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 a height of 1m94 and sensible position choice, 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.'s 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 the Euclidean distance formula because of its “ruler” property.

II. Project Processing step by step:

- Importing Libraries: First, the necessary libraries are imported.

```
# Import Libraries
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from scipy.spatial.distance import euclidean
import seaborn as sns
import time
import matplotlib.pyplot as plt
from sklearn.metrics import silhouette_score
from sklearn.cluster import KMeans
```

- Start the timer: We use function time() to calculate the project's running time.

```
# Start the timer
start_time = time.time()
```

- 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.

```
df = pd.read_csv('Data/2022-2023_Football_Players_stats.csv')
```

- Data Checking: Data checking is performed to find nad remove any missing values and convert the KPI columns to a numeric data type.

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

- 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.

```
# Normalize the KPI data using MinMaxScaler
scaler = MinMaxScaler()
kpi_data_norm = scaler.fit_transform(kpi_data)
```

- 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.

```
# Define a function to compute the Euclidean distance between two players
def euclidean_distance(p1, p2):
    return np.sqrt(np.sum((p1 - p2) ** 2))
```

- 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.

```python
# Find the player who is most similar to a given player
target_player = 'Virgil van Dijk'
target_data = kpi_data_norm[np.where(player_names == target_player)[0][0]]
distances = [euclidean(target_data, row) for row in kpi_data_norm]
most_similar_indices = np.argsort(distances)[1:11]
```

- 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.

```python
 Print the head of the list of the 10 most similar players
similar_players = []
for idx in most_similar_indices:
    row = df.iloc[idx]
    row_values = [row['Player'] ,row['Pos'], row['Squad'], row['Age']] + [
kpi_data_norm[idx, j] for j in range(len(kpi_columns))]
    similar_players.append(row_values)

similar_players_df = pd.DataFrame(similar_players, columns=['Player','Pos',
 'Squad', 'Age'] + list(kpi_columns.values()))
print(similar_players_df.head(10))
```

III. Data Analysis:

- Simple dataset descriptions:

```python
# Display the data
df.head()
```

[115] ✓ 0.0s Python

	Rk	Player	Nation	Pos	Squad	Comp	Age	Born	MP	Starts	...	Crs	TklW	PKwon	PKcon	OG	Recov	AerWon/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	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	
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	
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	
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	

5 rows × 133 columns

```python
# Descriptive statistics
df.describe()
```

[116] ✓ 0.2s Python

	Rk	Age	Born	MP	Starts	Min	90s	Goals	Shots	SoT
count	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
std	776.391761	4.446259	4.450108	6.864278	6.994383	591.094260	6.567484	2.013714	1.424619	0.754716
min	1.000000	15.000000	1981.000000	1.000000	0.000000	1.000000	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
50%	1345.000000	26.000000	1996.000000	13.000000	7.000000	684.000000	7.600000	0.000000	0.860000	0.180000
75%	2017.000000	29.000000	2000.000000	18.000000	14.000000	1245.000000	13.800000	1.000000	1.850000	0.590000
max	2689.000000	41.000000	2007.000000	23.000000	23.000000	2070.000000	23.000000	25.000000	15.000000	10.000000

