**PLAYER SIMILARITY TOOL FOR RECRUITMENT**

PROJECT REPORT

SATHISH PRASAD VT - 185001137

SREYAS V - 185001162

SRICHARAN PP - 185001163

SRIRAM M - 185001168

PROBLEM STATEMENT

We are aiming to build a player similarity model, which can be used by football teams for scouting, talent recruitment and even in transfers. This will mainly help teams to find players with similar traits and characteristics to top players in the world. This will be a really useful tool for teams mainly in lower tier leagues and those who can’t afford to bring in top players for their team.

OBJECTIVE

The main objective of this task is to find players with a similar style of play to a particular player from the dataset. The similarity can be calculated for all attributes of the player or certain select attributes provided by the user.

As another objective, we have performed interpolation of players of same/different positions and generated players with the combined features of both the selected players. This is achieved by taking the vector representation of one player, then mixing it with the vector of the second player to get the interpolated vector, and then looking for the closest players from the dataset. This feature is really helpful for teams who want to recruit multi- faceted players and also look for talents in similar areas.

DATASET

Our dataset consists of all the players from the top 5 football leagues (Premier League, La Liga, Bundesliga, Serie A and Ligue 1) and also the Dutch and Indian leagues. It contains the data of around 3500 players. The total number of attributes for each player is 105, regardless of his position. For example, an attacker may have zeros in goalkeeping attributes, but still those attributes will be present.

We have split the attributes based on whether they are pertaining to one of the three basic divisions : attacking, defence and passing.

OUR APPROACH & METHODOLOGY

Our basic approach to solving this problem was to adapt unsupervised learning. Here we take each player’s vector representation according to their attributes. We decided to relate the similar vectors based on cosine similarity rather than the conventional Euclidean distance. The extraction of similar players is performed using K-means clustering.

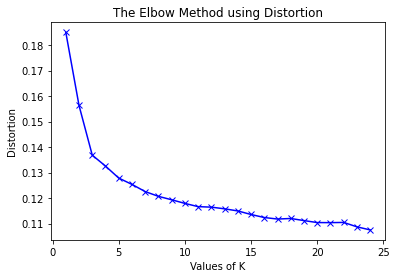
ELBOW METHOD

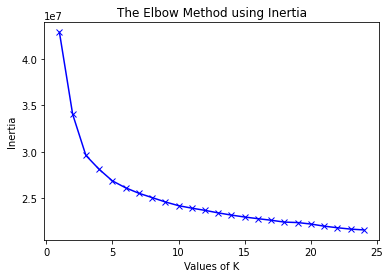
The elbow method is used to determine the number of clusters needed for effective performance of the model. The intuition is that increasing the number of clusters will naturally improve the fit (since there are more parameters to use) but that at some point this can lead to overfitting. The elbow model reflects this. For example, given data that actually consist of k labelled groups, clustering with more than k clusters will "explain" more of the variation (since it can use smaller, tighter clusters), but this is over-fitting, since it is subdividing the labelled groups into multiple clusters. The idea is that the K clusters will add much information, since the data actually consist of that many groups (so these clusters are necessary), but once the number of clusters exceeds the actual number of groups in the data, the added information will drop sharply, because it is just subdividing the actual groups. Assuming this happens, there will be a sharp elbow in the graph of explained variation versus clusters: increasing rapidly up to k (under-fitting region), and then increasing slowly after k (over-fitting region).

We now define the following: -

DISTORTION (Variance) : It is calculated as the average of the cosine distances from the cluster centers of the respective clusters.

INERTIA : It is the sum of distances of samples to their closest cluster centers.





From the above visualization, we can see that the optimal number of clusters should be around 4-5. But visualizing the data alone cannot always give the right answer.

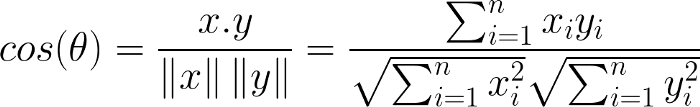
We iterated the values of k from 1 to 25 and calculate the values of distortions for each value of k and calculate the distortion and inertia for each value of k in the given range.

**SIMILARITY SCORES**

COSINE SIMILARITY

Cosine similarity measures the similarity between two vectors by calculating the cosine of the angle between the two vectors.

