I.J. Information Engineering and Electronic Business, **2023, 6, 30-47**
Published Online on December 8, 2023 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijieeb.2023.06.03

Optimal Playing Position Prediction in Football Matches: A Machine Learning Approach

Kevin Sander Utomo
School of Information Technology, Universitas Ciputra Surabaya, Surabaya 60219, Indonesia
Email: kevinsanderutomo@gmail.com
ORCID iD: https://orcid.org/0000-0001-9825-8779

Trianggoro Wiradinata*
School of Information Technology, Universitas Ciputra Surabaya, Surabaya 60219, Indonesia
Email: twiradinata@ciputra.ac.id
ORCID iD: https://orcid.org/0000-0002-7306-4254
*Corresponding Author

Received: 24 March, 2023; Revised: 18 April, 2023; Accepted: 07 May, 2023; Published: 08 December, 2023

Abstract: Deciding optimal playing position can sometimes a challenging task for anyone working in sport management industry, particularly football. This study will present a solution by implementing Machine Learning approach to find and help football managers determine and predict where to place individual existing football players/potential players into different positions such as Attacking Midfielder (AM), Defending Midfielder (DMC), All-Around Midfielder (M), Defender (D), Forward Winger (FW), and Goalkeeper (GK) in a specific team formation based on their attributes. To aid in this identification process, it may be beneficial to understand how a player's playstyle can affect where a player will be positioned in a team formation. The attributes used in facilitating the identification of the player position will be based on Passing Capabilities (AveragePasses), Offensive Capabilities (Possession, etc), Defensive Capabilities (Blocks, Through Balls, Tackles, etc), and Summary (Playtime, Goals, Assists, Passing Percentage, etc). The data that will be analysed upon will be scrapped manually from a popular football site that present football players statistics in a structured and ordered manner using a scrapping tool called Octoparse 8.0. Afterwards, the data that has been processed will be used to create a machine learning predictor modelled using various classification algorithms, which are KNN, Naive Bayes, Support Vector Machine, Decision Tree, and Random Forest ,coded using the Python programming language with the help of various machine learning and data science libraries, further enriched with copious graphs and charts which provides insight regarding the task at hand. The result of this study outputted in the form of the model predictor's evaluation metric proves the Decision Tree algorithm have both the highest accuracy and f1-score of 76% and 75% respectively, while Naïve Bayes sits the lowest at both 69% accuracy and f1-score. The evaluation has prioritized validating and filtering algorithms that have overfitting in copious amounts which are evident in both the KNN and Support Vector Machine algorithms. As a result, the model formed in this study can be used as a tool for prediction in facilitating and aiding football managers, team coaches, and individual football players in recognizing the performance of a player relative to their position, which in turn would help teams in acquiring a specific type of player to fill a systematic frailty in their existing team roster.

Index Terms: Talent Identification, Machine Learning, Classification, Web Scrapping, Team Formation, Football.

1. Introduction

Football is, without a doubt, the most popular sport worldwide. Other popular sports include basketball, tennis, and cricket, but nothing surpasses football's popularity. Globally renowned teams such as Manchester United and tournaments such as the Champions League have insured that football will continue to exist. Football has permeated all aspects of life, as we are continuously assaulted with advertisements and betting offers involving football celebrities. As football is played in numerous nations, it has produced dozens of superstars. Football's popularity rises as a result of the larger number of participants than in any other sport.

With the recent surge of football events in 2022, for example the World Cup, it is evident that football is returning to its former glory as one of the most glorified sports in the world. Incidentally, numerous amateur football players are interested in joining the professional football league. However, as an amateur football player trying to join a

This work is open access and licensed under the Creative Commons CC BY 4.0 License.

professional team on a whim to play professionally isn't realistic. On the other hand, it is the team manager that handles executive recruitment along with talent identification. Based on [1], talent identification is a very crucial avenue in effort to achieve top class sporting excellence and efficient guidance in any sports.

Usually, talent identification doesn't happen instantly, potential players would enlist in a practice program that would span from 3-6 months on average. Afterwards, at the end of the program, potential players who boast the most skill and potential will be enrolled in the team, hence enabling these players to play professionally. However, this approach isn't really recommended for small-medium scaled football teams, this is due to the exorbitant prices on hosting this program. Furthermore, talent searching also requires the skills of experienced analysers who are adequate at their field and discovery of sports talent using Artificial Intelligence (AI) is still incompetent [2]. Likewise, with the incorporation of talent identification, there also lies talent management, which possesses its own sets of importance in football management. In football management, talent identification and talent management go hand in hand. Talent management remains a challenge, especially in ensuring the best position for a player in a team formation. This may be due to the fact that these tasks requires a lot of experience in decision management and coaching. In general, there are no scientific equations or formulas adopted to recognize the most suitable position for each player in a particular team formation [1]. Football manager or team coaches needs to perform many experiments directly with the players and analyse them individually to get an idea of a player's skills. These featured skills would involve many different aspects such as offensive, defensive capabilities, and supporting capabilities. These areas would affect how the player would blend in into the team and be put in a suitable position [3].Given these various confines, professional sporting organizations are increasingly turning towards machine learning to assist with the identification of players who possess unique attributes in certain playing positions may offer a competitive advantage [4]. To utilize and make talent identification and position selection more accurate for football managers and team coaches, data mining techniques and machine algorithms, primarily classification algorithms could aid in making the correct decisions [3]. These techniques would require data that is accurate, structured, and processed in the first place [5]. For this reason, a model predictor created using various machine learning algorithms are needed and applied to help in giving positions recommendations for football managers and team coaches to players in a football team. The dataset provided for the training of the model predictor will be obtained using web-scraping from a popular football statistics website. Afterwards, the prediction model will then be compared with other common classification algorithms and the algorithm that achieves better results based on its evaluation metrics, using both accuracy and F1-Score, will be selected.

Furthermore, to make the task analysis at hand clarify able, the following analysis will be divided into 5 important points, some with their own particular sub-points, included with the purpose of supplying additional clarification and details to the main points. Incidentally, the 5 main points are: Target variable & Features Affecting the Target Variable, Data Understanding, Data Preparation, Modelling, and Evaluation.

In a previous study done by [4] using Machine Learning approach with the purpose to detect DDoS attacks using various Machine Learning Classification Algorithms has shown exceptional result with most evaluation metrics lying in the 98% mark. Having a high evaluation metrics shows that a model predictor is highly reliable, However the limitation of this study lies in the lack of explanation of how the Machine Learning approach is performed. To be able to understand how an analysis is performed, it should be explained briefly or in detail. Lastly, a past study done by [6] with the same purpose of predicting football player's position, uses an algorithm called the fuzzy logic, which was applied to 264 soccer players revealed that the system designed using fuzzy logic is able to identify the qualities among players exceptionally well. However, using fuzzy logic in itself requires a good deal of human intervention, primarily needing domain experts who are well-experienced in this area of expertise, therefore relying on another approach, for example Machine Learning using either Supervised Learning and Unsupervised Learning is preferrable because the iterative process is automatic and the machine is able to learn with more training, whereas using fuzzy logic, this process may be cluttered.

To demonstrate the analysis' strong points, the respective analysis will be compared with a past study performed and created in 2019 [7] with the same purpose of finding best playing positions for football players. However, after further examination and research, a difference can be spotted between the analysis performed by [7]. On one hand the analysis done by [7] uses regression, whereas for this current analysis, it uses classification algorithm to create a model predictor. One strength of this analysis, by using classification algorithm, the best playing position is able to be found instantly, however in the case of the analysis done by [7], the target variable that is needed to be predicted by the regression algorithm is based on four main blocks of features that have been filtered and separated depending on each possible playing positions. These features are then compiled as the dependent variables used to find the target variable, which is overall performance. Therefore, overall performance on each of these blocks must be found and then processed before determining the perfect playing position, meaning that the model doesn't predict the best playing position, yet individuals such as football managers is still the one doing the prediction. Another main strength lies in the application of trial and error, whereas in this analysis specifically, we will be using five classification algorithms to find the perfect algorithm to create a model predictor, whereas the analysis done by [7] only uses one regression algorithm, which is the linear regression. The final strength of this analysis compared to the study done by [7] lies in the utilization of cross-validation technique to help reduce bias and increase the effectiveness of training and testing of the model predictor.

2. Literature Review

Previous have shown various usage of machine learning and classification algorithms, primarily the Decision Tree algorithm in helping individuals and organizations perform prediction. These usages are prioritized and practiced widely in various domains, such as finance, healthcare, marketing, management, and many more. In finance, a study performed by [8] focused in developing a hybrid model called the Hybrid Credit Card Fraud Detection (HCCFD) system using anomaly detection techniques by applying genetic algorithm and multivariate normal distribution to demonstrate the effectiveness in detecting fraudulent credit card transactions. The results of the study showed that the HCCFD system outperformed other used machine learning algorithms such as artificial neural network, decision tree, and support vector machine, with an accuracy of 93.3%, including the Decision Tree algorithm performing relatively well at 79.4%. A separate study carried out by [9] for the cause of predicting football match winner outcomes from the output of the English Premier League using historical match statistic data shows suboptimal results with of the Decision Tree algorithm with an accuracy of 64.87%, hence showing that the model predictor is not suitable in performing the respective tasks. This limitation may be caused by incomplete or faulty data pre-processing. The author also mentioned that the problems were also caused due to overfitting and the trained model had limited data caused from the small size of the dataset with very few features. Therefore, in the study that will be performed these factors will be taken into account.

Moreover, a study by [10] explored various classification algorithms which are KNN, Naive Bayes, Support Vector Machine, Decision Tree, and Random Forest. The use of these algorithms assisted potential bicycle buyers in making informed choices. The classification model developed by the author, using data collected from 242 bicycle users in various communities in Indonesia. The findings suggest that Decision Trees has proven to be useful as a predictive tool for potential bicycle buyers in selecting the appropriate type of bicycle to purchase. Lastly a study by [11] was conducted to analyze the relationship between extracurricular activities and academic performance of students using Machine Learning prediction algorithms such as Decision Tree, Random Forest and KNN. The study focused on discovering the shortcomings in each algorithm and compares the prediction outcomes obtained by these algorithms. Results depicted that the Decision Tree algorithm outperforms other algorithms with an accuracy and F1-Score of 85% and 84% respectively. The author expects that the result of this study will help in paving the way for more specialized and in-depth studies in predicting academic performance of students using machine learning algorithms.

