Football Analytics: Harnessing big data to identify players with similar technical attributes

Kelvin Lim Wan (929715)

Research Project COMP90055
Semester 2, 2022

Supervisor:
Tilman Dingler

Industry Sponsor:
Daniel Pelchen, Traits Insights

School of Computing and Information Systems
Faculty of Engineering and Information Technology
The University of Melbourne
Australia
October 2022

CONTENTS

I Introduction 1
- I-A Motivation . . . 1
- I-B Research Questions . . . 1

II Related Works 1

III Methods 2
- III-A Dataset . . . 2
- III-B Exploratory Data Analysis . . . 2
- III-C Data Preprocessing . . . 3
- III-D Experiments . . . 4
  - III-D1 Similarity Measures . . . 4
  - III-D2 Self-supervised Learning . . . 4
  - III-D3 Clustering . . . 5
  - III-D4 Reinforcement Learning . . . 5
- III-E Evaluation Metrics . . . 5

IV Results 6

V Final Framework 7
- V-A Optimisation . . . 7
- V-B Additional Features . . . 7
  - V-B1 Traits Importance . . . 7
  - V-B2 Querying two players . . . 8
  - V-B3 Filters . . . 8
  - V-B4 Comparative visualisations . . . 8
- V-C Implementation . . . 9
- V-D Survey . . . 9

VI Conclusion 9
- VI-A Overview . . . 9
- VI-B Future Work . . . 10

References 10

Appendix A 11

Appendix B 12

List of Figures

1 Distribution of Appearances per Position . . . 2
2 Distribution of Positions among Leagues . . . 2
3 Distribution of Date of Births among Leagues . . . 3
4 Pairwise Plot of Rating Traits per Position . . . 3
5 Cluster Plot for Defensive Midfielders . . . 3
6 Fully-connected Neural Network Architecture . . . 5
7 Skip-connections Neural Network Architecture . . . 5
8 Two Players and Two Traits . . . 7
9 Combining Two Players . . . 8
10 Radar Chart . . . 9
11 Difference Bar Chart . . . 9
12 Find Similar Players . . . 11
13 Compare Players . . . 11

List of Tables

I Traits Aggregation . . . 2
II Experimental Results . . . 7
III Survey Results . . . 12

***Abstract*—For the task of quantifying how similar football players are, we benchmark and offer a thorough comparative analysis of numerous algorithms. Cosine Similarity uses the angle between vector points to measure similarity. As a result, it exhibits the best results overall in our evaluations. We also point out that Pearson Correlation and other similarity metrics, such as Euclidean Similarity, Manhattan Similarity and Adjusted Cosine Similarity, provide unsatisfactory results. Self-supervised Learning approaches, in contrast, yield respectable outcomes but do not support dynamic samples of attributes. The Gaussian Mixture Model and other clustering approaches work well to group similar players together, but they fall short in producing sufficient similarity ratings. We further improve the Cosine Similarity algorithm to make the scores easier to understand and interpret by leveraging the angles of the scores and min-max normalisation. In order to make the system more user-friendly, features such as allowing for trait priority, two-player query, filters, and comparative visualisations are also incorporated. The dataset and resulting algorithm are then published on an interactive dashboard using the Dash framework in Python. The dashboard is then used to obtain one hundred survey ratings of the system. The survey's overall grade is 70%.**

I. Introduction

A. Motivation

In recent years, clubs at every level of the football pyramid have become smarter and more efficient. How? Through the use of data and analytics [2]. Analysts are now recording data from thousands of actions during games and training sessions to help shape pre-match preparations and post-game debriefs, pinpoint transfer targets and develop young talents [21]. This is revolutionary for the transfer market, mainly because the financial gap in football's top leagues is constantly increasing [21]. To compete against the affluent clubs requires creative thinking from less wealthy and successful clubs. For instance, Southampton F.C. has created the 'black box': a live database collecting player metrics from every major league. This has enabled them to acquire players of undervalued talent and sell them for a profit. Virgil van Dijk, Sadio Mane and Luke Shaw are prime examples [2].

Football is an extremely complex sport: it is low-scoring, continuous, time-varying and free-flowing [21]. As a result, data points collected during matches are not as rich and informative as in other sports, which poses a challenge for football data analysts. To illustrate, basketball and Australian football are high-scoring, while tennis and baseball are time-segmented; this allows experts to derive accurate and informative insights to aid executives in their decision-making. Besides, there are aspects of a player such as their professionalism and football IQ that cannot be aggregated into data, and still requires watching live performances or talking to the player [2].

One of the main challenges for a football team is upon the unforeseen departure of a player; when there is no evident replacement in the current squad. After all, teams are trained to play in a specific system, where each player carries out a specific role [2]. The most sensible solution is to find a like-to-like replacement, that is, a new player with the same attributes as the departed player. Naturally, no two players are identical, but by leveraging data and analytics, it is possible to assess players similarity based on their technical profiles. An old-school approach is to send scouts to watch players in live matches across several locations [21]. This process is very time-consuming and not scalable, especially in the rapidly-growing football industry. In addition, there is a risk of unconscious bias in one's opinion of a player, due to cultural differences or a preferred style of play. With big data, clubs are able to analyse a plethora of players all over the world in just a few clicks, while also eliminating the cognitive bias from human scouts.

Further, the football transfer market has seen some remarkable transfers in recent years, with the current transfer fee record being €222 million (A$333 million) for the transfer of Neymar from FC Barcelona to Paris Saint-Germain F.C. in August 2017 [2]. One of the reasons for those excessive prices is that people get obsessed with one player; "they think 'this is the guy, we need to have him, and we are willing to pay over the odds'" explains Anderson [2]. What data can do is help generate options: to find players that are similar to another player, or who would fit into the team in a slightly different way. It allows clubs to walk away from a bad deal.

B. Research Questions

This research project aims to leverage data analytics tools and machine learning techniques to answer the following research questions:

1. Which algorithm is the most accurate and intuitive in outputting similar players and similarity scores given a query player?
2. How can a similarity algorithm be automated for dynamic samples of traits and differing frameworks?

II. Related Works

This section provides an overview of the research and work done in the field of player similarity. In 2019, Pimpaud [17] compared the use of euclidean and cosine distances to represent the similarity between players in football. His work showed that, when players are represented using vectors, it provided an easy framework to implement various distance measures as a similarity index. Further, he proposed the notion of interpolating players in the vector space. Player similarity is not limited to football; it is also necessary in other sports such as basketball, baseball and Australian football [11]. Jackson [11] introduced a novel measure of similarity between players in the Australian Football League (AFL). He looked at the linear transformation of the vector angle between the statistics representing two players, and uses that as the similarity score. He further expands this measure to graphically represent an AFL squad, to identify unique players and match players from other leagues to their AFL equivalents. Both [20] and [23] deployed a platform for football player similarity. The former uses the cosine distance as similarity algorithm and deployment is performed using Streamlit [20]. On the other hand, [23] promotes an industry-proven framework to identify similar players in European football leagues, which

TABLE I
TRAITS AGGREGATION

	Overall Rating			
Composite Measures	Scoring	Creating	Passing	Defending
Derived Metrics	Goals	Assists	Involvement	Aerial
	Shots	Crossing	Accuracy	On ball
	Conversion	Dribbling	Intent	Off ball
	Positioning	Carries	Receiving	Fouls

is used by Leicester City and Bayern Munich. Lastly, we reviewed [10], which approaches the similarity problem quite differently; they implement several clustering algorithms in order to group players. The clustering techniques include K-means clustering, Hierarchical clustering and Density-Based Spatial Clustering of Applications with Noise (DBSCAN). However, those clustering techniques do not output similarity scores between players.