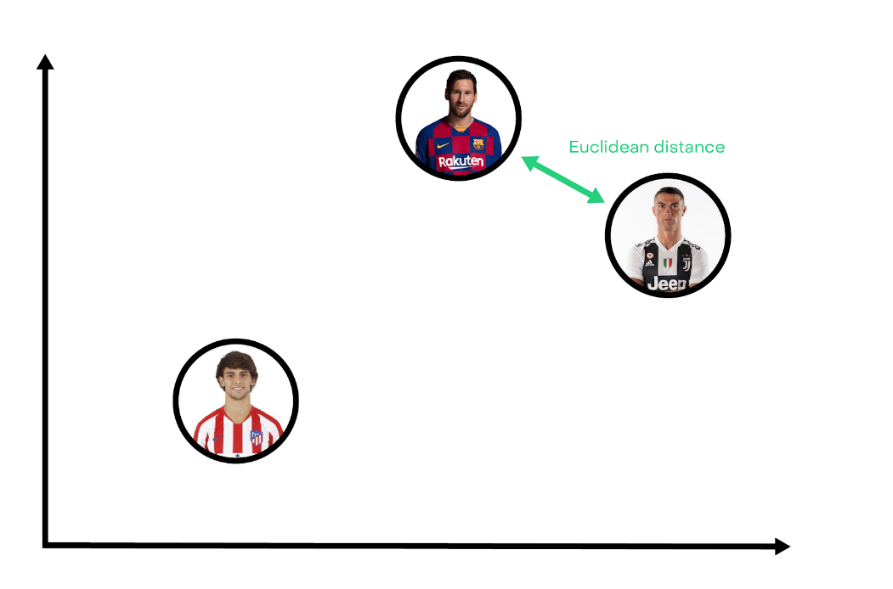
Cosine similarity is one of the most widely used and powerful similarity measures in Data Science. It is used in multiple applications such as finding similar documents in NLP, information retrieval, finding similar sequences to DNA in bioinformatics, detecting plagiarism and many more.

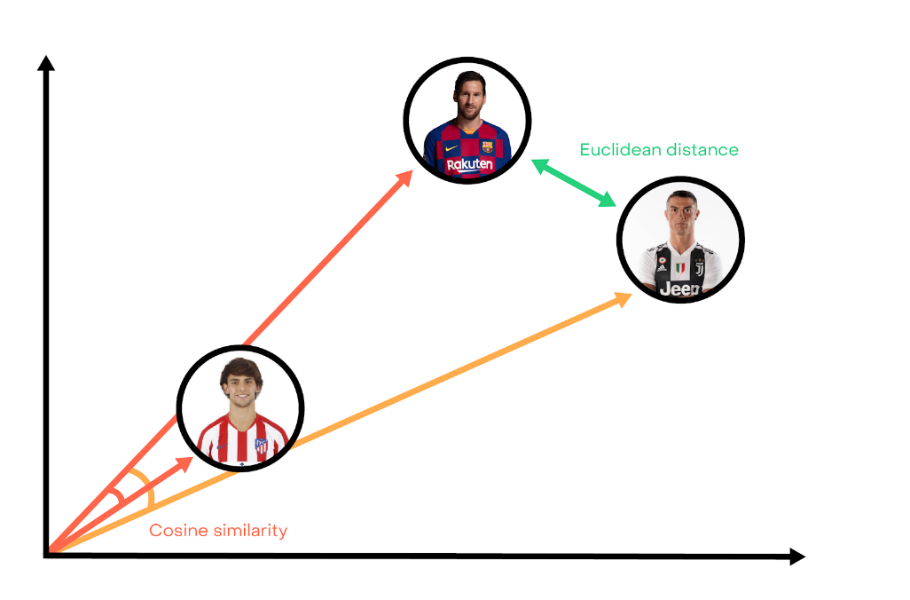
Cosine similarity is calculated as follows :

The cosine of 0° is 1, and it is less than 1 for any angle in the interval (0,π] radians.

COSINE vs EUCLIDEAN

Cosine similarity is a measure of similarity between two non-zero vectors in the feature space that measures the cosine of the angle between them. It is thus a judgment of orientation and not magnitude: two vectors with the same orientation have a cosine similarity of 1, two vectors oriented at 90° relative to each other have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude.

****Cosine similarity allows us to better capture “style” rather than pure “statistics” attributes. Cosine similarity is generally used as a metric for measuring distance when the magnitude of the vectors does not matter.