While these studies have demonstrated the effectiveness of Decision Trees in prediction tasks, there are some limitations that need to be considered. For instance, Decision Trees may overfit the training data, leading to poor performance on new, unseen data. To address this issue, researchers have developed various techniques, such as pruning, ensemble methods, and cross-validation, to improve the generalization of Decision Trees. Moreover, Decision Trees may be biased towards certain attributes or features, leading to suboptimal predictions. This issue can be addressed by careful feature selection and preprocessing. In conclusion, Decision Trees have emerged as a powerful tool in the field of machine learning, with various applications in prediction tasks.

3. Research Methodology

3.1 Block Diagram

The diagram of the Research Process depicts the data science methodology from business comprehension to model evaluation (see Fig. 1). Several parameter tuning procedures were iterated to improve the performance of a machine learning model.

Fig. 1. Diagram of the Research Process

3.2 Business Understanding

a. Target Variable:

1. Position: Suitable Position for football players

b. Features Affecting Target Variable:

2. KeyPasses: Key passes per game of player

3. Dribble: Dribbles per game of player
4. Dispossessed: Dispossessed per game of player
5. BadControl: Bad Control per game of player
6. Playtime: Total playtime of player
7. Tackles: Tackles per game of player
8. Intercepts: Intercepts per game of player
9. Fouls: Fouls per game of player
10. Offsides: Offsides per game of player
11. Clears: Clears per game of player
12. DribblePast: Dribble past per game of player
13. Blocks: Blocks per game of player
14. Goals: Total goals of player
15. Assists: Total assists of player
16. ShotsAveragePerGame: Shots average per game of player
17. PassSuccess: Pass success percentage of player
18. AveragePasses: Passes per game of player
19. Crosses: Crosses per game of player
20. Longballs: Long balls per game of player
21. ThroughBalls: Through balls per game of player

3.3 Data Understanding

In this point, analysis of the data will be explained step-by-step to help understand the process in understanding the data used to solve the task which have been explained in the first two points.

a. Data Collection

The dataset used for the analysis will be taken and scraped from https://www.whoscored.com/Statistics. The steps on how to replicate the process of web scraping will be explained in great detail in this subpoint. To perform web scraping on the consequent website given above, a free extension called Octoparse will be used. Octoparse is a cloud-based web data extraction solution that helps users extract relevant information from various types of websites. To begin, first open up Octoparse. Make sure the newest Octoparse has been installed, currently Octoparse is on its 8th version. If everything has been implemented and installed properly, the main page will be shown (see Fig. 2).

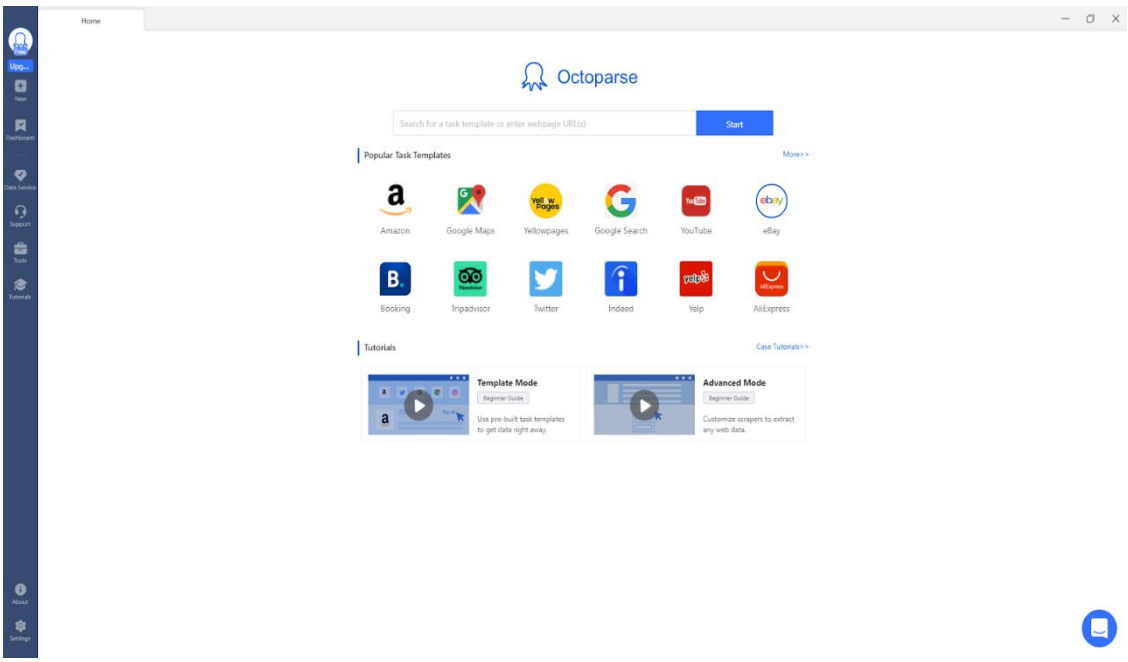


Fig. 2. Main Page of Octoparse

Afterwards, we can then enter in the link of the website that we want to scrap in the search box below the Octoparse logo. Now, after a link has been entered in the search box, Octoparse will redirect us to something called a "workflow" (see Fig. 3), this is the place where we are given the leniency to customize our own web-crawler and web-scraper.

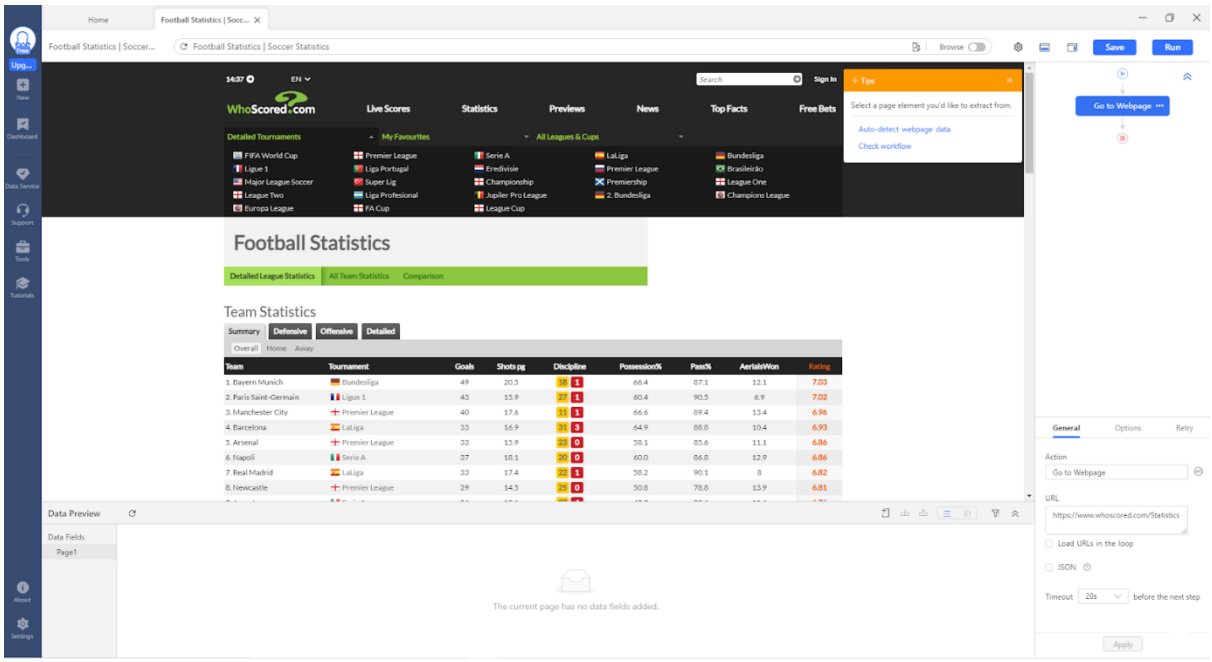
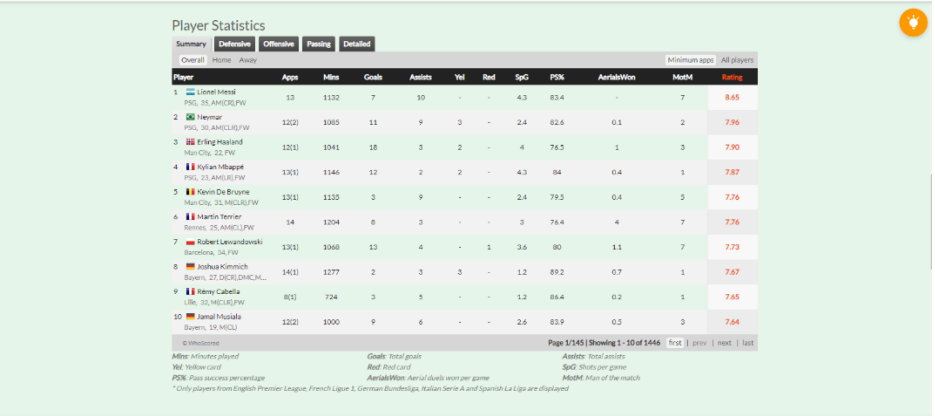


Fig. 3. Workflow of Octoparse

First before we start the crawling and scraping process, we want to analyze the page of the website, there are two main tables present on the website. Firstly, the table for team statistics and secondly the table for player statistics. Based on our task objective, we will take the player statistics table (see Table 1.) instead.

