MACHINE LEARNGING ANALYSIS OF PLAYER STASTICS

Presented by Mohamed Amr

OVERVIEW

01

Our Team

02

Our Mision

03

Data preprocessing

04

Data visualization

05

MI Models

OUR TEAM

Mohamed Amr

Artificial intelligence

Abdallah saad

Communication engineering

Abdulrahman Taha

Computer engineer

Khaled Zakria

Artificial intelligence

Mohamed Hamed

Computer and communication engineering

MISION

My mission in this project is to leverage the power of machine learning and data visualization to assist football scouts in efficiently identifying top-performing players across the top 5 leagues. By providing a user-friendly interface and insightful visualizations through Power BI, this project aims to simplify the scouting process, enabling clubs to make data-driven decisions when selecting players based on their unique requirements and performance metrics

DATA PREPROCESSING

Data Cleaning

is the process of correcting or removing inaccuracies, inconsistencies, and missing values in a dataset to ensure data quality and reliability for analysis

Data Exploration

is the initial process of analyzing and visualizing a dataset to discover patterns, trends, and insights, as well as to understand its structure and relationships between variables

DATA CLEANING

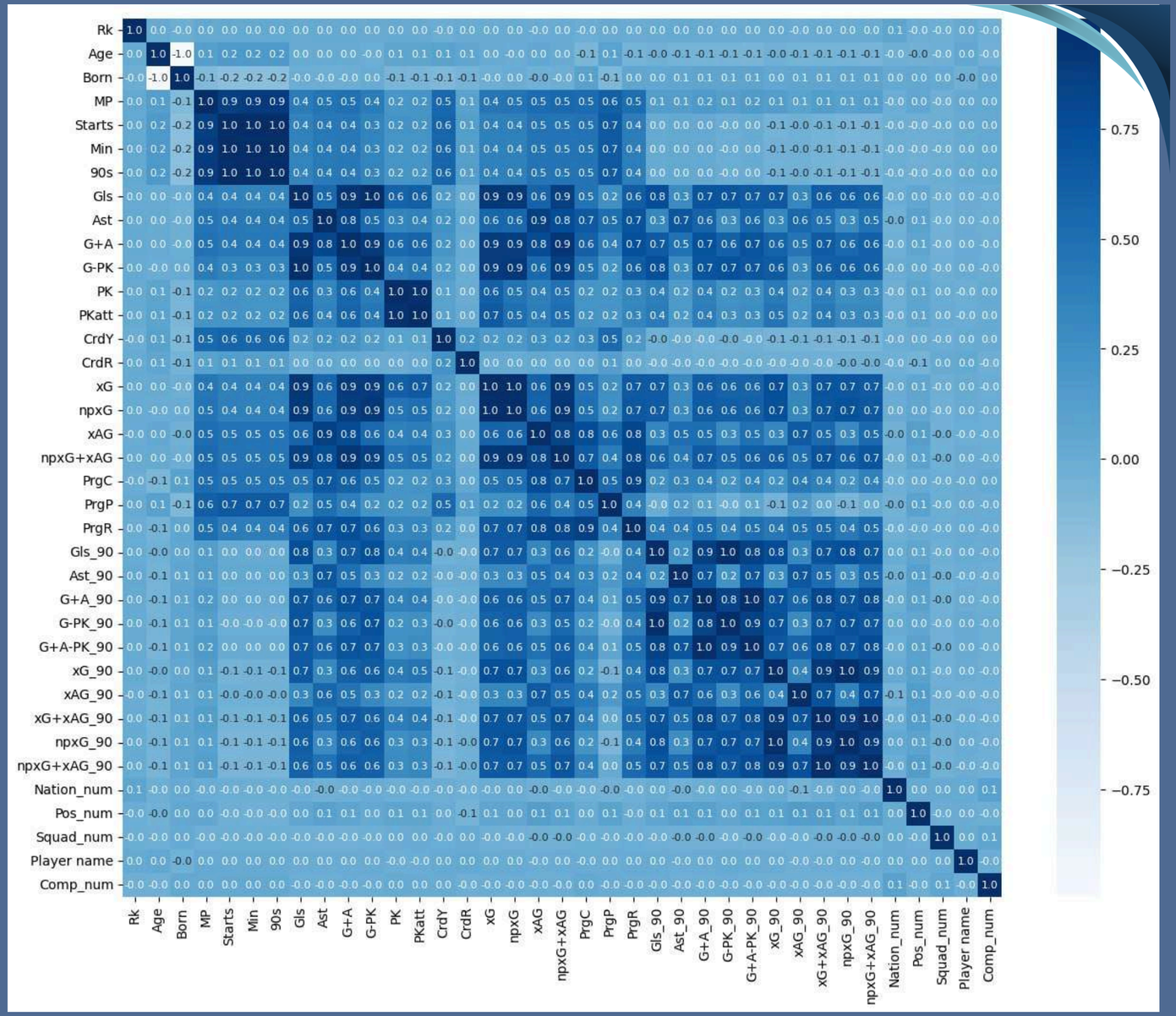
1. The first chart represents the data before cleaning, showing key metrics for players, including goals, assists, and expected goals per 90 minutes.
2. The second chart displays the data after cleaning, with the same set of players and metrics. The refined data offers clearer insights into their performances, such as goals, assists, and expected assists per 90 minutes

	Rk	Player	Nation	Pos	Squad	Comp	Age	Born	MP	Starts	...	Gls_90	Ast_90	G+A_90	G-PK_90	G+A-PK_90	xG_90	xAG_90	xG+xAG_90
0	1	Max Aarons	eng ENG	DF	Bournemouth	eng Premier League	23.0	2000.0	20	13	...	0.00	0.07	0.07	0.00	0.07	0.00	0.06	0.06
1	2	Brenden Aaronson	us USA	MF,FW	Union Berlin	de Bundesliga	22.0	2000.0	30	14	...	0.14	0.14	0.28	0.14	0.28	0.14	0.13	0.27
2	3	Paxten Aaronson	us USA	MF	Eint Frankfurt	de Bundesliga	19.0	2003.0	7	1	...	0.00	0.89	0.89	0.00	0.89	0.11	0.07	0.19

[8]:

	Rk	Player	Nation	Pos	Squad	Comp	Age	Born	MP	Starts	...	Gls_90	Ast_90	G+A_90	G-PK_90	G+A-PK_90	xG_90	xAG_90	xG+xAG_90
0	1	Max Aarons	ENG	DF	Bournemouth	Premier League	23.0	2000.0	20	13	...	0.00	0.07	0.07	0.00	0.07	0.00	0.06	0.06
1	2	Brenden Aaronson	USA	MF	Union Berlin	Bundesliga	22.0	2000.0	30	14	...	0.14	0.14	0.28	0.14	0.28	0.14	0.13	0.27
2	3	Paxten Aaronson	USA	MF	Eint Frankfurt	Bundesliga	19.0	2003.0	7	1	...	0.00	0.89	0.89	0.00	0.89	0.11	0.07	0.19

