THE BEST REPLACEMENT FOR VIRGIL VAN DIJK IN THE SEASON 2022-2023 (Player Recommender System using Python)

I. Motivation:

- Season 2022-2023 has been a horrific drop in form of arguably the best Centre defender: Virgil van Dijk, which lead to a decline in the strength of Liverpool F.C.'s defensive system. The cause may mainly lead back to Virgi van Dijkl's hamstring problem. Liverpool F.C.'s solution to the problem is using young academy players to replace Virgil van Dijk, but they are all underqualified to compete in an intense league like EPL. This project will use a python recommender system to find Virgil Van Dijk replacements.

Let's define the KPIs of a Central defender:

- Defensive actions:

Van Dijk has done a remarkable job at stopping opponents' attacks by tackles, interceptions, clearances, and stopping shots. These metrics can score these actions:

  + Number of dribblers tackled per 90: Tkl/90
  + Win percentage in tackles: TklWon%
  + Percentage of getting dribbled through: TklDriPast
  + Number of times blocking a shot by standing in its path per90: BlkSh/90
  + Number of interceptions per 90: Int/90
  + Clearances per 90: Clr/90

- Ariael action:

  With the height of 1m94 and sensibly position choosing, Van Dijk has proven to be a great aerial warrior both on defense and offense. These actions can be scored by these metrics:

  + Total of aerial duels per 90: AerDuels/90
  + Percentage of aerial duels won: AerWon%

- Passing action:

Van Dijk is a great deep-lying playmaker, his passes and crosses play a vital role in Liverpool F.C. success. These actions can be scored by these metrics:

  + Passes attempted per 90: PasTotAtt/90
  + Pass completion percentage: PasTotCmp%
  + Long passes attempted (>30 yards) per 90: PasLonAtt/90
  + Long pass completion (>30 yards) percentage: PasLonCmp%

Some notes when processing the data:

- The dataset is taken from Kaggle: 2022-2023 Football Player Stats | Kaggle, This dataset contains 2022-2023 football player stats per 90 minutes. Only players from the Premier League, Ligue 1, Bundesliga, Serie A, and La Liga are listed.
- The dataset has 2689 players, including players who are not central defenders.
- All the variables are scaled to minimize the deviation.
  + To avoid differences in ratio, we scale the data using the MinMaxScaler function, where the data is transformed into values from 0 to 1.
- The similarities between players can be found by using a “ruler” to find who is the player closest to Virgil van Dijk on the KPIs performance scale.
  + We choose to use Euclidean distance formula because of its “ruler” property.

II. Project Processing step by step:

- Importing Libraries: First, the necessary libraries are imported. These include NumPy, pandas, sci-kit-learn, and scipy.
- Loading Data: The football player stats data is loaded using the read_csv function from the pandas library. The file is in a CSV format and the file path is specified as an argument.
- Data Cleaning and Subset Selection: Data cleaning is performed to remove any missing values and convert the KPI columns to a numeric data type. A subset of the dataset is also selected to limit the number of players to 1048576. This is done using the iloc function from pandas.
- Normalizing Data: The KPI data is normalized using the MinMaxScaler function from scikit-learn. This is done to ensure that all KPIs are on the

same scale, where the data is transformed into values from 0 to 1 and can be compared fairly..

- Computing Distance Metrics: A function to compute the Euclidean distance between two players is defined using the euclidean function from scipy. This is the distance metric that is used to determine the similarity between two players.
- Finding Similar Players: The player who is most similar to Virgil van Dijk is determined by computing the Euclidean distance between the target player's KPI data and the KPI data of all other players. The 10 players with the smallest distances are selected as the most similar players. This is done using the argsort function from numpy.
- Creating a DataFrame of Similar Players (Similar Players under 25): A DataFrame is created to display the most similar players, along with their positions, squad, age, and KPI values. The DataFrame is created using the iloc function from pandas to select the rows for the most similar players.

III. Data Analysis:

In [1]:
```
# Import Libraries
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from scipy.spatial.distance import euclidean
import seaborn as sns
```

Load the Dataset

In [2]:
```
df = pd.read_csv('Data/2022-2023_Football_Players_stats.csv')
```

In [3]:
```
# Display the data
df.head()
```

Out[3]:

	Rk	Player	Nation	Pos	Squad	Comp	Age	Born	MP	Starts	...	Crs	TklW	PKwon	PKcon	OG	Recov	AerWon/90	AerLost	AerDuels/90	Ae
0	1	Brenden Aaronson	USA	MFFW	Leeds United	Premier League	22	2000	20	19	...	2.54	0.51	0.0	0.0	0.00	4.86	0.34	1.19	1.53	
1	2	Yunis Abdelhamid	MAR	DF	Reims	Ligue 1	35	1987	22	22	...	0.18	1.59	0.0	0.0	0.00	6.64	2.18	1.23	3.41	
2	3	Himad Abdelli	FRA	MFFW	Angers	Ligue 1	23	1999	14	8	...	1.05	1.40	0.0	0.0	0.00	8.14	0.93	1.05	1.98	
3	4	Salis Abdul Samed	GHA	MF	Lens	Ligue 1	22	2000	20	20	...	0.35	0.80	0.0	0.0	0.05	6.60	0.50	0.50	1.00	
4	5	Laurent Abergel	FRA	MF	Lorient	Ligue 1	30	1993	15	15	...	0.23	2.02	0.0	0.0	0.00	6.51	0.31	0.39	0.70	

