

**Draft Project Report**

**An Interactive Dashboard for Football Data Analysis**

by

Akash Dasmondal

(1847104)

Under the guidance of

Dr. Nizar Banu P K

A Draft Project Report document submitted in partial fulfillment of the requirements for the award of degree of Master of Science (Computer Science) of CHRIST (Deemed to be University)

August - 2019

**TABLE OF CONTENTS**

1. Synopsis 6

2. Introduction 7

2.1 Project Description 7

2.2 Existing System 7

2.2.1 Infiniti Red Bull Racing 7

2.2.2 Zebra Technologies 8

2.2.3 Sports UV 8

2.2.4 Catapult Team Tracking System 8

2.2.5 Future of Sports Analysis 8

2.3 Proposed System 9

2.4 Objectives 10

2.5 Purpose, Scope, and Applicability 10

2.5.1 Purpose 10

2.5.2 Scope 10

2.5.3 Applicability 10

3. System Analysis and Requirements 11

3.1 Problem Definition 11

3.2 Block Diagram 11

3.3 System Requirements 12

3.3.1 User Characteristics 12

3.3.2 Software and Hardware Requirements 12

3.3.3 Constraints 12

4. Dataset Description 13

5. User Interface Design 19

5.1 Modules 19

5.1.1 Analysis 19

5.1.2 Reports 19

5.1.3 Feedback 19

5.2 Software Tools Used 19

5.2.1 HTML 19

5.2.2 JavaScript 20

5.2.3 Bootstrap 20

5.2.4 CSS 20

5.2.5 Flask 21

5.3 User Interface 21

5.3.1 Draft Design 21

5.3.2 User Interfaces 22

6. Implementation 23

6.1 Player Positions 23

6.2 Pre-processing 25

6.3 Player Position Analyser 25

6.3.1 Pre-processing for Position Analyser 25

6.3.2 Analysis 26

6.3.3 Some Prime Attribute Inferences 29

6.4 Player Position Predictor 30

6.4.1 Decision Tree Classifier 30

6.4.2 Random Forest Classifier 34

6.5 BMI Analyser 37

6.6 Code 39

6.6.1 Position Analyser 39

6.6.2 Position Predictor Using Decision Tree 42

6.6.3 Position Predictor Using Random Forest 47

6.6.4 BMI Analyser 50

1. **SYNOPSIS**

This project aims to create a dashboard that would provide football-focused insights by making use of player performance and attribute statistics which are part of a dataset we have built using various sources. The insights are focused on three modules which perform their own focused analysis on the player dataset.

The first module focuses on player position analysis and prediction. The first part of the module will provide insights on the most valuable attributes for players playing in a particular position. The second part of the module will predict the optimal position for a player based on their attributes. The module will perform multiple classification and multi-label probabilistic classification.

The second module will assess the impact of BMI (Body Mass Index) on performance and attributes of high-level football players. The module will include an analysis model to help players of particular BMIs choose their best positions. The system analysis variations in BMI distributions across multiple performance and position parameters and provides graphical distributions as a reference to make BMI-based position decisions.

The third module is based on analysing progression of players in the world’s top professional clubs to determine the best player development systems. Using the data available in the player attribute dataset, we can chart progression of players from the start of their career at their professional club. Analysis of the dataset will help determine clubs which have produced/nurtured the best young talent. This can also be filtered based on player position, nationality and so on.

The project will implement predictive analysis using machine learning. Python and Python-based data analysis packages will be used to pre-process, manipulate, and analyse the dataset to achieve the specified objectives. Module 2 in particular will use Decision Tree Classification and Random Forest Classification. The front-end dashboard will be built on HTML/CSS with a BootStrap framework, with the assistance of AJAX and JavaScript for client-side scripting.

1. **INTRODUCTION**
   1. **Project Description**

This project aims to create a dashboard that would provide football-focused insights by making use of team performance statistics and player quality attributes which have been collected from sources like DataHub.io and EA Sports FIFA franchise player database. The insights provided by the project are focused on three modules.

The first module analysis position attribute importance and performs predictive analysis to determine optimal position.

The second module focuses on assessing the impact of BMI (Body Mass Index) on performance and attributes of high-level football players.

The third module analyses player progression over a period of time. This progression is analysed in relation to player position, age, formation, team ranking and so on. This project and its three modules will create a framework that works with data files that are fed into the system, provided they follow the prescribed format.

* 1. **Existing System**

The following companies and technologies are examples of data analysis employed in the various sporting realms, highlighting the value for analytical techniques in the modern sports industry.

**2.2.1 Infinti Red Bull Racing**

* Formula one is one of the most popular sports in the world, and one of the most data driven.
* Red Bull racing rely on real-time information to make enhancement to cars and in race strategies. Every lap driven by the F1 car.
* Sensors produce enough data like tire pressure, tire temperature, suspension performance, wheel alignment and whole lot of engine statistics, the pit team uses that data to make immediate decisions [1].

**2.2.2 Zebra Technologies**

* The company makes Radio Frequency Identification tags, as part of their Motion works.
* Sports Solution, that attach to equipment, balls and players to track movement, distance and speed.
* The tags blink 25 times per second and deliver data in 120 milliseconds.
* This data is used to analyse in the post-match session to understand what worked and what didn’t work for the teams [2].

**2.2.3 Sports UV**

* The Company has six cameras in each NBA arena that collect data on the movements on every player and movements of the basketball 25 times per second.
* This data is used by game developers to understand ball movements and incorporate them [3].

**2.2.4 Catapult Team Tracking System**

Originally formed from a partnership between the Australian Institute of Sport (AIS) and the Cooperative Research Centres (CRC) to maximise the performance of Australian athletes ahead of the Sydney Olympics, Catapult was officially founded in Melbourne in 2006 [4].

**2.2.5 Future of Sports Analysis**

The following technologies and tools are in the process of being developed to further advance the field of sports data analysis:

* Kit man Labs
* P3 Applied Sports Science
* Sparta Science
* SAP Labs
  1. **Proposed System**

The football analysis dashboard will make use of Python-based data analysis and statistical techniques to provide useful analytical output for the specified three modules.

All three modules will be built as analytical frameworks, which are designed to work for any data fed into the system. Users will be able to create analysis profiles for each of the modules. Each analysis profile has a single folder of data files. The scope of each data file is up to the user and the objective of the profile, provided each file follows the data and attribute format as specified by the module’s requirements. For example, the profile can focus solely on one team, or an entire league of twenty teams. The data file can be updated and reloaded to the module for an updated analysis. This system will be implemented across the three modules.

The modules will use a datasets of player ratings and attributes, drawn from EA(Electronic Arts) Sports.

The first module focuses on player position analysis and classification. Correlation is used to determine the most important positions for players of particular positions. Decision Tree Classification and Random Forest Classification are used to perform the classification necessary to predict optimal position of a player for a particular set of player attributes.

The second module works with player ratings and attributes, and seeks to connect them to the Body Mass Index (BMI) of the players. This analysis will relate BMI to effective player positions, effective formations, and the BMI of high-quality players in top teams, to give a good indication of BMI values and how they associate with particular player attributes.

The third module charts player progression. The module will be built to provide an analytical visualization of player progression in a flexible manner. For example, it can be limited to a particular team, or an entire league, or all players in Europe, provided the relevant statistics are provided in the associated CSV files.

* 1. **Objectives**

The interactive data analysis dashboard for football aims to provide a tool to help analyse trends and predict potential outcomes in the realm of the sport of football.

* The BMI-based module helps players determine the best position, league, formation, and team for their particular BMI value. This extends to assisting managers in determining the ideal strategy to use based on the team’s BMI trends.
* The player progression model can be useful in determining the scale of player progression with respect to teams, leagues, and nationalities. This could help players make right career decisions, and scouts and recruiters recognize problem areas in player advancement.
  1. **Purpose, Scope, and Applicability**
     1. **Purpose**

It is a collection of relevant, historical, statistics that when properly applied can provide a competitive advantage to a team or individual.

* + 1. **Scope**

Technique collects various strategies and historical statistics to provide a competitive benefit to teams or individuals. Globally, a noticeable market trend is evident “Heighten Viewership of Domestic League”.

* + 1. **Applicability**

This project can be used for amateur sports teams or low level sports league.

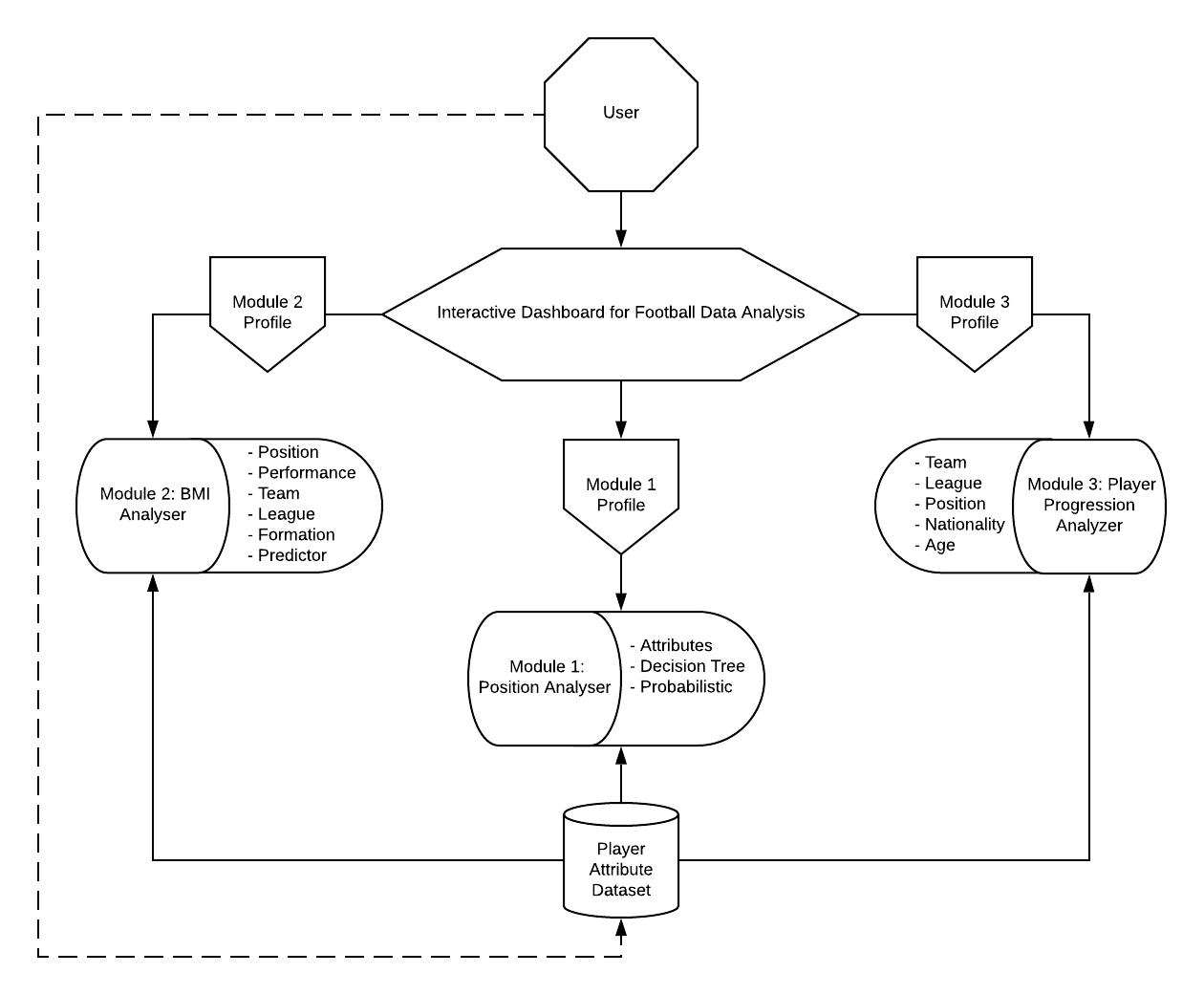
It can be used in school and colleges in order to develop the players with a decision making system to perform for better clubs and right fit for future.