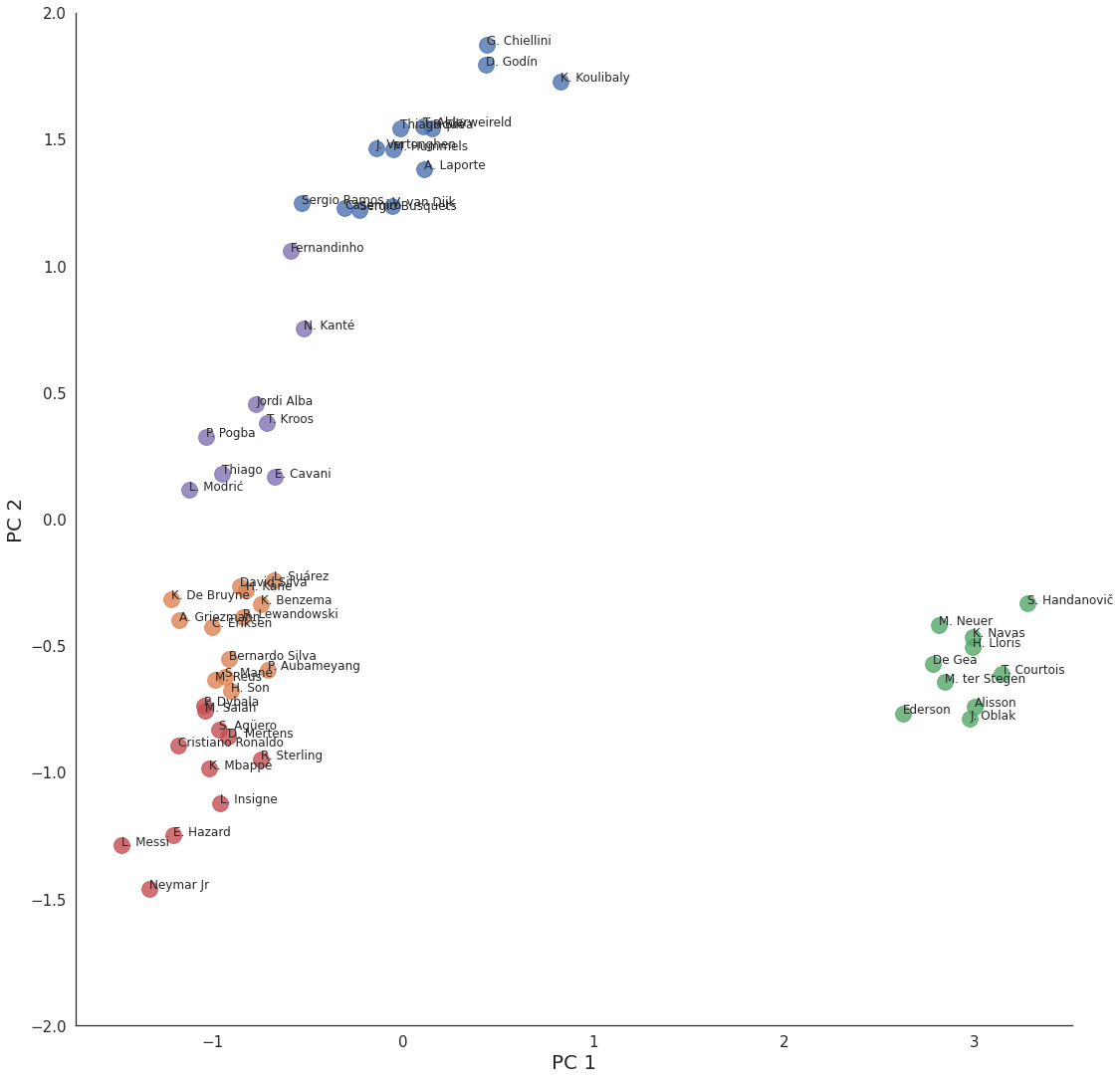


Euclidean distance is like using a ruler to measure the distance. However, choosing this distance is probably not the best option. For example, Ronaldo is close to Messi because they have high ratings in shoots, speed or dribbles. But a young player like Joao Felix who has the same profile to Messi will be further away because his attributes are weaker, but in the same proportion.

CLUSTERING

Clustering is one of the unsupervised learning techniques (PCA is another one).

We can cluster (or group) observations into the same subgroups so that observations within a subgroup are quite similar to each other and observations in different subgroups are quite different from each other.



**K-Means clustering** is one of the clustering algorithms.

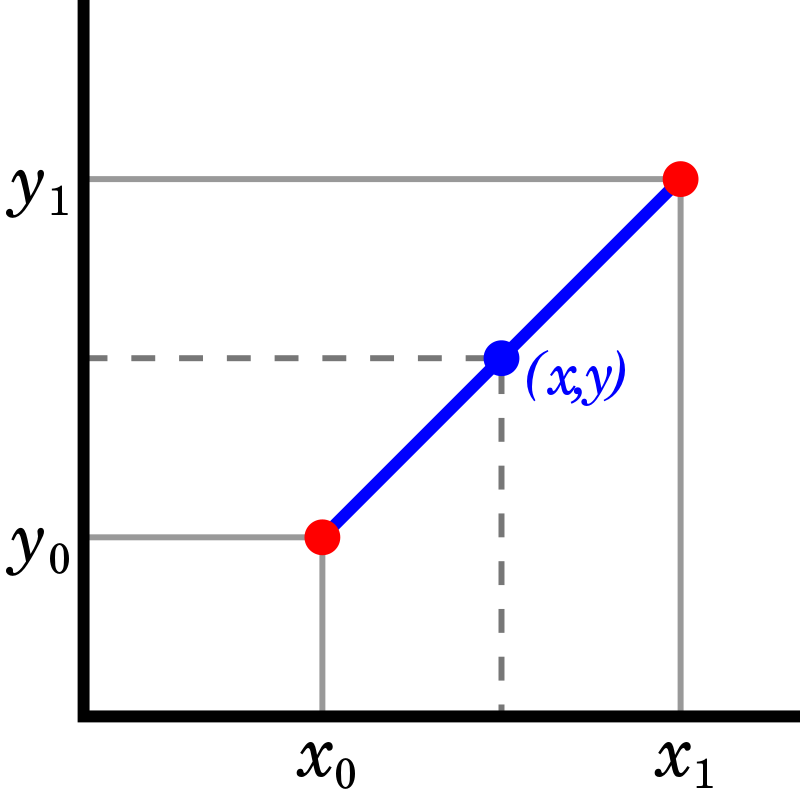
The basic algorithm :

