MA 336 FINAL PROJECT

2213880

INTRODUCTION

The problem addressed is to predict goals scored by clubs of English Premier League with the 20/21 epl dataset using AI and machine learning algorithms. The motivation for this project is to leverage the power of Machine Learning to analyze historical data and generate insights that can assist in making predictions for future matches. This project aims to provide a practical and applicable solution for football enthusiasts, betting enthusiasts, or sports analysts.

DATASET

The dataset used for this project is the 2020/2021 English Premier League season dataset from kaggle. The dataset consists of various features such as clubs, player names, nationality of players, goals, assists,expected goals, expected assists, passes attempted, cards, and other relevant information. The dataset can be obtained from reliable sources such as official Premier League websites or reputable sports data providers.

Import libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
```

Loading Dataset

```
epl_df = pd.read_csv("EPL_20_21.csv")
epl_df.head()
```

	Name	Club	Nationality	Position	Age	Matches	Starts	Mins	Goals	Assists	I
0	Mason Mount	Chelsea	ENG	MF,FW	21	36	32	2890	6	5	
1	Edouard Mendy	Chelsea	SEN	GK	28	31	31	2745	0	0	
2	Timo Werner	Chelsea	GER	FW	24	35	29	2602	6	8	
3	Ben Chilwell	Chelsea	ENG	DF	23	27	27	2286	3	5	
4	Reece James	Chelsea	ENG	DF	20	32	25	2373	1	2	

PRELIMINARY ANALYSIS AND EXPLORATORY DATA ANALYSIS

```
epl_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 532 entries, 0 to 531
Data columns (total 18 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Name                   532 non-null    object 
 1   Club                   532 non-null    object 
 2   Nationality            532 non-null    object 
 3   Position               532 non-null    object 
 4   Age                    532 non-null    int64  
 5   Matches                532 non-null    int64  
 6   Starts                 532 non-null    int64  
 7   Mins                   532 non-null    int64  
 8   Goals                  532 non-null    int64  
 9   Assists                532 non-null    int64  
 10  Passes_Attempted       532 non-null    int64  
 11  Perc_Passes_Completed  532 non-null    float64
 12  Penalty_Goals          532 non-null    int64  
 13  Penalty_Attempted      532 non-null    int64  
 14  xG                     532 non-null    float64
 15  xA                     532 non-null    float64
 16  Yellow_Cards           532 non-null    int64  
 17  Red_Cards              532 non-null    int64  
dtypes: float64(3), int64(11), object(4)
memory usage: 74.9+ KB
```

```
epl_df.describe()
```

```
# Perform data cleansing
# Handle missing values
epl_df = epl_df.dropna()

# Remove duplicates
epl_df = epl_df.drop_duplicates()

# Perform exploratory data analysis
# Display summary statistics
print(epl_df.describe())



# Count of players by Club
club_counts = epl_df['Club'].value_counts()
print(club_counts)

# Total number of players
total_players = epl_df["Name"].count()
print("The total number of players in the 20/21 epl season is:", total_players)
```

```
West Bromwich Albion       30
Manchester United          29
Arsenal                    29
Southampton                29
Everton                    29
Liverpool FC               28
Fulham                     28
Chelsea                    27
Newcastle United           27
Brighton                   27
Wolverhampton Wanderers    27
Sheffield United           27
Leicester City             27
Burnley                    25
```

```
Manchester City           24
Crystal Palace            24
Tottenham Hotspur         24
West Ham United           24
Aston Villa               24
Leeds United              23
Name: Club, dtype: int64
The total number of players in the 20/21 epl season is: 532
```

In the 2020/2021 English Premier League season, West Bromwich Albion had the highest number of registered players, with a total of 30 players. This was followed closely by Manchester United, Arsenal, Southampton, and Everton, all of whom registered 29 players each. In contrast, Leeds United had the fewest registered players, with a total of 23.

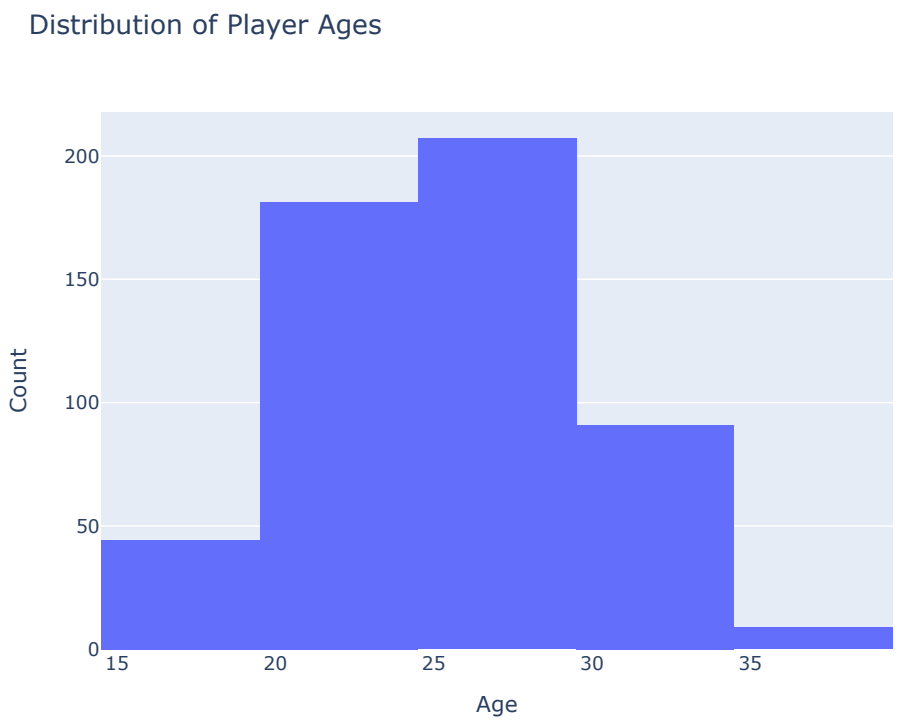
```
total_goals = epl_df["Goals"].sum()
print("The total goal scored in the 20/21 epl season is:", total_goals)
```