DATA EXPLORATION

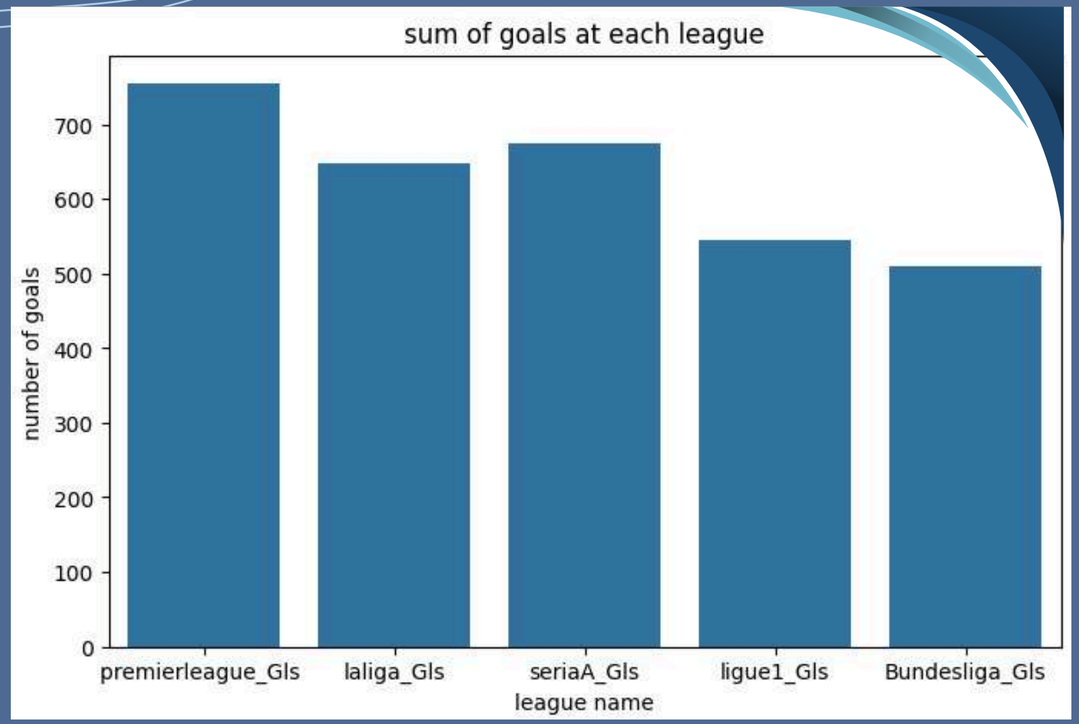


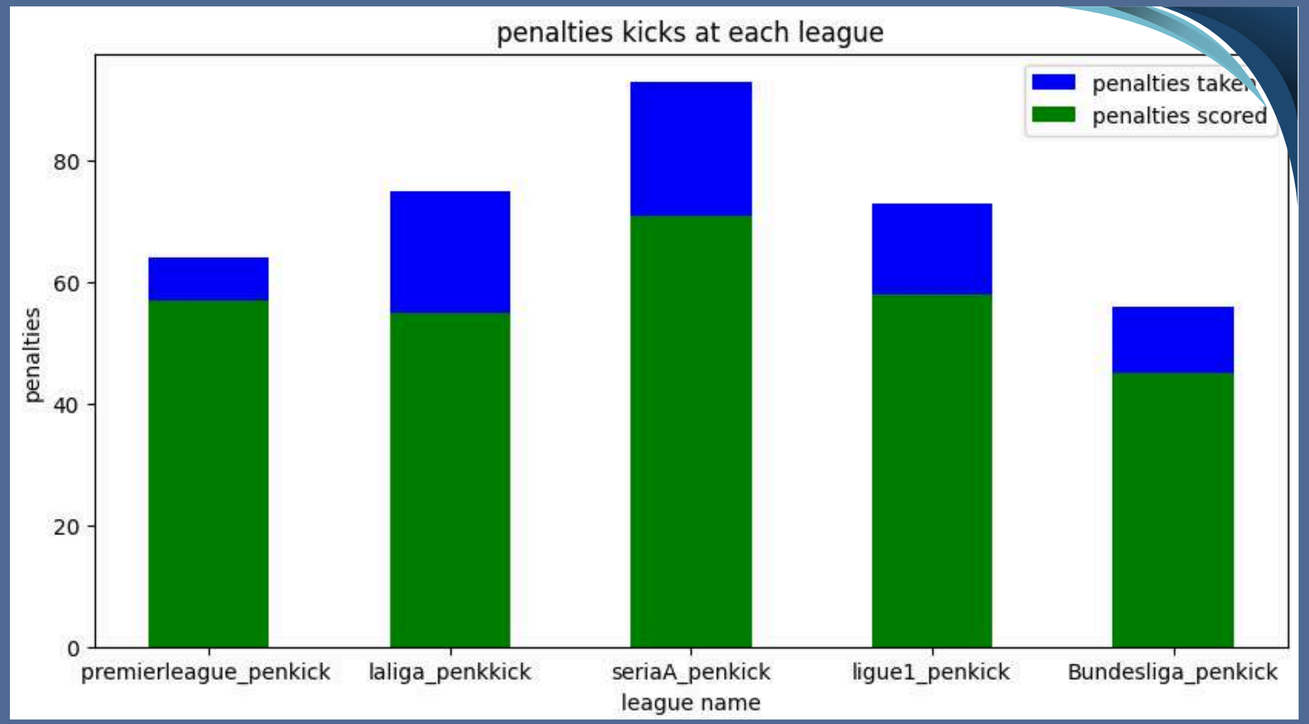
THIS IMAGE APPEARS TO BE A HEATMAP OR VISUALIZATION OF PLAYER STATISTICS FOR VARIOUS METRICS IN A SPORTING CONTEXT. SOME KEY OBSERVATIONS:

1. THE IMAGE IS DIVIDED INTO ROWS REPRESENTING DIFFERENT PLAYER METRICS, SUCH AS RK (RANK), AGE, BORN, MP (MINUTES PLAYED), STARTS, MIN (MINUTES), 90S, GLS (GOALS), AST (ASSISTS), G+A (GOALS + ASSISTS), G-PK (GOALS EXCLUDING PENALTY KICKS), PK (PENALTY KICKS), PKATT (PENALTY KICK ATTEMPTS), CRDY (YELLOW CARDS), CRDR (RED CARDS), AND SEVERAL ADVANCED STATS LIKE XG, XAG, NPXG, ETC.
2. THE COLUMNS SEEM TO REPRESENT INDIVIDUAL PLAYERS, WITH THEIR NAMES SHOWN AT THE BOTTOM OF THE IMAGE.
3. THE COLOR CODING RANGES FROM DARK BLUE (LOW VALUES) TO BRIGHT YELLOW (HIGH VALUES), ALLOWING FOR EASY VISUALIZATION OF THE PLAYER'S PERFORMANCE ACROSS DIFFERENT METRICS.
4. THE DATA APPEARS TO BE COMPREHENSIVE, COVERING VARIOUS ASPECTS OF PLAYER PERFORMANCE, INCLUDING GOALS, ASSISTS, CARDS, AND ADVANCED ANALYTICS LIKE EXPECTED GOALS AND EXPECTED ASSISTS.
5. THIS TYPE OF VISUALIZATION COULD BE USEFUL FOR ANALYZING AND COMPARING THE PERFORMANCE OF PLAYERS ACROSS DIFFERENT LEAGUES OR TEAMS, AS WELL AS IDENTIFYING THEIR STRENGTHS AND WEAKNESSES BASED ON THE VARIOUS METRICS DISPLAYED.

DATA EXPLORATION

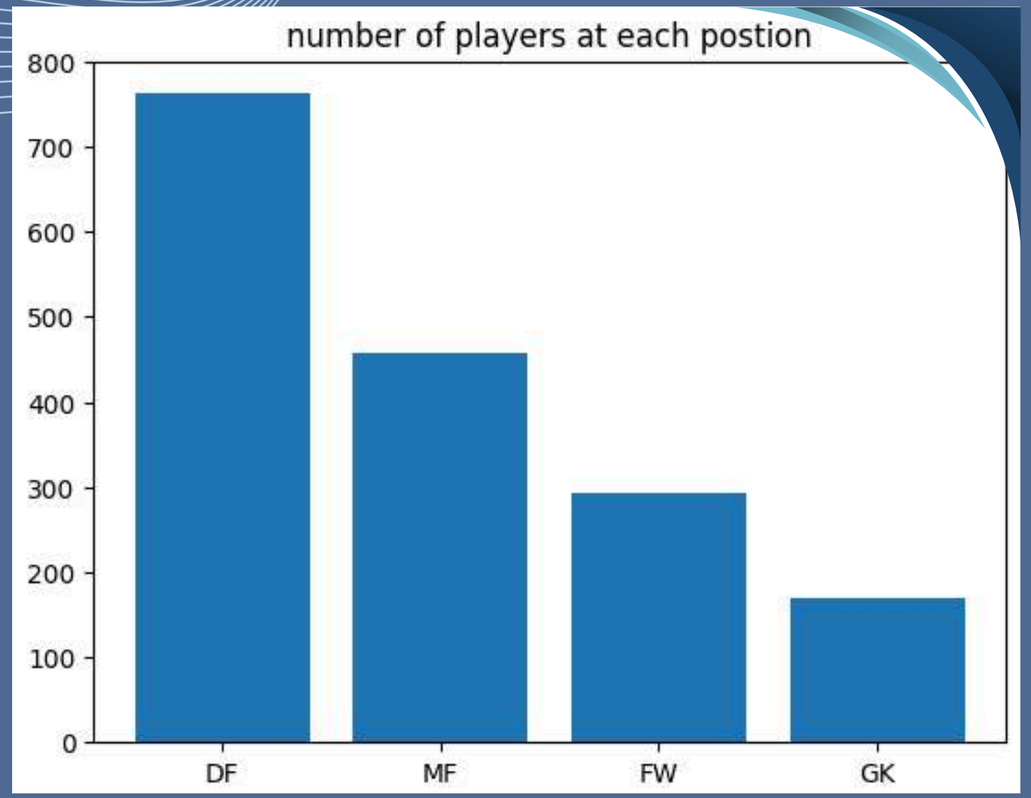
1. The first chart shows the total number of goals scored in each league, with the Premier League having the highest total goals and Bundesliga the lowest.
2. The second chart displays penalty kicks at each league, comparing the number of penalties taken and penalties scored, with Serie A leading in both metrics.