III. METHODS

A. Dataset

Information and statistics about players have been provided by Traits Insights [22]. In the dataset, each season is displayed as a year, which represents the season spanning from August of that year until May of the following year. Each instance has satisfied a minimum number of appearances to be included in the data, so that the traits have low estimation variance and better predictive performance [22]. Each row consists of traits, represented as sixteen derived metrics, which are aggregated into four composite measures, which are further combined into one overall rating. Table I illustrates this aggregation. The trait values are computed using a mixture of Principal Component Analysis (PCA) and further confidential techniques. The mean value for each trait is 2.5 [22]. The dataset is provided in an Excel file, which is separated into individual CSV files for each position and each file is imported in Python as a Pandas dataframe [16].

B. Exploratory Data Analysis

We first explore the data to identify patterns and trends in the dataset. This process allows us to gain a better understanding of the data prior to making assumptions.

The dataframe has 4668 rows and 31 columns. It consists of 1006 centre-backs, 875 full-backs, 443 defensive-midfielders, 784 midfielders, 856 wingers and 704 centre-forwards. There is a column for primary positions whereby some positions are split as such: full-backs have left-backs and right-backs, midfielders have central-midfielders, left-midfielders and right-midfielders, wingers have attacking-midfielders, left-wingers and right-wingers, and centre-forwards have second-strikers as well. There are five leagues: Bundesliga, La Liga, Ligue 1, Premier League and Serie A, three seasons: 2019, 2020, 2021, and 104 nationalities. Other information present in the dataset includes the player names, teams, date of births, number of appearances, average minutes played per appearance and the 21 traits mentioned in Table I. There is no empty, null or duplicate row.

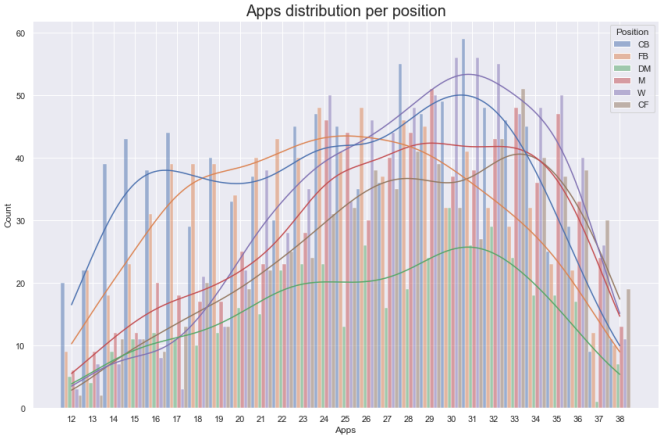


Fig. 1. Distribution of Appearances per Position

Upon examination of the distribution of appearances for each instance, we note that the minimum number of appearances is 12. This represents the threshold mentioned in the previous section. The maximum number of appearances is 38, which is the total number of matches in a league season [24]. From Fig. 1, we observe a slight negative skew in the histogram plot, suggesting that more players have a higher number of appearances. In addition, the mode values for centre-backs, full-backs, defensive-midfielders, midfielders, wingers and centre-forwards are 31, 25, 31, 29, 31 and 33 respectively. Those numbers are coherent; a position such as full-back demands more running than other positions, which demonstrates that full-backs generally play less matches as compared to players in other positions [24].

When analysing the positions among the leagues (Fig. 2), we note that most players are centre-backs and the defensive-midfielder position has the least number of players. There is a somewhat uniform distribution for all leagues in each position. Also, the Serie A consists of the highest number of centre-backs and midfielders, La Liga has the highest number of full-backs, wingers and centre-forwards and the Ligue 1 has the

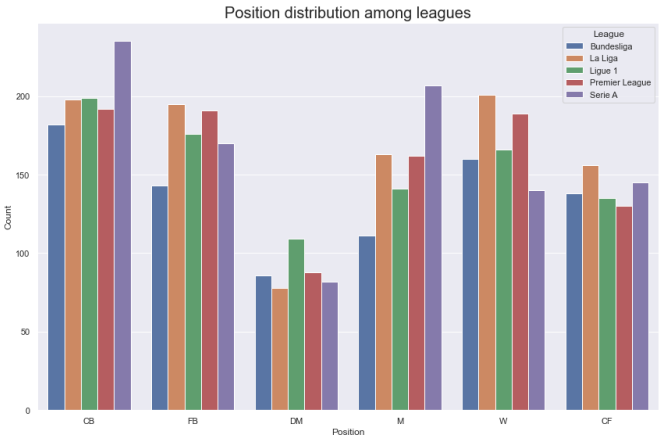


Fig. 2. Distribution of Positions among Leagues

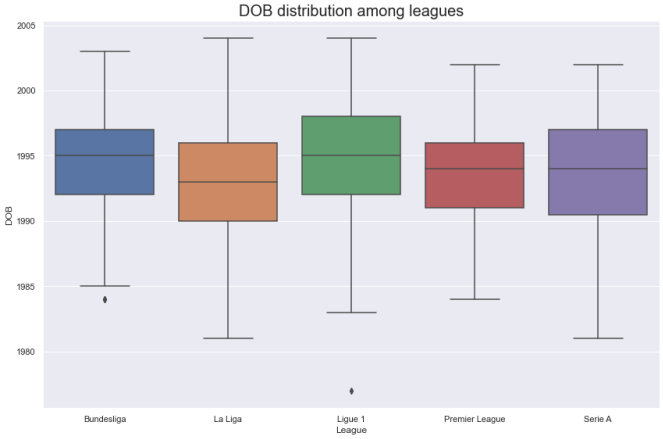


Fig. 3. Distribution of Date of Births among Leagues

highest number of defensive-midfielders.

From Fig. 3, we find that the majority of players were born between 1990 and 1998, which relates to the current age range of 24 to 32. The Ligue 1 has the youngest crop of players, with a median date of birth of 1995 and an upper quantile of 1998, with the Bundesliga a close second. La Liga has the oldest crop of players with a median date of birth of 1993 and a lower quantile of 1990. The Ligue 1 contains the extreme oldest players in the dataset as observed in the outliers of the boxplot (Fig. 3).

A further observation is that players do not change their position or primary position over time. However, when players are transferred or loaned mid-season, another row is added to the dataset, with the new club and new statistics. For instance, Dan Burn was transferred from Brighton to Newcastle United in the middle of the 2021 season [5], so there are two rows for Dan Burn in 2021, one at Brighton with 13 appearances and one at Newcastle United with 16 appearances. Besides, when players do not meet the minimum number of appearances in a season, or players are in a team outside the top-five leagues, they are not included in the dataset. For instance, Felipe Anderson has 25 appearances for West-Ham United in 2019 and 38 appearances for Lazio in 2021, but in 2020, he only made 2 appearances for West-Ham United and was transferred to FC Porto, who play in the Portuguese League, mid-season [8], therefore there is no row for Felipe Anderson in 2020.

Fig. 4 illustrates the relationships between the composite measures and the overall rating. As observed, the pairwise plots are quite scattered, so it is difficult to identify the relationships with the available data. Similarly, the pairwise plots for the derived metrics and composite measures are hard to interpret.

In Fig. 5, we leverage PCA and the K-means algorithm to produce a two-dimensional plot of thirty defensive-midfielders in the dataset. We notice that Joshua Kimmich has a unique style of play since his vector points are clustered far from the other players.