Table 1. Football Player Statistics Table



Player Statistics

Summary | Defensive | Offensive | Passing | Detailed

Overall | Home | Away

Minimum apps | All players

	Player	Apps	Mins	Goals	Assists	Yel	Red	SpG	PS%	AerialsWon	MotM	Rating
1	Lionel Messi PSG, 35, AM(CR),FW	13	1132	7	10	-	-	4.3	83.4	-	7	8.65
2	Neymar PSG, 30, AM(CLR),FW	12(2)	1085	11	9	3	-	2.4	82.6	0.1	2	7.96
3	Erling Haaland Man City, 22, FW	12(1)	1041	18	3	2	-	4	76.5	1	3	7.90
4	Kylian Mbappé PSG, 23, AM(LR),FW	13(1)	1146	12	2	2	-	4.3	84	0.4	1	7.87
5	Kevin De Bruyne Man City, 31, M(CLR),FW	13(1)	1135	3	9	-	-	2.4	79.5	0.4	5	7.76
6	Martin Terrier Rennes, 25, AM(CL),FW	14	1204	8	3	-	-	3	76.4	4	7	7.76
7	Robert Lewandowski Barcelona, 34, FW	13(1)	1068	13	4	-	1	3.6	80	1.1	7	7.73
8	Joshua Kimmich Bayern, 27, DC(R),DMC,M...	14(1)	1277	2	3	3	-	1.2	89.2	0.7	1	7.67
9	Rémy Cabella Lille, 32, M(CLR),FW	8(1)	724	3	5	-	-	1.2	86.4	0.2	1	7.65
10	Jamal Musiala Bayern, 19, M(CL)	12(2)	1000	9	6	-	-	2.6	83.9	0.5	3	7.64

© WhoScored

Page 1/145 | Showing 1 - 10 of 1446 | first | prev | next | last

Mins: Minutes played
Yel: Yellow card
PS%: Pass success percentage
Goals: Total goals
Red: Red card
AerialsWon: Aerial duels won per game
Assists: Total assists
SpG: Shots per game
MotM: Man of the match
* Only players from English Premier League, French Ligue 1, German Bundesliga, Italian Serie A and Spanish La Liga are displayed

On top of that, we want to decide the flow or in other words the logic for scraping the specific website. If we examine closely on the table of the player statistics, we can see that there are 5 tabs of which are: Summary, Defensive, Offensive, Passing, and Detailed. Currently we are on the Summary tab, each tab will have its own respective attributes. Hence, these attributes and also the player information from each tab will be scraped. For simplicity purposes the Detailed tab will not be scrapped because it contains real-time statistics and is not suitable for the task we are trying to accomplish. Another thing to note is the pagination present on the table. There are about 145 pages, each with 10 rows of data regarding each individual football player on each page. That being said, the logic for scraping the consequent table would have to take into account these tabs and also pagination routes and options. To make things easier, we will make use of the auto-detect feature on the top right corner of the workflow (see Fig. 4).

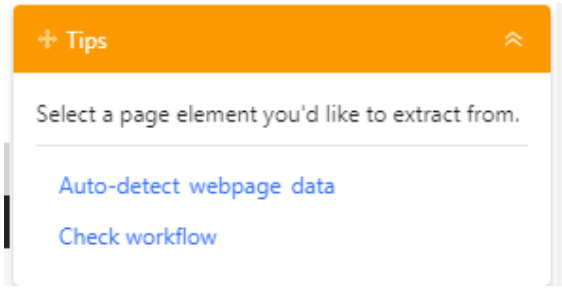


Fig. 4. Auto-detect Scraping Feature

Subsequently, we then want to click on the auto-detect webpage data link and Octoparse will automatically detect all the scrapable contents on the website (see Fig. 5). Octoparse shows 4 blocks of content that is potentially scrapable. However, since the main focus lies on scraping the player statistics table, we want to focus on this part. Octoparse auto-detect feature has already taken into account pagination, so we don't have to worry about it anymore. In spite of that, there is one caveat regarding this method of web-scraping, which is the inability to switch tabs. It is quite hard to switch tabs and scrape the data due to the structure of the table. Hence, we will create 4 workflows, each with the purpose of getting all the attributes and player information on each tab.

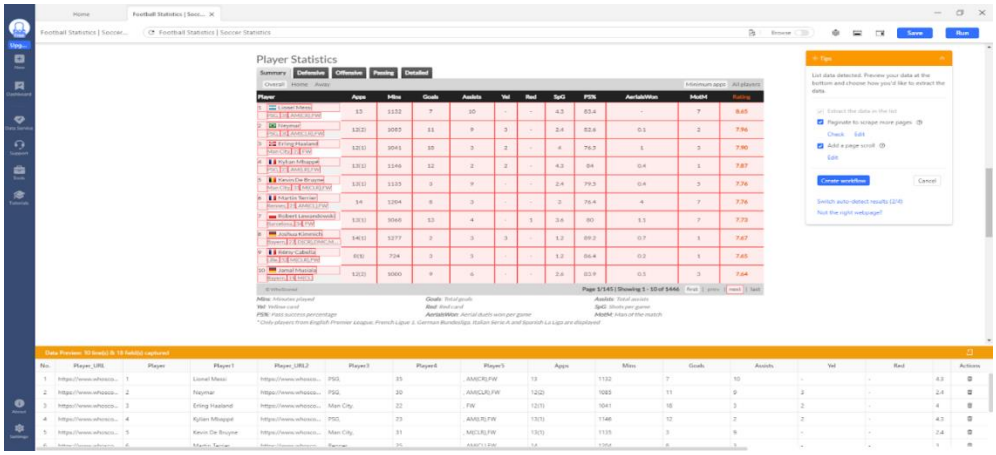


Fig. 5. Auto Detect Result

Following the usage of the auto-detect feature, we then want to click on the create workflow button whilst having the options above checked. Afterwards, we can see that a logic tree/flow tree has been created on the right side of the workflow (see Fig. 6). Indicating the content that we want to scrape and how it will be scrape.

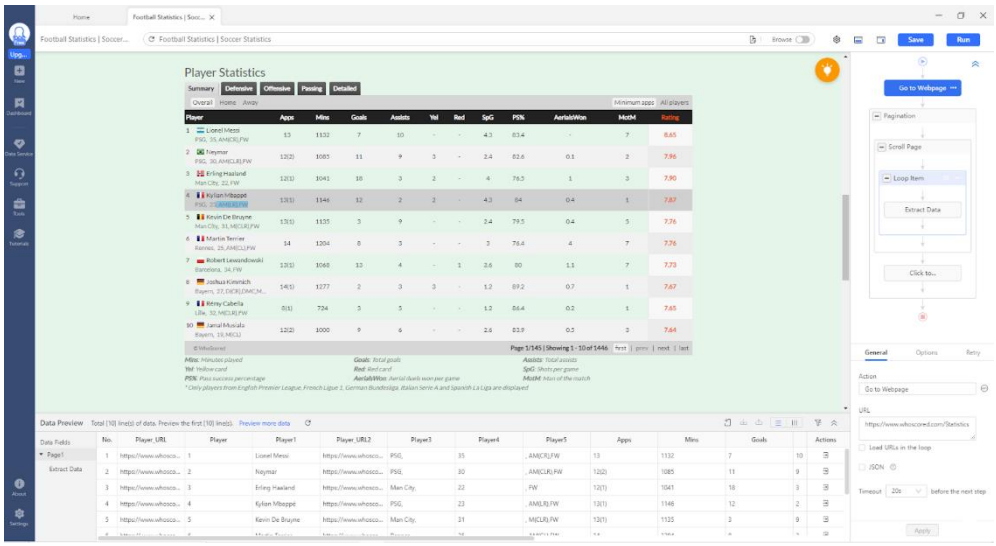


Fig. 6. Flow Tree at the Right Side of the Workflow

Coincidentally, after confirming the flow of the web-scraping process, we then want to proceed on starting the web-scraping process. To do this we can simply just press the run button at the top right corner of the workflow just below the exit program button. Octoparse will then display a modal message whether to run the scraping process via local or cloud, since we are currently using the free version of Octoparse, we will carry on with local devices. After confirming this option, Octoparse will then run the scraping process and scrape all the available content based on the flow tree (see Fig. 7). And after the scraping process is done, Octoparse will send out a notification indicating that the scraping process is finished along with the option to export the data in various different file formats such as xls, csv, etc. For the use of the analysis, we will export the scraped content/data into the csv format.

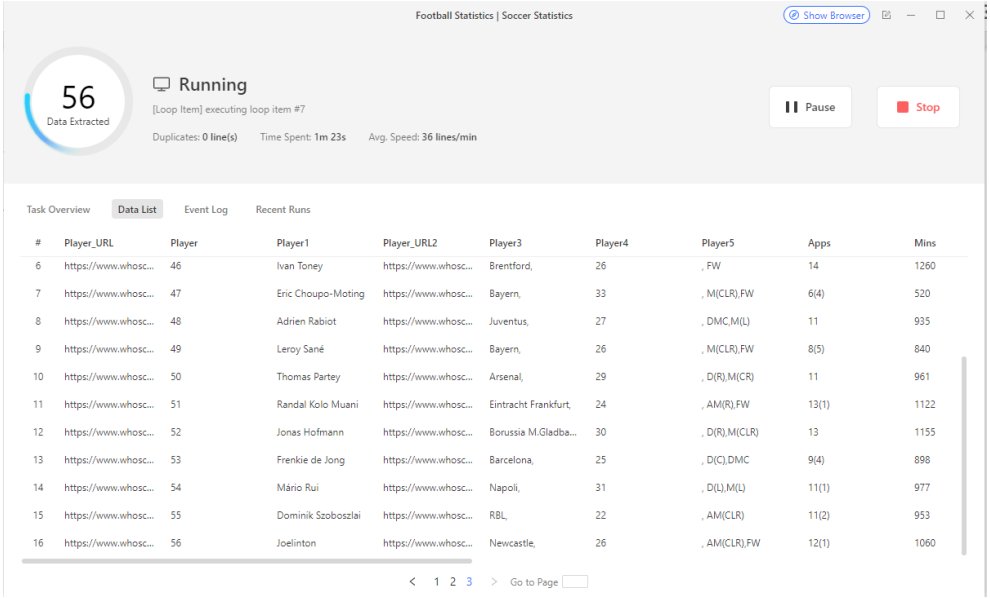


Fig. 7. Scraping Data from the Website

We then want to do this for all the remaining tabs, which are the Defensive, Offensive, and Passing Tab. The flow for scraping the content on each respective remaining tab is similar, however the difference lies in adding a simple click listener to navigate between these tabs (see Fig. 8). At first, we start out at the Summary Tab, hence we didn't need to add a click listener, however the opposite could be said when we are scraping from the other tabs.