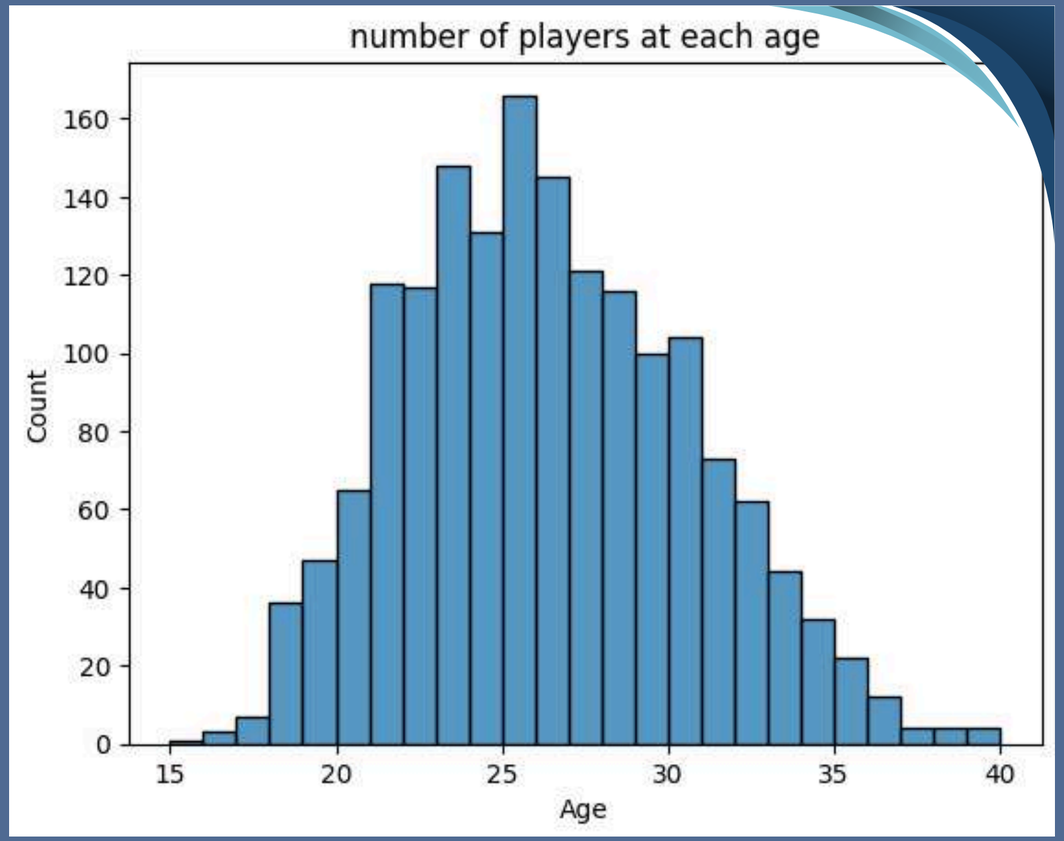




DATA EXPLORATION

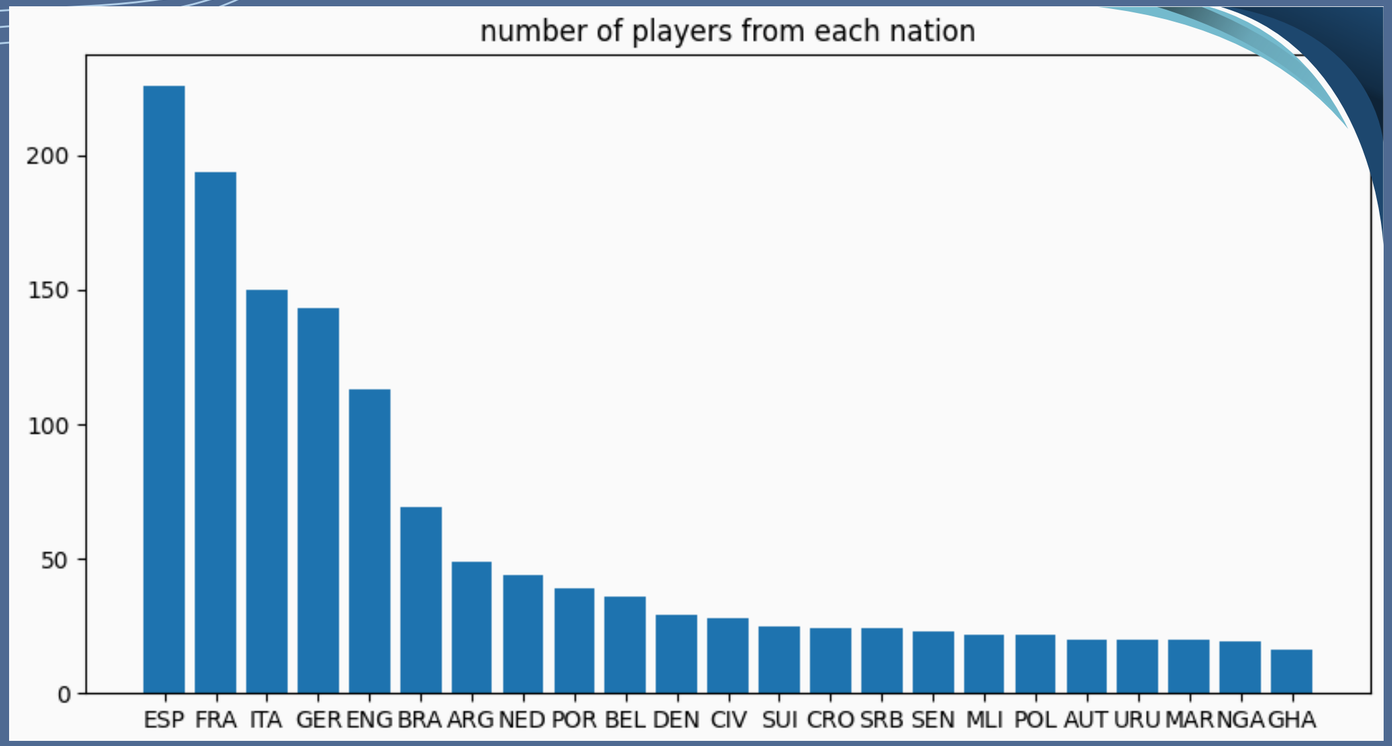
1. The first image shows a graph of the number of players at each age, with the highest concentration between the ages of 25 and 30. 2. The second image shows a bar graph of the number of players at each position, with the highest number of players being at the DF (Defender) position.

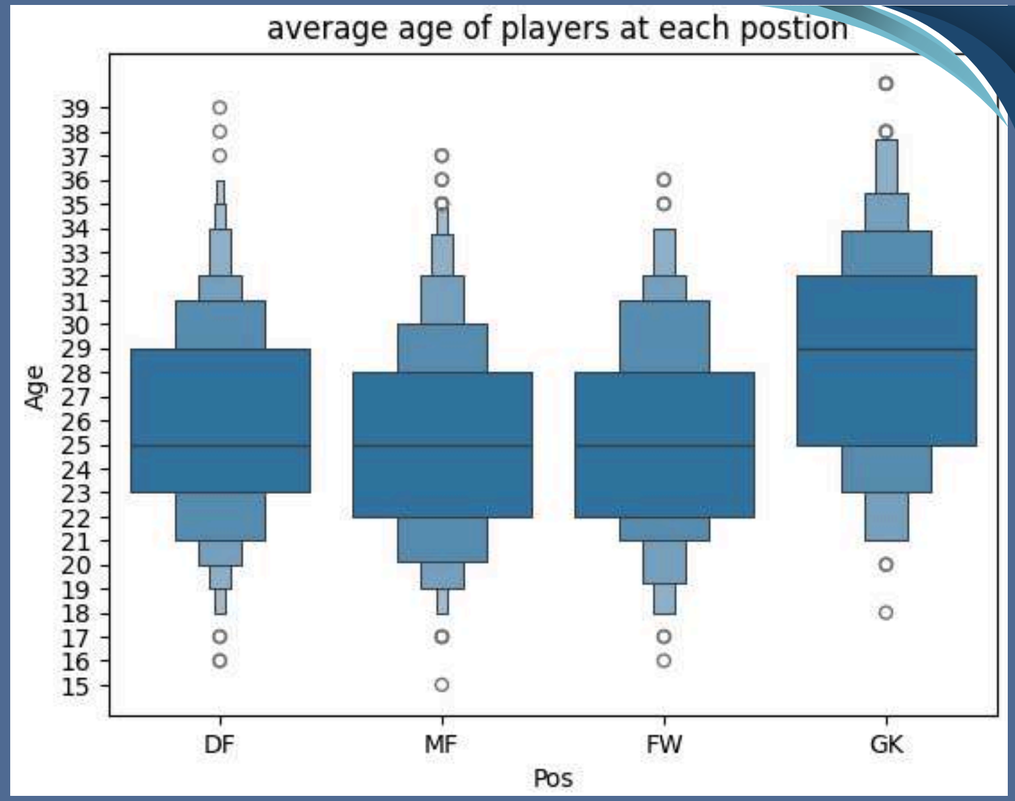




DATA EXPLORATION

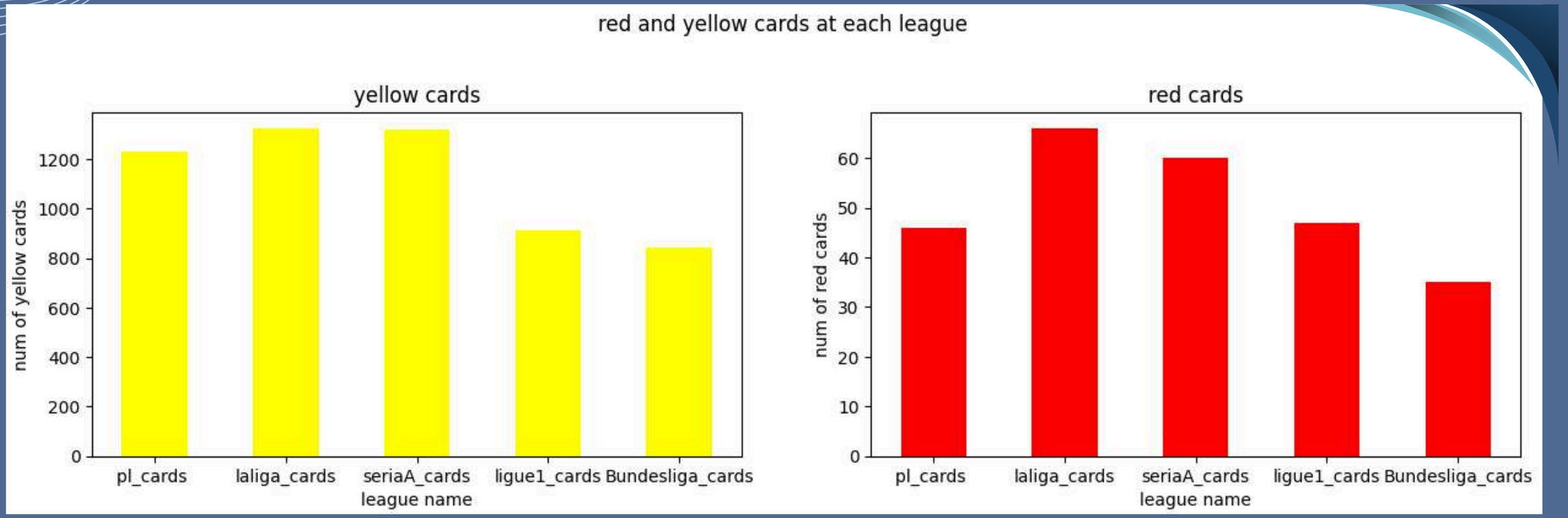
1. The first image shows a box plot of the average age of players at each position, with GK (Goalkeeper) having the highest average age and DF (Defender) having the lowest
2. . The second image shows a bar graph of the number of players from each nation, with ESP (Spain) having the highest number of players.





DATA EXPLORATION

1. The first image shows a bar graph of the number of yellow cards awarded in different leagues, with pl_cards having the highest number. The second image shows a bar graph of the number of red cards awarded in different leagues, with pl_cards having the highest number.



DATA VISUALIZATION

Power Bi

Power BI is a comprehensive business analytics service by Microsoft that enables users to connect, transform, visualize, and share data-driven insights and interactive reports.

Data visualization

1. calculate column

1. The code creates a new table called "Top1_Ligue1" with a single column.
2. Inside the "TOP" function, it filters the data to only include rows where the "cleaned_data[Comp]" column is equal to "Ligue 1".
3. It then selects the "cleaned_data[TotalScore]" column and sorts the results in descending order using the "DESC" function.
4. Finally, the code uses the "CONCATENATE" function to create a new column that combines the text "Total Score: " with the maximum value from the "cleaned_data[TotalScore]" column.

```
Top1_Ligue 1 =
VAR TopPlayerTable =
    TOPN(
        1,
        FILTER(
            cleaned_data,
            cleaned_data[Comp] = "Ligue 1"
        ),
        cleaned_data[TotalScore],
        DESC
    )
RETURN
    CONCATENATE(
        MAXX(TopPlayerTable, cleaned_data[Player]),
        "  Total Score: " & MAXX(TopPlayerTable, cleaned_data[TotalScore])


)
```

This code is designed to create a new table that shows the top player in the Ligue 1 league based on their total score. The same process is then repeated for the other four top leagues (Premier League, Bundesliga, Serie A, and LaLiga) to create similar tables for those leagues. The goal of this code is to provide a way to quickly identify the top-scoring player in each of the major European football leagues, which can be useful for analysis and comparison purposes.