5 rows × 133 columns

In [4]:
```
# Descriptive statistics
df.describe()
```

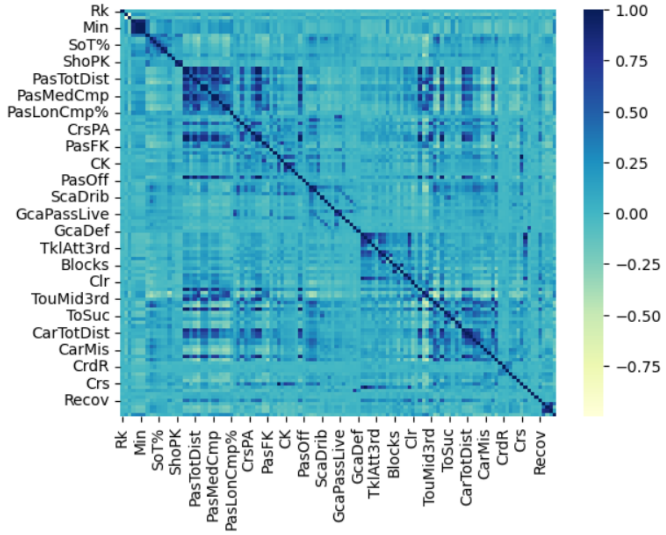
Out[4]:

	Rk	Age	Born	MP	Starts	Min	90s	Goals	Shots	SoT	...	Crs	26
count	2689.000000	2689.000000	2689.000000	2689.000000	2689.000000	2689.000000	2689.000000	2689.000000	2689.000000	2689.000000	...	2689.000000	
mean	1345.000000	26.011157	1996.155820	11.833023	8.476013	760.451097	8.450465	1.027520	1.245787	0.411261	...	1.661636	
std	776.391761	4.446259	4.450108	6.864278	6.994383	591.094260	6.567484	2.013714	1.424619	0.754716	...	2.319000	
min	1.000000	15.000000	1981.000000	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	
25%	673.000000	23.000000	1993.000000	5.000000	2.000000	194.000000	2.200000	0.000000	0.260000	0.000000	...	0.000000	
50%	1345.000000	26.000000	1996.000000	13.000000	7.000000	684.000000	7.600000	0.000000	0.860000	0.180000	...	0.760000	
75%	2017.000000	29.000000	2000.000000	18.000000	14.000000	1245.000000	13.800000	1.000000	1.850000	0.590000	...	2.500000	
max	2689.000000	41.000000	2007.000000	23.000000	23.000000	2070.000000	23.000000	25.000000	15.000000	10.000000	...	30.000000	

8 rows × 120 columns

In [5]:
```
# Correlation Matrix to identify any correlations between KPI columns
corr_matrix = df.corr()
sns.heatmap(corr_matrix, cmap="YlGnBu")
```

Out[5]: <AxesSubplot:>



```
In [6]: # Shape of the data

        df.shape

Out[6]: (2689, 133)

In [7]: # Taking subset of the dataset

        df=df.iloc[:1048576,0:]

In [8]: # Check the datatype

        df.dtypes

Out[8]: Rk              int64
        Player         object
        Nation         object
        Pos            object
        Squad          object
                        ...
        Recov         float64
        AerWon/90     float64
        AerLost       float64
        AerDuels/90   float64
        AerWon%       float64
        Length: 133, dtype: object

In [9]: df.info()

        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 2689 entries, 0 to 2688
        Columns: 133 entries, Rk to AerWon%
        dtypes: float64(113), int64(7), object(13)
        memory usage: 2.7+ MB
```

Check and handling missing values

```
In [10]: # Check for missing values

         print('Number of missing values across columns: \n',df.isnull().sum())

         Number of missing values across columns:
          Rk            0
         Player         0
         Nation         1
         Pos            0
         Squad          0
                       ..
         Recov          0
         AerWon/90      0
         AerLost        0
         AerDuels/90    0
         AerWon%        0
         Length: 133, dtype: int64
```

- For other data analysis process such as Distribution Plots, and Hypothesis Testing, we wish to conclude them because our laptops are cheap, we try but the fan in the laptop make too much noise while the jupyter notebook doesn't produce anything.