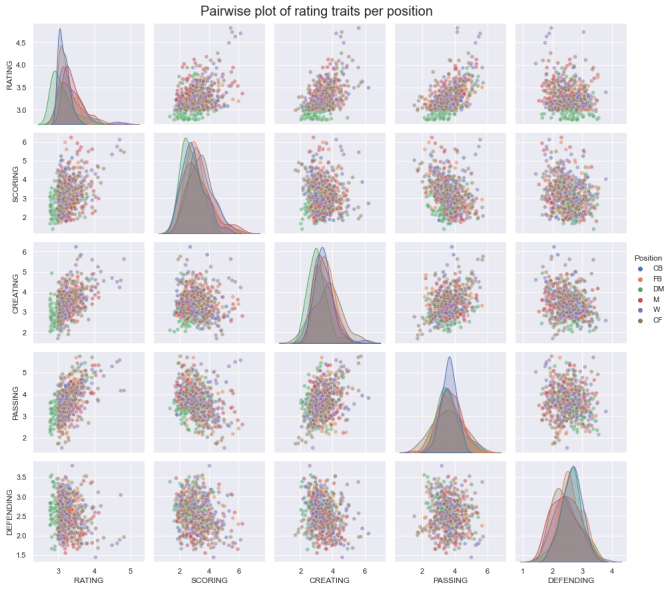


Fig. 4. Pairwise Plot of Rating Traits per Position

C. Data Preprocessing

We preprocess the dataset into a format which is easy and simple to use for future experiments. Firstly, we add a column for individual player names so that we can group instances by players. We also add a column for the total minutes played in a season since clubs are interested in a player's fitness, stamina level and injury proneness, and this metric is a good representation [7]. The players' ages are also computed from their date of births. Lastly, the season column is modified such that all statistics of a player in a particular season are

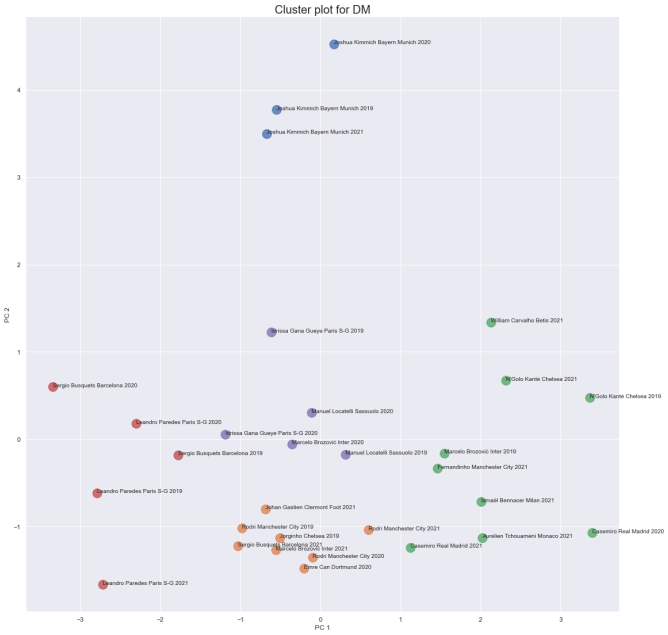


Fig. 5. Cluster Plot for Defensive Midfielders

aggregated into one set of statistics for the entire season. This is mostly relevant for loan spells and mid-season transfers. Further, traits for individual seasons for 2019 and 2020 are not representative of the current players' technical profile. Consequently, we aggregate seasons into periods 2019-2021 (last three seasons), 2020-2021 (last two seasons) and 2021 (last season). The trait values are combined using a weighted framework whereby more weight is placed on the most recent seasons. For 2019-2021, the statistic weights are $\frac{3}{6}$ for the 2021 season, $\frac{2}{6}$ for the 2020 season and $\frac{1}{6}$ for the 2019 season. For 2020-2021, the weights are $\frac{2}{3}$ for the 2021 season and $\frac{1}{3}$ for the 2020 season. The trait values for the 2021 season is unchanged.

D. Experiments

1) Similarity Measures:

- Euclidean Similarity: Euclidean distance is a standard metric for geometrical problems. It is the ordinary displacement between two vector points in a single or multi-dimensional space. It is a popular similarity measure in Natural Language Processing (NLP) and Computer Vision in domains such as document similarity and face recognition respectively [9]. Since Euclidean distance is unbounded, we take its reciprocal to denote similarity, hence the name Euclidean similarity. This measure favours players who have similar overall rating, that is, highly ranked players would have a high similarity score. For two vector points p and q with n dimensions, the Euclidean similarity is computed as:

$$d\left(p,q\right) = \frac{1}{\sqrt{\sum_{i=1}^{n}\left(q_i - p_i\right)^2}}$$

- Manhattan Similarity: Manhattan distance is similar to Euclidean distance, except that it measures the distance between two points measured along axes at right angles, or the sum of the absolute differences of their cartesian coordinates. This measure is based on the grid-like street geography of the New York borough of Manhattan. It is also a popular similarity measure in document similarity and face recognition [9]. Since the distance is unbounded, we again take its reciprocal to denote the Manhattan similarity. For two vector points p and q with n dimensions, the Manhattan similarity is computed as:

$$d\left(p,q\right) = \frac{1}{\sum_{i=1}^{n}\left|q_i - p_i\right|}$$

- Cosine Similarity: Cosine similarity is a measure of similarity between two vector points that looks at the angle between them. Cosine similarity allows us to better capture style rather than pure statistics attributes. It is also widely used in computing the similarity between text documents and also in numerous information retrieval applications [18]. An important property of the Cosine similarity measure is its independence on the vector length [18]. In other words, vectors with the same composition but different sizes would be treated identically. Given two vectors p and q with n dimensions, the Cosine similarity is represented using a dot product and magnitude as:

$$d\left(p,q\right) = \frac{\sum_{i=1}^{n} q_i p_i}{\sqrt{\sum_{i=1}^{n} q_i^2}\sqrt{\sum_{i=1}^{n} p_i^2}}$$

- Adjusted Cosine Similarity: One fundamental difference between Cosine similarity and Adjusted cosine similarity is that the former does not consider the scale of the vectors. The latter offsets this drawback by subtracting the mean from each vector pairs [25].
- Pearson Correlation: The Pearson correlation coefficient is the most common way of measuring a linear correlation. It is represented by a number between -1 and +1 that measures the strength and direction of the relationship between two variables. A value of -1 means a total negative linear correlation, a value of 0 means no correlation and a value of +1 means a total positive linear correlation. It is generally used in statistical settings such as hypothesis testing and identifying the association between two continuous variables [3]. Formally, the Pearson correlation for two vectors p and q with n dimensions is given as:

$$d\left(p,q\right) = \frac{\sum_{i=1}^{n}\left(q_i - \bar{q_i}\right)\left(p_i - \bar{p_i}\right)}{\sqrt{\sum_{i=1}^{n}\left(q_i - \bar{q_i}\right)^2}\sqrt{\sum_{i=1}^{n}\left(p_i - \bar{p_i}\right)^2}}$$

2) Self-supervised Learning: We devise two input-output configurations and two neural network architectures. Min-max normalised cosine similarity scores are used as labels as they are a good representation of similarity [18] and normalising them makes them more interpretive. Research from [14] shows that this trivial task actually provides powerful similarity features for prediction. Accordingly, this results in four methods to output similarity scores. We use a train-test split ratio of 85:15.

- Input-Output 1: The first input-output configuration uses the raw traits for each player as inputs and outputs a similarity score for each player in the queried player's position dataset. The training dataset is a two-dimensional array, with columns as traits for each player and rows representing players. The labels are compiled into an array of cosine similarity scores, of equal length to the number of players in the training dataset.
- Input-Output 2: The second input-output configuration augments the dataset by creating instances for each pair of players in the dataset. As a result, the size of the dataset increases exponentially to r^2, where r is the number of players in a position dataset. The raw traits for the player pairs are concatenated and used as inputs and the output is a similarity score for the two queried players. The training dataset is a two-dimensional array, with columns as the concatenation of the raw traits for the two players, and rows representing pairs of players. The labels are compiled into an array of cosine similarity scores, of length r^2.