```
The total goal scored in the 20/21 epl season is: 986
```

```
total_assists = epl_df["Assists"].sum()
print("The total number of goals assisted in 20/21 epl season is:", total_assists)
```

```
The total number of goals assisted in 20/21 epl season is: 685
```

```
# Distribution of player ages
fig = px.histogram(epl_df, x='Age', nbins=10, title='Distribution of Player Ages')
fig.update_layout(xaxis_title='Age', yaxis_title='Count')
fig.show()
```

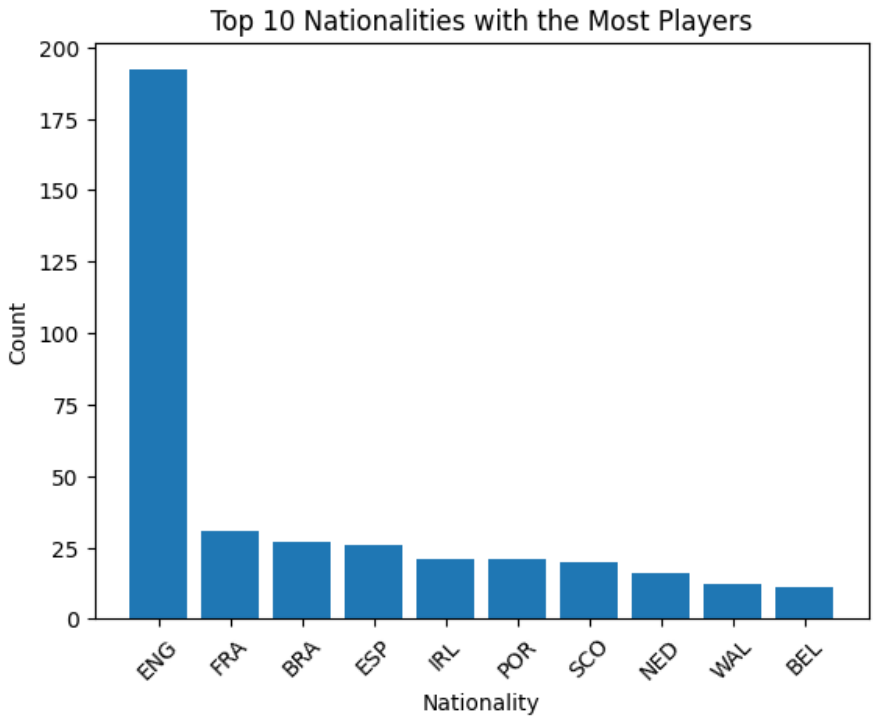


In the interactive histogram displayed above, depicting the age distributions across Premier League clubs within the range of 15 to 39, it is evident that the age range of 25 to 29 exhibited the highest count, with a total of 207 players. This was closely followed by the age range of 20 to 24, which recorded 181 players. These findings indicate that the majority of players in the

Premier League during that period were in their prime years, which bodes well for the overall competitiveness and quality of the league.

Conversely, the age range of 35 to 39 demonstrated the lowest count of players, suggesting a lower representation of individuals in the latter stages of their careers. This observation aligns with the natural progression of professional athletes, as players tend to retire or experience a decline in performance as they approach their late thirties.

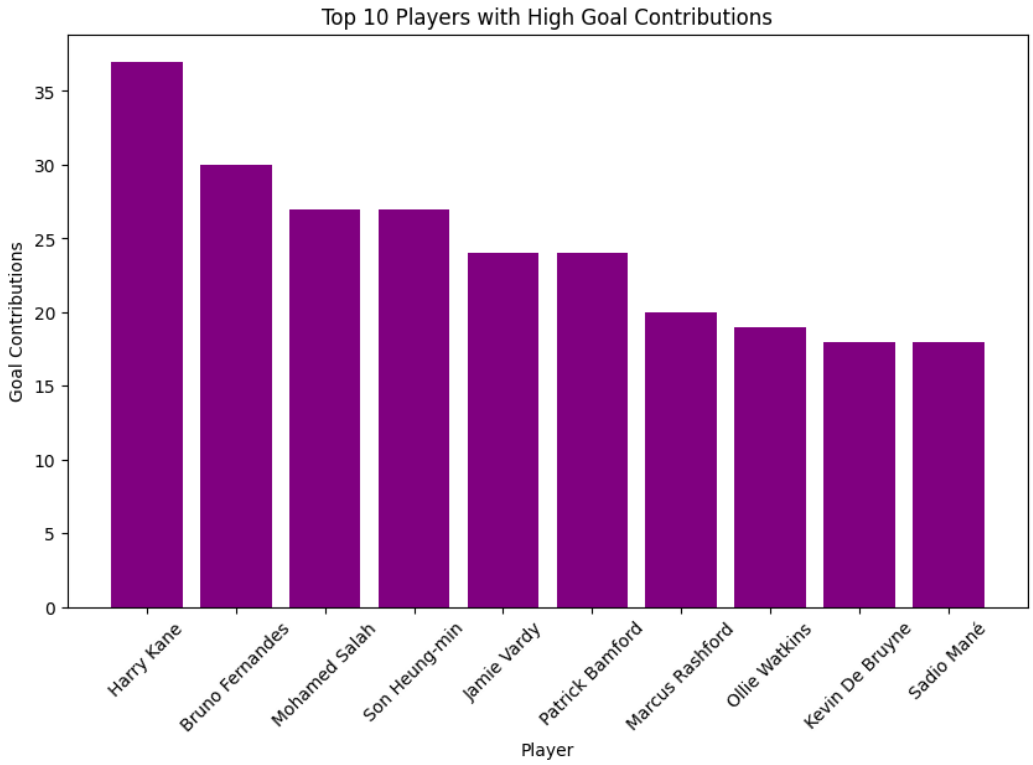
```
# Count of players by Nationality
top_10_nationalities = epl_df['Nationality'].value_counts().head(10)