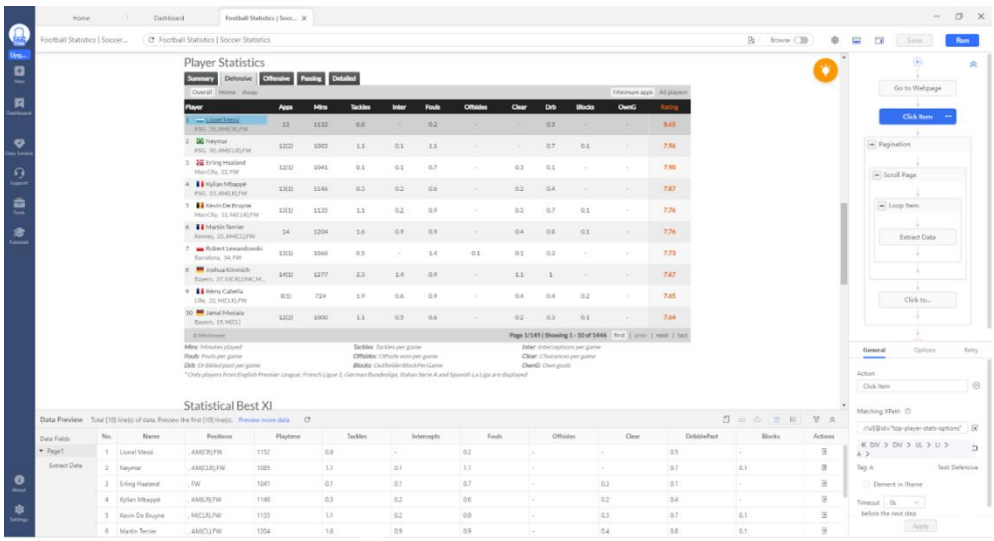


Fig. 8. Added Click Listener to Navigate Between Tabs

Having collected all the necessary information and converting it into csv format. We then want to head into python to do the analysis based on the task objective. But first before doing the analysis, the csv files need to be imported into python whilst also importing prerequisite libraries for the subsequent analysis (see Fig. 9).

Import General Libraries

```
[ ] #Import Libraries & CSV
    import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns
    algorithms_list = ["KNN","Naive Bayes", "Support Vector Machine", "Decision Tree", "Random Forest"]
    precision_list = []
    training_accuracy_list = []
    test_accuracy_list = []
    recall_list = []
    f1score_list = []
```

Data Collection

```
[ ] url1 = 'https://raw.githubusercontent.com/Sumahan7/FootballPlayers/main/FootballPlayersSummary.csv'
    url2 = 'https://raw.githubusercontent.com/Sumahan7/FootballPlayers/main/FootballPlayersPassing.csv'
    url3 = 'https://raw.githubusercontent.com/Sumahan7/FootballPlayers/main/FootballPlayersDefense.csv'
    url4 = 'https://raw.githubusercontent.com/Sumahan7/FootballPlayers/main/FootballPlayersOffensive.csv'

    footballplayer_summary = pd.read_csv(url1, delimiter=',')
    footballplayer_passing = pd.read_csv(url2, delimiter=',')
    footballplayer_defense = pd.read_csv(url3, delimiter=',')
    footballplayer_offense = pd.read_csv(url4, delimiter=',')
    # df = pd.DataFrame(dataset)
```

Fig. 9. Importing Csv and Libraries

After importing both the csv and libraries, we can then view the overall structure of the dataset using the head() function (see Table 2.). It is imminent that the tables are still relatively dirty and cleaning would need to be operated on each of the tables.

Table 2. View Overall Structure of Dataset Using head() Function

```
[ ] footballplayer_summary.head()
```

	Name	Positions	Playtime	Goals	Assists	ShotsAveragePerGame	PassSuccess
0	Lionel Messi	, AM(CR),FW	1132	7%	10%	4.3%	83.4
1	Neymar	, AM(CLR),FW	1085	11%	9%	2.4%	82.6
2	Erling Haaland	, FW	1041	18%	3%	4%	76.5
3	Kylian Mbappé	, AM(LR),FW	1146	12%	2%	4.3%	84.0
4	Kevin De Bruyne	, M(CLR),FW	1135	3%	9%	2.4%	79.5

```
[ ] footballplayer_passing.head()
```

	Name	Positions	AveragePasses	Crosses	LongBalls	ThroughBalls
0	Lionel Messi	, AM(CR),FW	59.9	0.8%	2.9%	1.3%
1	Neymar	, AM(CLR),FW	59.9	0.7%	1.9%	0.6%
2	Erling Haaland	, FW	14.4	-%	0.1%	0.2%
3	Kylian Mbappé	, AM(LR),FW	34.7	0.4%	0.3%	0.1%
4	Kevin De Bruyne	, M(CLR),FW	43.5	2.4%	2.9%	0.5%

```
footballplayer_defense.head()
```

	Name	Positions	Playtime	Tackles	Intercepts	Fouls	Offsides	Clear	DribblePast	Blocks
0	Lionel Messi	, AM(CR),FW	1132	0.8%	-%	0.2%	-%	-%	0.5%	-%
1	Neymar	, AM(CLR),FW	1085	1.1%	0.1%	1.1%	-%	-%	0.7%	0.1%
2	Erling Haaland	, FW	1041	0.1%	0.1%	0.7%	-%	0.3%	0.1%	-%
3	Kylian Mbappé	, AM(LR),FW	1146	0.3%	0.2%	0.6%	-%	0.2%	0.4%	-%
4	Kevin De Bruyne	, M(CLR),FW	1135	1.1%	0.2%	0.9%	-%	0.3%	0.7%	0.1%

```
[ ] footballplayer_offense.head()
```

	Name	Positions	KeyPasses	Dribble	Dispossed	BadControl
0	Lionel Messi	, AM(CR),FW	2.6%	3.6%	2.4%	1.5%
1	Neymar	, AM(CLR),FW	2.7%	1.6%	1.6%	2.1%
2	Erling Haaland	, FW	1.1%	0.3%	0.9%	1.3%
3	Kylian Mbappé	, AM(LR),FW	1.9%	1.9%	1.3%	2.4%
4	Kevin De Bruyne	, M(CLR),FW	3.4%	0.8%	1.1%	1.2%

b. Data Integration

Foremost, before doing any further action on the datasets, we must first turn the datasets into a dataframe. The purpose of transforming the dataset into a dataframe is so we are able to use functions/properties from pandas that are only available for data frames. Since the data frames are still separated, we must first merge them into a compound table with all the features formed. Hence by doing so, we are able to do our analysis based on the described features from the Target variable & Features Affecting the Target Variable point. After merging the datasets together into one compound table, duplicated features are present since each respective table prior to merging has similar features and therefore needs to be removed, as well as the "Name" feature, since it doesn't bring any weight/clarity on the analysis (see Table 3.). The total features resulting from the merging of data frames amounts to 21 features.

Table 3. Converting Data into Dataframe and Merging of Dataframes

Data Integration

```
[ ] #turn all dataset into dataframes
    df_footballplayer_summary = pd.DataFrame(footballplayer_summary)
    df_footballplayer_defense = pd.DataFrame(footballplayer_passing)
    df_footballplayer_passing = pd.DataFrame(footballplayer_defense)
    df_footballplayer_offense = pd.DataFrame(footballplayer_offense)

    #merge all dataframes together without deleting any rows nor columns
    df_footballplayer_summary_defense = df_footballplayer_summary.merge(df_footballplayer_defense, left_on="Name", right_on= "Name")
    df_footballplayer_passing_summary_defense = df_footballplayer_passing.merge(df_footballplayer_summary_defense, left_on="Name", right_on= "Name")
    df = df_footballplayer_offense.merge(df_footballplayer_passing_summary_defense, left_on="Name", right_on= "Name")
```

c. Pre-Cleaning

Thereafter the data integration process has eclipsed, a step called pre-cleaning will be performed upon the dataframe. The pre-cleaning step will involve cleaning the table to transform it into something “usable” or able to be processed by the machine learning model. Notice that some of the values on the table ends with “\t” this might have something to do with how pandas read our csv files. But no worries, we can simply replace all values or instances of all features that end with “\t” with a blank string (see Table 4.). Furthermore, we also see that some values are indicated with “-”, the website we scraped data from depicts values that are 0 as “-”, so for the time being we will be replacing all “-” values with NaN to make processing our data much easier in the later points.