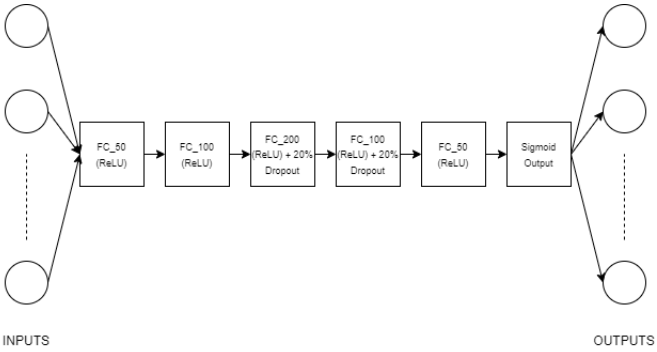


Fig. 6. Fully-connected Neural Network Architecture

- Fully-connected Neural Network: The first neural network architecture is a fully-connected neural network as illustrated in Fig. 6. We leverage the dropout regularisation technique to prevent overfitting and use the ReLU activation function in the dense layers. A sigmoid output is used since we want to output a number between 0 and 1, representing the similarity between players. We use the mean absolute error as the loss function so that the neural network is trained to minimise the absolute distance between the predicted similarity scores and the cosine similarity scores. Lastly, we use Adam as an optimiser as it can handle sparse gradients on noisy problems and provides an efficient and stable procedure in updating neuron weights [1].
- Skip-connections Neural Network: The second neural network architecture is inspired by Residual Neural Networks (ResNets); the inputs are fed into the left block, which feeds inputs into block 1, then the outputs from block 1 and the left block are added together (they have the same dimension) and are fed as inputs into the left block, which feed inputs into block 2 etc. until we reach block 5, then the outputs of the left block are fed into the right fully-connected layer for predictions. This helps dealing with saturation inaccuracy with deep neural networks [15]. The architecture is shown in Fig. 7. We use a ReLU activation function in the hidden layers and a sigmoid activation function in the output layer since we want to output a number between 0 and 1, representing the similarity between players. We use mean absolute error as the loss function, to encourage the model to minimise the absolute distance between the predicted similarity scores and the cosine similarity scores. Lastly, we use Adam as an optimiser as it can handle sparse gradients on noisy problems and provides an efficient and stable procedure in updating neuron weights [1]. We also use dropout to deal with overfitting.

3) Clustering:

- K-means: K-means is an iterative algorithm that partitions the dataset into k predetermined number of clusters with a centroid for each cluster and finds the distance between the data points and centroids [13]. However, since the K-means framework does not contain scores or probabilities, it is discarded for further experiments.
- Gaussian Mixture Model: A Gaussian Mixture Model (GMM) is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. It can be thought as generalising K-means to incorporate information about the covariance structure of the data as well as the centres of the latent Gaussians. GMM implements the expectation-maximisation algorithm for fitting the data points [19]. This framework is suitable for our dataset since traits follow a Gaussian distribution, with a mean of 2.5. For this task, we utilise the probability that a player belongs to another player's cluster as their similarity score.

4) Reinforcement Learning: Reinforcement learning is a machine learning umbrella term which rewards desired behaviours or penalises undesirable ones. A reinforcement learning agent can typically perceive and comprehend its surroundings, act and learn by making mistakes. We considered this technique based on Netflix's recommender system as detailed in [4]. However, since our framework has no reward or environment, this technique is disregarded.

E. Evaluation Metrics

Since we do not have ground-truth labels in our dataset, and because player similarity is highly subjective, we devise the following evaluation metrics to help us understand the algorithms, and assess the similarity scores they produce.

a) Average rank of queried player: When a player is queried in an algorithm, that player should ideally rank first since they are the most similar to themselves. This metric averages the predicted rank of the queried player for all players in the dataset. Therefore, the lower the metric value, the more precise the algorithm is.

b) Standard deviation of traits ratios for top 20: Consider three players: player 1 with statistics $[1, 1, 1]$, player 2 with statistics $[0.5, 0.5, 0.5]$ and player 3 with statistics

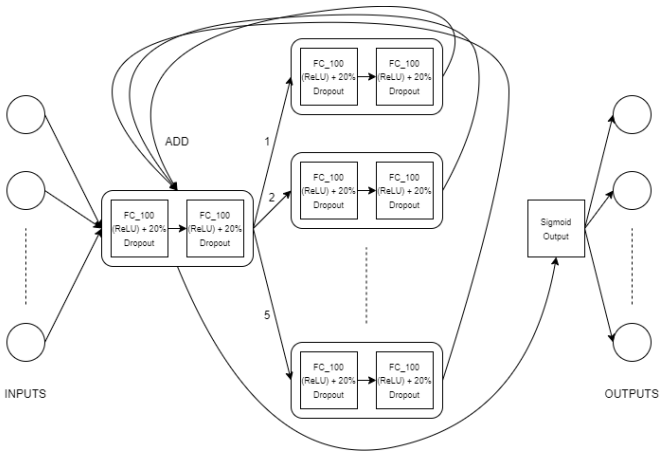


Fig. 7. Skip-connections Neural Network Architecture

$[0.2, 1.5, 0.7]$. By intuition, player 2 is highly similar to player 1 as compared to player 3. This is because the ratios of the player 2 and player 1 statistics are close in terms of value, whereas those between player 3 and player 1 statistics are highly variant. The former standard deviation of traits ratios is 0, and the latter standard deviation of traits ratios is 0.65. This metric averages the standard deviation of traits ratios for the top 20 most similar players. Hence, the lower the value, the more accurate the algorithm is.

c) Scores sparsity: The sparsity of scores is a measure of the overall distribution for an algorithm's predicted scores. It is measured as the average value of the variances of the similarity scores for each queried player. A high scores sparsity is desirable since it means that the algorithm is able to numerically differentiate similarities, and it provides the user with an interpretive score.

d) Range of scores: An intuitive range of scores is important for users to be able to interpret those scores. For instance, a range of scores $[-10, 1000]$ is confusing and unfamiliar, whereas scores between 0% and 100% are easier to decipher.

e) Skewness of scores: The skewness of the distribution of scores defines its asymmetry. A positive skew signifies a right skew, or a distribution where the long tail is on the right side and a negative skew means a left skew [12]. In our case, a small negative skew is desirable since it would mean that the scores taper off more slowly for lower values. As a result, extreme values would be far from the peak on the low side more frequently than the high side.

f) Kurtosis of scores: Kurtosis is a measure of whether the data is heavy-tailed or light-tailed relative to a normal distribution. That is, datasets with a positive kurtosis tend to have outliers and datasets with a negative kurtosis tend to have a lack of outliers [12]. Having a considerable amount of outliers means that the algorithm is able to discern unique cases of player similarity. Hence, a small positive kurtosis is advisable.

g) Dynamicity: Dynamicity refers to whether or not the algorithm allows for dynamic samples of traits and differing frameworks, which essentially answers the second research question. In other words, if a method is able to adjust to variable numbers of traits as well as different aggregation frameworks, it would evaluate to 'Yes', and 'No' otherwise.

IV. Results

Table II presents the results of our experimental methods in accordance with the evaluation metrics previously explained.

Observing the results, we notice that all similarity measures rank first for 'Average rank of queried player', with the third self-supervised learning algorithm a close second. This is because similarity measures, by definition, use computations of distance that are directly related to the traits values for each player; since the statistics of the queried player are exactly equal, the distance is 0, which translates to the highest similarity score, ranking the queried player first. For the 'Standard deviation of traits ratios for top 20', Cosine Similarity ranks first, Euclidean Similarity ranks second and I/O 2 + Fully-Connected (SSL) ranks third. Those ranks denote how accurate the techniques are in measuring the similarity in technical profiles for the football players. Both Adjusted Cosine Similarity and Pearson Correlation are best for 'Scores sparsity', but mainly due to their wide 'Range of scores'. For methods with intuitive 'Range of scores' such as Cosine Similarity and I/O 2 + Fully-Connected (SSL), their 'Scores sparsity' are relatively low. In terms of 'Skewness of scores' and 'Kurtosis of scores', we note that Cosine Similarity and the third self-supervised learning technique are the two best candidates, with both scoring a low negative value for skewness and a low positive kurtosis. This indicates that the similarity scores for those techniques have an adequate distribution, whereby extreme values are on the low side more, and there is a decent number of outliers. Lastly, all methods apart from the self-supervised learning techniques allow for dynamic samples of traits and differing frameworks.