The project module can be used to maintain the efficiency of BMI using the application.

1. **SYSTEM ANALYSIS AND REQUIREMENTS**
   1. **Problem Definition**

An interactive dashboard for football data analysis that:

* Analyses the impact of BMI on player and team performance
* Predicts match results based on previous results combined with team ratings
* Charts player progression over a range of variables
  1. **Block Diagram**

****

**Figure 1 : System Model**

* 1. **System Requirements**
     1. **User Characteristics**

There are approximately 3-4 types of users for the system based on the required functionality such as Coach, Scouts, Players, and others who are interested in football, commercially or personal interest. The user needs to know what exactly he/she is looking for and based on that needs to give proper inputs.

* + 1. **Software and Hardware Requirements**

Recommended Operating System

* Windows: 7 or newer
* MAC: OS X v10.7 or higher
* Linux: Ubuntu

Hardware Requirements

* Processor: Minimum 1 GHz; Recommended 2GHz or more
* Ethernet connection (LAN) OR a wireless adapter (Wi-Fi)
* Hard Drive: Minimum 32 GB; Recommended 64 GB or more
* Memory (RAM): Minimum 2 GB; Recommended 4 GB or above

Recommended Software

* HTML5, CSS3, JavaScript, Django, Python, Microsoft Excel, Visual Studio Code
  + 1. **Constraints**
* This application will not be able to give proper outputs to certain countries.
* Need to feed proper format of data into the algorithm for accurate results.
* Countries with low football growth are limited to this application.

1. **DATASET DESCRIPTION**

The objective of this project is to create a predictive model for player attributes and player progression. Considering the differing objectives of each of these two modules, there is a requirement for two separate datasets, as described below.

The Interactive Dashboard for Football Data analysis project will make use of the following two datasets:

1. A CSV file comprising player attributes, which includes player height and weight. Going forward, the document will refer to this file as Player Attribute Dataset. This dataset is limited to one year’s worth of player data and will help create a model to preform predictive classification.
   * This dataset will comprise 50 columns (attributes) and 18200 rows (players).
   * This dataset is sourced from EA Sports FIFA franchise data.
2. A CSV file comprising player ratings over time, which can be used to chart player progression. Going forward, the document will refer to this file as the Player Progression Dataset. This dataset comprises player data over a period of time which can help chart a predictive progression model.
   * This dataset is meant to be flexible. The initial dataset will be used to build a model, which can then be implemented for any future CSV file which contains the relevant attributes. The dataset will have 9 columns (attributes).
   * There exists no available dataset which provides player rating data over a span of time. Therefore, the project sources this data from EA Sports and combines them into a single CSV file which includes rating over a period of years.

The attributes specified above are elaborated on in the following tables:

**Player Attribute Dataset:**

|  |  |
| --- | --- |
| **Attribute Name** | **Attribute Description** |
| ID | Unique Identifier for player. Pre-processing required to convert raw IDs to updated ascending values. |
| Name | Player names. ID will help differentiate players with similar names. |
| Age | Player age. |
| Photo | A URL which specifies the player photograph. Can be used in the project GUI. |
| Nationality | Specifies player nationality. |
| Flag | A URL which specifies flag corresponding to player nationality. Can be used in the project GUI. |
| Overall | Specifies player rating, which signifies the quality of the player during the current year, and for the relevant position. Value ranges from 40-100.  This is a critical piece of information which forms the basis of analysing player quality/performance with respect to any other attributes, as well as charting player progression over time. |
| Club | Specifies the club the player plays for. |
| Club Logo | URL which specifies the logo for the club the player plays for. Can be used for the project GUI. |
| Value | Estimated market value of the player. No plan to include this attribute in analysis at the moment. |
| Weak Foot | An ordinal value (1-5) which specifies ability with weaker foot. |
| Skill Moves | An ordinal value (1-5) which specifies ability and frequency of performing complex dribbling and skill moves. |
| Work Rate | Specifies Attacking/Defending work rate of player. |
| Position | Specifies preferred position of player. |
| Height | Specifies player height. |
| Weight | Specifies player weight. |
| **Specific Player Attributes (Range from 0 to 100)** | |
| Crossing | Player crossing ability. |
| Finishing | Measure of player’s ability to finish in front of goal. |
| HeadingAccuracy | Player’s accuracy with headers in front of goal. |
| ShortPassing | Player’s short passing ability. |
| Volleys | Player’s effectiveness with volleys in front of goal. |
| Dribbling | Player’s dribbling ability. |
| Curve | Player’s ability to curve the ball with free-kicks and shots. |
| FKAccuracy | Player’s accuracy and efficiency with free-kicks. |
| LongPassing | Player’s long passing ability. |
| BallControl | Measure of player’s ball control. |
| Acceleration | Measure of player’s ability to reach top speed in sprint. |
| SprintSpeed | Measure of player’s top speed in sprint. |
| Agility | Measure of player’s agility in combination with dribbling and ball control. |
| Reactions | Player’s ability to react to sudden changes in play. |
| Balance | Measure of player’s balance in combination with dribbling, ball control, and agility. |
| ShotPower | Measure of player’s shot power in front of goal. |
| Jumping | Measure of player’s ability to jump in attempts to reach a high ball. |
| Stamina | Measure of player’s stamina. |
| Strength | Measure of player’s ability to apply strength effectively in competitive situations. |
| LongShots | Measure of player’s effectiveness and accuracy with long shots towards goal. |
| Aggression | Measure of player’s aggression when competing for ball and leading team in high pressure situations. |
| Interceptions | Measure of player’s ability to intercept passes by the opposition team. |
| Positioning | Measure of player’s situational awareness and effective positioning during attacking play. |
| Vision | Measure of player’s ability to see how play unfolds and make effective through passes. A measure of a playmaker. |
| Penalties | Measure of a player’s ability to take penalties. |
| Composure | Measure of a player’s ability to stay composed under pressure. |
| Marking | Measure of a defender’s ability to effectively mark attacking opposition. |
| StandingTackle | Measure of a defender’s ability with standing tackles. |
| SlidingTackle | Measure of a defender’s ability with sliding tackles. |
| **Goalkeeper Exclusive Attributes** | |
| GKDiving | Measure of goalkeeper’s ability to dive. |
| GKKicking | Measure of goalkeeper’s kicking accuracy and power. |
| GKHandling | Measure of goalkeeper’s ability and composure in handling the ball and palming away shots to safe locations. |
| GKPositioning | Measure of goalkeeper’s effectiveness in positioning between the posts, in order to make effective saves. |
| GKReflexes | Measure of goalkeeper’s reflexes in reacting to close range shots. |

**Player Progression Dataset:**

The player progression dataset makes use of similar attributes to those specified in the Player Attribute Dataset. However, the records for each player are repeated for each year of their professional career in a League. As such, using the Player Attribute Dataset here would lead to a large volume of redundant data. Therefore, the project maintains a separate dataset for player progression with the following attributes:

|  |  |
| --- | --- |
| **Attribute Name** | **Attribute Description** |
| Player ID | Unique ID for each player. However, this ID will repeat for each year’s record of the player. Useful to identify players uniquely. |
| Name | Player name. |
| Age | Player age. |
| Position | Player preferred position for that year. |
| Team | Player team for that year. |
| League | Player league for that year. |
| Nationality | Player nationality. |
| Rating | Specifies player rating, which signifies the quality of the player during the current year, and for the relevant position. Value ranges from 40-100.  This is a critical piece of information which forms the basis of analysing player quality/performance with respect to any other attributes, as well as charting player progression over time. |
| Year | Year of recording the particular players attributes. Since the projects aim to chart progression, the project will be collecting player data for multiple years to build our module. |

**Modules and Datasets:**

The following provides a description of the two modules which comprise this project, and how they make use of the above datasets:

**Player Position Analysis Module:**

This module has two parts. The first part uses correlation to analyse the most important player attributes for a particular position, to help determine focus areas for player training based on position preferences.

The second part predicts the optimal position for a player based on player attributes. This is performed using Decision Tree Classification and Random Forest Classification for probabilistic position prediction. The dataset used for this prediction consists of player numerical assessment values.

**BMI Analyser Module:**

This module will focus on an analysis of player attributes in relation to the player’s Body Mass Index (BMI), which is computed as a function of the player’s height and weight. The project will correlate the player’s BMI with all player attributes, both performance-related and descriptive, to recognise relevant patterns. An example would be a correlation between player BMI and position.

Using this the module can attempt to identify effective Positions and focus areas for players with particular BMI values (in combination with other attributes if required).

**Player Progression Module:**

The player progression module will make use of the Player Progression Dataset exclusively, to build a model which generates a report which highlights the measure of progression of players in relation to a flexible set of attributes, like team, position, league, age, nationality, and so on. An attempt will then be made to predict player progression using this module.

Decision Tree Regression will help create a predictive path for players based on the model generated. For example, we can generate a value or create a model that signifies progression of a player in a particular League or Team in a particular position. Applying this model to a new player can help predict future player rating values.

**5. USER INTERFACE DESIGN**

**5.1. Modules**

**5.1.1 Analysis**

This project consists of 2 modules as mentioned before.

This module requires 2 CSV files, where the data from the CSV files are given as input to an algorithm, processed and send or display back the generated output.

The major requirements of this module are that the CSV file has to be in a mentioned format to process the data in a or else the model won’t generate a proper output.

**5.1.2 Reports**

This section will display the output generated for the given modules.

The data is translated in a readable, more often in the form of graphs, videos, images, plain text, etc.

It is then stored for future use and when data is properly stored, it can be easily accessed by different users whenever needed.

**5.1.3 Feedback**

The user can contact the admin through the feedback module. This module is not only used to collect the feedback but also to solve queries. The user can send his or her query through this module which will be answered back via the mail. The mail will be sent to the registered email id of the user.

**5.2 Software Tools Used**

The website was built using HTML, CSS, and JavaScript to create the front-end webpages. Bootstrap was used make the website look more dynamic. Python and Data Analytics will be used to process the data, and various packages will be used to generate the output for the given CSV files. Django or Flask framework will be used as the bridge between HTML and Python.

Sublime, Notepad++ and Visual studio was used as the front-end code editor and Microsoft Excel, Anaconda, spyder will be used for processing of data.

**5.2.1 HTML**

Hyper Text Mark-up Language commonly abbreviated as HTML is the standard markup language used to create web pages. Along with CSS and JavaScript, HTML is a cornerstone technology used to create web pages as well as user interfaces for mobile and web applications. Web browser can read HTML files and render them into visible audible web pages. HTML describes the structure of a website semantically Cascading Style Sheet (CSS) is a style sheet language used for describing the presentation of a document written in mark-up language. The language can apply to any XML document including plain XML, SVG and XUL, and is applicable rendering in speech or on the other media. Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interface for web applications, and user interface for many mobile applications.

**5.2.2 JavaScript**

JavaScript is a dynamic computer programming language. It is light weight and most commonly used part of web pages; whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

**5.2.3 Bootstrap**

Bootstrap is a free and open-source front-end framework for designing websites and web applications. It contains HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. Unlike many earlier web frameworks, it concerns itself with front-end development only.