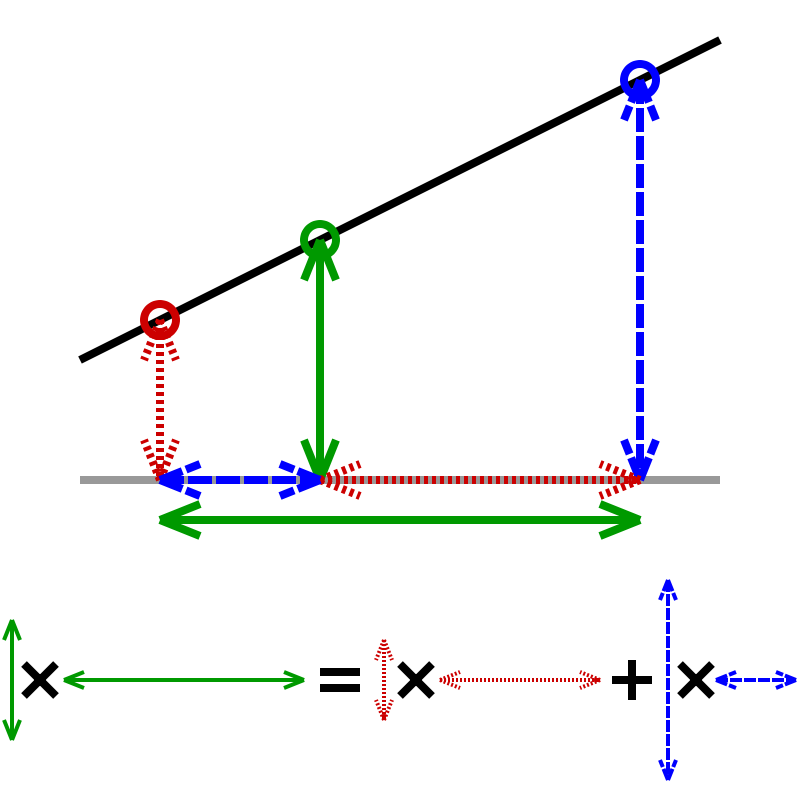
* Specify K-clusters and initialize random centroids
* Iterate until the cluster assignments stop changing. The method assigns each observation to exactly one of the K clusters
* For each K cluster, calculate the cluster mean
* Proceed through the list of observations and assign an observation to the cluster whose mean is nearest.
* The goal is to form the clusters in a way that the observations within the same cluster are as similar as possible.

VECTOR INTERPOLATION

Many a times we come across use cases where the we need to find players with attributes of two or more players, to solve this problem we use vector interpolation. In mathematics, linear interpolation is a method of curve fitting using linear polynomials to construct new data points within the range of a discrete set of known data points. This is the general formula for vector interpolation.







The resultant green vector is a new point which captures both attributes of both the blue and red vectors based on the value of alpha, which is a tuneable parameter.

**RESULTS**

1. PLAYER SIMILARITY

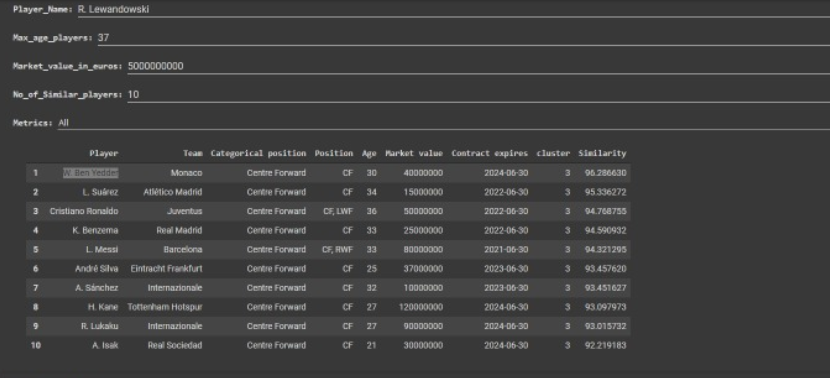
Chosen player :