Both Euclidean Similarity and Manhattan Similarity do not perform well, especially in terms of 'Skewness of scores' and 'Kurtosis of scores'. This is because the underlying distance measures used to compute the similarity scores are unbounded; therefore taking their reciprocal results in a highly right-skewed and heavy-tailed distribution. Furthermore, those distance functions measure the 'farness' between two vector points. This means that players with similarly-valued traits overall would have a small distance between one another, disregarding the proportion of those traits, which actually represents the technical profile of a player. This is where Cosine Similarity comes; since it looks at the angle between two vector points rather than how far they are from each other, it captures more the playing style rather than pure statistical attributes. Adjusted Cosine Similarity and Pearson Correlation are both variants of the Cosine Similarity function. The difference is that they normalise the traits values prior to computation. This results in a better scores distribution than those for Euclidean Similarity and Manhattan Similarity. However, they do not fare as well in terms of accuracy, denoted by the 'Standard deviation of traits ratios for top 20' since the normalisation distorts the values excessively. The self-supervised techniques are implemented to trial a deep-learning approach to predict cosine similarity scores by improving their low variance of scores. However, the results in Table II suggests that this computation is an overkill to simply using the cosine similarity function. I/O 2 + Fully-Connected (SSL) is the only self-supervised learning technique that performs adequately on the evaluation metrics. This indicates that the second Input-Output configuration fares better than the first configuration, which makes sense since there are more training and development instances from the data augmentation process. In addition, it suggests that a simple multi-layer perceptron, or fully-connected neural network is better in capturing and identifying relationships between traits values and devised features than the complex skip-connections architecture. Finally, the results for the Gaussian Mixture Model suggests that taking the probabilities that a vector point

TABLE II
EXPERIMENTAL RESULTS

Method	Average rank of queried player	Standard deviation of traits ratios for top 20	Scores sparsity	Range of scores	Skewness of scores	Kurtosis of scores	Dynamicity
Euclidean Similarity	**1.00**	0.371	0.101	[0.06, 10.0]	29.1	921	**Yes**
Manhattan Similarity	**1.00**	0.399	0.0978	[0.02, 10.0]	32.2	1058	**Yes**
Cosine Similarity	**1.00**	**0.360**	0.00135	[0.28, 1.00]	-1.35	**5.57**	**Yes**
Adjusted Cosine Similarity	**1.00**	0.376	**0.132**	**[-0.96, 1.00]**	0.0931	-0.704	**Yes**
Pearson Correlation	**1.00**	0.376	**0.132**	**[-0.96, 1.00]**	0.0931	-0.704	**Yes**
I/O 1 + Fully-Connected (SSL)	47.4	0.446	0.00120	[0.42, 0.99]	-1.67	7.84	No
I/O 1 + Skip-Connections (SSL)	174	0.529	0.00797	[0.54, 0.99]	-2.06	11.7	No
I/O 2 + Fully-Connected (SSL)	1.04	0.373	0.000830	[0.43, 1.00]	**-1.19**	6.33	No
I/O 2 + Skip-Connections (SSL)	541	0.786	0.00	[1.00, 1.00]	0.00	-3.00	No
Gaussian Mixture Model	13.3	0.462	0.0244	[0.00, 1.00]	6.76	48.5	**Yes**

(which represents a player's traits) belongs to another vector point's cluster is too broad to represent player similarity.

Despite having low 'Scores sparsity', it is clear that Cosine Similarity and the third self-supervised learning technique are the two best candidates overall. However, I/O 2 + Fully-Connected (SSL) does not satisfy the 'Dynamicity' criteria, and is only third for 'Standard deviation of traits ratios for top 20'. Therefore, our final method is:

Cosine Similarity

V. FINAL FRAMEWORK

Now that we have the Cosine Similarity method to output similar players and similarity scores accurately, and also allowing for dynamic sample traits and differing frameworks, we aim to optimise this method and add features to make it intuitive and usable.

A. Optimisation

The downside of Cosine Similarity is that it has a low 'Scores sparsity' of 0.00135 and a non-intuitive 'Range of scores' in $[0.28, 1.00]$. [11] proposed a novel measure: to take the angle of the cosine similarity score, for better scaling and representation. The suggested formula is:

$$Score_{Angle} = 1 - \frac{2}{\pi} arccos\left(Score_{Cos}\right)$$

This formula increases the score sparsity by 159%, to 0.00350 and puts the scores in the range $[0.18, 1.00]$.

To further improve those details, we normalise the scores using Min-max normalisation:

$$Score_{Norm} = \frac{Score_{Angle} - min(Score_{Angle})}{1 - min(Score_{Angle})}$$

Here, we use 1 as the maximum value since the similarity score between a query player and themself is 1. The player with the lowest similarity score to another player now has a similarity score of 0; this is sensible as similarity is relative to the players available in the dataset. Now, the range of scores sits in $[0.00, 1.00]$, which is highly intuitive as a similarity score can be represented as a percentage. The sparsity of scores improves a further 137%, to 0.00829.

B. Additional Features

We also implement aspects to the final algorithm so that it is more interpretive and facilitating for the end-users.

1) Traits Importance: For illustrative purposes, assume we only have two players: player 1 and player 2 and two traits: Assists and Goals.

As displayed in Fig. 8, player 1 has traits values of 1 and 0 for Assists and Goals respectively and player 2 has trait values of 1 for both Assists and Goals. Using the angle of cosine similarity, the similarity between player 1 and player 2 is:

$$1 - \frac{2}{\pi} arccos\left(\frac{1 \times 1 + 0 \times 1}{\sqrt{1^2 + 0^2}\sqrt{1^2 + 1^2}}\right) = \mathbf{50\%}$$

However, players have distinct roles within a squad which require them to excel in relevant aspects of the game. Thus, it would make sense to assign different weights to some traits when computing the similarity scores between football players [21]. If users want to place more emphasis on certain traits, we implement a weighted framework within the cosine similarity algorithm, which places more weight on the selected

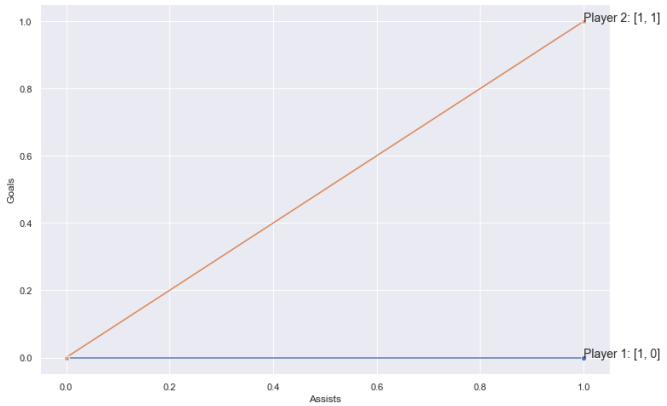


Fig. 8. Two Players and Two Traits

traits. For instance, if a user wants to place more importance on Goals at a ratio of 1:9 for Assists:Goals, the novel framework would return the similarity between players 1 and 2 as:

$$1 - \frac{2}{\pi} arccos\left(\frac{1 \times 1 + \mathbf{9}\,(0 \times 1)}{\sqrt{1^2 + \mathbf{9}\,(0^2)}\sqrt{1^2 + \mathbf{9}\,(1^2)}}\right) = \mathbf{20\%}$$