**5.2.4 CSS**

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript. CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

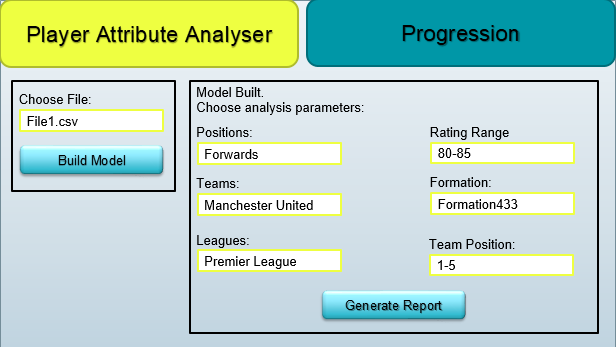
**5.2.5 Flask**

Flask is considered more Pythonic than the Django web framework because in common situations the equivalent Flask web application is more explicit.

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

**5.3 User Interface**

**5.3.1 Draft Design**

****

**Figure 2: UI Template**

**5.3.2 User Interfaces**

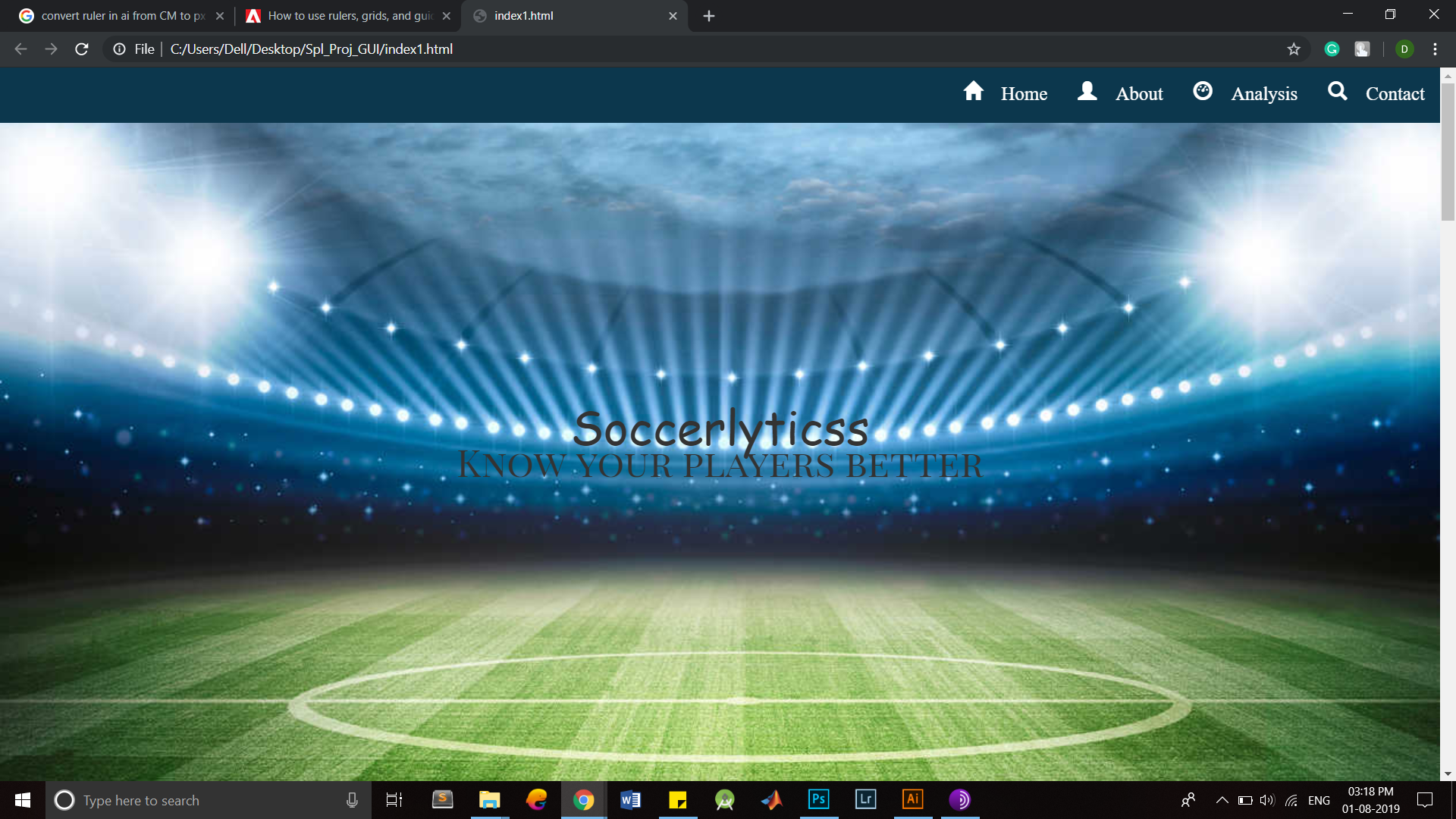


Figure 3: UI Template

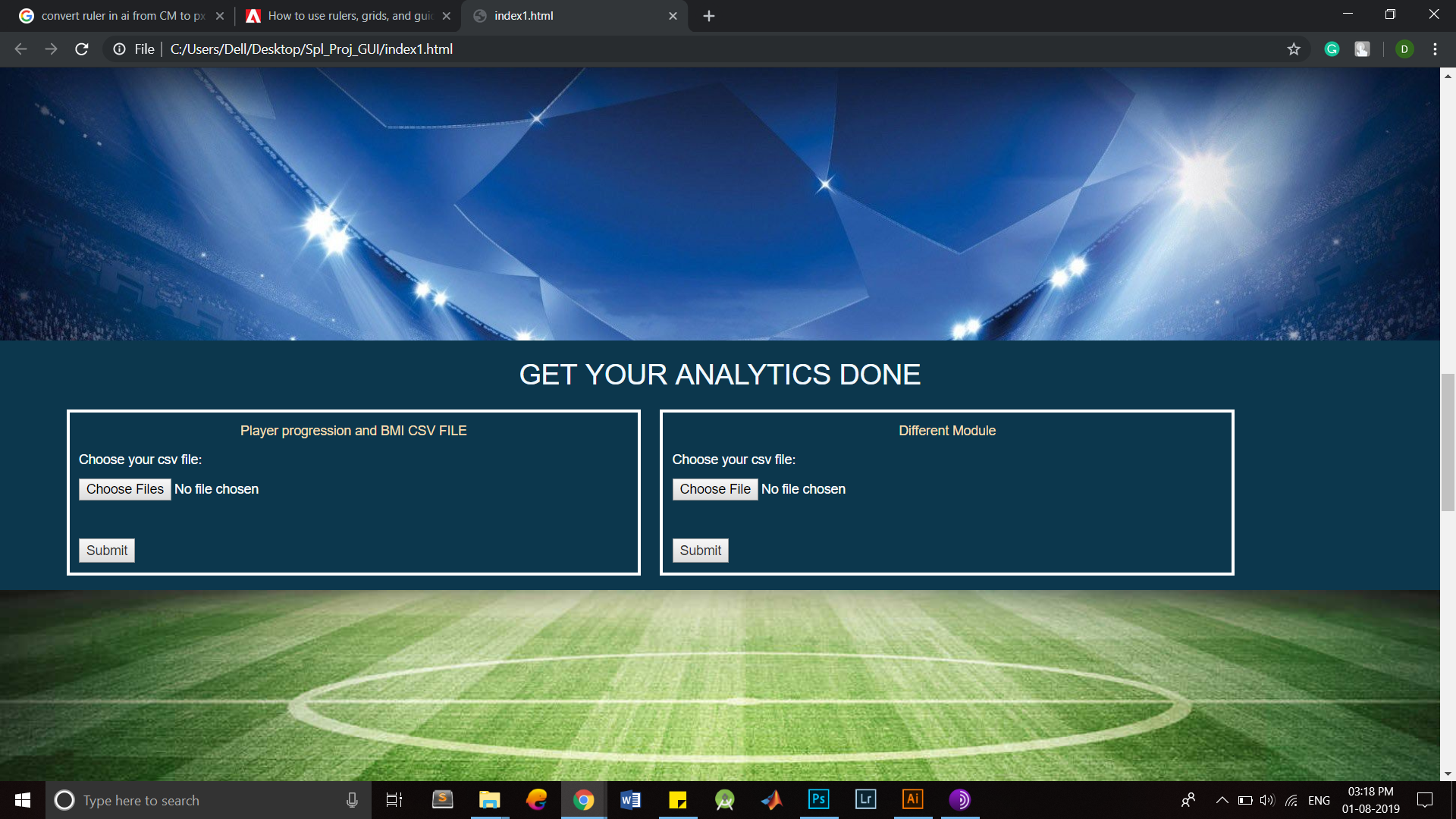


Figure 4: UI Template

**6. IMPLEMENTATION**

**6.1 Player Positions**

A football player can play in one of multiple positions for a team at a point of time. These positions can be grouped into generalized roles. These positions are listed below:

|  |  |  |
| --- | --- | --- |
| Role | Position | Description |
| Goalkeeper | GK | Goalkeeper. |
| Defender | RCB  CB  LCB | Centre-back. Can be left-sided or right-sided. All three if effectively similar. |
|  | LWB RWB | Wing-back. Can be left-sided or right-sided. Both effectively similar. |
|  | LB  RB | Full-back. Can be left-sided or right-sided. Both effectively similar. |
| Midfielder | RDM  CDM  LDM | Centre Defensive Midfield. Can be left-sided or right-sided. All three if effectively similar. |
|  | RCM  CM  LCM | Centre Midfield. Can be left-sided or right-sided. All three if effectively similar. |
|  | RAM  CAM  LAM | Centre Attacking Midfield. Can be left-sided or right-sided. All three if effectively similar. |
|  | LM  RM | Midfield Winger. Can be left-sided or right-sided. Both effectively similar. |
| Attacker | LW  RW | Attacking Winger. Can be left-sided or right-sided. Both effectively similar. |
|  | LF  CF  RF | Centre Forward. Can be left-sided or right-sided. All three if effectively similar. |
|  | LS  ST  RS | Striker. Can be left-sided or right-sided. All three if effectively similar. |

The following figure illustrates the player positions:

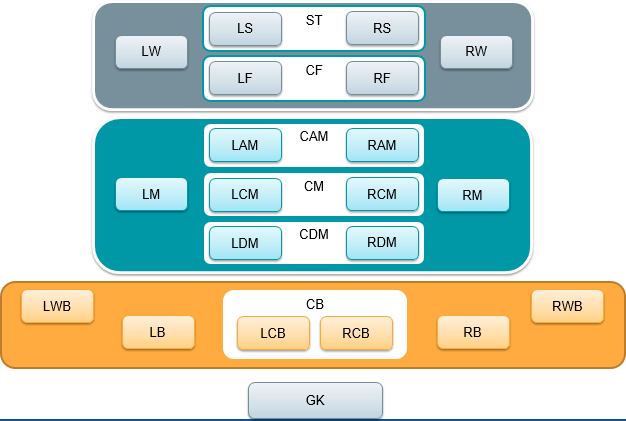
****

Figure 5: Football Positions

**6.2 Pre-processing**

Data fields which are not important have been discarded.

players.drop(['Unnamed: 0','ID','Name','Age','Photo','Nationality','Flag','Potential','Club','Club Logo','Value','Wage','Special',

'Preferred Foot', 'International Reputation','Weak Foot',

'Skill Moves','Work Rate','Body Type','Real Face','Jersey Number', 'Joined', 'Loaned From', 'Contract Valid Until',

'Height', 'Weight', 'LS', 'ST', 'RS', 'LW', 'LF', 'CF', 'RF', 'RW',

'LAM', 'CAM', 'RAM', 'LM', 'LCM', 'CM', 'RCM', 'RM', 'LWB', 'LDM',

'CDM', 'RDM', 'RWB', 'LB', 'LCB', 'CB', 'RCB', 'RB', 'Release Clause', 'Overall'],axis=1,inplace=True)

players.head()