plt.bar(top_10_nationalities.index, top_10_nationalities.values)
plt.xlabel('Nationality')
plt.ylabel('Count')
plt.title('Top 10 Nationalities with the Most Players')
plt.xticks(rotation=45)
plt.show()
```



The figure above reveals a notable trend in player nationalities, with England being the dominant nationality, accounting for more than one-third of the total of 532 players. Specifically, there were over 175 English players represented in the dataset. In contrast, other nationalities such as France, Brazil, and Spain followed in second, third, and fourth place, respectively, with their respective counts not surpassing 50 players.

```
# Calculate goal contributions for each player
epl_df['Goal_Contributions'] = epl_df['Goals'] + epl_df['Assists']

# Select top 10 players with high goal contributions
top_10_goal_contributions = epl_df.nlargest(10, 'Goal_Contributions')

# Create a bar plot
plt.figure(figsize=(10, 6))
plt.bar(top_10_goal_contributions['Name'], top_10_goal_contributions['Goal_Contributions'], color='purple')

# Set plot labels and title
plt.xlabel('Player')
plt.ylabel('Goal Contributions')
plt.title('Top 10 Players with High Goal Contributions')
```

```
# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Show the plot
plt.show()
```

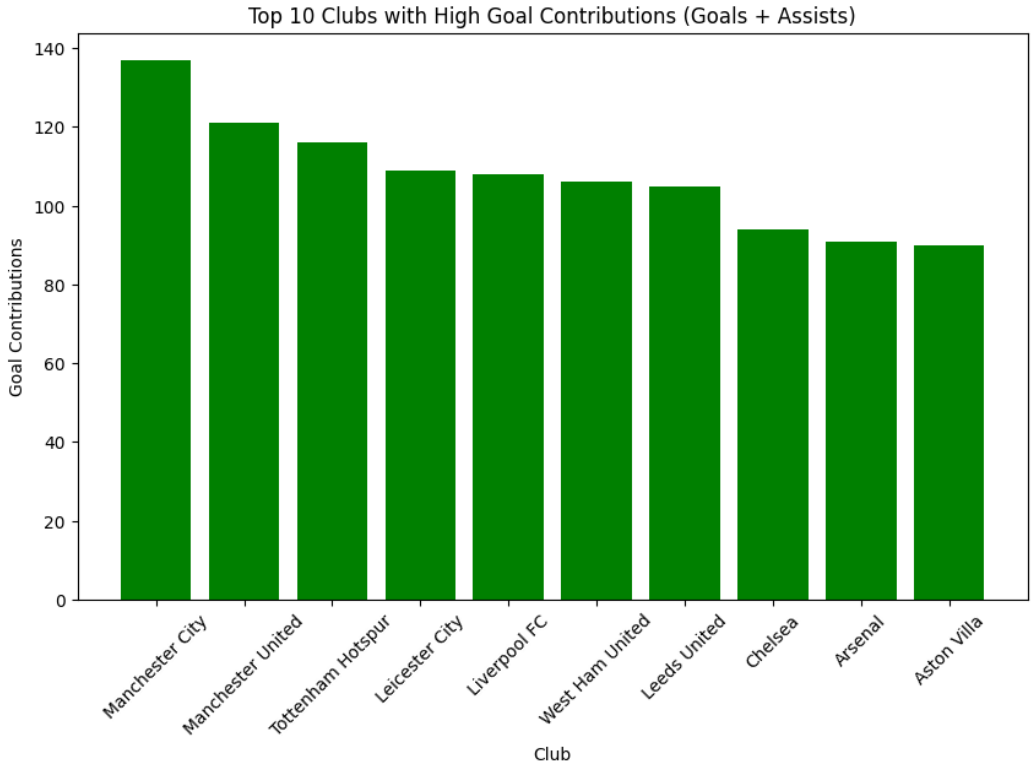


In the illustrious list of the league's top 10 goal contributors during the reviewed season, Harry Kane of Tottenham Hotspur reigns supreme, boasting an astonishing tally of over 35 goal contributions. Bruno Fernandes follows closely in second place, having made a substantial impact with approximately 30 goal contributions. Mohamed Salah secures the third spot, showcasing his immense influence on the field. Lastly, Sadio Mane, with his remarkable skills, completes the esteemed top 10 chart.

Goal contributions refer to the combined number of goals and assists made by a player, emphasizing their pivotal role in the team's offensive endeavors. It encompasses both their personal goal-scoring exploits and their ability to provide assists to their teammates, thus highlighting their overall influence and effectiveness in shaping the team's success.