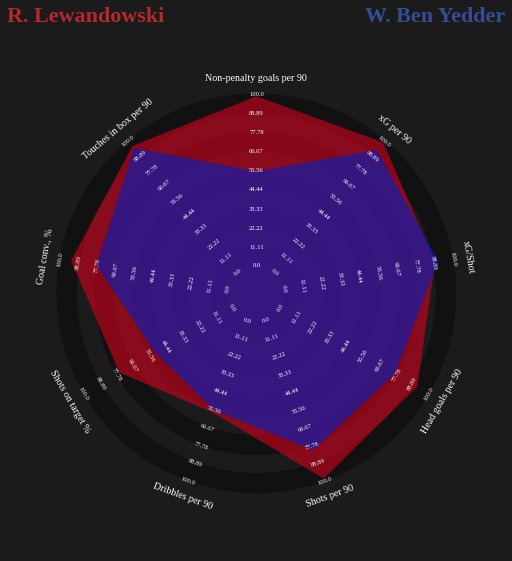
Name: Robert Lewandowski

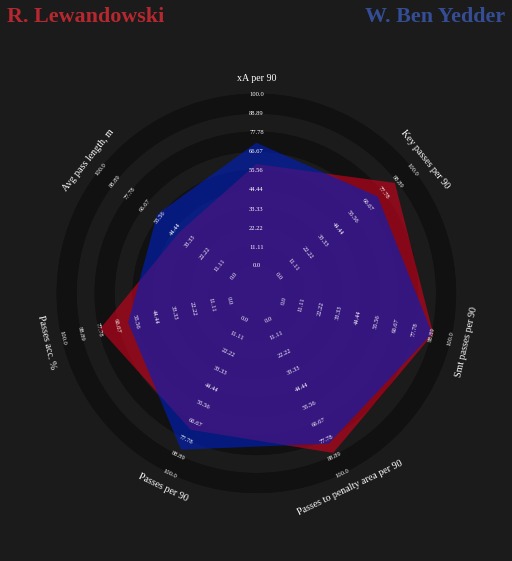
Position: Striker

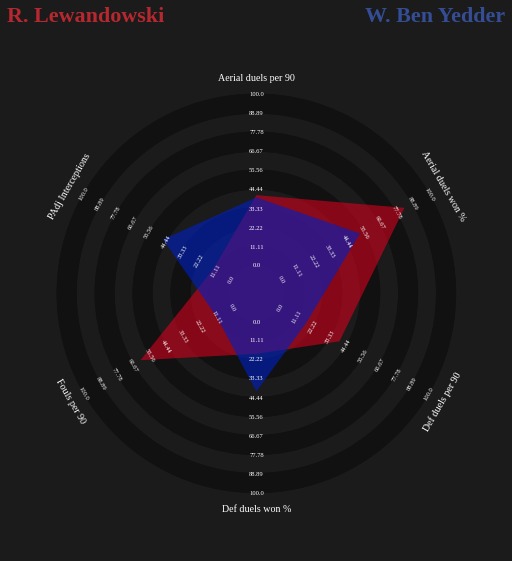
Nation: Poland

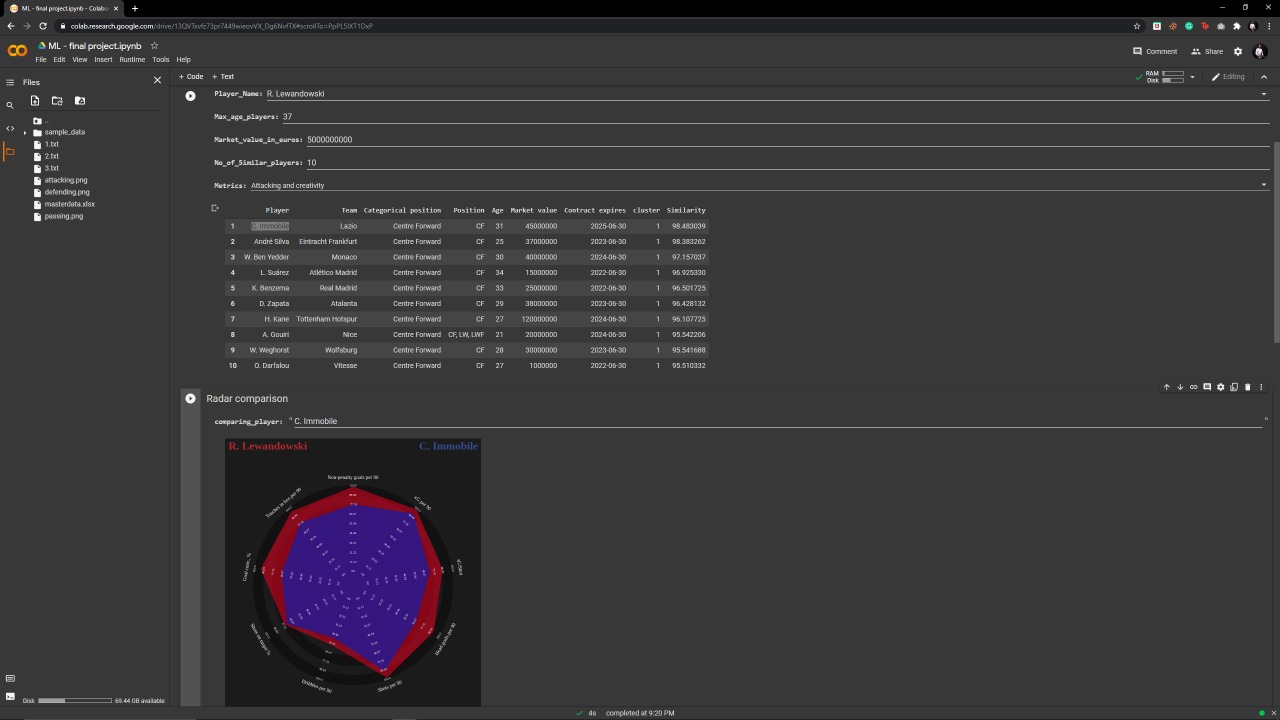
Club: Bayern Munich

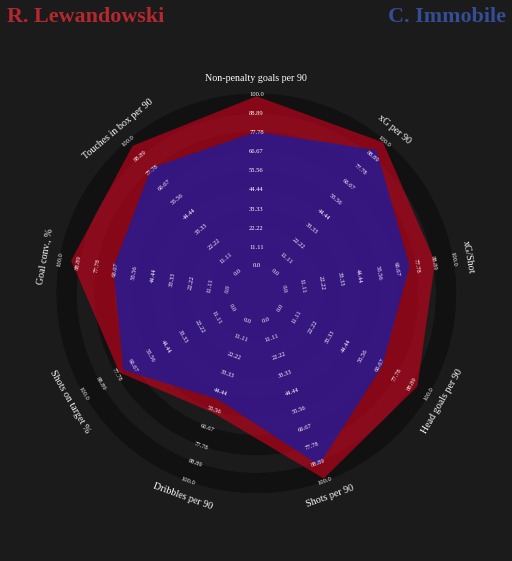


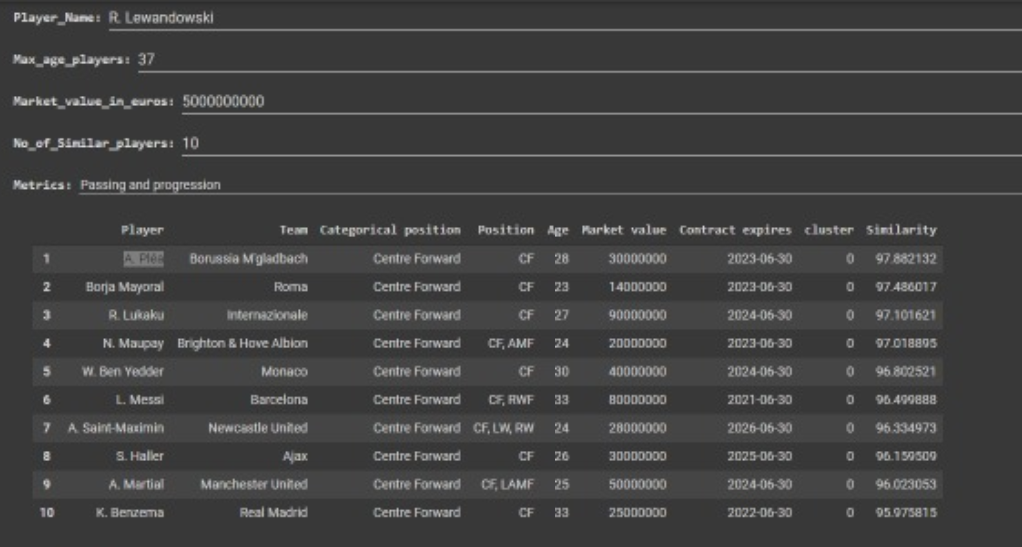


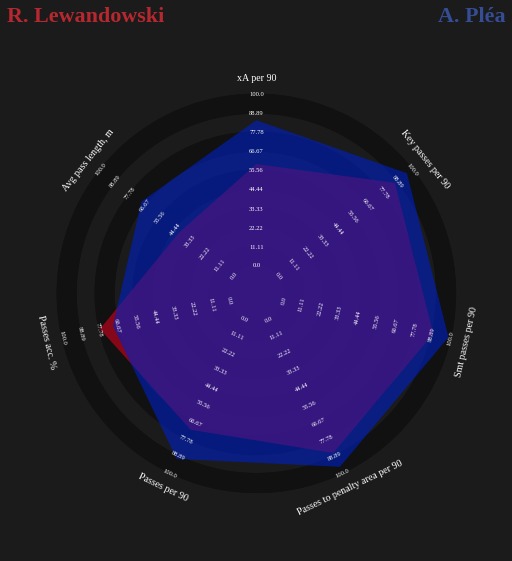


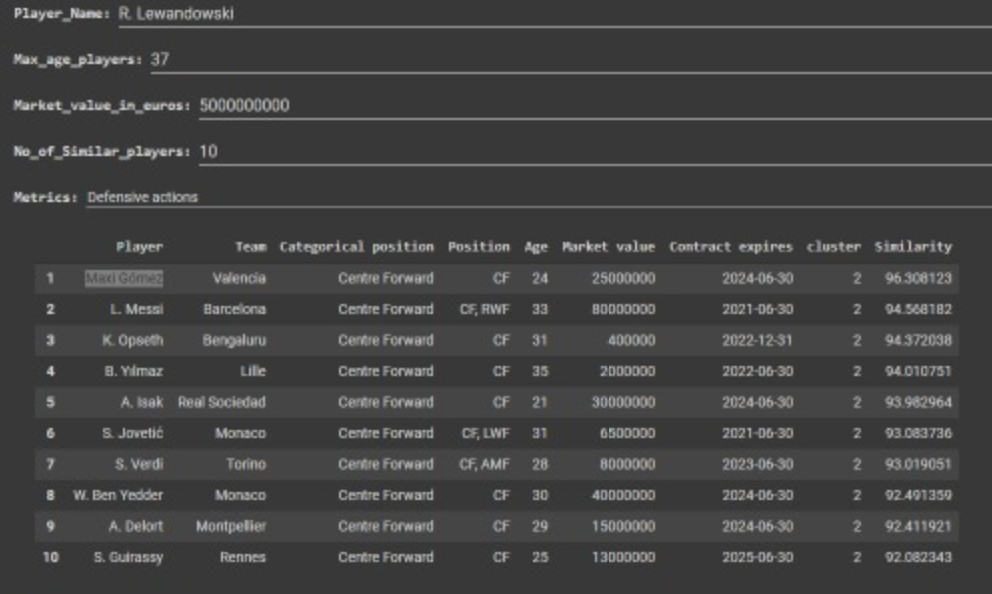


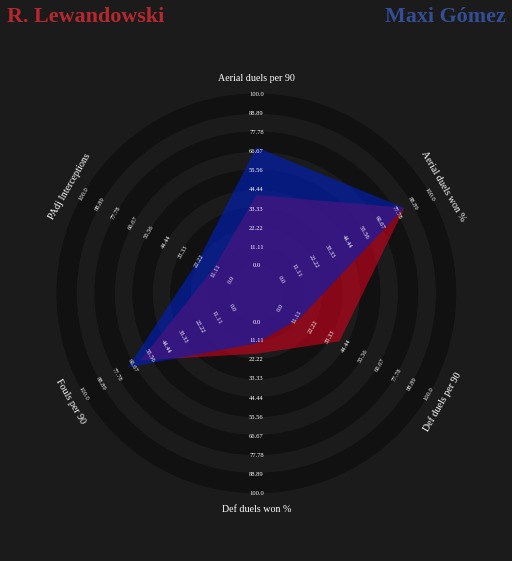