The data required for performing the predictive analysis is retained. These are the player attributes used for prediction:

players.columns

Index(['Position', 'Crossing', 'Finishing', 'HeadingAccuracy', 'ShortPassing',

'Volleys', 'Dribbling', 'Curve', 'FKAccuracy', 'LongPassing',

'BallControl', 'Acceleration', 'SprintSpeed', 'Agility', 'Reactions',

'Balance', 'ShotPower', 'Jumping', 'Stamina', 'Strength', 'LongShots',

'Aggression', 'Interceptions', 'Positioning', 'Vision', 'Penalties',

'Composure', 'Marking', 'StandingTackle', 'SlidingTackle', 'GKDiving',

'GKHandling', 'GKKicking', 'GKPositioning', 'GKReflexes'],

dtype='object')

**6.3 Player Position Analyser**

**6.3.1 Pre-processing for Postion Analyser**

Unimportant attributes are first removed. Position analyser is done on groups of positions based on how closely they relate to each other. The groups are:

* Strikers (ST, LS, RS)
* Forwards (CF, LF, RF)
* Attacking Wingers (RW, LW)
* Attacking Midfielders (CAM, LAM, RAM)
* Midfield Wingers (LM, RM)
* Centre Midfielders (CM, RCM, LCM)
* Defensive Midfielders (CDM, RDM, LDM)
* Wingbacks (LWB, RWB)
* Fullbacks (LB, RB)
* Centre Backs (RCB, LCB, CB)
* Goalkeepers (GK)

**6.3.2 Analysis**

Correlation metrics are used to perform this analysis. The module correlates player attribute statistics for each position, and these correlation values are sorted in descending order. The higher placed player attributes are more pertinent for the position under consideration. Since all player attributes positively contribute to the overall rating of the player, all correlation values are observed to be positive. Therefore, correlation values limited in the range of 0 to 1 have to be considered when performing comparative analysis.

Correlation matrices are generated for each position grouping. The following is an example (strikers):

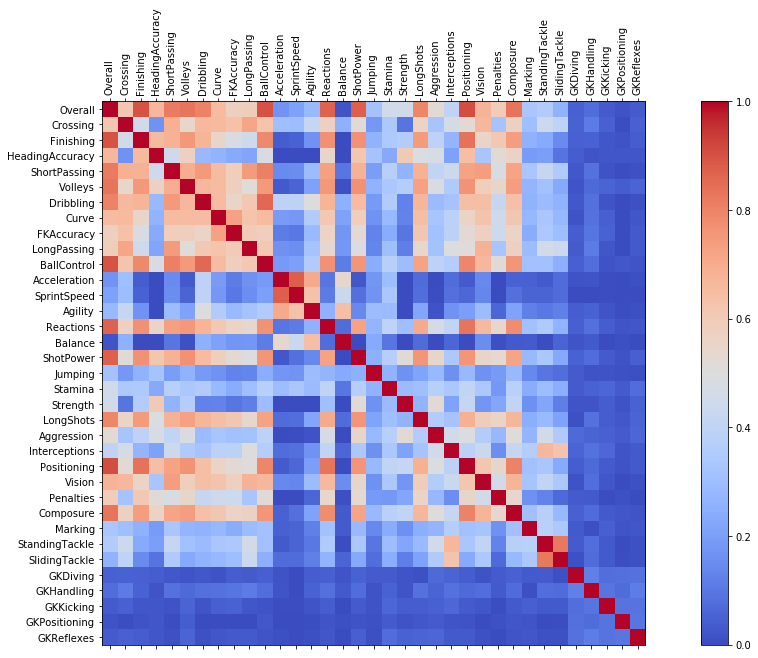


Figure 6: Correlation Matrix

We then list out the most important player attributes for the particular position. The following is an example (strikers):

|  |  |
| --- | --- |
| Attribute | Correlation Value |
| Overall | 1.000000 |
| Positioning | 0.909207 |
| Finishing | 0.902016 |
| BallControl | 0.901148 |
| ShotPower | 0.874362 |
| Reactions | 0.869161 |
| Composure | 0.831669 |
| Volleys | 0.830210 |
| ShortPassing | 0.820211 |
| Dribbling | 0.804175 |
| LongShots | 0.794537 |
| Vision | 0.677674 |
| HeadingAccuracy | 0.664870 |
| Curve | 0.654706 |
| Crossing | 0.614259 |
| Penalties | 0.592026 |
| LongPassing | 0.586801 |
| FKAccuracy | 0.577493 |
| Aggression | 0.521356 |
| Strength | 0.459841 |
| Stamina | 0.458522 |
| Interceptions | 0.398569 |
| StandingTackle | 0.358137 |
| Marking | 0.333553 |
| Jumping | 0.317552 |
| Agility | 0.283607 |
| SlidingTackle | 0.263553 |
| SprintSpeed | 0.209845 |
| Acceleration | 0.171982 |
| GKHandling | 0.076662 |
| GKDiving | 0.047834 |
| GKReflexes | 0.038505 |
| GKKicking | 0.037917 |
| Balance | 0.021691 |
| GKPositioning | 0.018420 |

**6.3.3 Some Prime Attribute Inferences**

Here are some prime attribute inferences we can obtain from the position analysis:

|  |  |  |
| --- | --- | --- |
| Attribute | Positions under Consideration | Pertinent Correlation Value |
| Crossing | WB, FB, WM, CAM | 0.78 –0.88 |
| Finishing | ST | 0.9 |
| Heading Accuracy | CB, ST | 0.8 and 0.6 respectively |
| Short Passing | CF, CM | > 0.9 |
| Volleys | ST, CF, CAM | 0.74 –0.83 |
| Dribbling | CF, AW | > 0.9 |
| Curve | CF | 0.78 |
| Free kick Accuracy | CF, CAM, CM | 0.6 –0.75 |
| Long Passing | CM, CDM | >0.8 |
| Ball Control | ST, CF, AW, CAM, WM, CM | >0.8 |
| Acceleration | WM, AW | 0.4-0.5 |
| Sprint Speed | WM, AW | 0.4-0.5 |
| Agility | AW | 0.72 |
| Reactions | All Positions | >0.8 |
| Balance | CF, AW | 0.25-0.4 |
| Shot Power | ST, CF, CAM | 0.87, 0.75, and 0.75 |
| Jumping | GK | 0.42 |
| Stamina | FB, WB | 0.6-0.7 |
| Strength | ST, CB | 0.45-0.55 |
| Long Shots | CAM, CF | >0.8 |
| Aggression | WB, FB, CB | 0.6-0.75 |
| Interceptions | CB | >0.9 |
| Positioning | ST, CF | >0.9 |
| Vision | CM, WM, CAM, AW, CF | >0.8 |
| Penalties | CF, CAM | 0.6-0.7 |
| Standing Tackle | CB | >0.9 |
| Sliding Tackle | CB, FB, WB | >0.8 |
| GK Diving | GK | 0.93 |
| GK Handling | GK | 0.91 |
| GK Kicking | GK | 0.77 |
| GK Positioning | GK | 0.93 |
| GK Reflexes | GK | 0.94 |

Such an analysis helps decision-makers and players decide the physical and skill-based attributes to focus on based on position of interest, or team requirements.

**6.4 Player Position Predictor**

**6.4.1 Decision Tree Classifier**

Decision tree algorithm falls under the category of supervised learning. They can be used to solve both regression and classification problems. Decision tree uses the tree representation to solve the problem in which each leaf node corresponds to a class label and attributes are represented on the internal node of the tree. Is a flowchart-like tree structure where an internal node represents feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome. The topmost node in a decision tree is known as the root node.



Figure 7: Decision Tree Classifier

Decision Tree is a white box type of ML algorithm. It shares internal decision-making logic, which is not available in the black box type of algorithms such as Neural Network. Its training time is faster compared to the neural network algorithm. The time complexity of decision trees is a function of the number of records and number of attributes in the given data.

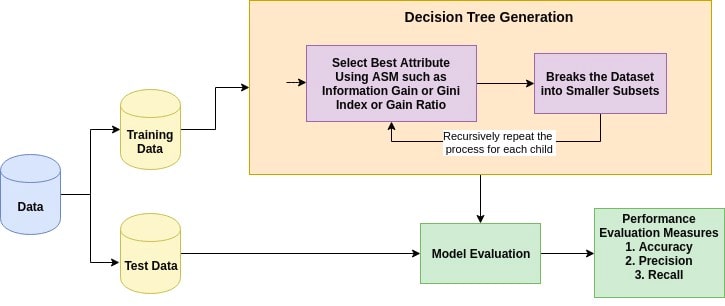


Figure 8: Decision Tree Generation

The dataset was split into a training and test set. The training set is used to obtain the Decision Tree rules, which are then tested for accuracy with the test set.

Decision Tree Classifier model is used with the training data set, and test the accuracy using the test dataset.

We obtain a Decision Tree Classifier model accuracy of 40% which is abysmally poor. K-Fold cross validation was attempted, with 10 iterations. The justification is the efficacy of K-Fold cross validation for relatively small datasets.

Accuracy using K-Fold Cross validation did not increase beyond 40%. Using domain knowledge, it can be hypothesised that the lack of accuracy is a result of the tendency for football players to be suited to multiple positions for the same set of attributes. Due to this, attempting to classify a player into a single position leads to highly inaccurate results.

An attempted solution to this problem is to group player positions into their respective roles which are Attacker, Midfielder, Defender, and Goalkeeper. This is because, while players may play in multiple positions, they tend to stick to these roles. The following diagram is an illustration of this process:

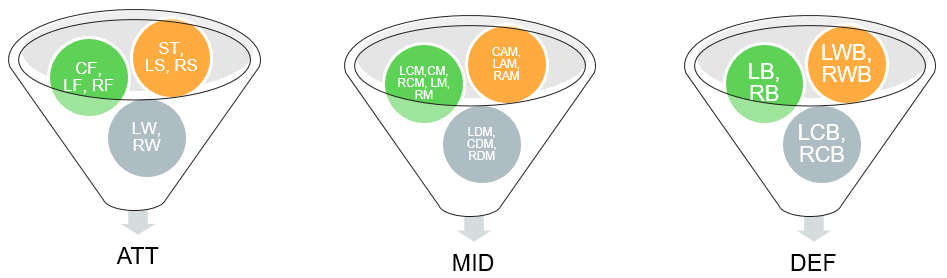


Figure 9: Generalization of Roles

The diagram illustrates the positions which are generalized to ATT, MID, DEF. Goalkeepers (GK) are treated as both a role and a position.

Applying Decision Tree Classification on the updated dataset with the Roles field results in significantly improved accuracy of 81%.

A prediction for the player role can be performed with this model as highlighted in this example:

Akash=[[50,45,70,65,30,45,50,68,72,65,35,55,45,55,50,80,65,70,75,65,35,45,55,65,50,50,70,50,50,0,0,0,0,0]];

genPos=genModel.predict(Akash)

genPos[0]

'MID'

Three additional models are created to identify the specialized positions within the previously identified role. These models exist for Attackers, Midfielders, and Defenders. Note that the data is further pre-processed to generalize the Left and Right sub-positions, where they exist, to the main central position. This is justified as the Left and Right sub-positions are effectively similar.

This brings the total number of prediction models being used to four.

The accuracy for the three specialized position classification models is as follows:

* Attacker Classification Accuracy: 80%
* Midfielder Classification Accuracy: 55%
* Defender Classification Accuracy: 89%

The following flow diagram illustrates the approach taken thus far.