Top1_Ligue 1	Top1_Premier League	Top1_Bundesliga	Top1_Serie A	Top1_LaLigaPlayerWithScore
Kylian Mbappé Total Score: 34	Cole Palmer Total Score: 33	Harry Kane Total Score: 44	Lautaro Martínez Total Score: 27	Artem Dovbyk Total Score: 32
Kylian Mbappé Total Score: 34	Cole Palmer Total Score: 33	Harry Kane Total Score: 44	Lautaro Martínez Total Score: 27	Artem Dovbyk Total Score: 32
Kylian Mbappé Total Score: 34	Cole Palmer Total Score: 33	Harry Kane Total Score: 44	Lautaro Martínez Total Score: 27	Artem Dovbyk Total Score: 32
Kylian Mbappé Total Score: 34	Cole Palmer Total Score: 33	Harry Kane Total Score: 44	Lautaro Martínez Total Score: 27	Artem Dovbyk Total Score: 32
Kylian Mbappé Total Score: 34	Cole Palmer Total Score: 33	Harry Kane Total Score: 44	Lautaro Martínez Total Score: 27	Artem Dovbyk Total Score: 32
Kylian Mbappé Total Score: 34	Cole Palmer Total Score: 33	Harry Kane Total Score: 44	Lautaro Martínez Total Score: 27	Artem Dovbyk Total Score: 32
Kylian Mbappé Total Score: 34	Cole Palmer Total Score: 33	Harry Kane Total Score: 44	Lautaro Martínez Total Score: 27	Artem Dovbyk Total Score: 32
Kylian Mbappé Total Score: 34	Cole Palmer Total Score: 33	Harry Kane Total Score: 44	Lautaro Martínez Total Score: 27	Artem Dovbyk Total Score: 32
Kylian Mbappé Total Score: 34	Cole Palmer Total Score: 33	Harry Kane Total Score: 44	Lautaro Martínez Total Score: 27	Artem Dovbyk Total Score: 32
Kylian Mbappé Total Score: 34	Cole Palmer Total Score: 33	Harry Kane Total Score: 44	Lautaro Martínez Total Score: 27	Artem Dovbyk Total Score: 32

Data visualization

1. visualization

1. This image provides information about the top players in the top 5 leagues. It displays details about 5 different players:
2. Kylian Mbappe – A player in France's Ligue 1 league with a total score of 34.
3. Harry Kane – A player in Germany's Bundesliga league with a total score of 44.
4. Cole Palmer – A player in England's Premier League with a total score of 33.
5. Artem Dovbyk – A player in Spain's La Liga with a total score of 32.
6. Lautaro Martínez – A player in Italy's Serie A with a total score of 27.
7. The image also shows the total contribution score of 8,360 for all the players combined.

Data visualization

1. visualization

1- Assist Breakdown:
- **Sum of Assists: 3470 (94.19%)**
- **Sum of Ast 90: 0.21 (5.81%)**

2- Goals Breakdown:
- **Sum of PK : 0K (8.27%)**
- **Sum of Gls: 5K (91.73%)**

3- Cards Breakdown:
- **Sum of CrdY: 8K (95.66%)**
- **Sum of CrdR: 0K (4.34%)**

Data visualization

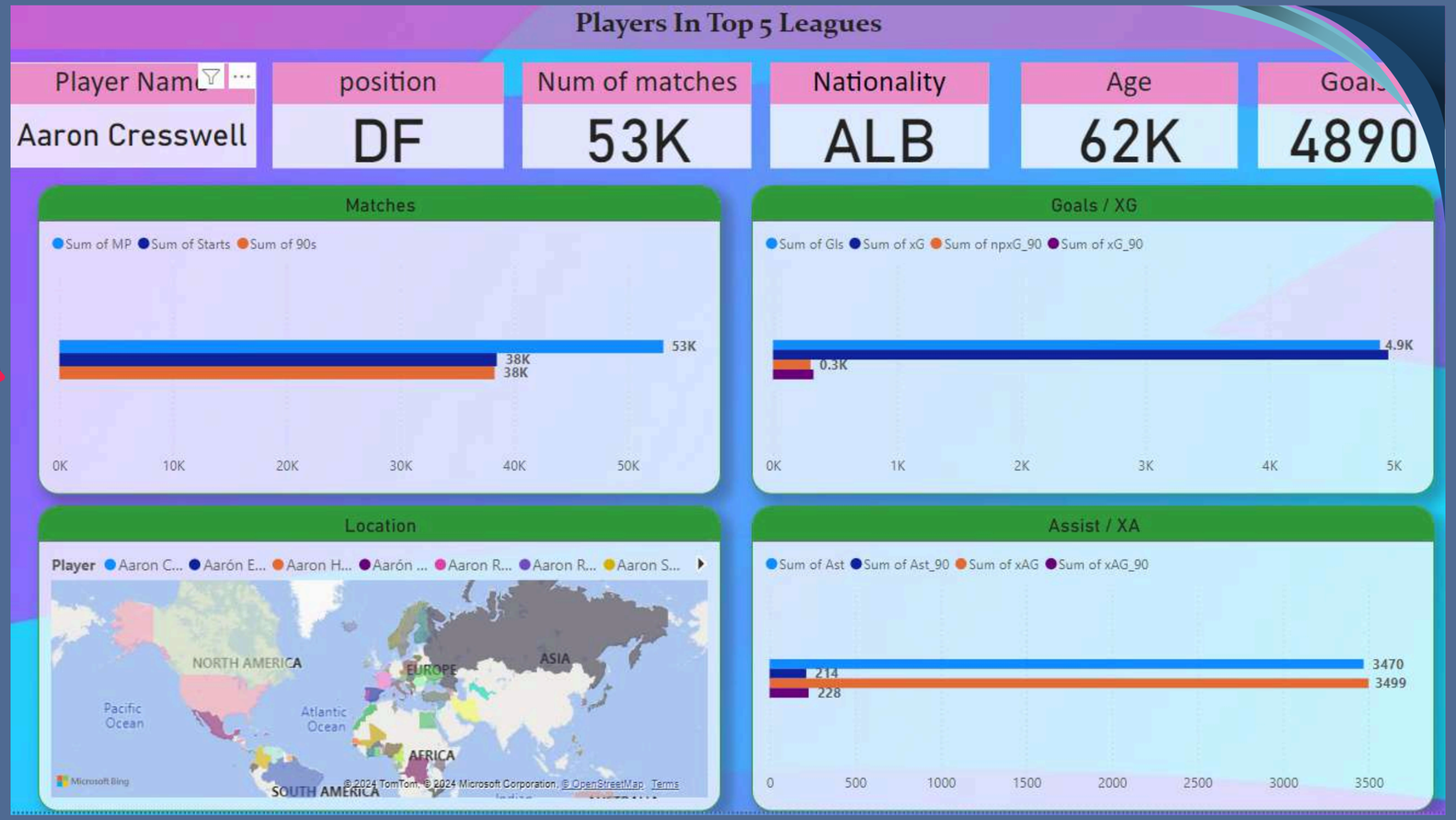
1. visualization

Matches:
- The chart shows the number of matches played (MP), starts, and 90s minutes played over the course of his career.

Goals/xG:
- This chart displays Cresswell's goal-scoring stats, including total goals scored, expected goals (xG), non-penalty expected goals (npxG), and expected goals per 90 minutes (xG/90).

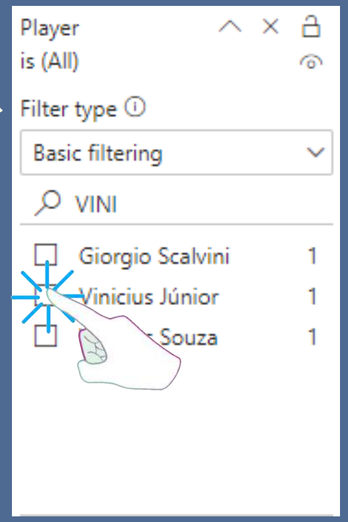
Assist/xA:
- The chart shows Cresswell's assist stats, including total assists, expected assists (xA), and the sum of his assist-related metrics.

Location:
- A world map shows the geographic distribution of the various "Aaron" players, highlighting Cresswell's position in Europe.

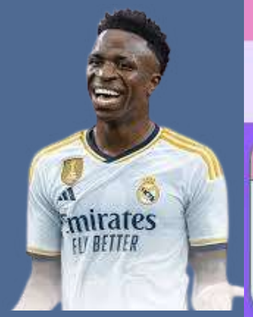


Data visualization

1. **Choose any player from the list**



2. It will show all the statistics of the selected player



Player Name	position	Num of matches	Nationality	Age	Goals
Vinicius Júnior	FW	26	BRA	23	15