8 rows × 120 columns

```python
# Shape of the data
df.shape
```

[117] ✓ 0.0s

```
(2689, 133)
```

```python
# Check the datatype
df.dtypes
```

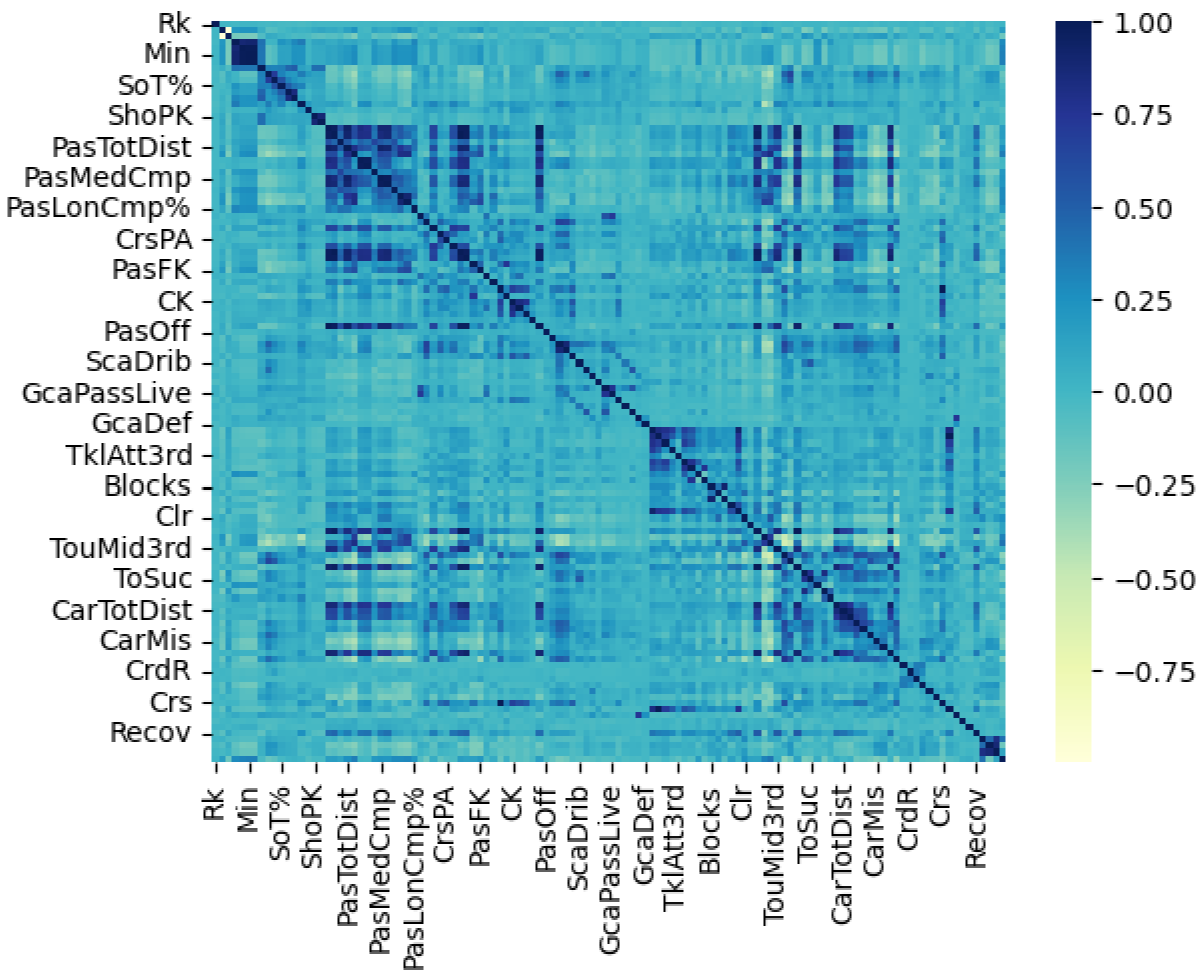
[118] ✓ 0.0s

```
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
```

```
df.info()
[119] ✓ 0.0s
... <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
```