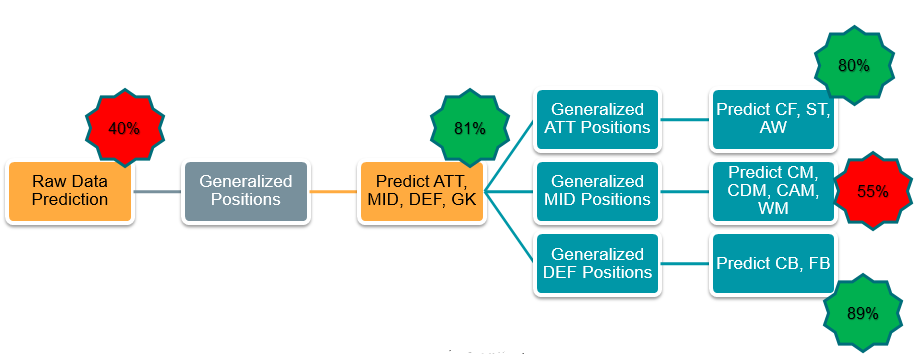


Figure 10: Classification Procedure

The following are examples of predictions carried out using this approach:

Messi=[[84.0, 95.0, 70.0, 90.0, 86.0, 97.0, 93.0, 94.0, 87.0, 96.0,

91.0, 86.0, 91.0, 95.0, 95.0, 85.0, 68.0, 72.0, 59.0, 94.0, 48.0,

22.0, 94.0, 94.0, 75.0, 96.0, 33.0, 28.0, 26.0, 6.0, 11.0, 15.0,

14.0, 8.0]]

Recommended Role: ATT

Recommended Position: CF

Neymar=[[79.0, 87.0, 62.0, 84.0, 84.0, 96.0, 88.0, 87.0, 78.0, 95.0,

94.0, 90.0, 96.0, 94.0, 84.0, 80.0, 61.0, 81.0, 49.0, 82.0, 56.0,

36.0, 89.0, 87.0, 81.0, 94.0, 27.0, 24.0, 33.0, 9.0, 9.0, 15.0,

15.0, 11.0]]

Recommended Role: ATT

Recommended Position: AW

While the accuracy obtained for the Attacker (ATT) and Defender (DEF) prediction algorithms is high (80% and 89%), Midfield prediction accuracy is still low (55%) for the same reason observed before. The reason being, within the subset of midfielders, the tendency to occupy varying positions is higher for a similar set of player attributes.

As a result of this, it would not be possible to further increase prediction accuracy by solely relying on multiple class classification. The approach of multi-label classification, which would return a probabilistic distribution of best suited positions for a player’s attributes, would help make a more accurate prediction. The probabilistic algorithm to be used here is Random Forest Classifier.

**6.4.2 Random Forest Classification**

Random forest classifier creates a set of decision trees from randomly selected subset of training set. It then aggregates the votes from different decision trees to decide the final class of the test object. • Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model’s prediction.



Figure 11: Random Forest Classifier

A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.

Before proceeding to implement the Random Forest Classification for probabilistic prediction of player position, it is necessary to generalize the player sub-positions into their effective positions, as mentioned before. The converted generalizations are as follows:

|  |  |
| --- | --- |
| **Sub Position** | **Position** |
| RCM | CM |
| LCM |  |
| LDM | CDM |
| RDM |  |
| LAM | CAM |
| RAM |  |
| LM | WM |
| RM |  |
| LS | ST |
| RS |  |
| LF | CF |
| RF |  |
| RCB | CB |
| LCB |  |
| LB | WB |
| RB |  |
| LWB |  |
| RWB |  |
| LW | AW |
| RW |  |

Applying Random Forest Classifier on the dataset for all specialized positions provides a probabilistic prediction for all positions, with an overall accuracy of 74%. This is significantly higher than the 40% the direct implementation of Decision Tree Classifier provided.

An example output of applying Random Forest Classifier is as shown below:

Tenzin=[[80,70,50,85,65,60,60,70,70,75,70,70,75,70,65,75,65,60,60,65,70,75,70,70,85,70,70,60,60,0,0,0,0,0]

{'AW': 0.0,

'CAM': 0.0,

'CB': 0.0,

'CDM': 0.6,

'CF': 0.0,

'CM': 0.4,

'GK': 0.0,

'ST': 0.0,

'WB': 0.0,

'WM': 0.0}

In this example, the player under consideration can play in either of the two positions CDM or CM. The probabilistic distribution implies the player is better suited for the CDM role with 60% of Decision Trees providing this output, with CM position being outputted by remaining 40% of trees.

On the other hand, using the previously created, and highly accurate, Decision Tree Models for the Attacker (ATT) and Defender (DEF) roles, and using Random Forest Classifier to solely predict Midfielders, Midfield specialization prediction accuracy is increased from 55% to 68%. The output of this approach, applied to a predicted MID role is as follows:

DeBruyne=[[93.0, 82.0, 55.0, 92.0, 82.0, 86.0, 85.0, 83.0, 91.0, 91.0,

78.0, 76.0, 79.0, 91.0, 77.0, 91.0, 63.0, 90.0, 75.0, 91.0, 76.0,

61.0, 87.0, 94.0, 79.0, 88.0, 68.0, 58.0, 51.0, 15.0, 13.0, 5.0,

10.0, 13.0]]

Recommended Role: MID

Recommended Position: {'CAM': 0.18, 'CDM': 0.08, 'CM': 0.66, 'WM': 0.08}

In this example, the professional footballer Kevin DeBruyne of Manchester City and Belgium is predicted to be suited for the CM role with 66% probability, followed by CAM with 18%. This is accurate to what is observed in the real world, thereby proving the model is reliable for use with new data. The following updated flow-diagram illustrates this new approach:

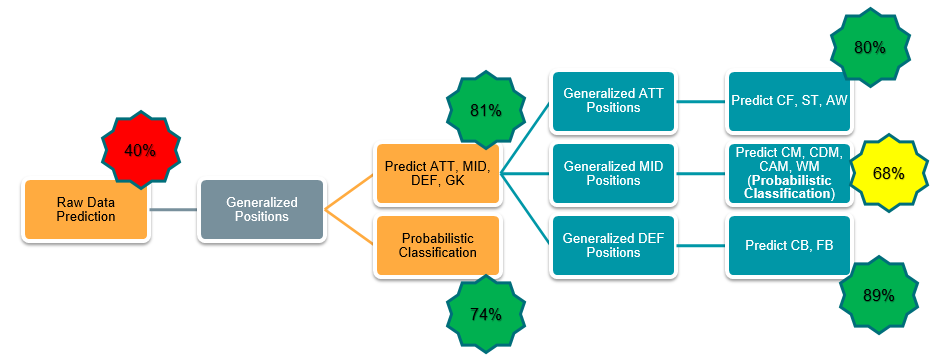


Figure 12: Updated Probabilistic Classification

**6.5 BMI Analyser**

Body Mass Index refers to the measure to body fat in the human body based on the height and weight metrics of the individual. BMI levels are an indicator of the fitness of the individual and are a valuable metric to determine levels of athletic performance.

The following are the main BMI categories:

|  |  |
| --- | --- |
| **BMI Level** | **Classification** |
| <18.5 | Underweight |
| 18.5 to 25 | Normal |
| 25-30 | Overweight |
| >30 | Obese |

For pre-processing the data, the formats of height and weight were converted from feet/inches and pounds to meters and kilograms respectively.

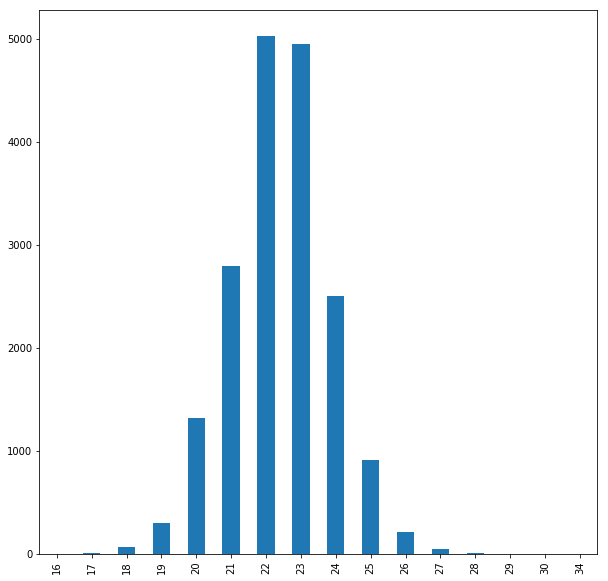
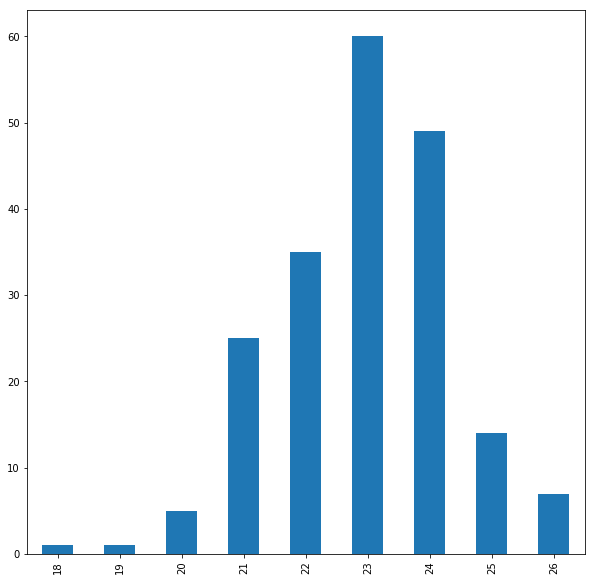
Analysis was carried out using multiple approaches, with mean BMI being the core metric under consideration.

Average BMI of all players (18147 players) is found to be 22.39

When trying to ascertain variations in BMI, multiple approaches were taken, each showing mean to be a weak differentiating factor:

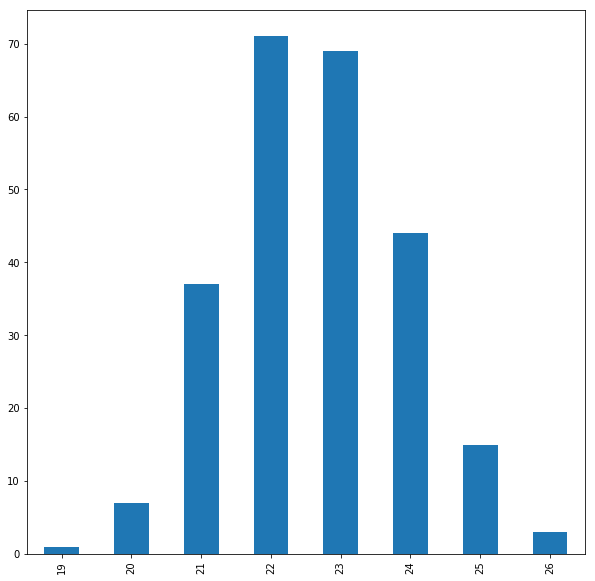
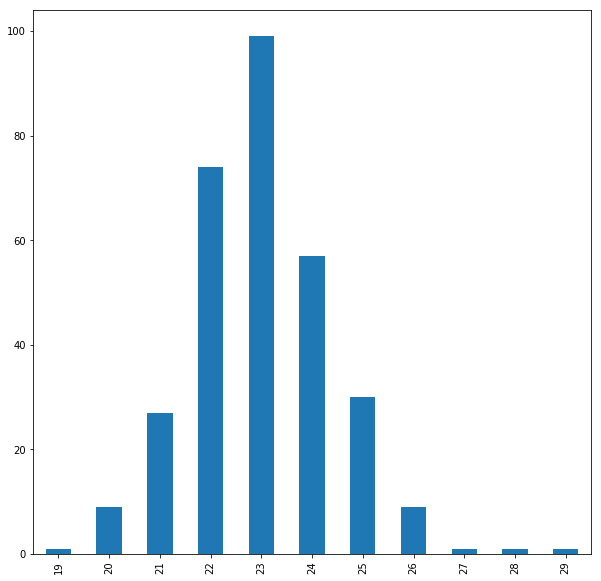
* Mean BMI values Per Position vary minimally
* Mean BMI values per player overall range vary minimally
* Mean BMI values for the above two metrics for only elite players with high ratings also vary minimally