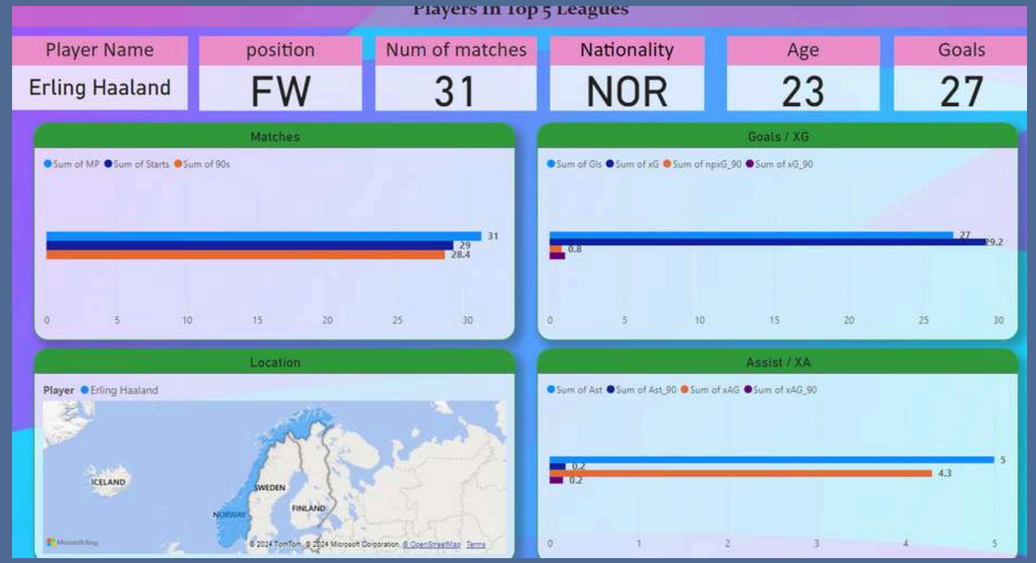
Total Contribution
20
Sum of TotalScore

Data visualization

It will show all the statistics of the selected player

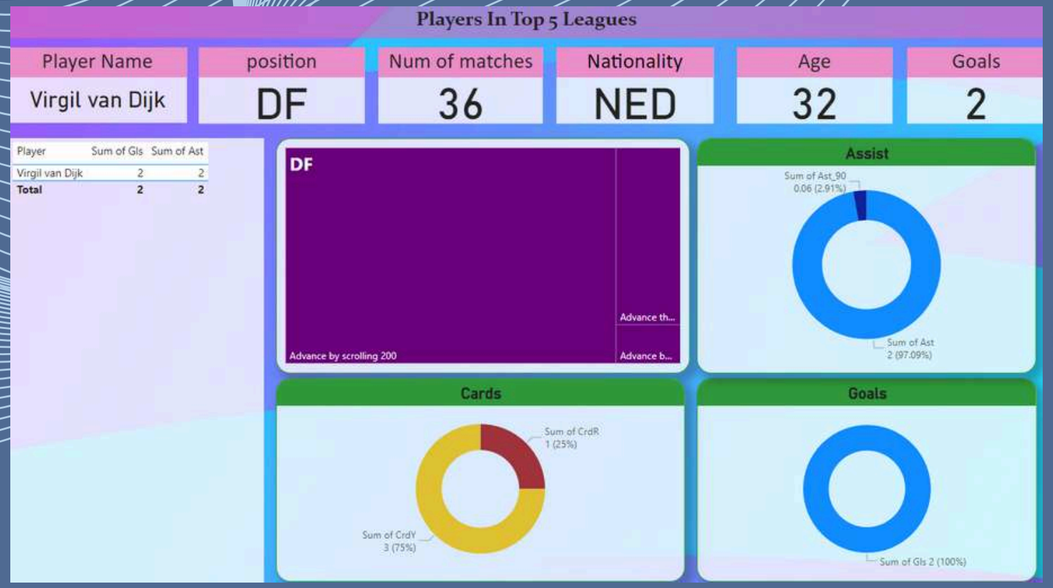
Data visualization

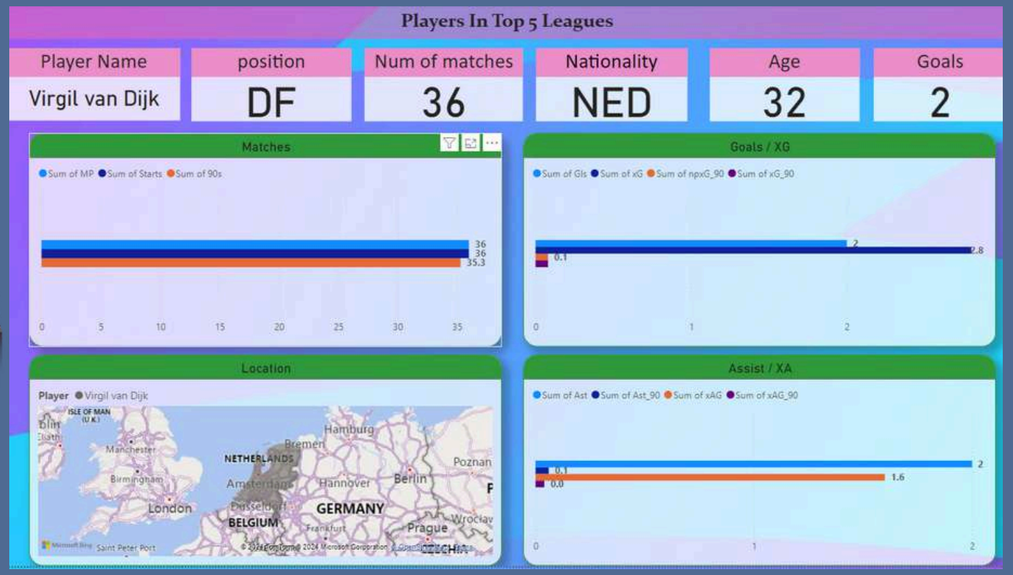
We will choose a player from each position to compare statistics



Data visualization

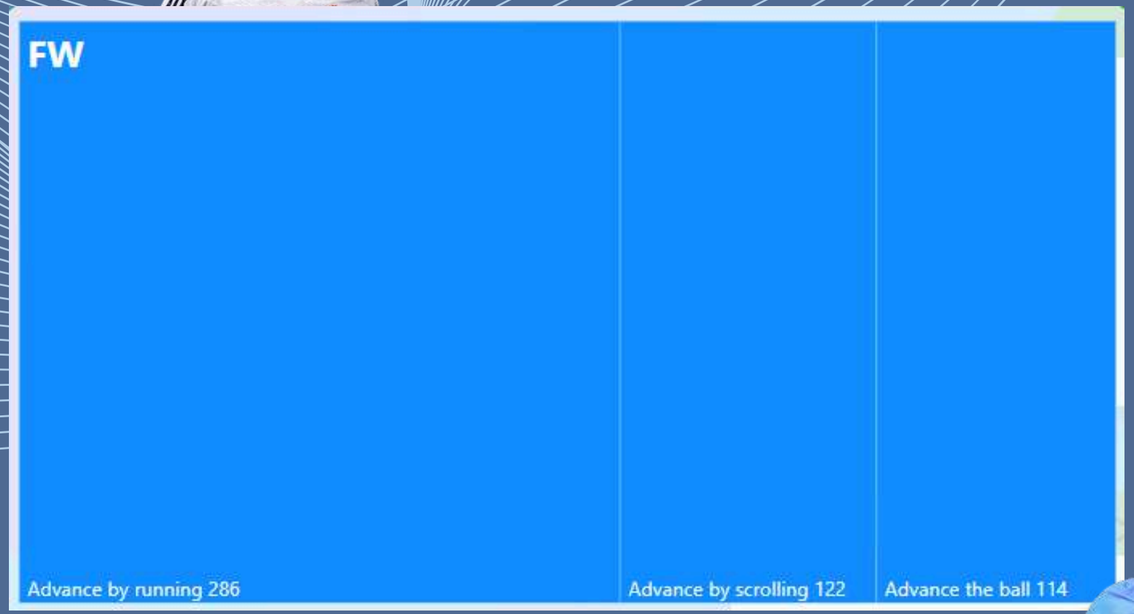
We will choose a player from each position to compare statistics

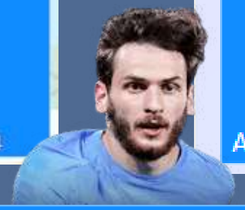


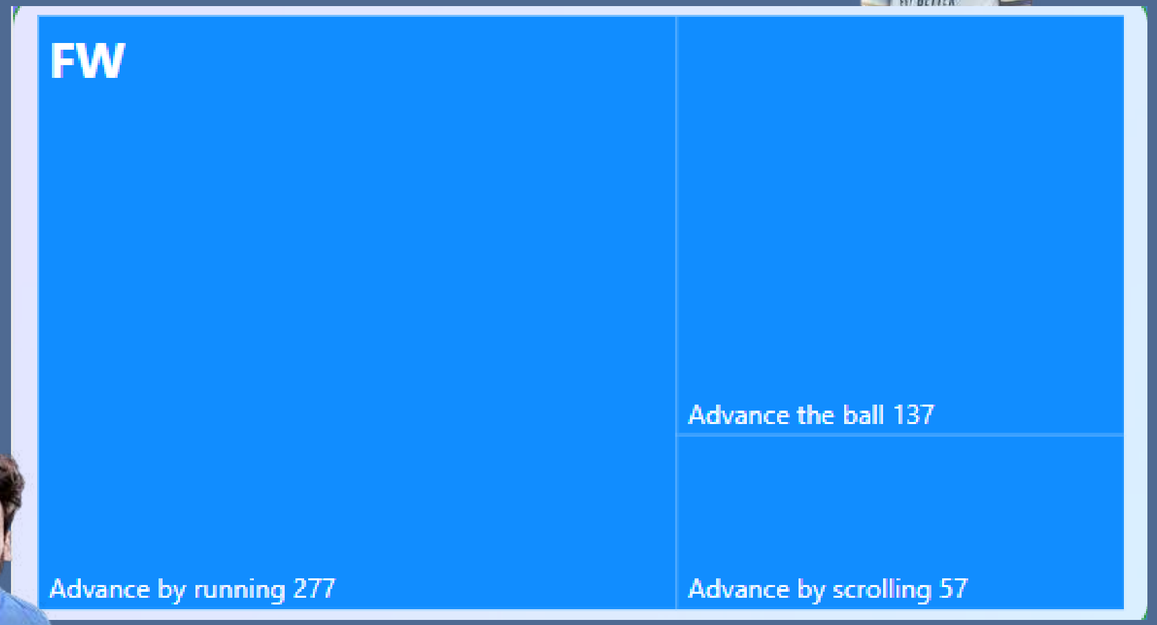


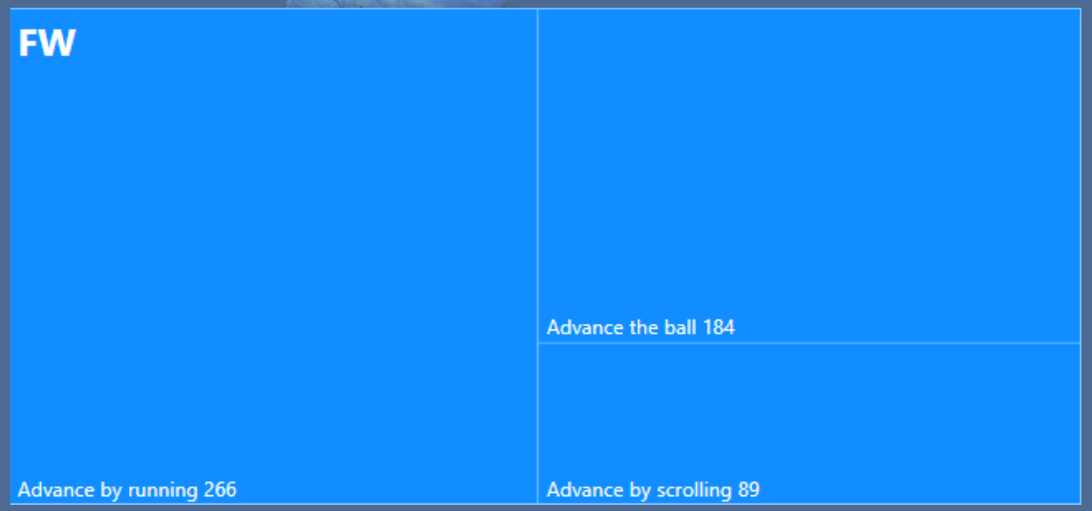
Data visualization

We will choose more than one player in the same position



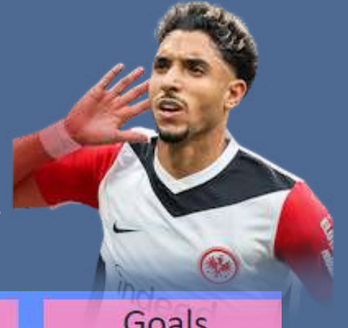






Data visualization

Predict an explosive season for any player based on his statistics



Player Name	position	Num of matches	Nationality	Age	Goals
Omar Marmoush	FW	29	EGY	24	12