Table 4. Replacing Instances that Contains “\t” and “-”

```
Pre-Cleaning

#remove unnecessary string "\t" from all columns with string values
df['KeyPasses'] = df['KeyPasses'].str.replace('\t', '')
df['Dribble'] = df['Dribble'].str.replace('\t', '')
df['Dispossed'] = df['Dispossed'].str.replace('\t', '')
df['BadControl'] = df['BadControl'].str.replace('\t', '')
df['Tackles'] = df['Tackles'].str.replace('\t', '')
df['Intercepts'] = df['Intercepts'].str.replace('\t', '')
df['Fouls'] = df['Fouls'].str.replace('\t', '')
df['Offsides'] = df['Offsides'].str.replace('\t', '')
df['Clear'] = df['Clear'].str.replace('\t', '')
df['DribblePast'] = df['DribblePast'].str.replace('\t', '')
df['Blocks'] = df['Blocks'].str.replace('\t', '')
df['Goals'] = df['Goals'].str.replace('\t', '')
df['Assists'] = df['Assists'].str.replace('\t', '')
df['ShotsAveragePerGame'] = df['ShotsAveragePerGame'].str.replace('\t', '')
df['Tackles'] = df['Tackles'].str.replace('\t', '')
df['Crosses'] = df['Crosses'].str.replace('\t', '')
df['LongBalls'] = df['LongBalls'].str.replace('\t', '')
df['ThroughBalls'] = df['ThroughBalls'].str.replace('\t', '')

#replace all "-" values in all columns with Nan to make it easier to visualize
df = df.mask(df == '-')
```

If we were to call the info() function, we can see that most of the features of the data is of type object Z, even though the values contained in all the features are all numerical, except for Position (see Fig. 10). Hence, we need to convert the datatype of the features with the type of object into their own respective data types (int/float).

```
df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1460 entries, 0 to 1459
Data columns (total 21 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Position            1460 non-null   object
 1   KeyPasses           1359 non-null   object
 2   Dribble             1263 non-null   object
 3   Dispossed           1313 non-null   object
 4   BadControl          1401 non-null   object
 5   Playtime            1460 non-null   int64
 6   Tackles             1387 non-null   object
 7   Intercepts          1309 non-null   object
 8   Fouls               1382 non-null   object
 9   Offsides            564 non-null    object
 10  Clear               1394 non-null   object
 11  DribblePast         1333 non-null   object
 12  Blocks              1000 non-null   object
 13  Goals               756 non-null    object
 14  Assists             682 non-null    object
 15  ShotsAveragePerGame 1354 non-null   object
 16  PassSuccess         1460 non-null   float64
 17  AveragePasses       1460 non-null   float64
 18  Crosses             951 non-null    object
 19  LongBalls           1419 non-null   object
 20  ThroughBalls        458 non-null    object
dtypes: float64(2), int64(1), object(18)
memory usage: 250.9+ KB

[ ] df[['KeyPasses', 'Dribble', 'Dispossed', 'BadControl',
        'Playtime', 'Tackles', 'Intercepts', 'Fouls', 'Offsides', 'Clear',
        'DribblePast', 'Blocks', 'Goals', 'Assists', 'ShotsAveragePerGame',
        'PassSuccess', 'AveragePasses', 'Crosses', 'LongBalls', 'ThroughBalls']] = df[['KeyPasses', 'Dribble', 'Dispossed', 'BadControl',
        'Playtime', 'Tackles', 'Intercepts', 'Fouls', 'Offsides', 'Clear',
        'DribblePast', 'Blocks', 'Goals', 'Assists', 'ShotsAveragePerGame',
        'PassSuccess', 'AveragePasses', 'Crosses', 'LongBalls', 'ThroughBalls']].apply(pd.to_numeric)
```

Fig. 10. Convert Object Features into Numerical

Something else we need to take into account is the position feature. The values of the position feature are still relatively dirty (leading commas, multiple positions of individual football players, whitespaces, and variation of the same values) (see Fig. 11). Based on the task objective we are trying to find the perfect position of a football players we only need one position to do our analysis. Hence, we need to clean and replace the inconsistent values in the position column.

	Position	KeyPasses	Dribble	Dispossed	BadControl	Playtime	Tackles	Intercepts	Fouls	Offsides	...	DribblePast	Blocks	Goals	Assists	ShotsAveragePerGame	PassSuccess	AveragePasses	Crosses	LongBalls	ThroughBalls
0	, AM(CR),FW	2.6	3.6	2.4	1.5	1132	0.8	NaN	0.2	NaN	...	0.5	NaN	7	10	4.3	83.4	59.9	0.8	2.9	1.3
1	, AM(CLR),FW	2.7	1.6	1.6	2.1	1085	1.1	0.1	1.1	NaN	...	0.7	0.1	11	9	2.4	82.6	59.9	0.7	1.9	0.6
2	, FW	1.1	0.3	0.9	1.3	1041	0.1	0.1	0.7	NaN	...	0.1	NaN	18	3	4	76.5	14.4	NaN	0.1	0.2
3	, AM(LR),FW	1.9	1.9	1.3	2.4	1146	0.3	0.2	0.6	NaN	...	0.4	NaN	12	2	4.3	84.0	34.7	0.4	0.3	0.1
4	, M(CLR),FW	3.4	0.8	1.1	1.2	1135	1.1	0.2	0.9	NaN	...	0.7	0.1	3	9	2.4	79.5	43.5	2.4	2.9	0.5

Fig. 11. Position Column Pre-Cleaning

After pre-cleaning the position column. The position column now only contains 6 unique values, which shows the position a football player can be assigned to in a team formation (see Fig. 12).

```
df["Position"].unique()

array(['AM', 'FW', 'M', 'D', 'DMC', 'GK'], dtype=object)
```

Fig. 12. All Unique Values of Position Column

d. Data Visualization

To get a better understanding of the data we are working with and also gain insight we will be performing Data Visualization, specifically EDA. Exploratory Data Analysis (EDA) refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypotheses and to check assumptions with the help of summary statistics and graphical representations [12-14]. We will be dividing the step of EDA Visualization

into two categories which are EDA (Univariate) and EDA (Multivariate) [15]. Univariate means the analysis of one variable whereas Multivariate means the analysis of more than one variable.

e. EDA (Univariate)

To get a brief layout and structure of the data we can get a sample of the data using the head() function to give us the first 5 instances of the data. Next, we can find the shape of the data using the shape() function which gives us the total number of columns and rows of the data. The data comprises 1460 rows/instances and 21 columns including the target variable. We can divide features of the data into both numerical and categorical. The numerical features are: KeyPasses, Dribble, Dispossessed, BadControl, Playtime, Tackles, Intercepts, Fouls, Offsides, Clears, DribblePast, Blocks, Goals, Assists, ShotsAveragePerGame, PassSuccess, AveragePasses, Crosses, Longballs, and ThroughBalls. The categorical feature of the data is Position. Subsequently, we are then going to research and analyse both numerical features and categorical features of the data.

f. EDA (Multivariate Numerical)

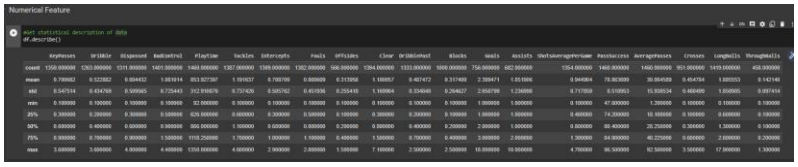


Fig. 13. Describe() Function to Get Statistical Description of Data

We can further enrich our knowledge and information about the numerical features of the data using the describe() function, which gives a lot of statistical information of the data (mean, range, std, etc) (see Fig. 13). A good insight to be drawn from these visualizations is regarding the range of the data: the range of the data of all the features lies between 0.1 until 1350. The difference between the ranges of these data may be major but we will normalize these values during the data pre-processing step. Another thing to note is that the count of the data is not the same for each of the features, this is because during the pre-cleaning step, we transformed all "-" values into NaN to make it easier for us to process the data later on.

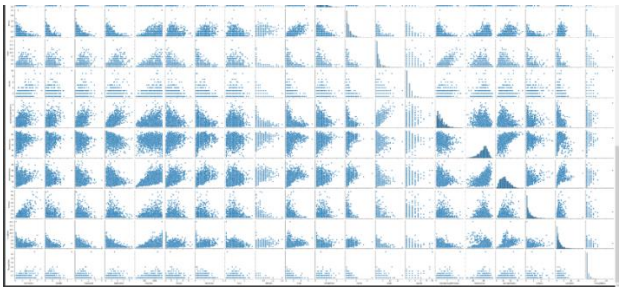


Fig. 14. Pair Plot between the Numerical Features of the Data

We can use a pair plot to analyse numerical features with respect to other numerical features (see Fig. 14). A good thing to note in this analysis is regarding the line of best fit. Hence, we can draw a line of best fit with respect to other numerical features, therefore we can categorize this visualization of the graph as linear (if one of the features is high, then the target variable is also high). We can detect a crucial thing regarding the graph, which are "outliers"; the outliers are present on the left side of the graph and may have a considerable impact on the prediction of the data.

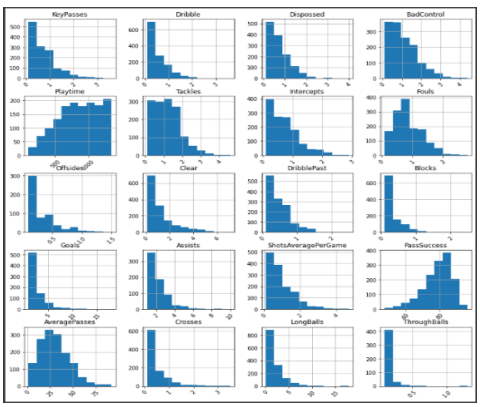


Fig. 15. Histogram of All Numerical Features

Next, we can see the data distribution of each feature using a histogram (see Fig. 15). From the histogram, we can conclude that most numerical features are skewed to the right. With the exception of PassSuccess and Playtime. Generally, if data distribution is skewed to the right then that means that potential outliers of the distribution curve are further out towards the left. And the opposite could be said if the data were to be skewed to the left.

g. EDA (Multivariate Categorical)

Similarly, with the EDA of the numerical features, we can use describe(include="object"), to get statistical information of the categorical features of the data (unique values, frequency, etc) (see Fig. 16).

Fig. 16. Categorical Feature Statistical Description

Afterwards, we can then view the distribution of unique values of the categorical features using a bar graph (see Fig. 17). The insights we got from the bar graph is that most of the football players of the dataset have a position of a Defender whilst Goalkeeper having the least. We can also then use a pie chart to illustrate the numerical proportion of data in percentages (see Fig. 18).