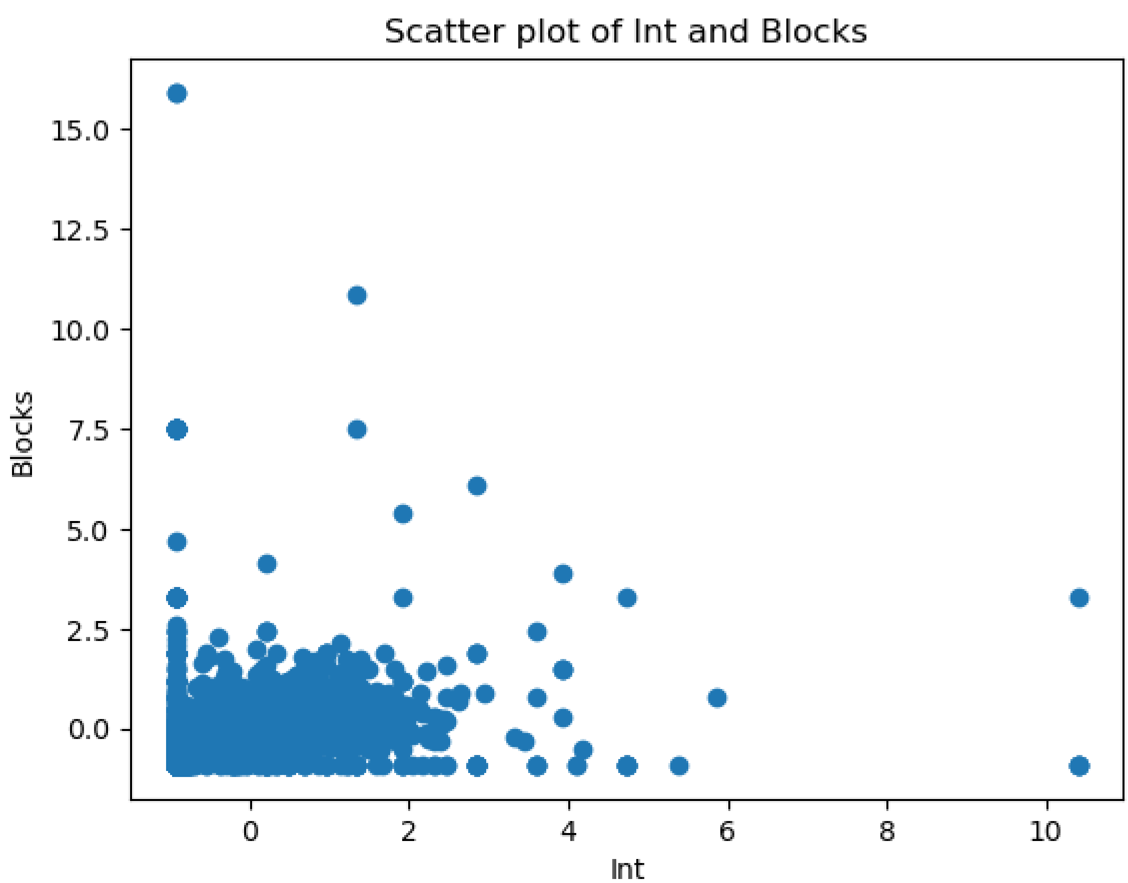
- Correlation Matrix:

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



- Scatter plot between of 'Int' and 'Blocks' KPI columns:

```
# Create a scatter plot of 'Int' and 'Blocks' KPI columns
df = df.dropna(subset=['Int', 'Blocks'])
scaler = StandardScaler()
kpi_data_norm = scaler.fit_transform(df[['Int', 'Blocks']])
Int = kpi_data_norm[:, 0]
Blocks = kpi_data_norm[:, 1]
plt.scatter(Int, Blocks)
plt.xlabel('Int')
plt.ylabel('Blocks')
plt.title('Scatter plot of Int and Blocks')
plt.show()
```



- Calculate Mean, Median, and Standard Deviation of “Int” column:

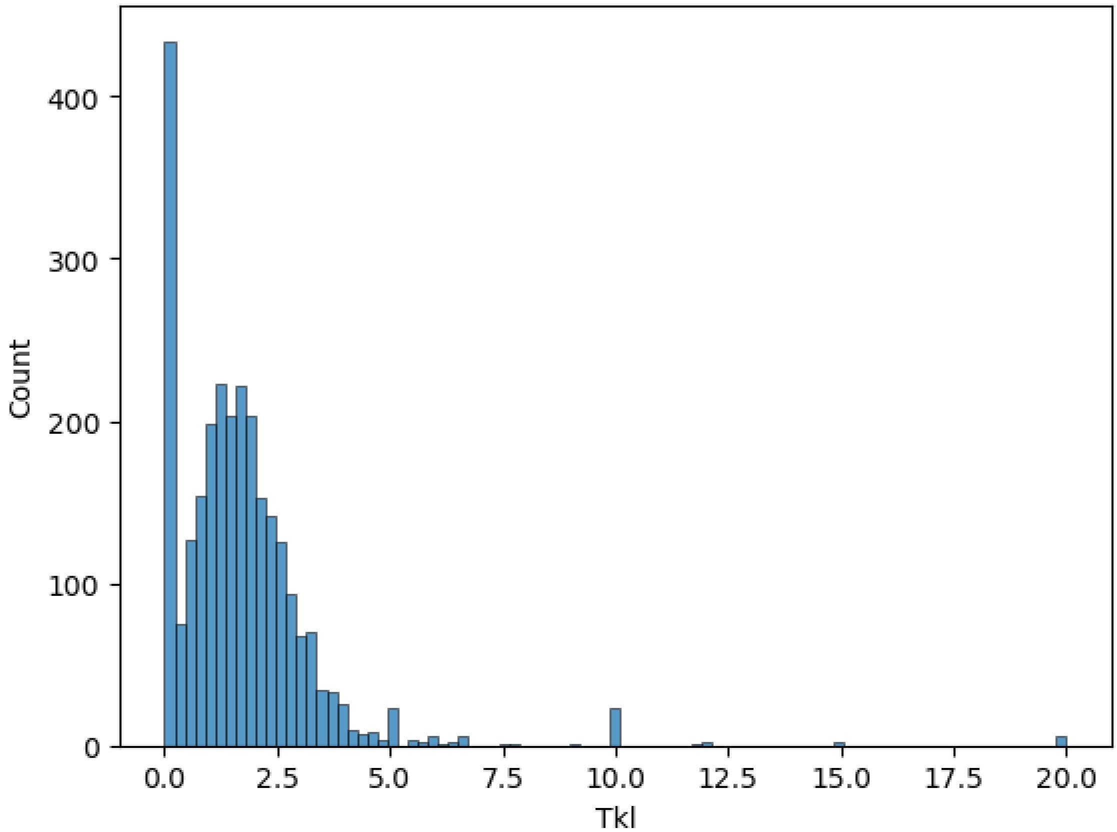
```python
# Calculate Mean, Median, and Standard Deviation of 'Int' KPI Column
tkl_col = df['Int'].apply(pd.to_numeric, errors='coerce').fillna(0)
mean_tkl = tkl_col.mean()
median_tkl = tkl_col.median()
std_tkl = tkl_col.std()

print('Mean Int:', mean_tkl)
print('Median Int:', median_tkl)
print('Standard Deviation Int:', std_tkl)
```

```
Mean Int: 0.8203309780587579
Median Int: 0.69
Standard Deviation Int: 0.8824212086752578
```

- Check the distribution of Tkl using a histogram:

```python
#Check the distribution of Tkl using a histogram
sns.histplot(df['Tkl'])
plt.xlabel('Tkl')
plt.show()
```