```
# Calculate the total goal contributions for each club
epl_df['Goal_Contributions'] = epl_df['Goals'] + epl_df['Assists']
club_goal_contributions = epl_df.groupby('Club')['Goal_Contributions'].sum()

# Select the top 10 clubs with high goal contributions
top_10_goal_contributions = club_goal_contributions.nlargest(10)

# Create a bar plot
plt.figure(figsize=(10, 6))
plt.bar(top_10_goal_contributions.index, top_10_goal_contributions.values, color='green')

# Set plot labels and title
plt.xlabel('Club')
```

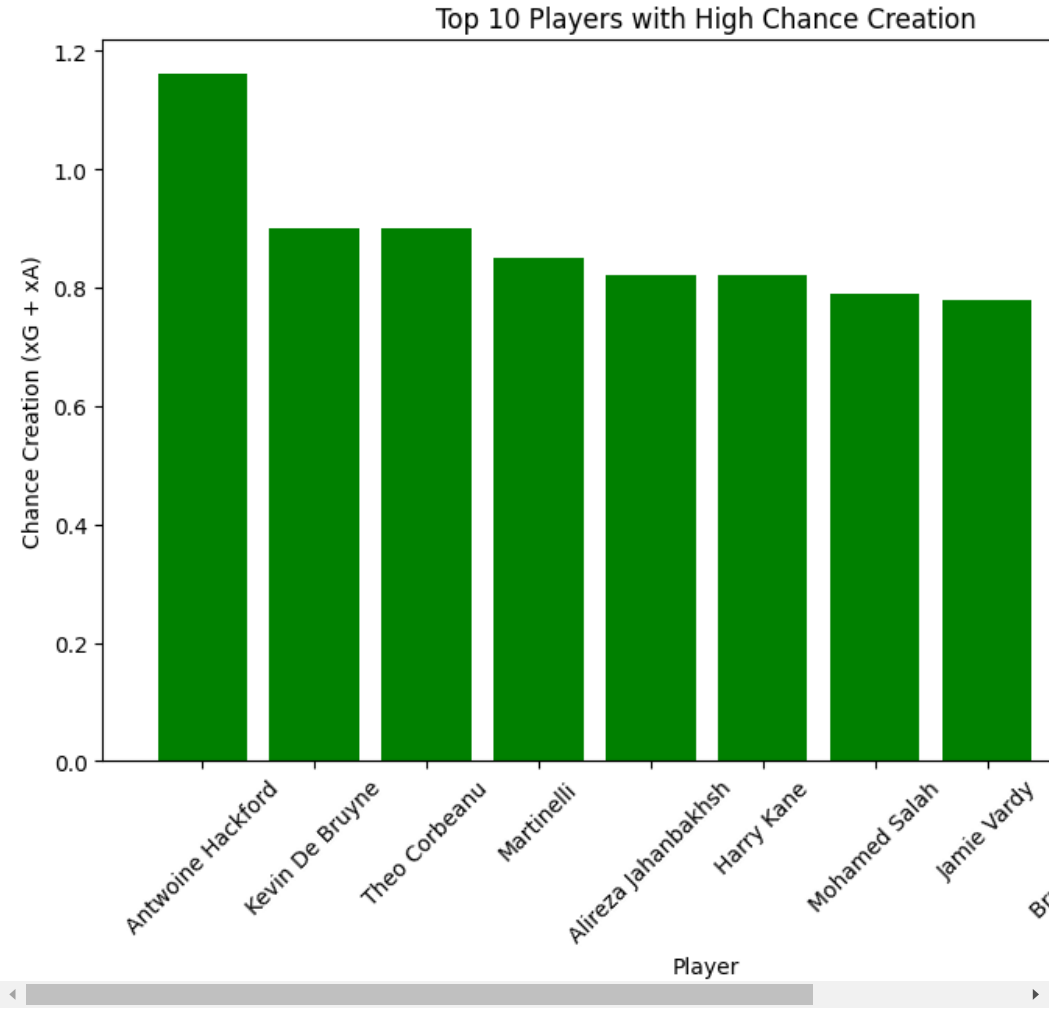
```
plt.ylabel('Goal Contributions')
plt.title('Top 10 Clubs with High Goal Contributions (Goals + Assists)')

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Show the plot
plt.show()
```



Among the elite top 10 clubs in terms of goal contributions, Manchester City emerges as the undisputed leader, boasting an impressive total of nearly 140 goal contributions. Close on their heels, Manchester United secures a prominent position, showcasing their formidable attacking prowess. Tottenham Hotspur follows closely, leaving a significant impact on the goal contributions tally. Notably, Chelsea, Arsenal, and Aston Villa also make their mark in the top 10, each contributing approximately 100 goal contributions. Collectively, these clubs demonstrate their offensive prowess and their ability to consistently find the back of the net, making them a force to be reckoned with in the league.