Total Contribution

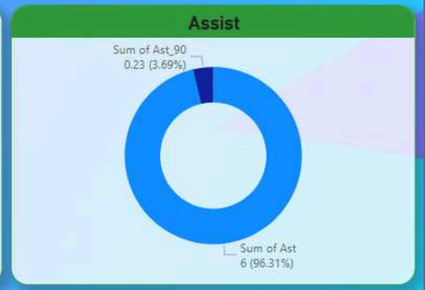
18

Sum of TotalScore

Players In Top 5 Leagues

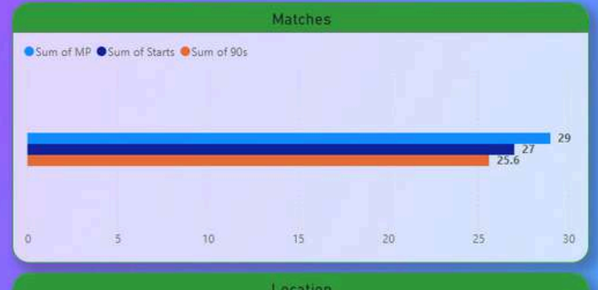
Player Name	position	Num of matches	Nationality	Age	Goals
Omar Marmoush	FW	29	EGY	24	12

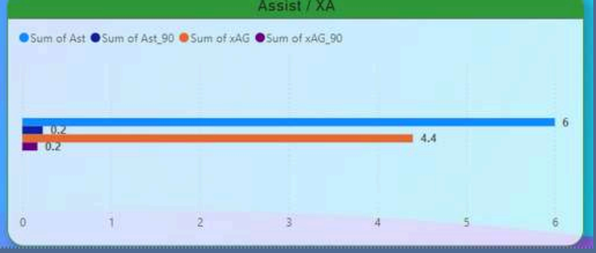
Player	Sum of Gls	Sum of Ast
Omar Marmoush	12	6
Total	**12**	**6**



Players In Top 5 Leagues

Player Name	position	Num of matches	Nationality	Age	Goals
Omar Marmoush	FW	29	EGY	24	12





ML MODELS

Random forest

You are sending and receiving too many words in a short period of time.

Decision tree

The decision tree model recursively partitions the feature space into regions and predicts the target variable based on the most informative features.

Linear regression

Linear regression is a statistical model that estimates the linear relationship between a dependent variable and one or more independent variables.

Svm

Support Vector Machine (SVM) is a supervised learning algorithm that finds the optimal hyperplane to separate data points of different classes in a high-dimensional space

neural network

computational model inspired by the human brain, consisting of interconnected layers of nodes (neurons) that process and learn from data to perform tasks such as classification, regression, and pattern recognition

DATA PREPATION

```
X_classification = df[['Gls','Ast','PK','CrdY','CrdR','xG','PrgC','PrgP','PrgR',]]
y_classification = df['Pos']

# Define features and target for regression
X_regression = df.drop(columns=['Player', 'Nation','Comp', 'Pos', 'Squad','Gls','G+A','G-PK','xG','npxG+xAG','G+A_90', 'G-PK_90', 'G+A-PK_90', 'xG_90'
y_regression = df['Gls']

# Data partitioning
X_train_class, X_test_class, y_train_class, y_test_class = train_test_split(X_classification, y_classification, test_size=0.2, random_state=42)
X_train_reg, X_test_reg, y_train_reg, y_test_reg = train_test_split(X_regression, y_regression, test_size=0.2, random_state=42)
```

LINEAR REGRESSION

CLASSIFICATION

```
]: from sklearn.linear_model import LogisticRegression
   from sklearn.preprocessing import StandardScaler
   from sklearn.metrics import accuracy_score
   import pandas as pd

   # Standardizing the classification data
   scaler_class = StandardScaler()
   X_train_class_scaled = scaler_class.fit_transform(X_train_class)
   X_test_class_scaled = scaler_class.transform(X_test_class)

   # Logistic Regression for classification
   model_classification_lr = LogisticRegression(random_state=42)
   model_classification_lr.fit(X_train_class_scaled, y_train_class)

   # Predict using the model
   y_pred_class_lr = model_classification_lr.predict(X_test_class_scaled)

   # Evaluate the model
   accuracy_class_lr = accuracy_score(y_test_class, y_pred_class_lr)
   print(f'Logistic Regression Classification Accuracy: {accuracy_class_lr}')
```

```
Logistic Regression Classification Accuracy: 0.6948453608247422
```

```
]: # Create results DataFrame for classification
   results_class_lr = pd.DataFrame({'Actual_Pos': y_test_class, 'Predicted': y_pred_class_lr})
   results_class_lr['Player_Name'] = df['Player']
   print(results_class_lr.head(10))
```

```
     Actual_Pos Predicted             Player_Name
1765         DF        MF           Ignasi Miquel
457          DF        DF           Isaac Carcelen
1675         DF        DF        Christian Mawissa
486          DF        DF  Jean-Charles Castelletto
576          MF        FW           Maxwel Cornet
106          MF        DF          Elliot Anderson
```

LINEAR REGRESSION

REGRESSION

[35]:
```
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
import pandas as pd

# Standardizing the regression data
scaler_reg = StandardScaler()
X_train_reg_scaled = scaler_reg.fit_transform(X_train_reg)
X_test_reg_scaled = scaler_reg.transform(X_test_reg)

# Linear Regression
model_regression_lr = LinearRegression()
model_regression_lr.fit(X_train_reg_scaled, y_train_reg)

# Predict using the model
y_pred_reg_lr = model_regression_lr.predict(X_test_reg_scaled)

# Ensure predictions are non-negative
y_pred_reg_lr = np.maximum(y_pred_reg_lr, 0)
# Convert predictions to int
y_pred_reg_lr = y_pred_reg_lr.astype(int)

# Evaluate the model
mse_reg_lr = mean_squared_error(y_test_reg, y_pred_reg_lr)
print(f'Linear Regression Mean Squared Error: {mse_reg_lr}')

r2_lr = r2_score(y_test_reg, y_pred_reg_lr)
print(f'Linear Regression R-squared (R²): {r2_lr}')
```

```
Linear Regression Mean Squared Error: 1.5690721649484536
Linear Regression R-squared (R²): 0.8446487724186698
```

[36]:
```
# Create results DataFrame for regression
results_reg_lr = pd.DataFrame({'Actual_goals': y_test_reg, 'Predicted': y_pred_reg_lr})
results_reg_lr['Player_Name'] = df['Player']
print(results_reg_lr.head(10))
```

	Actual_goals	Predicted	Player_Name
1765	1	1	Ignasi Miquel
457	0	0	Isaac Carcelen
1675	2	0	Christian Mawissa

DECISION TREE CLASSIFACATION

Decision Tree

[23]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
import pandas as pd

# Standardizing the classification data
scaler_class = StandardScaler()
X_train_class_scaled = scaler_class.fit_transform(X_train_class)
X_test_class_scaled = scaler_class.transform(X_test_class)

# Decision Tree Classifier
model_classification_dt = DecisionTreeClassifier(random_state=42)
model_classification_dt.fit(X_train_class_scaled, y_train_class)

# Predict using the model
y_pred_class_dt = model_classification_dt.predict(X_test_class_scaled)

# Evaluate the model
accuracy_class_dt = accuracy_score(y_test_class, y_pred_class_dt)
print(f'Decision Tree Classification Accuracy: {accuracy_class_dt}')
```

```
Decision Tree Classification Accuracy: 0.6989690721649484
```

[24]:
```
# Create results DataFrame for classification
results_class_dt = pd.DataFrame({'Actual_Pos': y_test_class, 'Predicted': y_pred_class_dt})
results_class_dt['Player_Name'] = df['Player']
print(results_class_dt.head(10))
```

```
     Actual_Pos Predicted             Player_Name
1765         DF        DF            Ignasi Miquel
457          DF        MF           Isaac Carcelen
1675         DF        MF        Christian Mawissa
486          DF        DF  Jean-Charles Castelletto
576          MF        FW            Maxwel Cornet
106          MF        FW          Elliot Anderson
1090         DF        DF         Massadio Haïdara
1166         DF        FW             Dean Huijsen
1606         DF        MF         Faitout Maouassa
```

REGRESSION

DECISION TREE

[25]:
```python
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
import numpy as np
import pandas as pd

# Standardizing the regression data
scaler_reg = StandardScaler()
X_train_reg_scaled = scaler_reg.fit_transform(X_train_reg)
X_test_reg_scaled = scaler_reg.transform(X_test_reg)

# Decision Tree Regressor for regression
model_regression_dt = DecisionTreeRegressor(random_state=42)
model_regression_dt.fit(X_train_reg_scaled, y_train_reg)

# Predict using the model
y_pred_reg_dt = model_regression_dt.predict(X_test_reg_scaled)

# Ensure predictions are non-negative
y_pred_reg_dt = np.maximum(y_pred_reg_dt, 0)

# Convert predictions to int
y_pred_reg_dt = y_pred_reg_dt.astype(int)

# Evaluate the model
mse_reg_dt = mean_squared_error(y_test_reg, y_pred_reg_dt)
print(f'Decision Tree Regression Mean Squared Error: {mse_reg_dt}')
r2_dt = r2_score(y_test_reg, y_pred_reg_dt)
print(f'Decision Tree Regression R-squared (R²): {r2_dt}')
```

```
Decision Tree Regression Mean Squared Error: 3.063917525773196
Decision Tree Regression R-squared (R²): 0.6966466173641831
```

[26]:
```python
# Create results DataFrame for regression
results_reg_dt = pd.DataFrame({'Actual_goals': y_test_reg, 'DecisionTree_Pred': y_pred_reg_dt})
results_reg_dt['Player_Name'] = df['Player']
print(results_reg_dt.head(10))
```

```
      Actual_goals  DecisionTree_Pred             Player_Name
1765             1                  0           Ignasi Miquel
457              0                  1          Isaac Carcelen
1675             2                  1       Christian Mawissa
486              2                  0  Jean-Charles Castelletto
```

RANDOM FOREST CLASSIFICATION

[18]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
import pandas as pd

# Standardizing the classification data
scaler_class = StandardScaler()
X_train_class_scaled = scaler_class.fit_transform(X_train_class)
X_test_class_scaled = scaler_class.transform(X_test_class)

# Random Forest Classifier
model_classification_rf = RandomForestClassifier(n_estimators=100, random_state=42)
model_classification_rf.fit(X_train_class_scaled, y_train_class)

# Predict using the model
y_pred_class_rf = model_classification_rf.predict(X_test_class_scaled)

# Evaluate the model
accuracy_class_rf = accuracy_score(y_test_class, y_pred_class_rf)
print(f'Random Forest Classification Accuracy: {accuracy_class_rf}')
```

```
Random Forest Classification Accuracy: 0.756701030927835
```