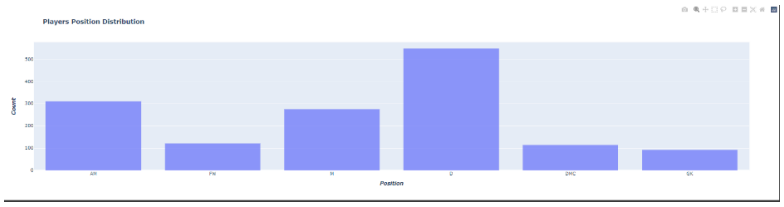


Fig. 17. Position Distribution using Bar Graph

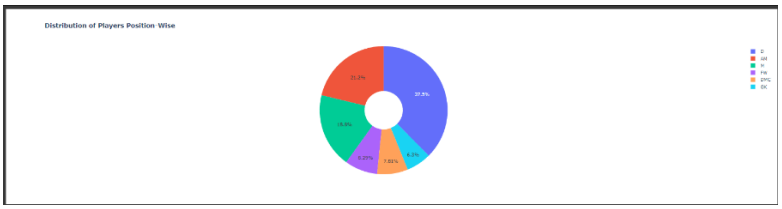


Fig. 18. Position Distribution Percentages using Pie Chart

h. Data Preparation

In this point, pre-processing of data will be explained in detail. Pre-processing of the data is done to prepare (cleaning and organizing) the data to make it suitable for building and training the Machine Learning model [16]. Afterwards, the determining and dividing of data into dependent and independent variables, along with splitting the data into training and testing will be performed. Lastly, oversampling will be applied due to imbalance data across the values of the features and feature scaling will be applied on the independent variables to make the process more accurate.

i. Filling NaN Values

As touched upon the earlier parts of the analysis, specifically on the Data Understanding phase. We have converted all "-" values into NaN, therefore we will fill these null values with 0 (see Fig. 19).

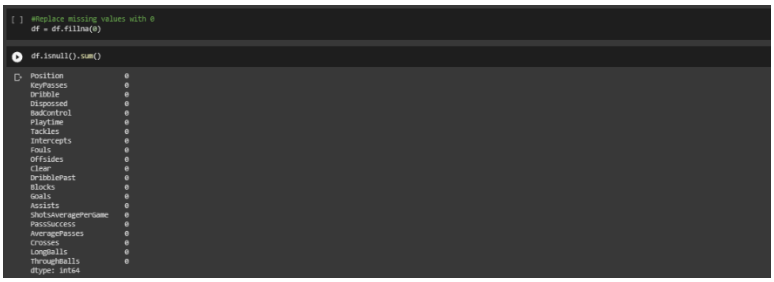


Fig. 19. Filling NaN Values with 0

j. Deciding Dependent and Independent Variables

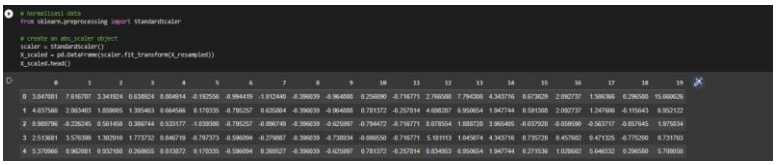
The next step of data pre-processing is splitting our features into dependent variables and independent variables. The target variable (y) is the Position of the player because we want to classify football players into the perfect position based on team formation, underpriced or standard priced based on the dependent variables. Hence the other variables affecting the target variable will be the dependent variables (X). Since we are trying to make a classification model to predict a player's position based on these dependent variables.

k. Balancing Imbalanced Data

Subsequently, we want to then balance the classes in our target variable. Balancing imbalance data is important because it makes training a Machine Learning model easier as it prevents the model from being biased towards one class just because it contains more data [17]. The balancing method that will be used is Random Oversampling. If we see the classes of the Position Column, we can see that Defender has 548 instances and Goalkeeper has 92 instances. Hence, balancing of these classes are needed due to the reason above. To do this, we will be using Random Oversampling. As the name suggests, this is a technique that randomly selects points from the minority class and duplicates them to increase the number of data points in the minority class. Oversampling is the most suited instead of under sampling because, if we were to delete part of the majority class then the accuracy of training the model will be reduced and clarity will be lost [17]. Subsequently, the sample might not accurately represent the real world and may cause the analysis to be inaccurate.

l. Feature Scaling

After balancing our target variable value, we then want to do feature scaling on our variables. Feature scaling is one of the most important data pre-processing steps in Machine Learning. Algorithms that compute distances between features tend to have numerically large values if the data is not scaled [18]. Hence, we will be applying a feature scaling technique called Standardization (see Fig. 20). This is made possible using a sklearn library called StandardScaler. The reason why we would want to standardize our data is because our data comes in varying ranges and has a normal distribution. By standardizing our data, we do not have to take in account data distribution anymore. Hence, using standardization we are able to transform every feature of our data so they have the same dimension to remove redundancy and repetitiveness [19]. Furthermore, standardization is not affected by outliers, because there is no predefined range of transformed features.



```
# Normalizing data
from sklearn.preprocessing import StandardScaler

# create an abs_scaler object
scaler = StandardScaler()
X_scaled = pd.DataFrame(scaler.fit_transform(X_resampled))
X_scaled.head()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	3.847681	7.616707	3.341924	0.628524	0.884914	-0.192556	-0.994419	-1.012440	-0.390039	-0.964888	0.258890	-0.716771	2.766580	7.794388	4.343716	0.673829	2.092737	1.586366	0.296500	15.680628
1	4.037566	2.865403	1.859005	1.395403	0.884566	0.176335	-0.795257	0.635804	-0.390039	-0.964888	0.781372	-0.257814	4.696207	6.950854	1.947744	0.591308	2.092737	1.247686	-0.115643	0.952122
2	0.988796	-0.226245	0.561458	0.386744	0.533177	-1.039380	-0.795257	-0.096749	-0.390039	-0.625887	-0.784472	-0.716771	0.878554	1.888728	3.965485	-0.037929	-0.698590	-0.563717	-0.857645	1.975834
3	2.513681	-3.576389	1.302910	1.773732	0.846719	-0.707373	-0.586094	-0.279887	-0.390039	-0.738834	-0.088550	-0.716771	5.581115	1.045074	4.343716	0.735728	0.457682	0.471325	-0.775200	0.731763
4	5.370966	0.962081	0.932188	0.208655	0.913072	0.170335	-0.586094	0.269527	-0.390039	-0.625887	0.781372	-0.257814	0.834953	0.950854	1.947744	0.271536	1.028082	1.648532	0.296500	5.708050

Fig. 20. Feature Scaling with Standardization

m. Train and Test Split

After standardizing the data, we want to split our data into training and testing data. We split our training data and testing data with a ratio of 0.7 and 0.3 respectively.

n. Modeling

In this point the process of creating a machine learning model will be elaborated. Five classification algorithms will be used to create our Machine Learning model. The classification algorithms used are: KNN, Naive Bayes, Support Vector Machine, Decision Tree, and Random Forest. Afterwards during the Evaluation step, the best model based on the classification algorithms used will be chosen and explained.

o. Maximizing Parameter

Before we create a model, it is advisable to first find the best parameter to maximize the performance of the Machine Learning model. To achieve this, a sklearn library called GridSearch will be used. A good thing to note is that each classification algorithm has their own set of parameters and differs from each other. Therefore, when we use GridSearch, we won't be maximizing the same parameters over and over.

a) Parameters to be maximized in the KNN Algorithm as recommended by [20] (see Fig. 21)*:*

- n_neighbors: Number of neighbors defined

Fig. 21. Finding Best Parameters for KNN Algorithm

b) Parameters to be maximized in the Naive Bayes Algorithm as recommended by [21] (see Fig. 22):

- var_smoothing: The portion of the largest variance across all features

Fig. 22. Finding Best Parameters for Na ïve Bayes Algorithm

c) Parameters to be maximized in the Support Vector Machine Algorithm as recommended by [23] (see Fig. 27):

- kernel: Kernel type to be used in the algorithm
- C: Strength of regularization
- gamma: Kernel coefficients

Fig. 23. Finding Best Parameters for Support Vector Machine Algorithm

d) Parameters to be maximized in the Decision Tree Algorithm as recommended by [23] (see Fig. 24):

- criterion: The function to measure the quality of a split
- max_depth: Maximum depth of tree
- min_samples_split: Minimum number of samples to split an internal node
- min_samples_leaf: Minimum number of samples required to be at a leaf node

Fig. 24. Finding Best Parameters for Decision Tree Algorithm

e) Parameters to be maximized in the Random Forest Algorithm as recommended by [24] (see Fig. 25):

- max_features: The number of features when looking for the best split
- max_depth: Maximum depth of tree
- min_samples_split: Minimum number of samples to split an internal node
- min_samples_leaf: Minimum number of samples required to be at a leaf node

Fig. 25. Finding Best Parameters for Random Forest Algorithm

p. Creating and Training Model

After having found the best parameters for each algorithm, we then want to create our model. Hence, why we used GridSearch to find the best parameter when we want to create our Machine Learning model. To create a model, we need to use the respective algorithms classifier from sklearn (see Fig. 26). Afterwards, we can then train our model so that

our model will learn how to understand and deal with unknown data when we test our model, to predict the best position for a football player (independent variable) based on the features (dependent variables) [25]. Or in other words we train our model so that it can learn patterns in the data.