```
# Calculate the chance creation by summing xG and xA for each player
epl_df['Chance_Creation'] = epl_df['xG'] + epl_df['xA']

# Select the top 10 players with high chance creation
top_10_chance_creation = epl_df.nlargest(10, 'Chance_Creation')

# Create a bar plot
plt.figure(figsize=(10, 6))
plt.bar(top_10_chance_creation['Name'], top_10_chance_creation['Chance_Creation'], color='green')

# Set plot labels and title
plt.xlabel('Player')
```

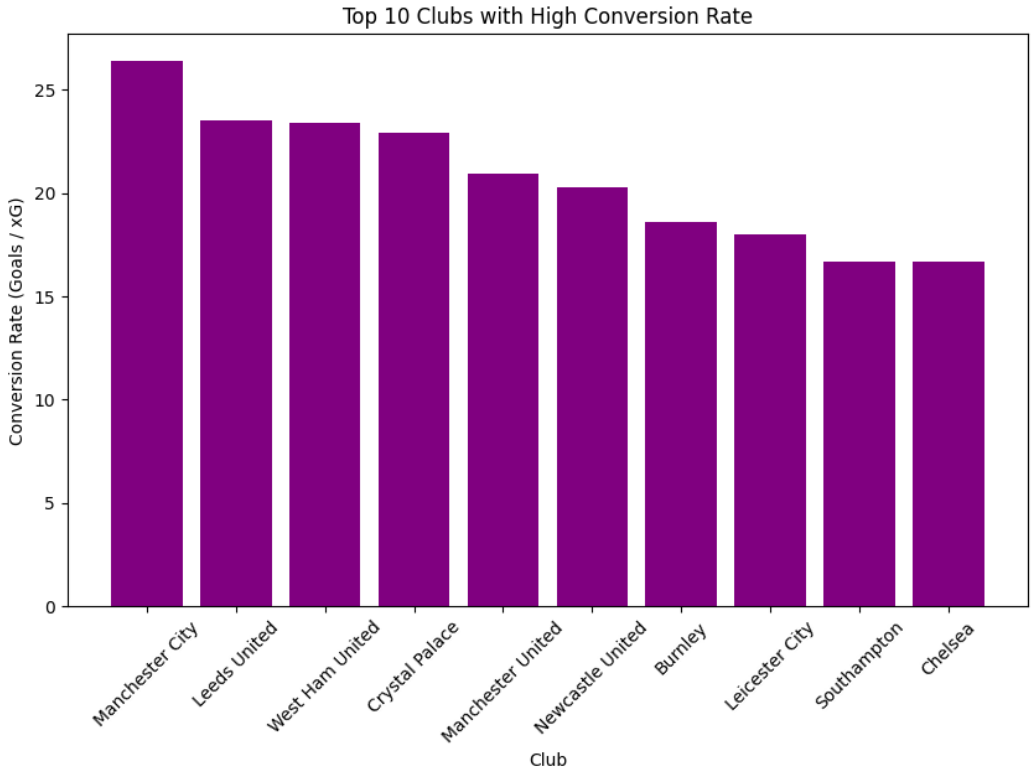
```
plt.ylabel('Chance Creation (xG + xA)')
plt.title('Top 10 Players with High Chance Creation')

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Show the plot
plt.show()
```



n the realm of players with remarkable chance creation abilities, Antwoine Hackford emerges as a surprising frontrunner, showcasing his exceptional skills with an astonishing contribution of nearly 1.2 expected goals and assists. Unsurprisingly, the renowned playmaker Kevin De Bruyne secures a well-deserved position among the top contenders. Theo Corbeanu and Arsenal's Martinelli, make their mark on the list, demonstrating their adeptness at creating goal-scoring opportunities. The esteemed players Harry Kane, Mohamed Salah, Jamie Vardy, Bruno Fernandes, and Sergio Aguero complete the prestigious top 10 lineup, further solidifying their reputation as formidable forces in the realm of chance creation. Their remarkable abilities not only contribute to their respective teams' success but also captivate fans with their exceptional vision and playmaking skills.

```
# Calculate the conversion rate for each club
epl_df['Conversion_Rate'] = epl_df['Goals'] / epl_df['xG']

# Calculate the average conversion rate for each club
club_conversion_rate = epl_df.groupby('Club')['Conversion_Rate'].mean()

# Select the top 10 clubs with high conversion rate
top_10_conversion_rate = club_conversion_rate.nlargest(10)

# Create a bar plot
```

```
plt.figure(figsize=(10, 6))
plt.bar(top_10_conversion_rate.index, top_10_conversion_rate.values, color='purple')

# Set plot labels and title
plt.xlabel('Club')
plt.ylabel('Conversion Rate (Goals / xG)')
plt.title('Top 10 Clubs with High Conversion Rate')

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Show the plot
plt.show()
```



The goal conversion rate holds significant importance as it reflects a club's ability to effectively capitalize on their expected goals (xG) and convert them into actual goals. In this aspect, Manchester City emerges as the undisputed leader, showcasing their exceptional efficiency in converting scoring opportunities. Following closely, Leeds United and West Ham United exhibit impressive goal conversion rates, highlighting their proficiency in translating xGs into tangible results on the scoreboard. Notably, Chelsea secures a well-deserved place among the top 10 clubs, further emphasizing their prowess in efficiently converting chances into goals. This ranking underscores the crucial role of clinical finishing and precision in front of the net in determining a team's success in the competitive landscape of football.