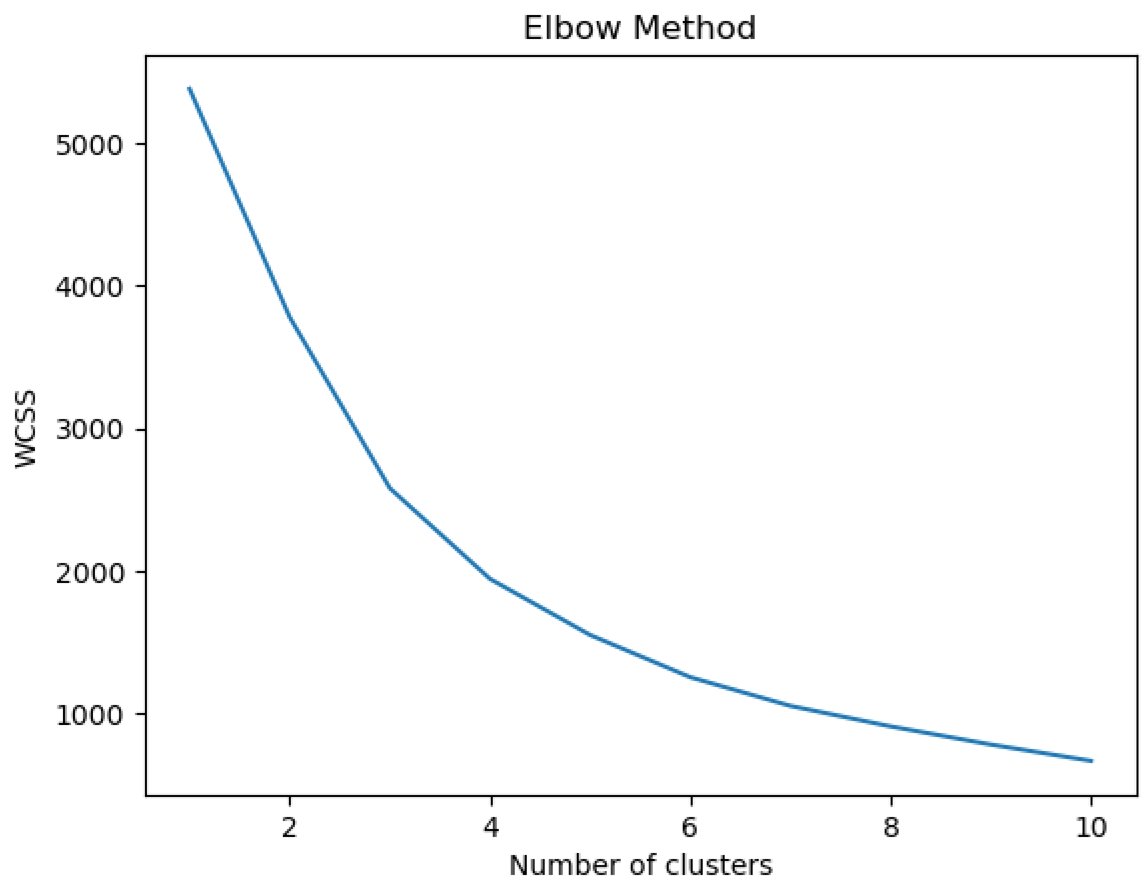


- Use K-means to group players:

```python
# Perform k-means clustering to group players based on their KPI values

# Determine the optimal number of clusters using the elbow method
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(kpi_data_norm)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()

# Use k-means clustering with 5 clusters
kmeans = KMeans(n_clusters=5, init='k-means++', random_state=42)
kmeans.fit(kpi_data_norm)
df['Cluster'] = kmeans.labels_
```



- Evaluate the clustering results:

```
# Evaluate the clustering results using silhouette score
silhouette_score(kpi_data_norm, kmeans.labels_)
```

```
0.38176879591189283
```

- For other data analysis processes such as PCA, train split test, we couldn't perform those because the outcome of the lost of interpretability, nonlinear relationships, and overfitting.

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

```
# Calculate the running time
print("%s seconds" % (time.time() - start_time))
```

```
10.830374479293823 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. 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	Fodé Ballo-Touré	DF	Milan	26
2	Oleksandr Zinchenko	DF	Arsenal	26
3	Iñigo Martínez	DF	Athletic Club	31
4	Malick Thiaw	DF	Schalke 04	21
5	Houssem Aouar	MF	Lyon	24
6	Giulio Donati	DF	Monza	33
7	Jonny Evans	DF	Leicester City	35
8	Joe Worrall	DF	Nott'ham Forest	26
9	Roger Ibanez	DF	Roma	24

- Also, here is the top 10 most similar players under 25:

	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 understand that using data is just one part of finding the perfect replacement for Virgil van Dijk, as the centre-back position requires many uncollectable data. But these players above do have the potential to fill Virgil van Dijk's shoe.

And that is the end of our report!
Thanks you for your attention ❣️

Collaborators on this project:
Nguyễn Đức Bảo Minh BI12-278
Nguyễn Bá Ngọc Minh BA9-042
Đào Trọng Lê Thái BA9-055