(c) Calculating Total points

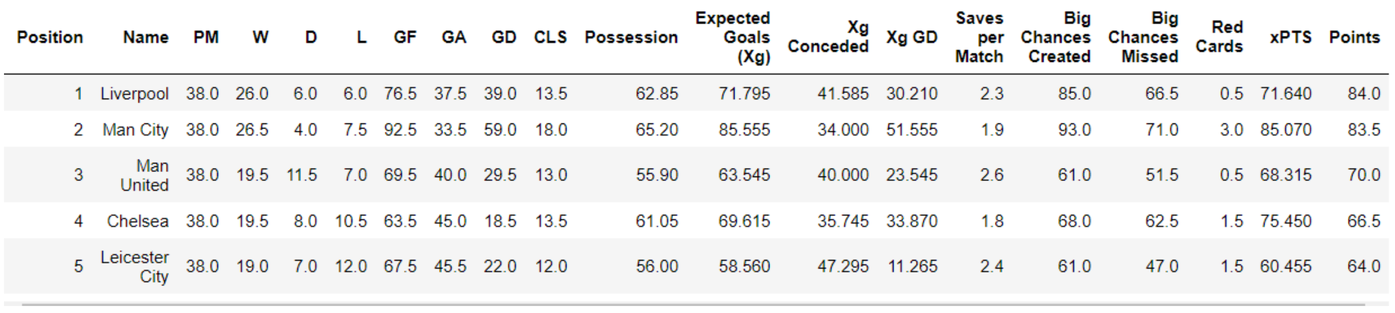
Concatenation of both datasets by taking their mean.



```
In [46]: dataset2['Position'] = [3,1,2,4,5,7,13,8,20,17,15,10,12,14,16,6,11,9,18,19]
         dataset2 = dataset2.sort_values(by = ['Position'],  ascending = True)
         dataset2.reset_index(drop = True,inplace = True)

In [47]: dataset2['Name'].replace("Bournemouth", "Leeds United", inplace = True)
         dataset2['Name'].replace("Sheffield United", "Brentford", inplace = True)
         dataset['Name'].replace("Sheffield United", "Brentford", inplace = True)
         dataset['Name'].replace("West Brom", "Watford", inplace = True)

In [48]: array =dataset2['Name']

In [49]: df = pd.concat([dataset, dataset2]).groupby('Position', as_index=False).mean()

In [50]: df['Name'] = array
         df = df[['Position','Name','PM','W','D','L','GF','GA','GD','CLS','Possession','Expected Goals (Xg)','Xg Conceded','Xg GD','Saves
         df = df.sort_values(by = ['Points'], ascending = False)
         df["Position"] = np.arange(1,21,1)
         df.head()
```

Figure 4.4(d) Concat func

New df dataset –

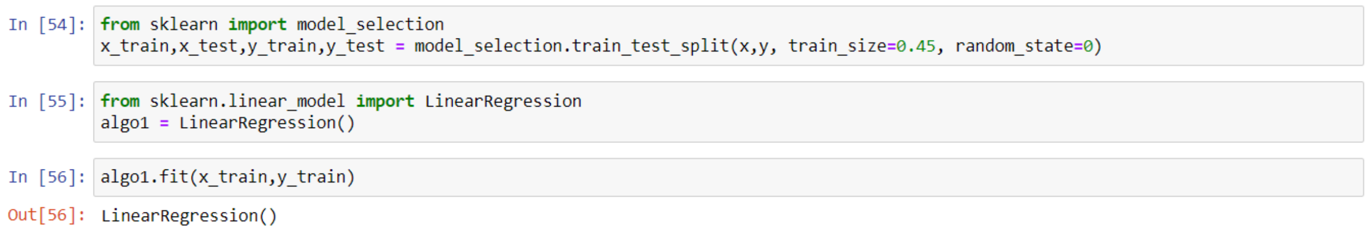


Position	Name	PM	W	D	L	GF	GA	GD	CLS	Possession	Expected Goals (Xg)	Xg Conceded	Xg GD	Saves per Match	Big Chances Created	Big Chances Missed	Red Cards	xPTS	Points
1	Liverpool	38.0	26.0	6.0	6.0	76.5	37.5	39.0	13.5	62.85	71.795	41.585	30.210	2.3	85.0	66.5	0.5	71.640	84.0
2	Man City	38.0	26.5	4.0	7.5	92.5	33.5	59.0	18.0	65.20	85.555	34.000	51.555	1.9	93.0	71.0	3.0	85.070	83.5
3	Man United	38.0	19.5	11.5	7.0	69.5	40.0	29.5	13.0	55.90	63.545	40.000	23.545	2.6	61.0	51.5	0.5	68.315	70.0
4	Chelsea	38.0	19.5	8.0	10.5	63.5	45.0	18.5	13.5	61.05	69.615	35.745	33.870	1.8	68.0	62.5	1.5	75.450	66.5
5	Leicester City	38.0	19.0	7.0	12.0	67.5	45.5	22.0	12.0	56.00	58.560	47.295	11.265	2.4	61.0	47.0	1.5	60.455	64.0