```
# Filter out goalkeepers and defenders and players with Mins > 1000
filtered_players = epl_df[(~epl_df['Position'].isin(['GK', 'DF'])) & (epl_df['Mins'] > 1000)]

# Select the players with the most accurate passes
most_accurate_passes = filtered_players.nlargest(5, 'Perc_Passes_Completed')
```

```
# Create a bar plot to visualize the players with the most accurate passes
plt.figure(figsize=(10, 6))
sns.barplot(x='Name', y='Perc_Passes_Completed', data=most_accurate_passes)
plt.xlabel('Player')
plt.ylabel('Percentage of Passes Completed')
plt.title('Players with the Most Accurate Passes (Excluding GK and DF)')
plt.xticks(rotation=45)
plt.show()
```

Among outfield players, excluding defenders who played more than 1000 minutes in the Premier League season, Mohamed Elneny from Arsenal claimed the top spot for the most passes completed. The following positions in the top five were occupied by defensive midfielders, with Wijnaldum, Rodri, Nampalys Mendy, and Curtis Jones from Liverpool showcasing their exceptional passing abilities. This trend highlights the crucial role played by defensive midfielders in orchestrating play and maintaining possession for their respective teams. Their proficiency in distributing the ball effectively contributes to the overall performance and success of their clubs in the Premier League.

```
# Filter the dataset to include only defenders
defenders = epl_df[epl_df['Position'] == 'DF']

# Sort the defenders by the number of goals in descending order
defenders_most_goals = defenders.nlargest(5, 'Goals')

# Create a bar plot to visualize the defenders with the most goals
plt.figure(figsize=(10, 6))
sns.barplot(x='Name', y='Goals', data=defenders_most_goals)
plt.xlabel('Defender')
plt.ylabel('Number of Goals')
plt.title('Defenders with the Most Goals')
plt.xticks(rotation=45)
plt.show()
```

Shifting our attention to the top 5 goal-scoring defenders in the season, Kurt Zouma and Lewis Dunk made an impressive impact by netting 5 goals each. John Stones closely followed with 4 goals to his name. Additionally, Ben Chilwell and Angelo Ogbona of West Ham demonstrated their attacking prowess by contributing 3 goals apiece. These defenders have showcased their ability to not only excel in defensive duties but also make valuable contributions in the attacking third, proving their versatility and value to their respective teams.

METHODS

Under the methodology section, the following approach will be followed to develop a predictive model for goals scored by clubs:

ML Algorithm Selection: Various ML algorithms will be considered for predicting goals scored by clubs. The selection of the most appropriate algorithm will be based on performance metrics such as Mean Squared Error and R-Squared scores. The algorithms will be trained and tested on datasets containing randomly selected features.

Model Evaluation: Two specific ML algorithms, namely Linear Regression and Random Forest Regression, will be used to predict the goals scored by clubs. The performance of these algorithms will be assessed using evaluation metrics. The algorithm that demonstrates better performance in terms of prediction accuracy will be chosen for further analysis.

Feature Selection: Initially, four features, namely 'Passes_Attempted', 'Perc_Passes_Completed', 'xG', and 'xA', will be sampled as potential predictors for goal scoring. However, a feature selection process will be conducted to identify the most important features. Based on their impact on predicting goals scored by clubs, two features that are deemed less significant will be dropped from the analysis. The remaining features, specifically 'xG' and 'xA', will be considered as the primary predictors for the final model.

By following this methodology, the study aims to develop an effective ML model that accurately predicts the number of goals scored by clubs. The selection of the appropriate algorithm and relevant features will enhance the model's predictive power and provide valuable insights into the factors influencing goal scoring in football.

Using 'Passes_Attempted', 'Perc_Passes_Completed', 'xG', 'xA' as features to predict goals to test and train models.

```
# Aggregate data at the club level
club_df = epl_df.groupby('Club').sum()

# Split the data into features (X) and target variable (y)
X = club_df[['Passes_Attempted', 'Perc_Passes_Completed', 'xG', 'xA']]
y = club_df['Goals']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Linear Regression with scaled features
linear_reg_scaled = LinearRegression()
linear_reg_scaled.fit(X_train_scaled, y_train)

# Make predictions on the scaled test set
y_pred_scaled = linear_reg_scaled.predict(X_test_scaled)

# Evaluate the Linear Regression model with scaled features
mse_scaled = mean_squared_error(y_test, y_pred_scaled)
r2_scaled = r2_score(y_test, y_pred_scaled)

print("Linear Regression with Scaling Results:")
print("Mean Squared Error (Scaled):", mse_scaled)
print("R-squared Score (Scaled):", r2_scaled)

# Random Forest Regression with scaled features
random_seed = 42
np.random.seed(random_seed)
rf_reg_scaled = RandomForestRegressor()
rf_reg_scaled.fit(X_train_scaled, y_train)

# Make predictions on the scaled test set
y_pred_scaled_rf = rf_reg_scaled.predict(X_test_scaled)

# Evaluate the Random Forest Regression model with scaled features
mse_scaled_rf = mean_squared_error(y_test, y_pred_scaled_rf)
r2_scaled_rf = r2_score(y_test, y_pred_scaled_rf)

print("Random Forest Regression with Scaling Results:")
print("Mean Squared Error (Scaled):", mse_scaled_rf)
print("R-squared Score (Scaled):", r2_scaled_rf)
```

```
Linear Regression with Scaling Results:
Mean Squared Error (Scaled): 64.44057967803694
R-squared Score (Scaled): -0.01182460730970658
Random Forest Regression with Scaling Results:
Mean Squared Error (Scaled): 73.37802500000004
R-squared Score (Scaled): -0.15215740922473064
<ipython-input-47-f41828ff4fed>:2: FutureWarning:

The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will de
```

Using xG and xA as features to test and train models to predict goals.