Attacking Attributes



Passing Attributes



Defensive Attributes



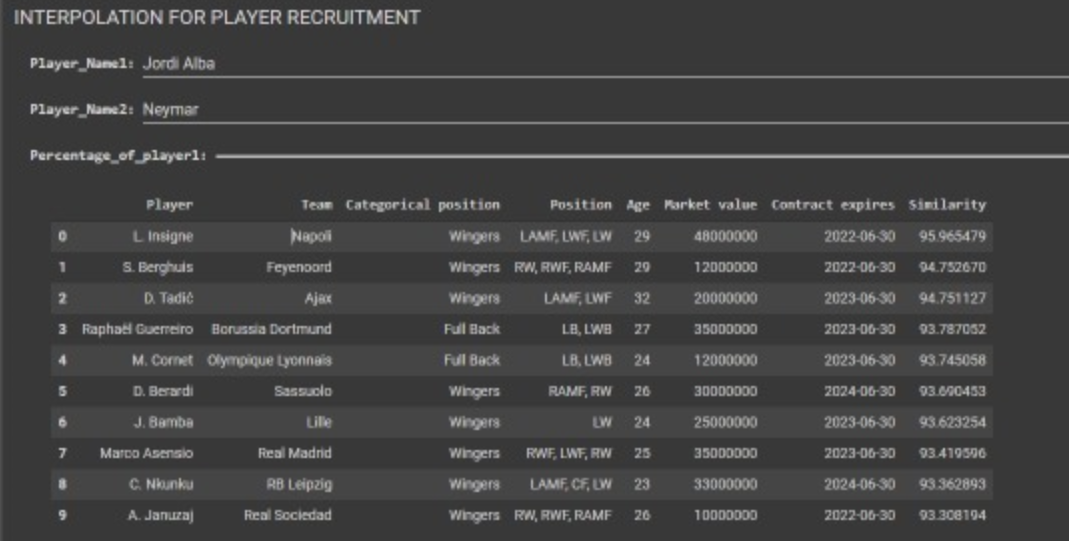
1. INTERPOLATION

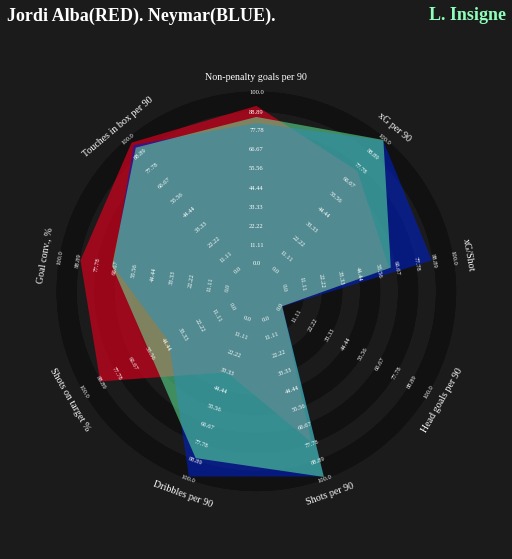
Chosen Players : Jordi Alba Neymar

Left Back Forward

Spain Brazil

FC Barcelona PSG

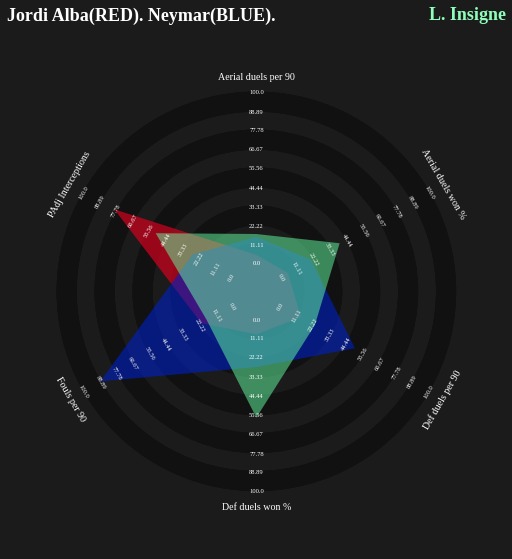


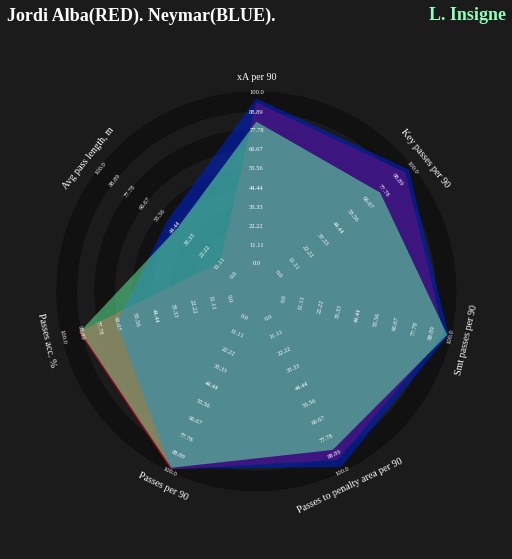


Lorenzo Ingsine (95.9%)

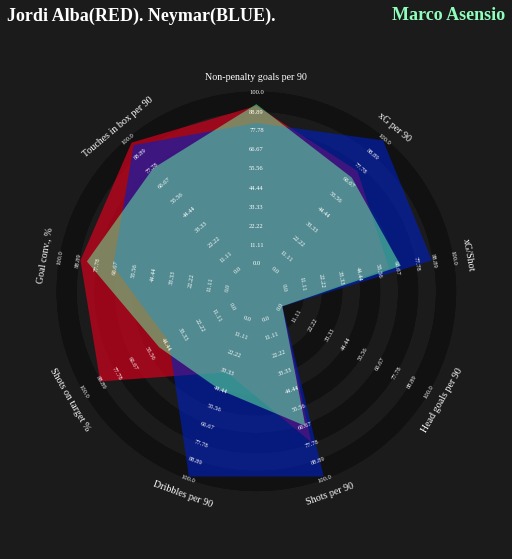
Attacking Attributes

Defensive Attributes