Visually, it is clear that, if importance is place on the y-axis, which relates to Goals, the two players are less similar. Comparably, we can also place more importance on Assists, which results in a similarity score of:

$$1 - \frac{2}{\pi} arccos\left(\frac{\mathbf{9}\,(1 \times 1) + 0 \times 1}{\sqrt{\mathbf{9}\,(1^2) + 0^2}\sqrt{\mathbf{9}\,(1^2) + 1^2}}\right) = \mathbf{80\%}$$

This framework can be expanded to any size of sample traits, therefore allowing the users to tailor the framework to their own needs and priorities. For an importance vector w, the algorithm can be formulated generally for players p and q as:

$$d\,(p, q) = \frac{\sum_{i=1}^{n} w_i q_i p_i}{\sqrt{\sum_{i=1}^{n} w_i q_i^2}\sqrt{\sum_{i=1}^{n} w_i p_i^2}}$$

2) Querying two players: We can further expand the model to allow users to query two players; the algorithm would output a similarity score between the combination of the two queried players and another player. This is possible since players are represented using vectors of traits. By aggregating their traits values into one player, we are able to general a novel player with aggregated traits from the two query players. The general formula, with two player-vectors p and q, the combined vector r and α being the proportion parameter, is:

$$r = \alpha q + (1 - \alpha)\, p$$

Then, computing the similarity score between the combined-vector player and another player is equivalent to before. For instance, consider the same vector space of Assists and Goals as shown in Fig. 9, the similarity scores between Player 1 and Player 3+4, and Player 2 and Player 3+4 are computed analogously.

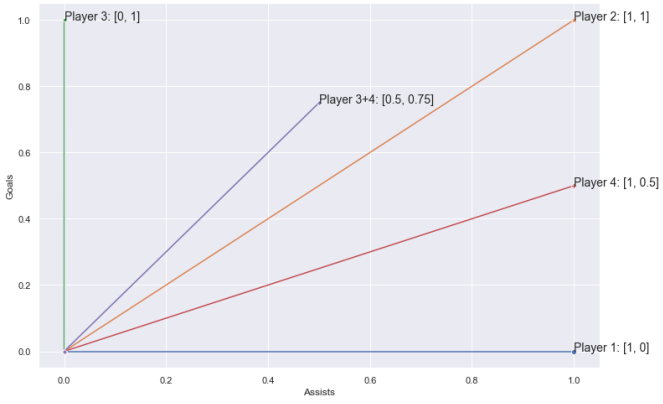


Fig. 9. Combining Two Players

3) Filters: We also allow users to filter the similar players based on the seasons they have played in, the leagues they have played in, their primary positions, their age, the total minutes they have played in a season, and their overall rating.

- *Seasons:* As discussed previously, the player traits have been aggregated into the last three seasons, last two seasons and last season from the years 2019, 2020 and 2021 in data preprocessing. For instance, if a user is looking for players that have good fitness levels and are not injury-prone, they could filter for player performances in the last three seasons [7]. Contrarily, if a user is more risk-averse and is after younger players or 'one-season wonders', filtering for the last season only is justified.
- *Leagues:* Some users may seek players who have experience in specific leagues. For instance, if a user wants a player who has played alongside and against physically strong players, they could filter the Premier League, which is regarded as one of the most physical and intense leagues [7].
- *Primary positions:* In our dataset, some positions contain primary positions that have slightly distinct requirements. For example, the full-back position contains both left-backs and right-backs. If a user wants a player similar to a query left-back, they may want to filter the primary position to left-back and disregard the right-backs, mainly because of their preferred foot [24].
- *Age:* The age filter gives users the option to filter their preference to younger players, perhaps to fill a position for the long-term, or older players, to bring experience to the current squad.
- *Total minutes:* This filter is also a metric for fitness, stamina level and injury proneness. Players who play more minutes in a season generally have better fitness and stamina. However, the total minutes played also depends on the position, since positions such as full-backs and wingers generally play less minutes because of the higher demand of their positions [7].
- *Overall rating:* If a user is tolerant to risk, they may want to take a gamble on players with potential, that is, those who have the desired traits proportion, similar to another player, but have a relatively low overall rating. Moreover, a user may want a player who is already-proven, that is, has a high overall rating.

4) Comparative visualisations: Charts contrasting two players is an effective way to visually evaluate the difference between similar players based on our framework. They give users an extra layer of information, and allows them to make their own judgement on whether a player is suitable or not.

- *Radar chart:* A radar chart is a useful way to display multiple traits and the variation between them. It can be used to compare values for several traits all at once. By using a radar chart, we can show how a player ranks in many different areas in one image, instead of looking at each statistic individually [21]. We can also overlap the radar charts for multiple players to assess their similarities

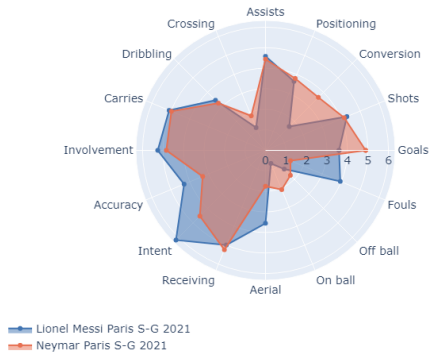


Fig. 10. Radar Chart

and differences, as seen in Fig. 10, which contrasts the traits between Lionel Messi and Neymar in 2021. We clearly see that Lionel Messi was better than Neymar in terms of Accuracy and Intent, and that Neymar was better in Conversion and On ball traits.

- *Difference bar chart:* A difference bar chart is simply a bar chart of the difference between the values for each traits, for each players being compared. In Fig. 11, it is easy to discern that Neymar was better than Lionel Messi in Goals, Conversion, Positioning, Crossing, Receiving, On ball and Off ball by roughly 1.2, 2, 0.1, 0.4, 0.2, 1.4 and 0.6 respectively.

C. Implementation

In order to exhibit the final framework, we created an interactive dashboard using the Dash framework in Python [6], to collate all our findings and features.

Consider the case where a user wants to find players similar to 70% of Lionel Messi and 30% of Neymar at Paris S-G in 2021. They also want to put more importance as weights of 9 and 7 on the Goals and Assists traits respectively. They filter the output players by the seasons 2019-2021, the Bundesliga, La Liga, and the Premier League, and filter the primary

Fig. 11. Difference Bar Chart

positions as attacking-midfielder, left-winger and right-winger. They set the maximum age to 25 , the minimum average minutes played to 900 (equivalent to 10 matches) and the minimum overall rating to 2. As a result, we can observe the top 20 similar players computed by the algorithm, with the top 3 being Rodrygo from Real Madrid, Jadon Sancho from Manchester Utd and Mason Mount from Chelsea, with respective similarity scores of 73.9%, 73.7% and 71.2%.

The radar chart and difference bar chart compares the query players (70% of Lionel Messi and 30% of Neymar at Paris S-G in 2021) and the most similar player, Rodrygo from Real Madrid. From the difference bar chart, we observe that the bars have relatively low values, except for the Involvement and Intent traits. This indicates that the players are quite similar since the differences in their traits have a relatively low variance. We also note that Rodrygo is better at Crossing, On ball and Off ball, but is worse in all other traits.

Refer to screenshots of the dashboard in Fig. 12 and Fig. 13 of Appendix A.

D. Survey

Lastly, we conducted a survey in order to provide a general evaluation for the final framework. Since we do not have labels in our dataset, and football player similarity is highly subjective [7], a survey was conducted where ten people who have been following European football for at least the last five years have been given ten random pairs of players each, along with their corresponding similarity scores as per our framework. They were then asked to rate those similarity scores subjectively out of 10. Consequently, we collected a hundred ratings for our framework, which averaged to an overall rating of:

$$7.02/10$$

See the survey results in Table III in Appendix B.