[19]:
```
# Create results DataFrame for classification
results_class_rf = pd.DataFrame({'Actual_Pos': y_test_class, 'Predicted': y_pred_class_rf})
results_class_rf['Player_Name'] = df['Player']
print(results_class_rf.head(10))
```

```
     Actual_Pos Predicted             Player_Name
1765         DF        DF           Ignasi Miquel
457          DF        DF           Isaac Carcelen
1675         DF        DF        Christian Mawissa
486          DF        MF  Jean-Charles Castelletto
576          MF        FW            Maxwel Cornet
```

RANDOM FOREST REGRESSION

```
# Standardizing the regression data
scaler_reg = StandardScaler()
X_train_reg_scaled = scaler_reg.fit_transform(X_train_reg)
X_test_reg_scaled = scaler_reg.transform(X_test_reg)

# Random Forest Regressor
model_regression_rf = RandomForestRegressor(random_state=42)
model_regression_rf.fit(X_train_reg_scaled, y_train_reg)

# Predict using the model
y_pred_reg_rf = model_regression_rf.predict(X_test_reg_scaled)

# Ensure predictions are non-negative
y_pred_reg_rf = np.maximum(y_pred_reg_rf, 0)
# Convert predictions to int
y_pred_reg_rf = y_pred_reg_rf.astype(int)

# Evaluate the model
mse_reg_rf = mean_squared_error(y_test_reg, y_pred_reg_rf)
print(f'Random Forest Regression Mean Squared Error: {mse_reg_rf}')

r2_rf = r2_score(y_test_reg, y_pred_reg_rf)
print(f'Random Forest Regression R-squared (R²): {r2_rf}')
```

```
Random Forest Regression Mean Squared Error: 1.688659793814433
Random Forest Regression R-squared (R²): 0.8328086000143109
```

[21]:

```
# Create results DataFrame for regression
results_reg_rf = pd.DataFrame({'Actual_goals': y_test_reg, 'Predicted': y_pred_reg_rf})
results_reg_rf['Player_Name'] = df['Player']
print(results_reg_rf.head(10))
```

```
      Actual_goals  Predicted             Player_Name
1765             1          1           Ignasi Miquel
457              0          0          Isaac Cancelo
```

NEURAL NETWORK

CLASSIFICATION

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
import pandas as pd
import numpy as np


# Standardizing the data
scaler = StandardScaler()
X_train_class_scaled = scaler.fit_transform(X_train_class)
X_test_class_scaled = scaler.transform(X_test_class)

# Define the neural network model for multi-class classification
model_classification = Sequential()
model_classification.add(Dense(256, activation='relu', input_dim=X_train_class_scaled.shape[1]))
model_classification.add(BatchNormalization())
model_classification.add(Dropout(0.4))
model_classification.add(Dense(128, activation='relu'))
model_classification.add(BatchNormalization())
model_classification.add(Dropout(0.4))
model_classification.add(Dense(64, activation='relu'))
model_classification.add(BatchNormalization())
model_classification.add(Dropout(0.4))
model_classification.add(Dense(len(np.unique(y_train_class)), activation='softmax'))

# Compile the model
optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)
model_classification.compile(optimizer=optimizer, loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Train the model
model_classification.fit(X_train_class_scaled, y_train_class, epochs=100, batch_size=32, verbose=1)

# Predict using the model
y_pred_class_probs = model_classification.predict(X_test_class_scaled)
y_pred_class = np.argmax(y_pred_class_probs, axis=1)

# Evaluate the model
accuracy_class_NN = accuracy_score(y_test_class, y_pred_class)
print(f'Classification Accuracy: {accuracy_class_NN}')
```

NEURAL NETWORK CLASSIFICATION

```
Classification Accuracy: 0.7505154639175258
```

```
# Create results DataFrame
results = pd.DataFrame({'Actual_Pos': y_test_class, 'Predicted': y_pred_class})
results['Player_Name'] = df['Player']
print(results.head(10))
```

```
      Actual_Pos  Predicted              Player_Name
1765           0          0           Ignasi Miquel
457            0          0           Isaac Carcelen
1675           0          0        Christian Mawissa
486            0          3  Jean-Charles Castelletto
576            3          3            Maxwel Cornet
106            3          0          Elliot Anderson
1090           0          0         Massadio Haïdara
1166           0          3             Dean Huijsen
1606           0          1         Faitout Maouassa
194            0          0          Alejandro Balde
```

NEURAL NETWORK

REGRESSION

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping

# Standardizing the data
scaler = StandardScaler()
X_train_reg_scaled = scaler.fit_transform(X_train_reg)
X_test_reg_scaled = scaler.transform(X_test_reg)
# Define the improved neural network model for regression
model_regression = Sequential()
model_regression.add(Dense(256, activation='relu', input_dim=X_train_reg_scaled.shape[1]))
model_regression.add(BatchNormalization())
model_regression.add(Dropout(0.4))
model_regression.add(Dense(128, activation='relu'))
model_regression.add(BatchNormalization())
model_regression.add(Dropout(0.4))
model_regression.add(Dense(64, activation='relu'))
model_regression.add(BatchNormalization())
model_regression.add(Dropout(0.4))
model_regression.add(Dense(1, activation='linear'))
# Compile the model
optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)
model_regression.compile(optimizer=optimizer, loss='mean_squared_error')

# Early stopping to avoid overfitting
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

# Train the model with validation split
model_regression.fit(X_train_reg_scaled, y_train_reg, epochs=100, batch_size=32, validation_split=0.2, callbacks=[early_stopping], ve

# Predict using the model
y_pred_reg = model_regression.predict(X_test_reg_scaled)

# Ensure predictions are non-negative
y_pred_reg = np.maximum(y_pred_reg, 0)

# Convert predictions to int
y_pred_reg = y_pred_reg.astype(int)

# Evaluate the model
mse_reg_NN = mean_squared_error(y_test_reg, y_pred_reg)
print(f'Regression Mean Squared Error: {mse_reg_NN}')

r2_NN = r2_score(y_test_reg, y_pred_reg)
print(f'R-squared (R²): {r2_NN}')
```

NEURAL NETWORK

REGRESSION

```
Regression Mean Squared Error: 1.97319587628866
R-squared (R²): 0.8046371553280776
```

```
# Create results DataFrame
results = pd.DataFrame({'Actual_goals': y_test_reg, 'Predicted': y_pred_reg.ravel()})
results['Player_Name'] = df['Player']
print(results.head(10))
```

	Actual_goals	Predicted	Player_Name
1765	1	0	Ignasi Miquel
457	0	0	Isaac Carcelen
1675	2	0	Christian Mawissa
486	2	0	Jean-Charles Castelletto
576	1	0	Maxwel Cornet
106	0	1	Elliot Anderson
1090	0	0	Massadio Haïdara
1166	2	0	Dean Huijsen
1606	0	0	Faitout Maouassa
194	0	0	Alejandro Balde

SVM

CLASSIFICATION

SVM model

[28]:
```
from sklearn.svm import SVC, SVR
from sklearn.metrics import accuracy_score, mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
import pandas as pd

# Standardizing the classification data
scaler_class = StandardScaler()
X_train_class_scaled = scaler_class.fit_transform(X_train_class)
X_test_class_scaled = scaler_class.transform(X_test_class)

# Building the SVM model for classification
model_classification = SVC(kernel='rbf', random_state=42)
model_classification.fit(X_train_class_scaled, y_train_class)

# Predict using the model
y_pred_class = model_classification.predict(X_test_class_scaled)

# Evaluate the model
accuracy_class_SVC = accuracy_score(y_test_class, y_pred_class)
print(f'Classification Accuracy: {accuracy_class_SVC}')
```

```
Classification Accuracy: 0.7237113402061855
```

[29]:
```
# Create results DataFrame for classification
results_class = pd.DataFrame({'Actual_Pos': y_test_class, 'Predicted': y_pred_class})
results_class['Player_Name'] = df['Player']
print(results_class.head(10))
```

```
     Actual_Pos Predicted             Player_Name
1765         DF        DF            Ignasi Miquel
457          DF        DF           Isaac Carcelen
1675         DF        DF        Christian Mawissa
486          DF        DF  Jean-Charles Castelletto
576          MF        FW            Maxwel Cornet
106          MF        DF          Elliot Anderson
1090         DF        DF         Massadio Haïdara
```

SVM

REGRESSION

[30]:
```python
# Standardizing the regression data
scaler_reg = StandardScaler()
X_train_reg_scaled = scaler_reg.fit_transform(X_train_reg)
X_test_reg_scaled = scaler_reg.transform(X_test_reg)

# Building the SVM model for regression
model_regression = SVR(kernel='rbf')
model_regression.fit(X_train_reg_scaled, y_train_reg)

# Predict using the model
y_pred_reg = model_regression.predict(X_test_reg_scaled)

# Ensure predictions are non-negative
y_pred_reg = np.maximum(y_pred_reg, 0)

# Convert predictions to int
y_pred_reg = y_pred_reg.astype(int)

# Evaluate the model
mse_reg_SVR = mean_squared_error(y_test_reg, y_pred_reg)
print(f'Regression Mean Squared Error: {mse_reg_SVR}')

r2_SVR = r2_score(y_test_reg, y_pred_reg)
print(f'R-squared (R²): {r2_SVR}')
```