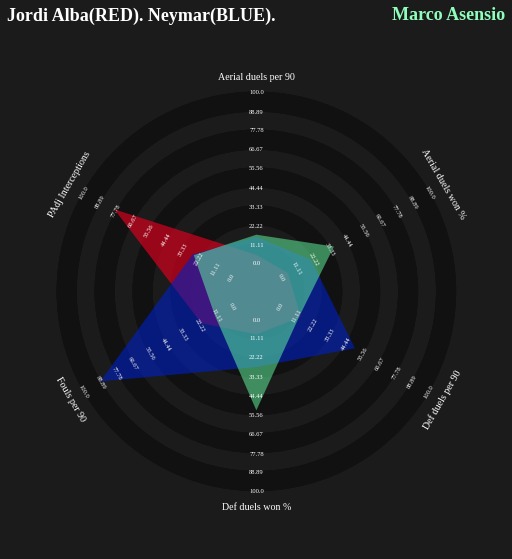


Passing Attributes

Marco Asensio (93.4%)



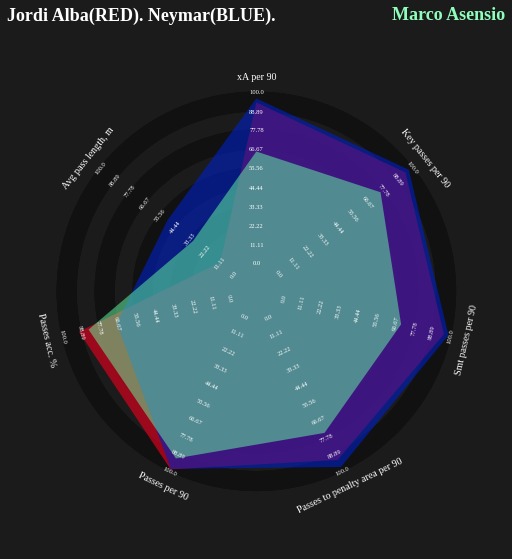
Attacking Attributes



Defensive

Attributes

Passing Attributes



OUTCOMES

* We were able to achieve player similarity with more than a rate of 95%
* Both the similarity and interpolation models were tested for different outcomes
* Finding youngsters/ developing players with attributes similar to top players using cosine similarity and K-means clustering was also successful.

FURTHER IMPROVEMENTS / SCOPE :

* The model can be constructed for larger datasets with distributed / parallel programming architectures.
* Further methods of dimensionality reductions methods like PCA, autoencoders etc. can be used.
* To improve dynamic requests, methods of data compression can be employed.
* To improve the run-time performance the model can also be modelled to run with cuda and other GPU frameworks.