IV. Calculate the running time using `import time`:

```
In [16]: # Start the timer
         start_time = time.time()
```

```
In [29]: # Calculate the running time
         print("--- %s seconds ---" % (time.time() - start_time))

         --- 23.66548442840576 seconds ---
```

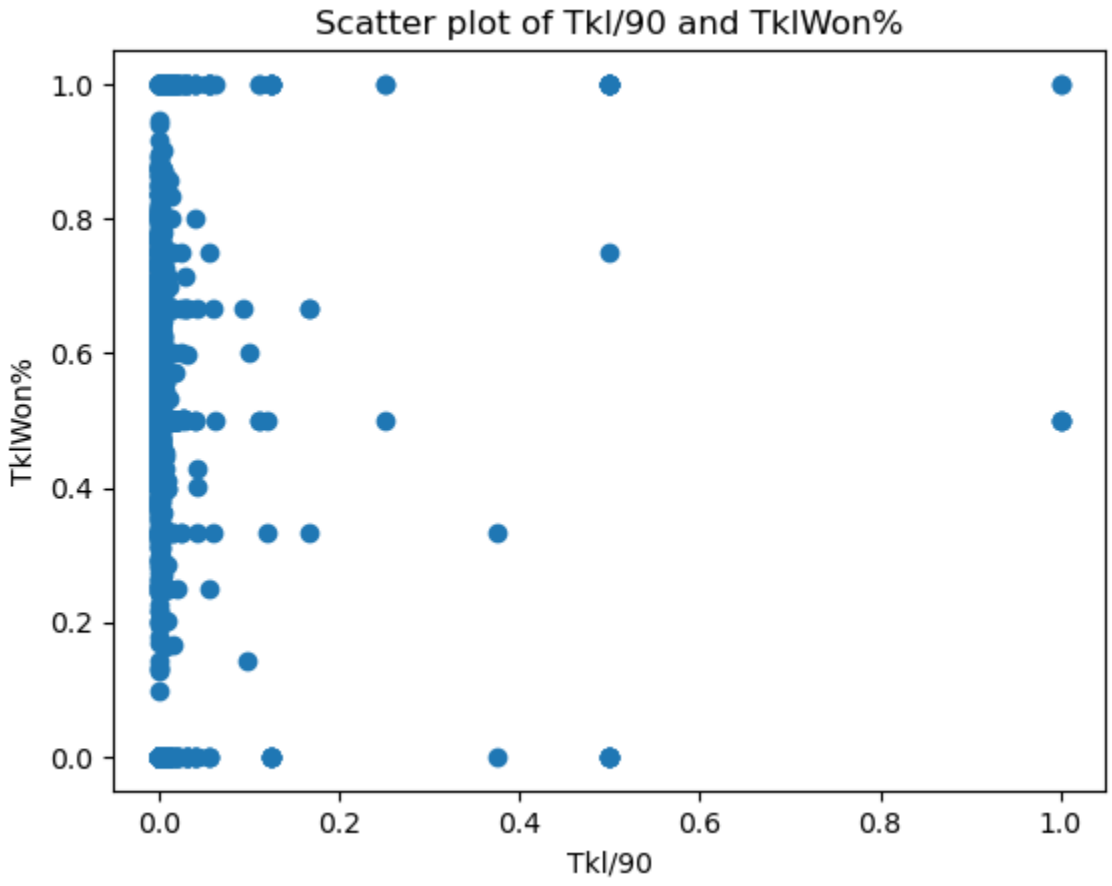
V. Calculate the complexity of the program:

- Because we use the Euclidean distance algorithm, due to the nested loop, the time complexity is O(n^2) where n is the number of players in the dataset.

VI. The mean, median, and standard deviation of an example KPI column 'TKL/90':

- Mean Tkl/90: 1.8800869328994851
- Median Tkl/90: 0.1504132231
- Standard Deviation Tkl/90: 12.207425727951167

VII. The scatter plot between column 'Tkl/90' and 'TklWon%':



VIII. Conclusion:

- After running the program, here are the top 10 most similar players to Virgil van Dijk, using 2022-2023 Van Dijk's stats as Model-based collaborative filtering:

	Player	Pos	Squad	Age
0	Benoît Badiashile	DF	Chelsea	21
1	Malick Thiaw	DF	Schalke 04	21
2	Houssem Aouar	MF	Lyon	24
3	Roger Ibanez	DF	Roma	24
4	Julian Ryerson	DF	Dortmund	25
5	Batista Mendy	MFDF	Angers	23
6	Pape Gueye	MFDF	Marseille	24
7	Dodô	DF	Fiorentina	24
8	Alessandro Bastoni	DF	Inter	23
9	Aleix García	MF	Girona	25

- We find out that finding the player that fits Van Dijk's shoe immediately is nearly impossible because his impact in and out of the field is simply unmeasurable by these metrics and data. We hope that Jurgen Klopp and his data scientists can overcome this challenge and find the perfect player that fits the system.

That is the end of our report!

Nguyễn Đức Bảo Minh BI12-278
Nguyễn Bá Ngọc Minh BA9-042
Đào Trọng Lê Thái BA9-055