```
# Pilih K terbaik dari data percobaan di atas
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(X_train, y_train)
prediction = knn.predict(X_test)
```

Fig. 26. Creating Various Models Using Best Parameters from GridSearch

3. Evaluation

In this point the metric evaluation of the model will be provided, the explanation for choosing the metrics, and choosing as well as explanation of the best model based on the evaluation metrics. Evaluation acts as the final step of Machine Learning. The reason as to why that is the case is because, if we have a Machine Learning model and have trained it beforehand. How are we going to test whether our model has successfully solved the problem at stake? How do we know whether our model did find a solution to the problem and how efficiently did they do it? These are some of the questions asked during the evaluation step. If all these questions are able to be answered correctly, then we can say that our model is marked as worthy and reliable.

A. Reasons of Using Specific Evaluation Metric

The evaluation metrics used for the specific analysis will be related to the type of task that is used. Because we are creating a Machine Learning model to do classification, therefore we will be using evaluation metrics that are suitable to evaluate a classification model. Every evaluation metric in a classification model will be based on the confusion matrix. Confusion matrix is essentially a performance measurement/indicator that provides details in the form of true/false labels which classes are being predicted correctly, incorrectly, and the errors that may have been made by the model [26]. There are four outcomes in a confusion matrix. First case is where a prediction is said to be true and it comes out as true (True Positive), second case is where a prediction is said to true and it comes out as false (False Negative), third case is where a prediction is said to be false and it comes out as true (False Positive), and lastly the case where a prediction is said to be false and it comes out as false (True Negative) [26]. After receiving and determining a confusion matrix can we then use the other evaluation metrics of a classification task. Hence, the evaluation metrics used in these analyses will be: Accuracy, Precision, Recall, and F1-Score. Accuracy is an evaluation metric in the classification task that looks at the True Positives and True Negatives of the confusion matrix and measures how many times a model got the prediction right [27, 28]. The reason as to why Accuracy is used as one of the evaluation metrics is because Accuracy is a very popular evaluation metric in classification and it is very easy to interpret for people who may not have a Machine Learning background. Another reason is because of the nature of our task analysis, since we are trying to predict the perfect position for a football player in a team formation, it is necessary for the model to be able to predict correctly most of the time, so it can maximize a team's performance by assigning talented football players in their ideal position. Precision is another evaluation metric that is used, which focuses on how precise the model is in predicting positive labels, therefore True Positives and False Positive outcomes from the confusion matrix will be used [27, 28]. Precision is helpful in calculating how many times when a model predicts as positive and counted as correct. This is especially a very crucial thing to take into consideration, if the model were to predict in the case of a Forward Winger, for example, the football player's best position to be a Defender, whereas in reality he is a very good Forward Winger. This is proven to be detrimental in managing a football team's quality and performance. These two positions are very different from each other, where Defenders mostly have a very defensive playstyle and usually play in the backline, stopping opponents from reaching the goal. Whereas Forward Winger plays upfront trying to look for every opportunity to make key passes to their teammates and score assists and even goals. The next evaluation metric that will be used is Recall. Recall is the calculation of percentages of actual positive the model identified as correct [27, 28]. Recall is very useful as an evaluation metric, let's say out of 10 predictions in the case of Defender. The model was able to correctly identify all the players as Defender, hence it will maximize a team performance with this regard because more roles are being filled in a football team and the team's quality is maximized. Lastly, the evaluation metric used is F1-Score. F1-Score is regarded as one of the most important evaluation metrics in a classification task. F1-Score is the harmonic mean of precision and recall and therefore is considered as the sum of predictive performance of the model based on precision and recall values [27, 28]. In other words, it combines both precision and recall values into a single number. F1-Score also takes into account both False Positives and False Negatives, meaning that if Recall and Precision score are high that means F1-Score is also high. The closer the F1-Score is to one then the model is considered good/reliable.

B. Cross-Validation

To make the most out of the evaluation metrics, a method called Cross validation will be performed on the model. Cross validation is a step that is performed right after we train our dataset. If we were to do just testing after training our model, it may not be real-world worthy, because it only represents a small set of all the possible data. Since the main

objective, is that the model we have trained should be adaptable to the real-world and applied without any faults. Cross validation will be used to estimate the performance of a Machine Learning model based on the previously defined evaluation metrics and is especially useful in protecting our model from overfitting [29], [30]. In cross validation, we make a fixed number of folds (or partitions) of the data, run the analysis on each fold, and then get the average of the evaluation metric [29], 30]. The cross validation that will be performed on the data will have a fixed number of 10 folds. Hence, we will perform cross validation on all the classification models (KNN, Naive Bayes, etc) and get the average score of the evaluation metrics after running the partition 10 times (see Fig. 27).

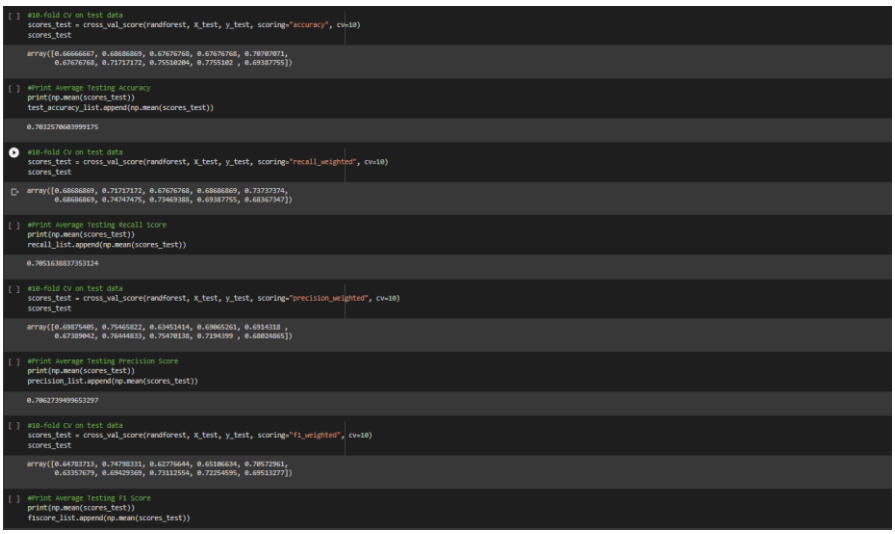


Fig. 27. Performing Cross Validation of the Evaluation-Metrics

C. Summary of All Models Evaluation Metrics and Choosing of Best Model

Based on the evaluation metrics that have been obtained from the Cross Validation step (see Table 5.), we can see that the Naive Bayes model has the lowest score of all evaluation metrics. Therefore, we are crossing off Naive Bayes from the selection of the best model. Following up is the Support Vector Machine model. Although the model may have high evaluation metrics, it is overfitting the model by approximately 9% based on the difference of the testing and training accuracy. The difference of 9% here is a big deal, because even if all Machine Learning models overfit, we want to minimize overfitting to a small margin whilst also having a reliable evaluation of the model. As a consensus, we are crossing the Support Vector Machine model for the candidate of best model. Similarly, due to the high value of difference, which causes overfitting in copious amounts, we are also crossing off KNN from the selection. Now we are just left with both Decision Tree and Random Forest. These two models are very similar, due to the fact that the Random Forest model is just a collection of decision trees. However, we will not be focusing on how similar a model is to each other, comparison will only be done based on the evaluation of the models. From both of the models, we can see that they experience overfitting on the range of 1.4% to 2.0% respectively based on the training accuracy and testing accuracy difference. The difference between these values is minimal, this is why we must also take into account the evaluation metrics of both the models. For example, the evaluation score of the Random Forest model is inferior to the evaluation score of the Decision Tree model. Taking this into account, we will pick the Decision Tree model as the best model used to classify the perfect position of football players based on a team formation. Hence, the evaluation score of the Decision Tree model is as follows:

1. Testing Accuracy: 0.760998
2. Precision Weighted: 0.769015
3. Recall Weighted: 0.767089
4. F1-Score: 0.753640

Table 5. Result of the Evaluation Metrics of Each Algorithms

	Algorithms	Training Accuracy	Testing Accuracy	Precision	Recall	F1-Score
0	KNN	0.774882	0.718367	0.721657	0.718367	0.714414
1	Naive Bayes	0.688869	0.690961	0.694605	0.690961	0.679167
2	Support Vector Machine	0.896569	0.804504	0.805040	0.804504	0.800687
3	Decision Tree	0.780975	0.760998	0.769015	0.767089	0.753640
4	Random Forest	0.717090	0.703257	0.706274	0.705164	0.685706

5. Conclusion

The study that has been conducted utilizes various machine learning algorithms of which are k-nearest neighbor, naïve bayes, decision tree, support vector machine, and random forest. Creation of the model predictor, data collection, data analysis, and data processing were accomplished using the Python programming language, machine learning library (sklearn), and Octoparse for the purpose of data collection through web crawling and web scraping.

As a conclusion, the model predictor that has been designed proves to be competent enough to the extent that it is able to predict and find best playing positions of football players. By using multiple classification algorithms, we were able to conduct trial and error, by finding the best model out of the various models that have been used. This result of this study created using machine learning will prove to be useful in helping football managers, team coaches, and individual football players to discover most suitable playing positions of football based on various attributes that has been defined. Although various confines exists as it has been proven difficult to reach extravagant evaluation metrics or rather exceptional performance due to the fact that in the real world, this result can also be affected by personal decision of individual football players or football managers. Regardless of this fact, we can draw various conclusions from the result of the analysis that has been conducted:

1. The implementation of machine learning can be done to predict and find football player's playing positions with an accuracy and f1-score of 76% and 75% respectively..