VI. Conclusion

A. Overview

In summary, we have benchmarked and provided a detailed comparative analysis of various algorithms for the task of football player similarity. From our evaluations, we have found that Cosine Similarity displays the best results overall thanks to its framework of utilising the angle between vector points to define similarity. We also note that other similarity measures: Euclidean Similarity, Manhattan Similarity, Adjusted Cosine Similarity and Pearson Correlation produce subpar results. In contrast, Self-supervised Learning techniques produce decent results but do not allow for dynamic samples of traits. Clustering techniques such as the Gaussian Mixture Model perform well to group similar players together, however they fail to produce adequate similarity scores. We further optimise the Cosine Similarity algorithm by taking the angle of the score and using min-max normalisation in order to make the scores more interpretive and intuitive. Features such as allowing for traits importance, two-player query, filters and comparative

visualisations are also added to improve the system's user-friendliness. Lastly, we leverage the Dash framework in Python to deploy the dataset and final algorithm onto an interactive dashboard, and use it to get one hundred survey ratings of the system. The overall survey rating is 70%.

B. Future Work

In the future, it would be interesting to have additional features representing each player in order to capture most features present in the complex sport of football. Such features could include a player's preferred foot and vectors representing the style of play in the league and team they play in. With time, the data collected will be more rich and informative; we could tweak and automate the weights for each trait in the Cosine Similarity algorithm to account for the traits importance for each position. A potential technique could be to use a neural network with back-propagation, where the weights are learned via deep learning and big data. In addition, with the knowledge of how trait values are derived, one could expand the model to identify players in other positions who have the same technical profile as a query player in a certain position. Finally, it would be compelling to use our framework in conjunction with age curves to display the performance of similar players by age.

ACKNOWLEDGEMENTS

I wish to express my gratitude towards Tilman Dingler from the University of Melbourne, Daniel Pelchen and Alexander Ekkel from Traits Insights, and my fellow colleague Nicholas Young for their guidance and supervision throughout this research project. Without their regular contribution of novel ideas and directions to explore, this thesis would not have been possible.

REFERENCES

[1] Ajani, B. and Bharadwaj, A, 2019. Adaptive Moment Estimator (Adam) Optimizer in ITK v3. The Insight Journal.

[2] Anderson, C. and Sally, D., 2013. The Numbers Game: Why Everything You Know About Football is Wrong. Penguin Books.

[3] Benesty, J., Chen, J., Huang, Y. and Cohen, I., 2009. Pearson correlation coefficient. In Noise reduction in speech processing (pp. 1-4). Springer, Berlin, Heidelberg.

[4] Chong, D., 2021. Deep dive into Netflix's Recommender System, Medium. Towards Data Science. Available at: https://towardsdatascience.com/deep-dive-into-netflixs-recommender-system-341806ae3b48 (Accessed: October 27, 2022).

[5] Dan Burn, 2022. Wikipedia. Wikimedia Foundation. Available at: https://en.wikipedia.org/wiki/Dan_Burn (Accessed: October 27, 2022).

[6] Dash overview, no date. Plotly. Available at: https://plotly.com/dash/ (Accessed: October 27, 2022).

[7] Feess, E. and Muehlheusser, G., 2003. The impact of transfer fees on professional sports: an analysis of the new transfer system for European football. Scandinavian Journal of Economics, 105(1), pp.139-154.

[8] Felipe Anderson, 2022. Wikipedia. Wikimedia Foundation. Available at: https://en.wikipedia.org/wiki/Felipe_Anderson (Accessed: October 27, 2022).

[9] Greche, L. et al., 2017. "Comparison between euclidean and Manhattan distance measure for facial expressions classification," 2017 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS) [Preprint]. Available at: https://doi.org/10.1109/wits.2017.7934618.

[10] Import Data, no date. YouTube. YouTube. Available at: https://www.youtube.com/c/ImportData1 (Accessed: October 27, 2022).

[11] Jackson, Karl., 2016. "Measuring the similarity between players in Australian Football".

[12] Joanes, D.N. and Gill, C.A., 1998. Comparing measures of sample skewness and kurtosis. Journal of the Royal Statistical Society: Series D (The Statistician), 47(1), pp.183-189.

[13] Likas, A., Vlassis, N. and Verbeek, J.J., 2003. The global k-means clustering algorithm. Pattern recognition, 36(2), pp.451-461.

[14] Misra, I. and Maaten, L.V.D., 2020. Self-supervised learning of pretext-invariant representations. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 6707-6717).

[15] Oyedotun, O., Ismaeil, K., and Aouada, D., 2021. Training very deep neural networks: Rethinking the role of skip connections. Neurocomputing, 441(12).

[16] Pandas, no date. pandas. Available at: https://pandas.pydata.org/ (Accessed: October 28, 2022).

[17] Pimpaud, B., 2019. Player similarities & interpolation, Medium. Towards Data Science. Available at: https://towardsdatascience.com/player-similarities-interpolation-aecbf6423c72 (Accessed: October 27, 2022).

[18] Rahutomo, F., Kitasuka, T. and Aritsugi, M., 2012, October. Semantic cosine similarity. In The 7th international student conference on advanced science and technology ICAST (Vol. 4, No. 1, p. 1).

[19] Reynolds, D.A., 2009. Gaussian mixture models. Encyclopedia of biometrics, 741(659-663).

[20] Saini, A.S., 2021. Building a player Recommender Tool, Medium. Analytics Vidhya. Available at: https://medium.com/analytics-vidhya/building-a-player-recommender-tool-666b5892336f (Accessed: October 27, 2022).

[21] Thakkar, P. and Shah, M., 2021. An Assessment of Football Through the Lens of Data Science. Annals of Data Science, 8(4), pp.823-836.

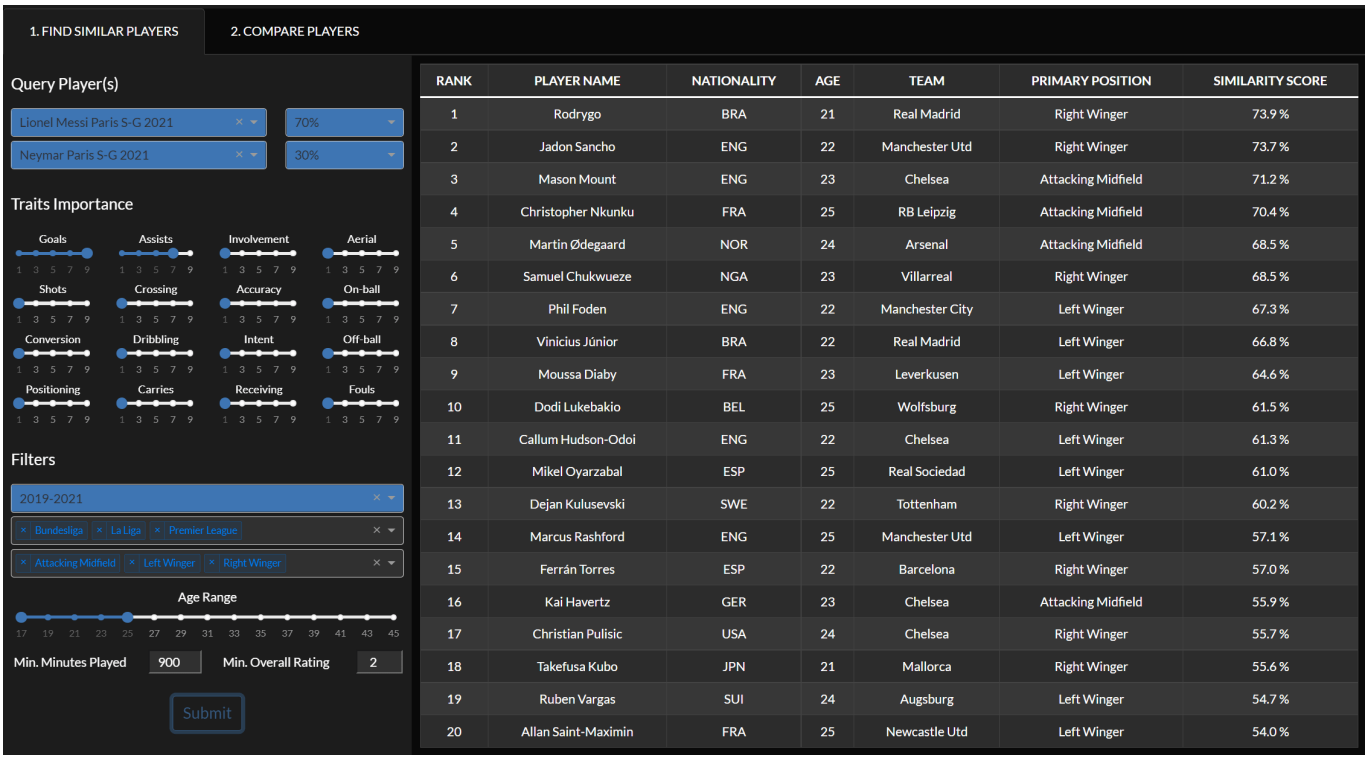
[22] Traits Insights. 2022. [online] Available at: https://www.traitsinsights.com/.