Figure 4.4(e) New modified dataset

Splitting data for training 2019-20 dataset:

Training size is 0.45 that means we trained only 45% of the data randomly.



```
In [54]: from sklearn import model_selection
         x_train,x_test,y_train,y_test = model_selection.train_test_split(x,y, train_size=0.45, random_state=0)

In [55]: from sklearn.linear_model import LinearRegression
         algo1 = LinearRegression()

In [56]: algo1.fit(x_train,y_train)
Out[56]: LinearRegression()
```

Figure 4.4(f) Data splitting

4.5 Model Evaluation

```
In [61]: tra_data_pred = algo1.predict(x_train)
```

```
In [62]: from sklearn import metrics
         r2_train = metrics.r2_score(y_train,tra_data_pred)
         print(r2_train*100)

         100.0
```

```
In [63]: from sklearn import metrics
         r2_test = metrics.r2_score(y_test_for_eval,y_predicted_for_eval)
         print(r2_test*100)

         86.7151247511712
```

```
In [64]: import matplotlib.pyplot as plt
         m = 1
         c = 0
         x_line = np.arange(15,90,5)
         y_line = m*x_line + c
         plt.plot(x_line,y_line,'r')
         plt.scatter(y_test_for_eval , y_predicted_for_eval)
         plt.show()
```