The updated approach is to use bar charts to plot the distribution of BMI values to determine optimal BMI metrics for each position:



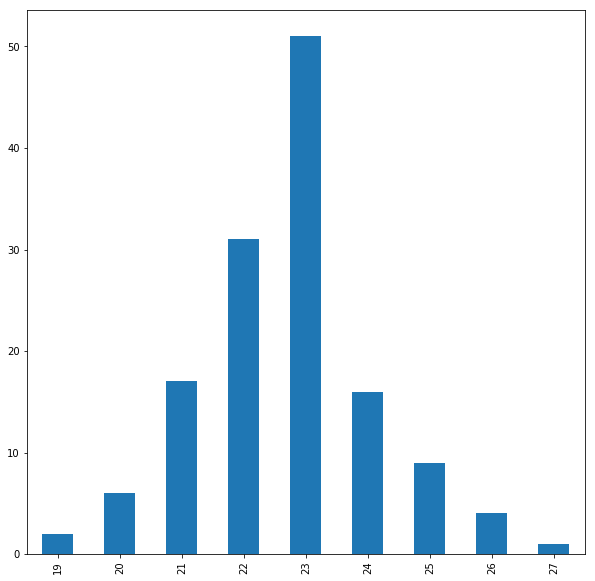
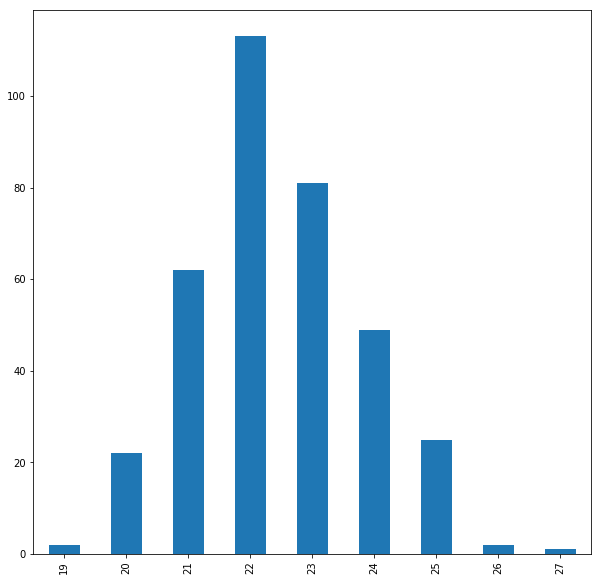
OVR

GK



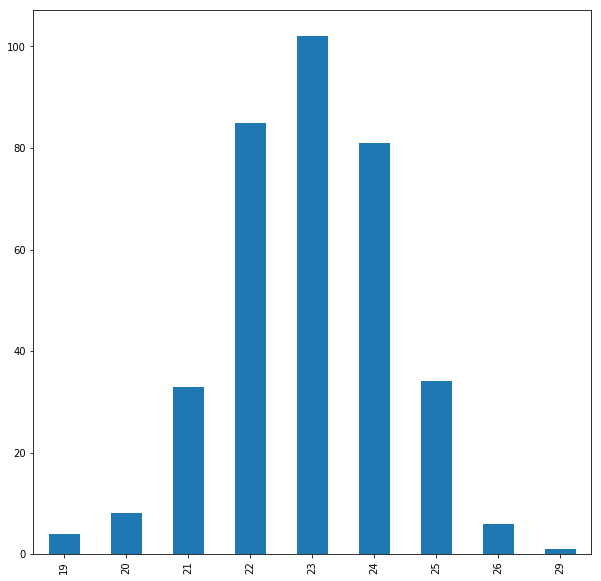
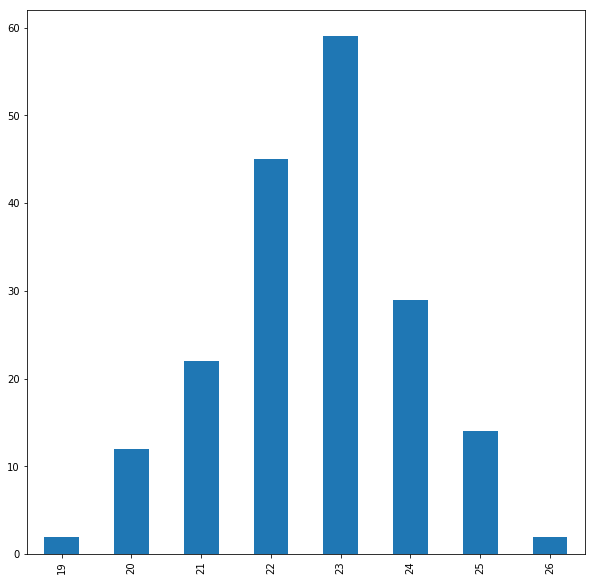
WB

ST



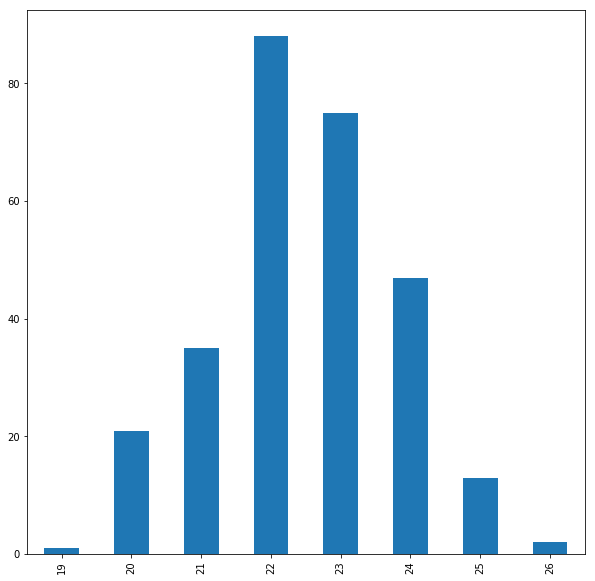
CAM

AW



CB

CDM



CM

Figure 13:Per-Position BMI Distributions Using Bar Charts

These distributions help determine the most effective BMI values for particular positions, even if the respective mean values indicate some other effective BMI value due to high variance distributions.

**6.6 Code**

**6.6.1 Position Analyser**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

players=pd.read\_csv("playerAttributes.csv")

players.drop(['Unnamed:0','ID','Name','Age','Photo','Nationality','Flag','Potential','Club','Club Logo','Value','Wage','Special','Preferred Foot', 'International Reputation','Weak Foot','Skill Moves','Work Rate','Body Type','Real Face','Jersey Number', 'Joined', 'Loaned From', 'Contract Valid Until','Height', 'Weight', 'LS', 'ST', 'RS', 'LW', 'LF', 'CF', 'RF', 'RW','LAM', 'CAM', 'RAM', 'LM', 'LCM', 'CM', 'RCM', 'RM', 'LWB', 'LDM',

'CDM', 'RDM', 'RWB', 'LB', 'LCB', 'CB', 'RCB', 'RB', 'Release Clause'],axis=1,inplace=True)

strikers=players.loc[(players['Position']=='ST')|(players['Position']=='LS')|(players['Position']=='RS')].copy()

strikers.corr().iloc[0].sort\_values(ascending=False)

**Output:**

**Overall 1.000000**

**Positioning 0.909207**

**Finishing 0.902016**

**BallControl 0.901148**

**ShotPower 0.874362**

**Reactions 0.869161**

**Composure 0.831669**

**Volleys 0.830210**

**ShortPassing 0.820211**

**Dribbling 0.804175**

**LongShots 0.794537**

**Vision 0.677674**

**HeadingAccuracy 0.664870**

**Curve 0.654706**

**Crossing 0.614259**

**Penalties 0.592026**

**LongPassing 0.586801**

**FKAccuracy 0.577493**

**Aggression 0.521356**

**Strength 0.459841**

**Stamina 0.458522**

**Interceptions 0.398569**

**StandingTackle 0.358137**

**Marking 0.333553**

**Jumping 0.317552**

**Agility 0.283607**

**SlidingTackle 0.263553**

**SprintSpeed 0.209845**

**Acceleration 0.171982**

**GKHandling 0.076662**

**GKDiving 0.047834**

**GKReflexes 0.038505**

**GKKicking 0.037917**

**Balance 0.021691**

**GKPositioning 0.018420**

**Name: Overall, dtype: float64**

corr=strikers.corr()

fig=plt.figure(figsize=(20,10))

ax=fig.add\_subplot(111) #1x1 grid, 1st subplot

cax=ax.matshow(corr,cmap='coolwarm',vmin=0,vmax=1)

fig.colorbar(cax)

ticks=np.arange(0,len(strikers.columns),1)

ax.set\_xticks(ticks)

plt.xticks(rotation=90)

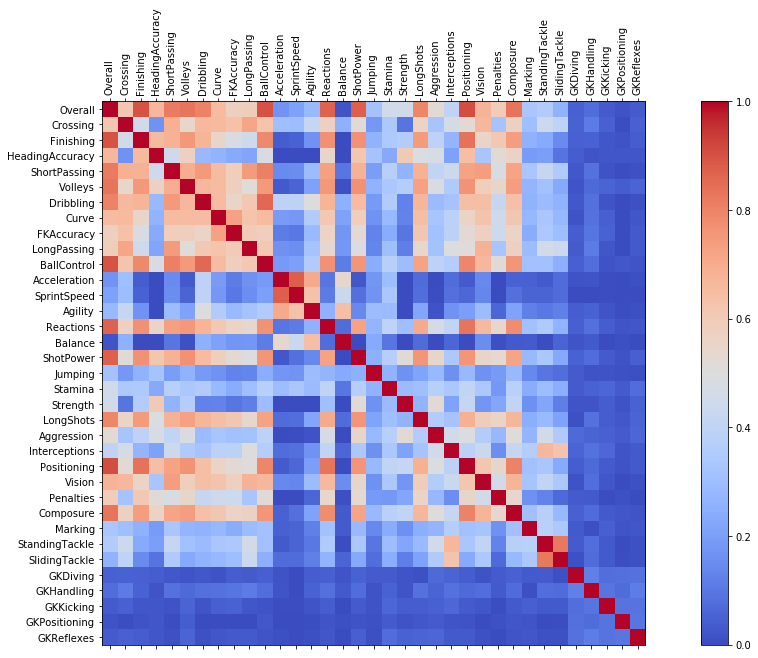
ax.set\_yticks(ticks)

ax.set\_xticklabels(strikers.columns)

ax.set\_yticklabels(strikers.columns)

plt.show()