2. The classification Algorithm most suitable to create the model predictors is by using the Decision Tree Algorithm with the highest accuracy, precision, recall, and F1-Score after having taken into account overfitting and underfitting.
3. Algorithms that are not recommended for creating the model predictor are K-Nearest Neighbor and Naive Bayes.
4. High values of overfitting are eminent in both the KNN and Support Vector Machine algorithms.
5. In this analysis, data balancing was proven effective because the number of data copied from the minority class by using the oversampling technique was not done repetitively in large numbers.
6. Proving the best implementation of machine learning algorithms and implementation of data processing can only be carried out by trial and error, this is due to the varied characteristics of the data.

As for future improvements, data sets should be prepared with more features and advanced data cleaning should be performed to maximize the performance of the model predictor in finding the best playing positions for football players. Another improvement would be to use data set that is move varied due to the fact that the data used was obtained using web-scraping from only one source, therefore multiple sources should be used to have more variation in the data to mimic real life scenarios better, hence being able to be formally applied in the football industry in the future.

References

[1] N. Razali, A. Mustapha, F. A. Yatim, and R. Ab Aziz, "Predicting Player Position for Talent Identification in Association Football," in IOP Conference Series: Materials Science and Engineering, Aug. 2017, vol. 226, no. 1. doi: 10.1088/1757-899X/226/1/012087.

[2] C. T. Woods, J. Veale, J. Fransen, S. Robertson, and N. F. Collier, "Classification of playing position in elite junior Australian football using technical skill indicators," J Sports Sci, vol. 36, no. 1, pp. 97–103, Jan. 2018, doi: 10.1080/02640414.2017.1282621.

[3] D. Abidin, "A case study on player selection and team formation in football with machine learning," Turkish Journal of Electrical Engineering and Computer Sciences, vol. 29, no. 3, pp. 1672–1691, 2021, doi: 10.3906/elk-2005-27.

[4] J. Pion, V. Segers, J. Stautemas, J. Boone, M. Lenoir, and J. G. Bourgois, "Position-specific performance profiles, using predictive classification models in senior basketball," Int J Sports Sci Coach, vol. 13, no. 6, pp. 1072–1080, Dec. 2018, doi: 10.1177/1747954118765054.

[5] D. R. Anamisa *et al.*, "A selection system for the position ideal of football players based on the AHP and TOPSIS methods," *IOP Conf Ser Mater Sci Eng*, vol. 1125, no. 1, p. 012044, May 2021, doi: 10.1088/1757-899x/1125/1/012044.

[6] M. Bazmara, "A Novel Fuzzy Approach for Determining Best Position of Soccer Players," International Journal of Intelligent Systems and Applications, vol. 6, no. 9, pp. 62–67, Aug. 2014, doi: 10.5815/ijisa.2014.09.08.

[7] RVS Technical Campus, IEEE Electron Devices Society, and Institute of Electrical and Electronics Engineers, Proceedings of the Second International Conference on Electronics, Communication and Aerospace Technology (ICECA 2018) : 29-31, May 2018.

[8] A. Makolo and T. Adeboye, "Credit Card Fraud Detection System Using Machine Learning," *International Journal of Information Technology and Computer Science*, vol. 13, no. 4, pp. 24–37, Aug. 2021, doi: 10.5815/IJITCS.2021.04.03.

[9] Y. F. Alfredo and S. M. Isa, "Football Match Prediction with Tree Based Model Classification," *International Journal of Intelligent Systems and Applications*, vol. 11, no. 7, pp. 20–28, Jul. 2019, doi: 10.5815/ijisa.2019.07.03.

[10] T. Wiradinata, "Folding Bicycle Prospective Buyer Prediction Model," *International Journal of Information Engineering and Electronic Business*, vol. 13, no. 5, pp. 1–8, Oct. 2021, doi: 10.5815/ijieeb.2021.05.01.

[11] N. Sharma, S. Appukutti, U. Garg, J. Mukherjee, and S. Mishra, "Analysis of Student's Academic Performance based on their Time Spent on Extra-Curricular Activities using Machine Learning Techniques," *International Journal of Modern Education and Computer Science*, vol. 15, no. 1, pp. 46–57, Feb. 2023, doi: 10.5815/ijmecs.2023.01.04.
[12] P. Patil, "What is exploratory data analysis?," Medium, 30-May-2022. [Online]. Available: https://towardsdatascience.com/exploratory-data-analysis-8fc1cb20fd15 [Accessed: 19 Jan 2023].
[13] S. R. Sahoo and B. B. Gupta, "Classification of various attacks and their defence mechanism in online social networks: a survey," Enterprise Information Systems, vol. 13, no. 6. Taylor and Francis Ltd., pp. 832–864, Jul. 03, 2019. doi: 10.1080/17517575.2019.1605542.
[14] G. N. Nguyen, N. H. le Viet, M. Elhoseny, K. Shankar, B. B. Gupta, and A. A. A. El-Latif, "Secure blockchain enabled Cyber–physical systems in healthcare using deep belief network with ResNet model," *J Parallel Distrib Comput*, vol. 153, pp. 150–160, Jul. 2021, doi: 10.1016/j.jpdc.2021.03.011.
[15] A. Mishra, B. B. Gupta, D. Peraković, F. José, and G. Peñalvo, "A Survey on Data mining classification approaches," 2021. [Online]. Available: http://ceur-ws.org
[16] S. Christian, "The importance of data preprocessing for machine learning in the e-commerce industry," School of Information Systems, 11-Jul-2022. [Online]. Available: https://sis.binus.ac.id/2022/07/11/the-importance-of-data-preprocessing-for-machine-learning-in-the-e-commerce-industry/.
[17] R. Mohammed, J. Rawashdeh, and M. Abdullah, "Machine Learning with Oversampling and Undersampling Techniques: Overview Study and Experimental Results," in 2020 11th International Conference on Information and Communication Systems, ICICS 2020, Apr. 2020, pp. 243–248. doi: 10.1109/ICICS49469.2020.239556.
[18] D. U. Ozsahin, M. Taiwo Mustapha, A. S. Mubarak, Z. Said Ameen and B. Uzun, "Impact of feature scaling on machine learning models for the diagnosis of diabetes," 2022 International Conference on Artificial Intelligence in Everything (AIE), Lefkosa, Cyprus, 2022, pp. 87-94. doi:10.1109/AIE57029.2022.00024.
[19] M. Butwall, "Data Normalization and standardization: Impacting classification model accuracy," International Journal of Computer Applications, vol. 183, no. 35, pp. 6–9, 2021.
[20] A. Martulandi, "K-nearest neighbors in python + hyperparameters tuning," Medium, 24-Oct-2019. [Online]. Available: https://medium.datadriveninvestor.com/k-nearest-neighbors-in-python-hyperparameters-tuning-716734bc557f. [Accessed: 19-Jan-2023].
[21] A. Sharma, "Naive bayes: Gaussian naive Bayes with Hyperpameter tuning in Python," Analytics Vidhya, 27-Jan-2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/01/gaussian-naive-bayes-with-hyperpameter-tuning/. [Accessed: 19-Jan-2023].
[22] M. B. Fraj, "In depth: Parameter tuning for SVC," Medium, 05-Jan-2018. [Online]. Available: https://medium.com/all-things-ai/in-depth-parameter-tuning-for-svc-758215394769. [Accessed: 19-Jan-2023].
[23] M. Mithrakumar, "How to tune a decision tree?," Medium, 12-Nov-2019. [Online]. Available: https://towardsdatascience.com/how-to-tune-a-decision-tree-f03721801680. [Accessed: 19-Jan-2023].
[24] W. Koehrsen, "Hyperparameter tuning the random forest in python," Medium, 10-Jan-2018. [Online]. Available: https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74. [Accessed: 19-Jan-2023].
[25] M. K. Uçar, M. Nour, H. Sindi, and K. Polat, "The Effect of Training and Testing Process on Machine Learning in Biomedical Datasets," Math Probl Eng, vol. 2020, 2020, doi: 10.1155/2020/2836236.
[26] Bharathi, "Confusion matrix for multi-class classification," Analytics Vidhya, 30-Nov-2022. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/06/confusion-matrix-for-multi-class-classification/. [Accessed: 08-Jan-2023].
[27] S. Gupta, K. Saluja, A. Goyal, A. Vajpayee, and V. Tiwari, "Comparing the performance of machine learning algorithms using estimated accuracy," Measurement: Sensors, vol. 24, Dec. 2022, doi: 10.1016/j.measen.2022.100432.
[28] A. Tasnim, Md. Saiduzzaman, M. A. Rahman, J. Akhter, and A. S. Md. M. Rahaman, "Performance Evaluation of Multiple Classifiers for Predicting Fake News," Journal of Computer and Communications, vol. 10, no. 09, pp. 1–21, 2022, doi: 10.4236/jcc.2022.109001.
[29] D. Berrar, "Cross-validation," in Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics, vol. 1–3, Elsevier, 2018, pp. 542–545. doi: 10.1016/B978-0-12-809633-8.20349-X.
[30] M. de Rooij and W. Weeda, "Cross-Validation: A Method Every Psychologist Should Know," Adv Methods Pract Psychol Sci, vol. 3, no. 2, pp. 248–263, Jun. 2020, doi: 10.1177/2515245919898466.

Authors' Profiles

Kevin Sander Utomo is a passionate and driven individual with a keen interest in technology. His primary focus lies in the fields of Application Development and Artificial Intelligence (AI). He aspires to become a world-class developer and is dedicated to continually learning and improving his skills. He is currently enrolled in the prestigious Apple Developer Academy @ UC, where he is actively honing his abilities. Through this program, he is exposed to cutting-edge technologies and industry best practices, empowering him to excel in his chosen field.With a specific research interest in AR (Augmented Reality) technology, Kevin is actively involved in exploring and developing applications based on his investigations. He recognizes the potential of AR to transform various industries and is committed to pushing the boundaries of this exciting technology.

Trianggoro Wiradinata is the Dean of School of Information Technology at Ciputra University. He received a bachelor's degree in electrical engineering (major in Computing) from Petra Christian University, Indonesia, and a master's degree in computer science from University of New South Wales, Australia and completed his Ph.D. program in Information Technology from Assumption University of Thailand. His main research interests are data science, software engineering, technology adoption, and technology-based entrepreneurship. He was actively involved in many large-scale enterprise software development projects before joined as researcher at Ciputra University, Surabaya, Indonesia. Assoc. Prof. Dr. Wiradinata is currently a. member of Association for Computing Machinery (ACM) and Association for Information Systems Indonesia (AISINDO).

How to cite this paper: Kevin Sander Utomo, Trianggoro Wiradinata, "Optimal Playing Position Prediction in Football Matches: A Machine Learning Approach", International Journal of Information Engineering and Electronic Business(IJIEEB), Vol.15, No.6, pp. 30-47, 2023. DOI:10.5815/ijieeb.2023.06.03