[23] TransferLab, 2022. Analytics FC. Available at: https://analyticsfc.co.uk/transferlab/ (Accessed: October 27, 2022).

[24] Vrooman, J., 2007. "Theory of the beautiful game: The unification of European football," Scottish Journal of Political Economy, 54(3), pp. 314–354. Available at: https://doi.org/10.1111/j.1467-9485.2007.00418.x.

[25] Xia, P., Zhang, L. and Li, F., 2015. Learning similarity with cosine similarity ensemble. Information Sciences, 307, pp.39-52.

APPENDIX A



RANK	PLAYER NAME	NATIONALITY	AGE	TEAM	PRIMARY POSITION	SIMILARITY SCORE
1	Rodrygo	BRA	21	Real Madrid	Right Winger	73.9 %
2	Jadon Sancho	ENG	22	Manchester Utd	Right Winger	73.7 %
3	Mason Mount	ENG	23	Chelsea	Attacking Midfield	71.2 %
4	Christopher Nkunku	FRA	25	RB Leipzig	Attacking Midfield	70.4 %
5	Martin Ødegaard	NOR	24	Arsenal	Attacking Midfield	68.5 %
6	Samuel Chukwueze	NGA	23	Villarreal	Right Winger	68.5 %
7	Phil Foden	ENG	22	Manchester City	Left Winger	67.3 %
8	Vinicius Júnior	BRA	22	Real Madrid	Left Winger	66.8 %
9	Moussa Diaby	FRA	23	Leverkusen	Left Winger	64.6 %
10	Dodi Lukebakio	BEL	25	Wolfsburg	Right Winger	61.5 %
11	Callum Hudson-Odoi	ENG	22	Chelsea	Left Winger	61.3 %
12	Mikel Oyarzabal	ESP	25	Real Sociedad	Left Winger	61.0 %
13	Dejan Kulusevski	SWE	22	Tottenham	Right Winger	60.2 %
14	Marcus Rashford	ENG	25	Manchester Utd	Left Winger	57.1 %
15	Ferrán Torres	ESP	22	Barcelona	Right Winger	57.0 %
16	Kai Havertz	GER	23	Chelsea	Attacking Midfield	55.9 %
17	Christian Pulisic	USA	24	Chelsea	Right Winger	55.7 %
18	Takefusa Kubo	JPN	21	Mallorca	Right Winger	55.6 %
19	Ruben Vargas	SUI	24	Augsburg	Left Winger	54.7 %
20	Allan Saint-Maximin	FRA	25	Newcastle Utd	Left Winger	54.0 %

Fig. 12. Find Similar Players

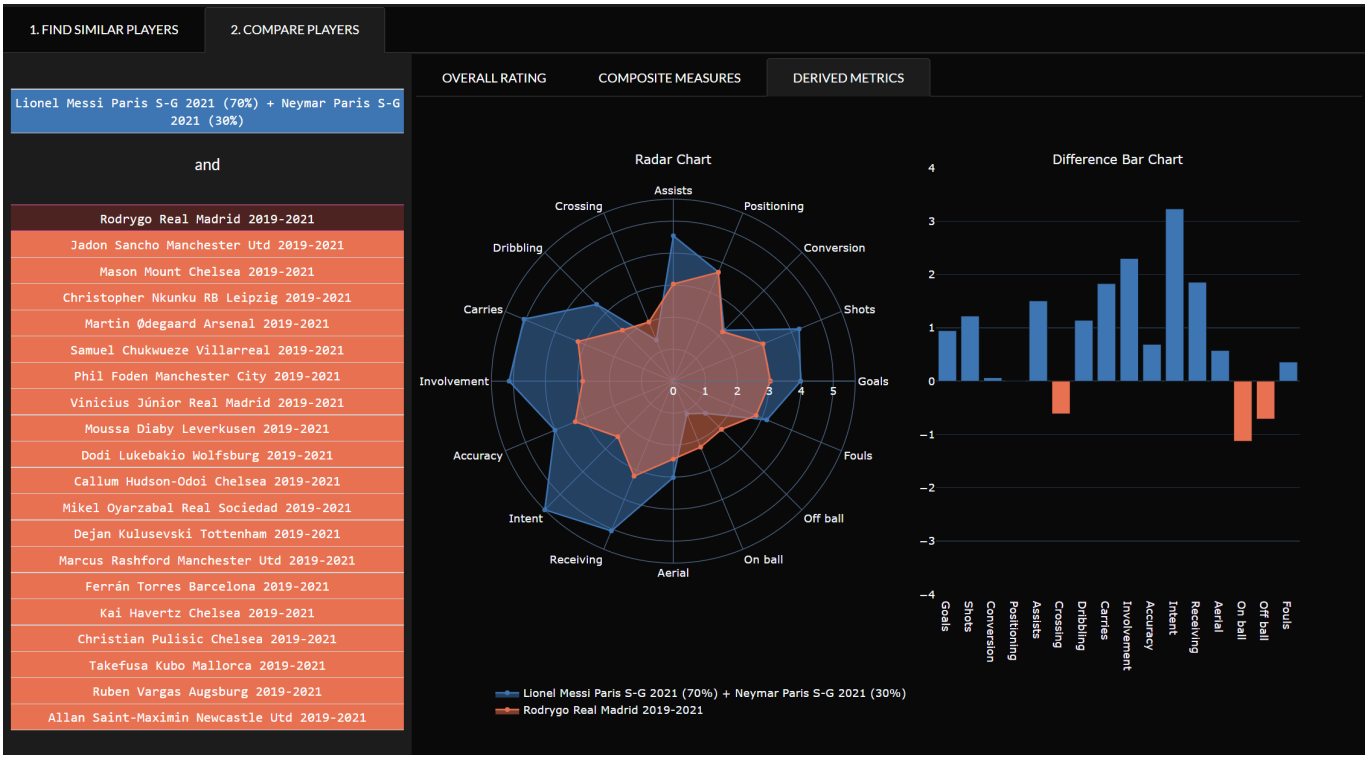


Fig. 13. Compare Players

APPENDIX B

TABLE III
SURVEY RESULTS

Surveyee	Score 1	Score 2	Score 3	Score 4	Score 5	Score 6	Score 7	Score 8	Score 9	Score 10
S1	8	7	4	9	8	8	7	6	6	7
S2	6	9	8	5	6	5	8	10	7	9
S3	7	8	7	9	8	7	8	9	9	6
S4	3	6	8	5	8	6	9	6	7	9
S5	5	7	8	4	4	5	8	6	7	5
S6	10	10	9	7	8	6	7	4	5	4
S7	6	8	5	9	10	8	5	7	6	9
S8	8	10	9	6	3	10	7	4	6	5
S9	8	7	7	7	6	8	7	6	9	7
S10	5	9	6	5	7	8	9	8	7	8