Output:



forwards=players.loc[(players['Position']=='CF')|(players['Position']=='RF')|(players['Position']=='LF')].copy()

forwards.corr().iloc[0].sort\_values(ascending=False)

attWingers=players.loc[(players['Position']=='RW')|(players['Position']=='LW')].copy()

attWingers.corr().iloc[0].sort\_values(ascending=False)

attmid=players.loc[(players['Position']=='CAM')|(players['Position']=='RAM')|(players['Position']=='LAM')].copy()

attmid.corr().iloc[0].sort\_values(ascending=False)

midWing=players.loc[(players['Position']=='LM')|(players['Position']=='RM')].copy()

midWing.corr().iloc[0].sort\_values(ascending=False)

cenMid=players.loc[(players['Position']=='CM')|(players['Position']=='RCM')|(players['Position']=='LCM')].copy()

cenMid.corr().iloc[0].sort\_values(ascending=False)

defMid=players.loc[(players['Position']=='CDM')|(players['Position']=='RDM')|(players['Position']=='LDM')].copy()

defMid.corr().iloc[0].sort\_values(ascending=False)

wingback=players.loc[(players['Position']=='LWB')|(players['Position']=='RWB')].copy()

wingback.corr().iloc[0].sort\_values(ascending=False)

fullback=players.loc[(players['Position']=='LB')|(players['Position']=='RB')].copy()

fullback.corr().iloc[0].sort\_values(ascending=False)

cback=players.loc[(players['Position']=='RCB')|(players['Position']=='LCB')|(players['Position']=='CB')].copy()

cback.corr().iloc[0].sort\_values(ascending=False)

keeper=players.loc[(players['Position']=='GK')].copy()

keeper.corr().iloc[0].sort\_values(ascending=False)

**6.6.2 Position Predictor Using Decision Tree Classifier**

feature\_names=['Crossing', 'Finishing', 'HeadingAccuracy', 'ShortPassing','Volleys', 'Dribbling', 'Curve', 'FKAccuracy', 'LongPassing','BallControl', 'Acceleration', 'SprintSpeed', 'Agility', 'Reactions','Balance', 'ShotPower', 'Jumping', 'Stamina', 'Strength', 'LongShots','Aggression', 'Interceptions', 'Positioning', 'Vision', 'Penalties','Composure', 'Marking', 'StandingTackle', 'SlidingTackle', 'GKDiving','GKHandling', 'GKKicking', 'GKPositioning', 'GKReflexes']

X=players[feature\_names]

y=players['Position']

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,random\_state=1)

from sklearn.tree import DecisionTreeClassifier

clf=DecisionTreeClassifier().fit(X\_train,y\_train) #Creating the Decision Tree Model

print('Accuracy of Decision Tree classifier on training set:{:.2f}'.format(clf.score(X\_train,y\_train)))

print('Accuracy of Decision Tree classifier on test set:{:.2f}'.format(clf.score(X\_test,y\_test)))

**Output:**

**Accuracy of Decision Tree classifier on training set:1.00**

**Accuracy of Decision Tree classifier on test set:0.40**

from sklearn.model\_selection import cross\_val\_score

clf=DecisionTreeClassifier().fit(X\_train,y\_train)

scores = cross\_val\_score(clf, X, y, cv=10)

scores

**Output:**

**array([0.27501367, 0.35471491, 0.36133992, 0.37768004, 0.37389868,**

**0.38024283, 0.37217007, 0.4079602 , 0.41375485, 0.40765391])**

gen\_players['Role']='ANY'

for index,row in players.iterrows():

if (row['Position'] == 'ST') | \

(row['Position'] == 'LS') | \

(row['Position'] == 'RS') | \

(row['Position'] == 'CF') | \

(row['Position'] == 'LF') | \

(row['Position'] == 'RF') | \

(row['Position'] == 'LW') | \

(row['Position'] == 'RW') :

gen\_players.at[index,'Role'] = 'ATT'

elif (row['Position'] == 'CAM') | \

(row['Position'] == 'RAM') | \

(row['Position'] == 'LAM') | \

(row['Position'] == 'RM') | \

(row['Position'] == 'LM') | \

(row['Position'] == 'CM') | \

(row['Position'] == 'LCM') | \

(row['Position'] == 'RCM') | \

(row['Position'] == 'CDM') | \

(row['Position'] == 'LDM') | \

(row['Position'] == 'RDM') :

gen\_players.at[index,'Role'] = 'MID'

elif (row['Position'] == 'LWB') | \

(row['Position'] == 'RWB') | \

(row['Position'] == 'LB') | \

(row['Position'] == 'RB') | \

(row['Position'] == 'CB') | \

(row['Position'] == 'LCB') | \

(row['Position'] == 'RCB') :

gen\_players.at[index,'Role'] = 'DEF'

else:

gen\_players.at[index,'Role'] = 'GK'

gen\_players['Role'].unique()

**Output:**

**array(['ATT', 'GK', 'MID', 'DEF'], dtype=object)**

feature\_names=['Crossing', 'Finishing', 'HeadingAccuracy', 'ShortPassing','Volleys', 'Dribbling', 'Curve', 'FKAccuracy', 'LongPassing','BallControl', 'Acceleration', 'SprintSpeed', 'Agility', 'Reactions','Balance', 'ShotPower', 'Jumping', 'Stamina', 'Strength', 'LongShots','Aggression', 'Interceptions', 'Positioning', 'Vision', 'Penalties','Composure', 'Marking', 'StandingTackle', 'SlidingTackle', 'GKDiving','GKHandling', 'GKKicking', 'GKPositioning', 'GKReflexes']

X=gen\_players[feature\_names]

y=gen\_players['Role']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,random\_state=1)

genModel=DecisionTreeClassifier().fit(X\_train,y\_train)

print('Accuracy of Decision Tree classifier on training set:{:.2f}'.format(genModel.score(X\_train,y\_train)))

print('Accuracy of Decision Tree classifier on test set:{:.2f}'.format(genModel.score(X\_test,y\_test)))

**Output:**

**Accuracy of Decision Tree classifier on training set:1.00**

**Accuracy of Decision Tree classifier on test set:0.81**

Akash=[[50,45,70,65,30,45,50,68,72,65,35,55,45,55,50,80,65,70,75,65,35,45,55,65,50,50,70,50,50,0,0,0,0,0]];

genPos=genModel.predict(Akash)

genPos[0]

**Output:**

**'MID'**

midfielders=gen\_players.loc[(gen\_players['Role']=='MID')].copy()

for index,row in midfielders.iterrows():

if (row['Position'] == 'RCM') | \

(row['Position'] == 'LCM') :

midfielders.at[index,'Position'] = 'CM'

elif (row['Position'] == 'LDM') | \

(row['Position'] == 'RDM') :

midfielders.at[index,'Position'] = 'CDM'

elif (row['Position'] == 'LAM') | \

(row['Position'] == 'RAM') :

midfielders.at[index,'Position'] = 'CAM'

elif (row['Position'] == 'LM') | \

(row['Position'] == 'RM') :

midfielders.at[index,'Position'] = 'WM'

midfielders['Position'].unique()

**Output:**

**array(['CM', 'CDM', 'CAM', 'WM'], dtype=object)**

X=midfielders[feature\_names]

y=midfielders['Position']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,random\_state=1)

midfieldModel=DecisionTreeClassifier().fit(X\_train,y\_train)

print('Accuracy of Decision Tree classifier on training set:{:.2f}'.format(midfieldModel.score(X\_train,y\_train)))

print('Accuracy of Decision Tree classifier on test set:{:.2f}'.format(midfieldModel.score(X\_test,y\_test)))

**Output:**

**Accuracy of Decision Tree classifier on training set:1.00**

**Accuracy of Decision Tree classifier on test set:0.55**

attackers=gen\_players.loc[(gen\_players['Role']=='ATT')].copy()

for index,row in attackers.iterrows():

if (row['Position'] == 'RF') | \

(row['Position'] == 'LF') :

attackers.at[index,'Position'] = 'CF'

elif (row['Position'] == 'RS') | \

(row['Position'] == 'LS') :

attackers.at[index,'Position'] = 'ST'

elif (row['Position'] == 'LW') | \

(row['Position'] == 'RW') :

attackers.at[index,'Position'] = 'AW'

attackers['Position'].unique()

**Output:**

**array(['CF', 'ST', 'AW'], dtype=object)**

X=attackers[feature\_names]

y=attackers['Position']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,random\_state=1)

attackerModel=DecisionTreeClassifier().fit(X\_train,y\_train)

print('Accuracy of Decision Tree classifier on training set:{:.2f}'.format(attackerModel.score(X\_train,y\_train)))

print('Accuracy of Decision Tree classifier on test set:{:.2f}'.format(attackerModel.score(X\_test,y\_test)))

**Output:**

**Accuracy of Decision Tree classifier on training set:1.00**

**Accuracy of Decision Tree classifier on test set:0.79**

defenders=gen\_players.loc[(gen\_players['Role']=='DEF')].copy()

for index,row in defenders.iterrows():

if (row['Position'] == 'RCB') | \

(row['Position'] == 'LCB') :

defenders.at[index,'Position'] = 'CB'

elif (row['Position'] == 'LB') | \

(row['Position'] == 'LWB') | \

(row['Position'] == 'RWB') | \

(row['Position'] == 'RB') :

defenders.at[index,'Position'] = 'FB'

defenders['Position'].unique()

**Output:**

**array(['CB', 'FB'], dtype=object)**

X=defenders[feature\_names]

y=defenders['Position']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,random\_state=1)

defenderModel=DecisionTreeClassifier().fit(X\_train,y\_train)

print('Accuracy of Decision Tree classifier on training set:{:.2f}'.format(defenderModel.score(X\_train,y\_train)))

print('Accuracy of Decision Tree classifier on test set:{:.2f}'.format(defenderModel.score(X\_test,y\_test)))

**Output:**

**Accuracy of Decision Tree classifier on training set:1.00**

**Accuracy of Decision Tree classifier on test set:0.89**

Messi=[[84.0, 95.0, 70.0, 90.0, 86.0, 97.0, 93.0, 94.0, 87.0, 96.0,

91.0, 86.0, 91.0, 95.0, 95.0, 85.0, 68.0, 72.0, 59.0, 94.0, 48.0,

22.0, 94.0, 94.0, 75.0, 96.0, 33.0, 28.0, 26.0, 6.0, 11.0, 15.0,

14.0, 8.0]]

genPos=genModel.predict(Messi)

if genPos[0]=='ATT':

specPos=attackerModel.predict(Messi)

elif genPos[0]=='MID':

specPos=midfieldModel.predict(Messi)

elif genPos[0]=='DEF':

specPos=defenderModel.predict(Messi)

else:

specPos[0]='GK'

print('Recommended Role: ',genPos[0])

print('Recommended Position: ',specPos[0])

**Output:**

**Recommended Role: ATT**

**Recommended Position: CF**

Neymar=[[79.0, 87.0, 62.0, 84.0, 84.0, 96.0, 88.0, 87.0, 78.0, 95.0,

94.0, 90.0, 96.0, 94.0, 84.0, 80.0, 61.0, 81.0, 49.0, 82.0, 56.0,

36.0, 89.0, 87.0, 81.0, 94.0, 27.0, 24.0, 33.0, 9.0, 9.0, 15.0,

15.0, 11.0]]

genPos=genModel.predict(Neymar)

if genPos[0]=='ATT':

specPos=attackerModel.predict(Neymar)

elif genPos[0]=='MID':

specPos=midfieldModel.predict(Neymar)

elif genPos[0]=='DEF':

specPos=defenderModel.predict(Neymar)

else:

specPos[0]='GK'

print('Recommended Role: ',genPos[0])

print('Recommended Position: ',specPos[0])