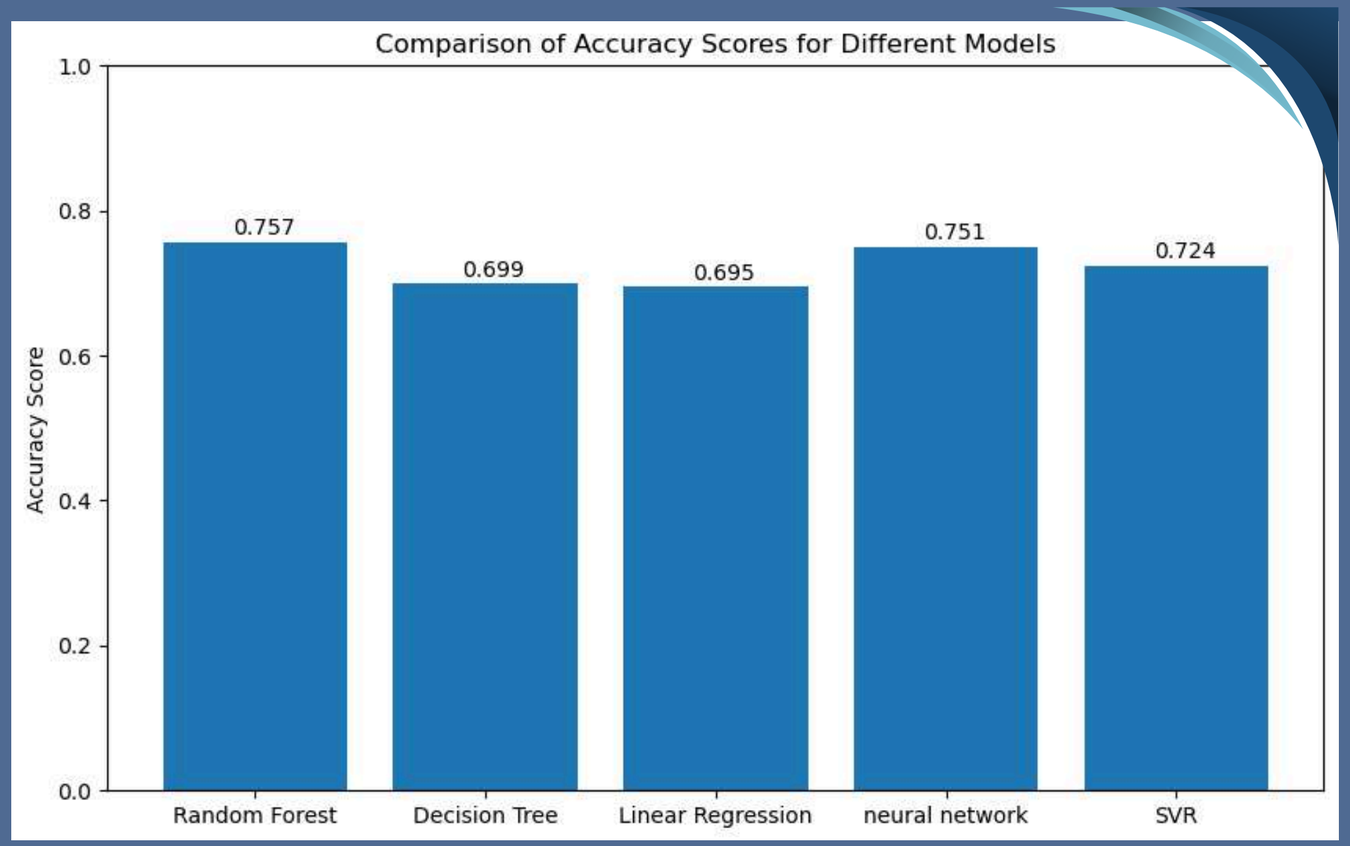
```
Regression Mean Squared Error: 3.181443298969072
R-squared (R²): 0.6850105858633476
```

[31]:
```python
# Create results DataFrame for regression
results_reg = pd.DataFrame({'Actual_goals': y_test_reg, 'Predicted': y_pred_reg})
results_reg['Player_Name'] = df['Player']
print(results_reg.head(10))
```

```
      Actual_goals  Predicted             Player_Name
1765             1          0           Ignasi Miquel
457              0          0           Isaac Carcelen
1675             2          0        Christian Mawissa
486              2          0  Jean-Charles Castelletto
576              1          0           Maxwel Cornet
106              0          1          Elliot Anderson
1090             0          0         Massadio Haïdara
```

COMPARISON BETWEEN MODELS

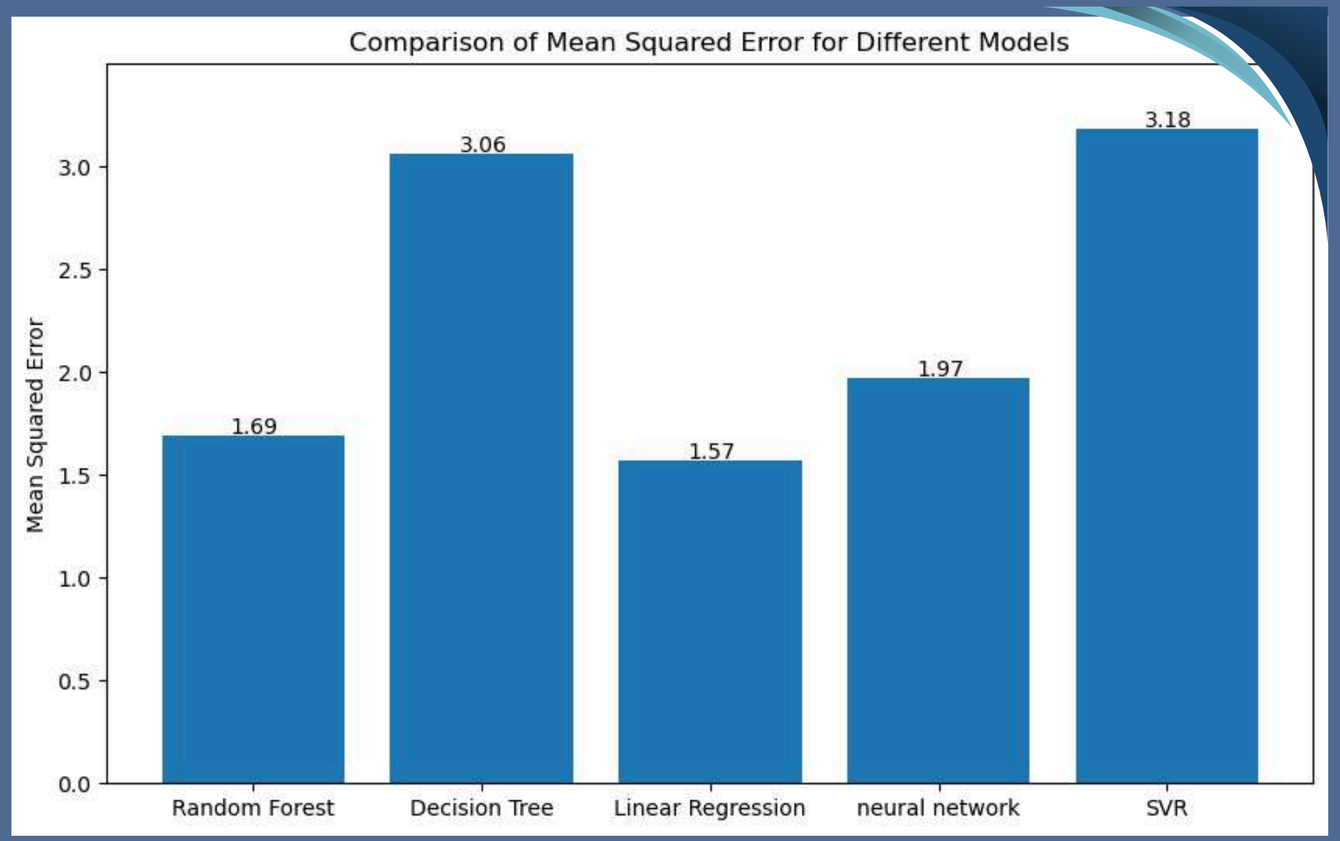
ACCURACY



RANDOM FOREST IS THE BEST ONE FOR CLASSIFACATION

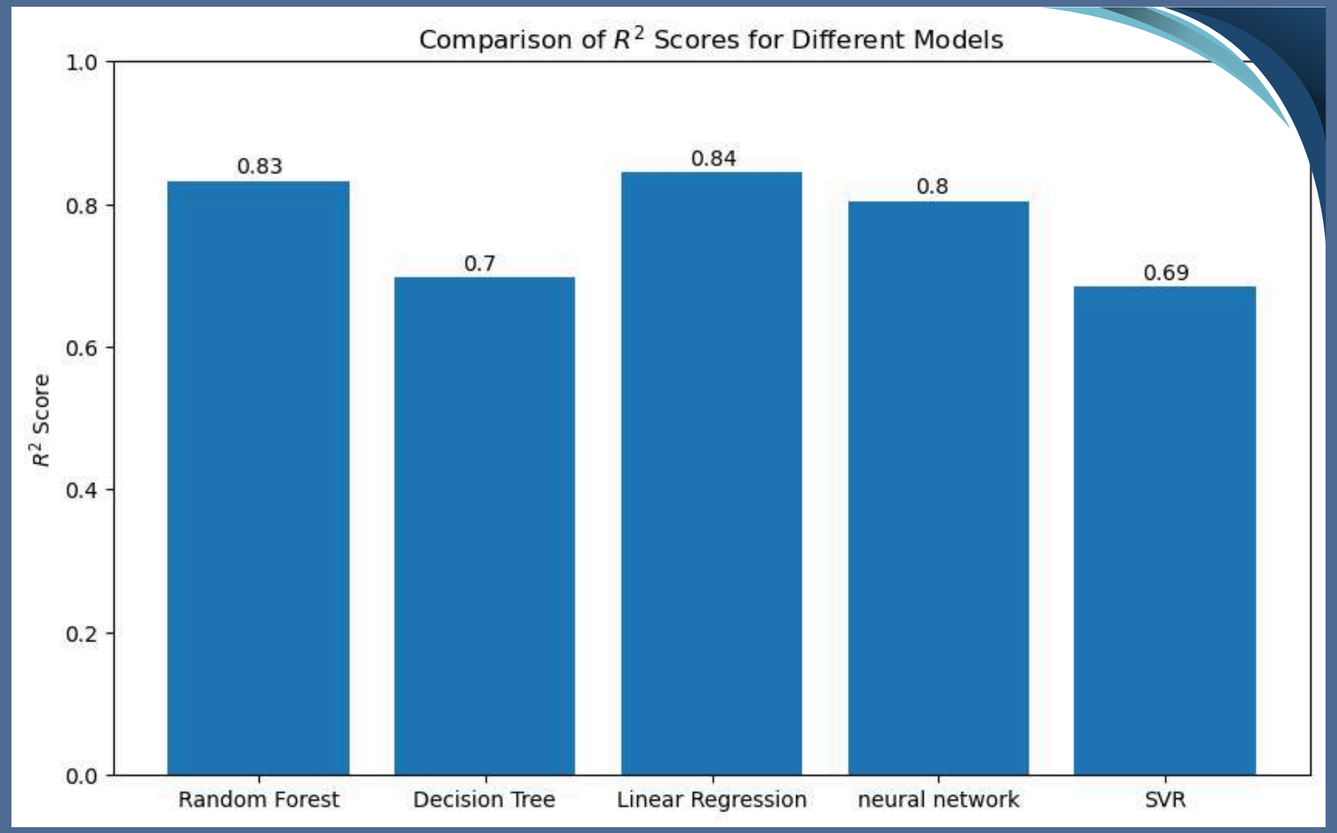
COMPARISON BETWEEN MODELS

MEAN SQUARED ERROR



COMPARISON BETWEEN MODELS

R2 SCORES



LINEARREGRESSION IS THE BEST ONE FOR REGRESSION

Thank You