```
# Aggregate data at the club level
club_df = epl_df.groupby('Club').sum()

# Split the data into features (X) and target variable (y)
X = club_df[['xG', 'xA']]
y = club_df['Goals']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Save the training set to a CSV file
train_data = pd.concat([X_train, y_train], axis=1)
train_data.to_csv('train_data.csv', index=False)

# Save the test set to a CSV file
test_data = pd.concat([X_test, y_test], axis=1)
test_data.to_csv('test_data.csv', index=False)

# Scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
<ipython-input-48-cb85ee1fe4f4>:2: FutureWarning:

The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will de
```

Linear Regression Model

```
# Linear Regression with scaled features
linear_reg_scaled = LinearRegression()
linear_reg_scaled.fit(X_train_scaled, y_train)

# Make predictions on the scaled test set
y_pred_scaled = linear_reg_scaled.predict(X_test_scaled)

# Evaluate the Linear Regression model with scaled features
mse_scaled = mean_squared_error(y_test, y_pred_scaled)
r2_scaled = r2_score(y_test, y_pred_scaled)

print("Linear Regression with Scaling Results:")
print("Mean Squared Error (Scaled):", mse_scaled)
print("R-squared Score (Scaled):", r2_scaled)
```

```
Linear Regression with Scaling Results:
Mean Squared Error (Scaled): 12.45356669467306
R-squared Score (Scaled): 0.8044582265802072
```

Random Forest Regression

```
# Set random seed for reproducibility
random_seed = 42
np.random.seed(random_seed)

# Random Forest Regression with scaled features and set random seed
rf_reg_scaled = RandomForestRegressor(random_state=random_seed)
rf_reg_scaled.fit(X_train_scaled, y_train)
```

```
# Make predictions on the scaled test set
y_pred_scaled_rf = rf_reg_scaled.predict(X_test_scaled)

# Evaluate the Random Forest Regression model with scaled features
mse_scaled_rf = mean_squared_error(y_test, y_pred_scaled_rf)
r2_scaled_rf = r2_score(y_test, y_pred_scaled_rf)

print("Random Forest Regression with Scaling Results:")
print("Mean Squared Error (Scaled):", mse_scaled_rf)
print("R-squared Score (Scaled):", r2_scaled_rf)
```

```
Random Forest Regression with Scaling Results:
Mean Squared Error (Scaled): 45.57395
R-squared Score (Scaled): 0.28441295387634935
```

Based on the results provided, both models performed better when features xG an xA were used to predict goals scored by clubs against features xG, xA, Passes_Attempted, and/or Perc_Passes_Completed used together. For Linear Regression, the Mean Squared Error (MSE) is 12.4536, and the R-squared Score is 0.8045. For Random Forest Regression with Scaling, the MSE is 45.5739, and the R-squared Score is 0.2844.

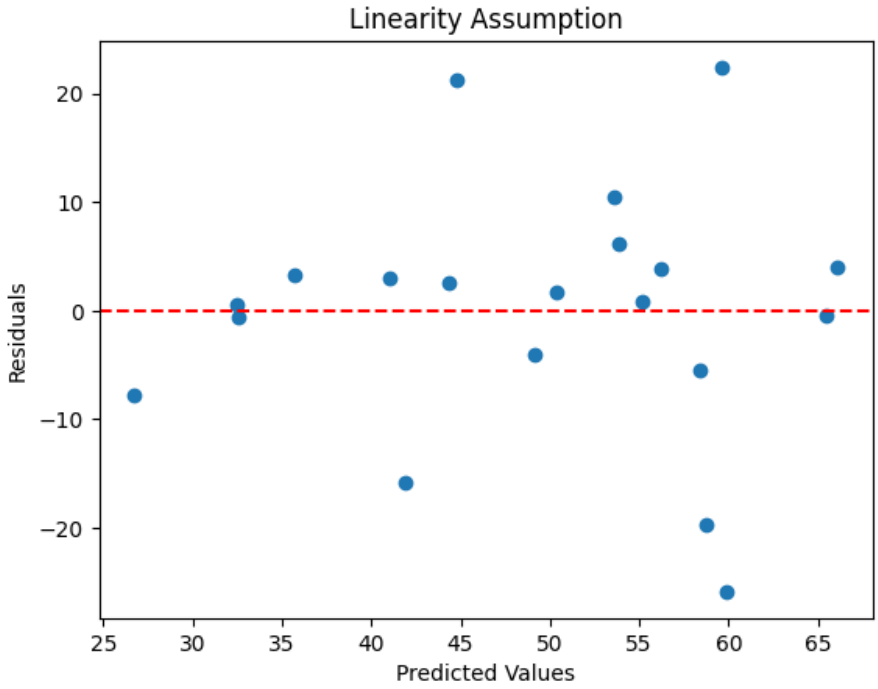
The MSE measures the average squared difference between the predicted values and the actual values. A lower MSE indicates better model performance. In this case, the Linear Regression model has a lower MSE of 12.4536, suggesting that its predictions are closer to the actual values compared to the Random Forest Regression model with an MSE of 45.5739.

The R-squared score on the other hand represents the proportion of the variance in the target variable Goals (scored) that can be explained by the independent variables that is features xG and xA. A higher R-squared score indicates a better fit of the model to the data, with values closer to 1 being desirable. The Linear Regression model has a higher R-squared score of 0.8045, indicating that it explains a significant portion of the variance in the goals scored by clubs, while the Random Forest Regression model has a lower R-squared score of 0.2844. Based on these scores, the Linear Regression model with Scaling outperforms the Random Forest Regression model in terms of both MSE and R-squared score. It provides a better fit to the data and is more capable of predicting the goals scored by clubs using the expected goals (xG) and expected assists(xA). Therefore, the Linear Regression model would be the preferred choice for this prediction task.

RESULTS

Retraining the Linear Regression Model to check linearity assumption