**Output:**

**Recommended Role: ATT**

**Recommended Position: AW**

**6.6.3 Position Predictor Using Random Forest Classifier**

for index,row in players.iterrows():

if (row['Position'] == 'RCM') | \

(row['Position'] == 'LCM') :

players.at[index,'Position'] = 'CM'

elif (row['Position'] == 'LDM') | \

(row['Position'] == 'RDM') :

players.at[index,'Position'] = 'CDM'

elif (row['Position'] == 'LAM') | \

(row['Position'] == 'RAM') :

players.at[index,'Position'] = 'CAM'

elif (row['Position'] == 'LM') | \

(row['Position'] == 'RM') :

players.at[index,'Position'] = 'WM'

elif (row['Position'] == 'LS') | \

(row['Position'] == 'RS') :

players.at[index,'Position'] = 'ST'

elif (row['Position'] == 'LF') | \

(row['Position'] == 'RF') :

players.at[index,'Position'] = 'CF'

elif (row['Position'] == 'RCB') | \

(row['Position'] == 'LCB') :

players.at[index,'Position'] = 'CB'

elif (row['Position'] == 'LB') | \

(row['Position'] == 'RB') | \

(row['Position'] == 'RWB') | \

(row['Position'] == 'LWB') :

players.at[index,'Position'] = 'WB'

elif (row['Position'] == 'LW') | \

(row['Position'] == 'RW') :

players.at[index,'Position'] = 'AW'

players['Position'].unique()

**Output:**

**array(['CF', 'ST', 'AW', 'GK', 'CM', 'CB', 'CDM', 'CAM', 'WM', 'WB'],**

**dtype=object)**

feature\_names=['Crossing', 'Finishing', 'HeadingAccuracy', 'ShortPassing','Volleys', 'Dribbling', 'Curve', 'FKAccuracy', 'LongPassing','BallControl', 'Acceleration', 'SprintSpeed', 'Agility', 'Reactions','Balance', 'ShotPower', 'Jumping', 'Stamina', 'Strength', 'LongShots','Aggression', 'Interceptions', 'Positioning', 'Vision', 'Penalties','Composure', 'Marking', 'StandingTackle', 'SlidingTackle', 'GKDiving','GKHandling', 'GKKicking', 'GKPositioning', 'GKReflexes']

X=players[feature\_names]

y=players['Position']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,random\_state=1)

clf = RandomForestClassifier(n\_estimators=100, max\_depth=None, min\_samples\_split=2, random\_state=0)

classifier = clf.fit(X\_train,y\_train)

x=dict(zip(classifier.classes\_,classifier.predict\_proba(Akash)[0]))

x

**Output:**

**{'AW': 0.05,**

**'CAM': 0.09,**

**'CB': 0.15,**

**'CDM': 0.25,**

**'CF': 0.01,**

**'CM': 0.3,**

**'GK': 0.0,**

**'ST': 0.03,**

**'WB': 0.07,**

**'WM': 0.05}**

print('Accuracy of Random Forest classifier on training set:{:.2f}'.format(classifier.score(X\_train,y\_train)))

print('Accuracy of Random Forest classifier on test set:{:.2f}'.format(classifier.score(X\_test,y\_test)))

**Output:**

**Accuracy of Decision Tree classifier on training set:1.00**

**Accuracy of Decision Tree classifier on test set:0.74**

from sklearn.ensemble import RandomForestClassifier

feature\_names=['Crossing', 'Finishing', 'HeadingAccuracy', 'ShortPassing','Volleys', 'Dribbling', 'Curve', 'FKAccuracy', 'LongPassing','BallControl', 'Acceleration', 'SprintSpeed', 'Agility', 'Reactions','Balance', 'ShotPower', 'Jumping', 'Stamina', 'Strength', 'LongShots','Aggression', 'Interceptions', 'Positioning', 'Vision', 'Penalties','Composure', 'Marking', 'StandingTackle', 'SlidingTackle', 'GKDiving',

'GKHandling', 'GKKicking', 'GKPositioning', 'GKReflexes']

X=midfielders[feature\_names]

y=midfielders['Position']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,random\_state=1)

clf = RandomForestClassifier(n\_estimators=50, max\_depth=None, min\_samples\_split=2, random\_state=0)

midfieldModel= clf.fit(X\_train,y\_train)

print('Accuracy of Decision Tree classifier on training set:{:.2f}'.format(midfieldModel.score(X\_train,y\_train)))

print('Accuracy of Decision Tree classifier on test set:{:.2f}'.format(midfieldModel.score(X\_test,y\_test)))

**Output:**

**Accuracy of Decision Tree classifier on training set:1.00**

**Accuracy of Decision Tree classifier on test set:0.68**

DeBruyne=[[93.0, 82.0, 55.0, 92.0, 82.0, 86.0, 85.0, 83.0, 91.0, 91.0,78.0, 76.0, 79.0, 91.0, 77.0, 91.0, 63.0, 90.0, 75.0, 91.0, 76.0,61.0, 87.0, 94.0, 79.0, 88.0, 68.0, 58.0, 51.0, 15.0, 13.0, 5.0,10.0, 13.0]]

genPos=genModel.predict(DeBruyne)

if genPos[0]=='ATT':

specPos=attackerModel.predict(DeBruyne)

print('Recommended Role: ',genPos[0])

print('Recommended Position: ',specPos[0])

elif genPos[0]=='MID':

specPos=dict(zip(midfieldModel.classes\_,midfieldModel.predict\_proba(DeBruyne)[0]))

print('Recommended Role: ',genPos[0])

print('Recommended Position: ',specPos)

elif genPos[0]=='DEF':

specPos=defenderModel.predict(DeBruyne)

print('Recommended Role: ',genPos[0])

print('Recommended Position: ',specPos[0])

else:

specPos[0]='GK'

print('Recommended Role: ',genPos[0])

print('Recommended Position: ',specPos[0])

**Output:**

**Recommended Role: MID**

**Recommended Position: {'CAM': 0.18, 'CDM': 0.08, 'CM': 0.66, 'WM': 0.08}**

**6.6.4 BMI Analyser**

for index,row in players.iterrows():

conv=players.loc[index]['Height']

conv\_search=re.search('(.).(.\*)',conv)

feet=int(conv\_search.group(1))

inches=int(conv\_search.group(2))

players.at[index,'Height']=((feet\*12)+inches)/39.37

for index,row in players.iterrows():

conv=players.loc[index]['Weight']

conv\_search=re.search('(.\*)lbs',conv)

pounds=int(conv\_search.group(1))

players.at[index,'Weight']=pounds/2.205

players['bmi']=0

for index,row in players.iterrows():

players.at[index,'bmi']=players.loc[index]['Weight']/(players.loc[index]['Height']\*players.loc[index]['Height'])

sorted(players['bmi'].unique().tolist())

**Output:**

**[16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 34]**

forChart=playerBMIs['bmi'].value\_counts().sort\_index()

forChart

**Output:**

**16 2**

**17 7**

**18 67**

**19 304**

**20 1324**

**21 2789**

**22 5021**

**23 4942**

**24 2505**

**25 908**

**26 212**

**27 47**

**28 12**

**29 5**

**30 1**

**34 1**

forChart.plot.bar(figsize=(10,10))

playerBMIs['bmi'].describe()

tempPlayers=players[['Overall','bmi']].copy()

tempPlayers.loc[(tempPlayers['Overall']>90)]['bmi'].describe()

tempPlayers.loc[(tempPlayers['Overall']>85) & (tempPlayers['Overall']<=90)]['bmi'].describe()

tempPlayers.loc[(tempPlayers['Overall']>80) & (tempPlayers['Overall']<=85)]['bmi'].describe()

tempPlayers.loc[(tempPlayers['Overall']>75) & (tempPlayers['Overall']<=80)]['bmi'].describe()

tempPlayers.loc[(tempPlayers['Overall']>70) & (tempPlayers['Overall']<=75)]['bmi'].describe()

tempPlayers.loc[(tempPlayers['Overall']<=70)]['bmi'].describe()

tempPlayers.loc[(tempPlayers['Position']=='GK')]['bmi'].describe()

tempPlayers.loc[(tempPlayers['Position']=='ST') | \

(tempPlayers['Position']=='LS') | (tempPlayers['Position']=='RS') | \

(tempPlayers['Position']=='LF') | (tempPlayers['Position']=='RF') | \

(tempPlayers['Position']=='CF')]['bmi'].describe()

tempPlayers.loc[(tempPlayers['Position']=='CB') | \

(tempPlayers['Position']=='RCB') | (tempPlayers['Position']=='LCB')] \

['bmi'].describe()

tempPlayers.loc[(tempPlayers['Position']=='LB') | \

(tempPlayers['Position']=='RB') | (tempPlayers['Position']=='LWB') |\

(tempPlayers['Position']=='RWB')]['bmi'].describe()

tempPlayers.loc[(tempPlayers['Position']=='CM') | \

(tempPlayers['Position']=='LCM') | (tempPlayers['Position']=='RCM') |\

(tempPlayers['Position']=='LAM') | (tempPlayers['Position']=='RAM') |\

(tempPlayers['Position']=='CAM')]['bmi'].describe()

tempPlayers.loc[(tempPlayers['Position']=='LM') | \

(tempPlayers['Position']=='RM') | (tempPlayers['Position']=='RW') |\

(tempPlayers['Position']=='LW')]['bmi'].describe()

tempPlayers.loc[(tempPlayers['Position']=='CDM') | \

(tempPlayers['Position']=='LDM') | (tempPlayers['Position']=='RDM')\

]['bmi'].describe()

tempPlayers=players.loc[(players['Overall']>=75)]

tempPlayers=tempPlayers[['Position','bmi']].copy()

tempPlayers.head()

keepers=tempPlayers.loc[(tempPlayers['Position']=='GK')]

keepers=keepers['bmi'].value\_counts().sort\_index()

keepers.plot.bar(figsize=(10,10))

cbs=tempPlayers.loc[(tempPlayers['Position']=='CB') | \

(tempPlayers['Position']=='RCB') | (tempPlayers['Position']=='LCB')]

cbs=cbs['bmi'].value\_counts().sort\_index()

cbs.plot.bar(figsize=(10,10))

wbs=tempPlayers.loc[(tempPlayers['Position']=='LB') | \

(tempPlayers['Position']=='RB') | (tempPlayers['Position']=='LWB') |\(tempPlayers['Position']=='RWB')]

wbs=wbs['bmi'].value\_counts().sort\_index()

wbs.plot.bar(figsize=(10,10))

cdms=tempPlayers.loc[(tempPlayers['Position']=='CDM') | \

(tempPlayers['Position']=='LDM') | (tempPlayers['Position']=='RDM')]

cdms=cdms['bmi'].value\_counts().sort\_index()

cdms.plot.bar(figsize=(10,10))

cms=tempPlayers.loc[(tempPlayers['Position']=='CM') | \

(tempPlayers['Position']=='LCM') | (tempPlayers['Position']=='RCM')]

cms=cms['bmi'].value\_counts().sort\_index()

cms.plot.bar(figsize=(10,10))

cams=tempPlayers.loc[(tempPlayers['Position']=='CAM') | \

(tempPlayers['Position']=='LAM') | (tempPlayers['Position']=='RAM')]

cams=cams['bmi'].value\_counts().sort\_index()

cams.plot.bar(figsize=(10,10))

ws=tempPlayers.loc[(tempPlayers['Position']=='LM') | \

(tempPlayers['Position']=='RM') | (tempPlayers['Position']=='RW') |\

(tempPlayers['Position']=='LW')]

ws=ws['bmi'].value\_counts().sort\_index()

ws.plot.bar(figsize=(10,10))

st=tempPlayers.loc[(tempPlayers['Position']=='ST') | \

(tempPlayers['Position']=='LS') | (tempPlayers['Position']=='RS')| \

(tempPlayers['Position']=='LF') | (tempPlayers['Position']=='RF')| \

(tempPlayers['Position']=='CF')]

st=st['bmi'].value\_counts().sort\_index()

st.plot.bar(figsize=(10,10))