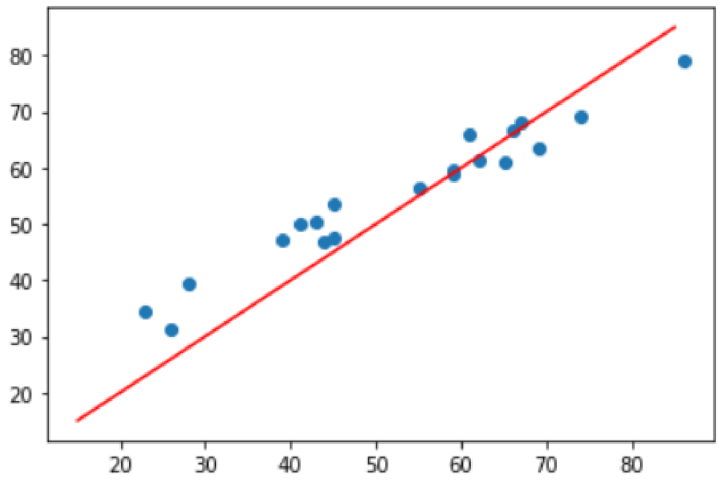


Figure 4.5(a) Model evaluation in graph

4.6 Final Prediction –

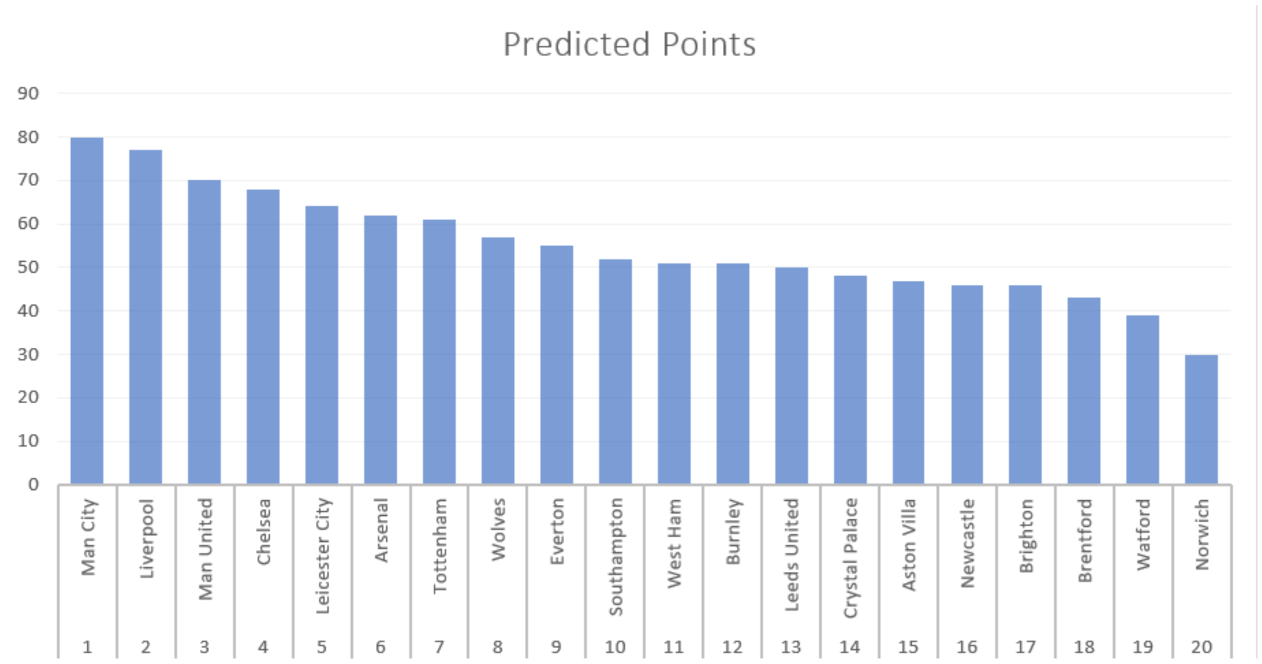
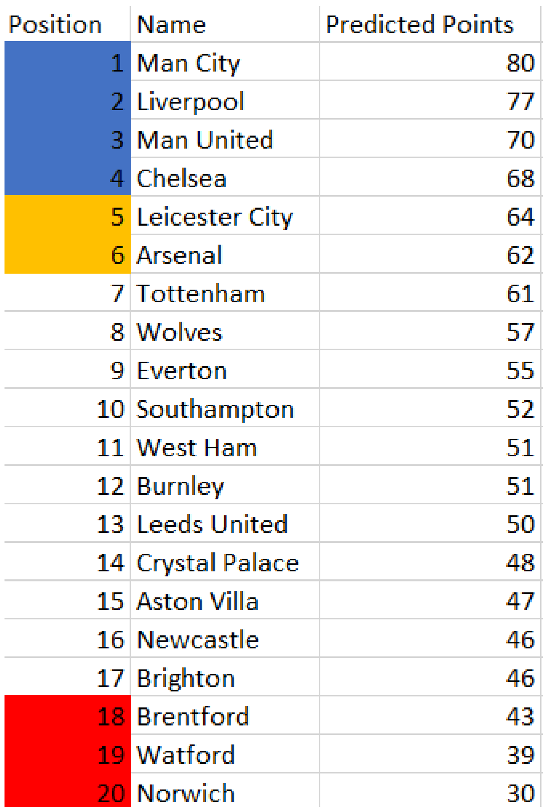


Figure 4.6(a) Points Table of PL 21-22 season visualization



Position	Name	Predicted Points
1	Man City	80
2	Liverpool	77
3	Man United	70
4	Chelsea	68
5	Leicester City	64
6	Arsenal	62
7	Tottenham	61
8	Wolves	57
9	Everton	55
10	Southampton	52
11	West Ham	51
12	Burnley	51
13	Leeds United	50
14	Crystal Palace	48
15	Aston Villa	47
16	Newcastle	46
17	Brighton	46
18	Brentford	43
19	Watford	39
20	Norwich	30

Figure 4.6(b) Full PL Table with predicted points

Chapter 5 -Conclusion / Future Work

Conclusion –

The result of this project which has the metrics score of 86%. This model is very different when compared to models that are traditionally used by betting companies. It’s fast and easy to understand. When only training 45% of one season’s dataset, it is 40% accurate which is very decent in sports prediction. On the other hand in this project the features which are related to the players are not included so the confidence is not measured and the accuracy is lower. Machine learning can be used to predict in different field, including sports, especially football. Our model is good enough to determine betting odds but obviously it’s not extremely accurate to predict the exact table and exact points scored by each team.

Future Work -

It would be better to add more datasets features like corner goals scored, penalties conceded and do a comparative work to get a more accurate and confident model. By adding some more features like transfer market activities of each club and in which area (defense, offense, midfield). Analyzing more to get which team needs strengthening in which area of the pitch.
Testing on more ML algorithms to find out which algorithm will create the most accurate model.

Chapter 6 – References

[1] https://matplotlib.org/
[2] https://en.wikipedia.org/wiki/Matplotlib
[3] https://pandas.pydata.org/
[4] https://en.wikipedia.org/wiki/Pandas_(software)
[5] https://en.wikipedia.org/wiki/Python_(programming_language
[6] https://medium.com/analytics-vidhya/
[7] https://Youtube.com