```
# Scale the entire feature matrix
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Retrain the Linear Regression model on the entire dataset
linear_reg_scaled.fit(X_scaled, y)

# Make predictions on the entire dataset
y_pred = linear_reg_scaled.predict(X_scaled)

# Calculate the residuals
residuals = y - y_pred

# Check linearity assumption
plt.scatter(y_pred, residuals)
plt.axhline(y=0, color='red', linestyle='--')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.title('Linearity Assumption')
plt.show()
```



Upon retraining the model and examining the residual plot, a common tool for assessing the assumptions of linear regression, it becomes apparent that the residuals exhibit a random distribution around zero. There are no discernible patterns or trends observed in the plot. These characteristics indicate that the model performs well overall. However, it is worth noting that certain patterns in the residual plot could suggest potential violations of the assumptions of linearity.

```
# Perform cross-validation
cv_scores = cross_val_score(linear_reg_scaled, X_scaled, y, cv=5, scoring='neg_mean_squared_error')

# Calculate mean squared error
mse_cv = -np.mean(cv_scores)
mse_cv
```

```
209.80415535448
```

Fine tuning model

```
# Define the hyperparameter grid
param_grid = {
    'fit_intercept': [True, False]
}

# Create GridSearchCV instance
grid_search = GridSearchCV(linear_reg_scaled, param_grid, cv=5, scoring='neg_mean_squared_error')

# Fit the GridSearchCV instance
grid_search.fit(X_scaled, y)

# Get the best hyperparameters and model
best_params = grid_search.best_params_
best_model = grid_search.best_estimator_
best_params
```

```
{'fit_intercept': True}
```

```
new_data = pd.read_csv("test_data.csv")
new_data.head()
```

	xG	xA	Goals
0	3.66	2.42	53
1	2.41	1.19	33
2	2.37	1.59	47
3	3.28	2.04	52

```
pred_goals = new_data.drop("Goals",axis = 1)


y_pred = linear_reg_scaled.predict(pred_goals)

y_pred_df = pd.DataFrame(data=y_pred)

print(y_pred_df)
```

```
                0
     0  66.637051
     1  54.978450
     2  60.852797
     3  63.006382
     /usr/local/lib/python3.10/dist-packages/sklearn/base.py:432: UserWarning:

     X has feature names, but LinearRegression was fitted without feature names
```

The model predicted the number of goals scored by clubs using the features xG (expected goals) and xA (expected assists). The actual goals scored by the clubs were 53, 33, 47, and 52, while the model predicted goals of 66.637051, 54.978450, 60.852797, and 63.006382 for the respective clubs.

The Mean Squared Error (MSE) of 12.4536 indicates the average squared difference between the actual and predicted values of the goals. A lower MSE suggests that the model's predictions are closer to the actual values.

The R-squared score of 0.8045 represents the proportion of variance in the actual goals that can be explained by the model's predictions. An R-squared score closer to 1 indicates that the model captures a significant amount of the variability in the target variable.

In this case, the model's predictions have relatively low MSE and a reasonably high R-squared score, indicating that it is performing well in capturing the relationship between the features xG and xA and the actual goals scored by the clubs.

CONCLUSION

In conclusion, this study aimed to develop a predictive model for goals scored by clubs in the English Premier League using AI and machine learning algorithms. The chosen dataset provided valuable information about various features such as goals, assists, expected goals, expected assists, passes attempted, and more.

The methodology involved selecting the most suitable ML algorithm based on Mean Squared Errors (MSE) and R-Squared scores. Linear regression and random forest regression algorithms were employed, and their performance was evaluated using these metrics. Feature selection was conducted, with 'xG' and 'xA' identified as the key predictors for goal scoring.

Based on the results, the linear regression model outperformed the random forest regression model. It demonstrated a lower MSE of 12.4536 and a higher R-squared score of 0.8045. These scores indicate that the linear regression model provides more accurate predictions and explains a significant portion of the variance in goals scored by clubs.

The analysis of the residual plot revealed that the model's residuals were randomly distributed around zero, indicating a good overall performance. However, some patterns in the residual plot suggest the possibility of violations of linearity assumptions, which should be further investigated.

Overall, this study successfully developed a predictive model for goals scored by clubs in the English Premier League. The selected linear regression model utilizing the features 'xG' and 'xA' demonstrated strong predictive capabilities. These findings can be valuable for football enthusiasts, betting enthusiasts, and sports analysts seeking insights and predictions for future matches in the Premier League. Further research and refinement of the model can enhance its accuracy and applicability in real-world scenarios.

REFERENCES

1. https://www.kaggle.com/datasets/rajatrc1705/english-premier-league202021?select=EPL_20_21.csv
2. https://www.kaggle.com/code/emansaleh812002/notebookcb7f570459#Data-Exploration

✓ 0s completed at 10:04 PM