comprehensive_team_comparison_eda

July 10, 2025

1 Manchester City vs Real Madrid - Comprehensive EDA

1.1 Overview

This notebook provides a comprehensive exploratory data analysis comparing Manchester City and Real Madrid's 2023-24 season performance across multiple dimensions:

- **Team Performance Analysis**
- **Player Statistics Comparison**
- **Tactical Analysis**
- **Competition Performance**
- **Advanced Metrics & Insights**

1.2 Setup and Data Loading

[1]:
```python
# Import required libraries
import pandas as pd
import numpy as np
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import warnings
warnings.filterwarnings('ignore')

# Configure display options
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)
pd.set_option('display.max_colwidth', None)

print("Libraries imported successfully!")
```

```
Libraries imported successfully!
```

[2]:
```python
# Load Manchester City data
mc_path = "../data/fbref_scraped/final_exports/"
mc_matches = pd.read_csv(f"{mc_path}manchester_city_match_results_2023_24.csv")
mc_performances = pd.
↪read_csv(f"{mc_path}manchester_city_player_match_performances_2023_24.csv")
```

```
mc_season_stats = pd.
    ↪read_csv(f"{mc_path}manchester_city_player_season_aggregates_2023_24.csv")
mc_competition = pd.
    ↪read_csv(f"{mc_path}manchester_city_competition_summary_2023_24.csv")

# Load Real Madrid data
rm_path = "../data/real_madrid_scraped/final_exports/"
rm_matches = pd.read_csv(f"{rm_path}real_madrid_match_results_2023_24.csv")
rm_performances = pd.
    ↪read_csv(f"{rm_path}real_madrid_player_match_performances_2023_24.csv")
rm_season_stats = pd.
    ↪read_csv(f"{rm_path}real_madrid_player_season_aggregates_2023_24.csv")
rm_competition = pd.read_csv(f"{rm_path}real_madrid_competition_summary_2023_24.
    ↪csv")

# Add team identifiers
mc_season_stats['team'] = 'Manchester City'
rm_season_stats['team'] = 'Real Madrid'
mc_matches['team'] = 'Manchester City'
rm_matches['team'] = 'Real Madrid'

print("Data loaded successfully!")
print(f"Manchester City: {len(mc_season_stats)} players, {len(mc_matches)}␣
    ↪matches")
print(f"Real Madrid: {len(rm_season_stats)} players, {len(rm_matches)} matches")
```

```
Data loaded successfully!
Manchester City: 31 players, 57 matches
Real Madrid: 24 players, 46 matches
```

1.3 Data Overview and Quality Check

[3]:
```
# Data structure overview
print("MANCHESTER CITY DATA STRUCTURE")
print("=" * 50)
print(f"Season Stats Shape: {mc_season_stats.shape}")
print(f"Match Results Shape: {mc_matches.shape}")
print(f"Player Performances Shape: {mc_performances.shape}")

print("\nREAL MADRID DATA STRUCTURE")
print("=" * 50)
print(f"Season Stats Shape: {rm_season_stats.shape}")
print(f"Match Results Shape: {rm_matches.shape}")
print(f"Player Performances Shape: {rm_performances.shape}")

# Check for missing values
print("\nMISSING VALUES CHECK")
```

```
print("=" * 50)
print("Manchester City Season Stats:")
print(mc_season_stats.isnull().sum().sum(), "total missing values")
print("\nReal Madrid Season Stats:")
print(rm_season_stats.isnull().sum().sum(), "total missing values")
```

```
MANCHESTER CITY DATA STRUCTURE
==================================================
Season Stats Shape: (31, 32)
Match Results Shape: (57, 33)
Player Performances Shape: (784, 37)

REAL MADRID DATA STRUCTURE
==================================================
Season Stats Shape: (24, 32)
Match Results Shape: (46, 33)
Player Performances Shape: (672, 37)

MISSING VALUES CHECK
==================================================
Manchester City Season Stats:
0 total missing values

Real Madrid Season Stats:
0 total missing values
```

[4]:

```
# Display sample data
print("SAMPLE PLAYER DATA")
print("=" * 50)
print("\nManchester City Top 5 Players (by minutes):")
display(mc_season_stats.nlargest(5, 'total_minutes')[['player_name',␣
↪'position', 'goals', 'assists', 'avg_rating', 'total_minutes']])

print("\nReal Madrid Top 5 Players (by minutes):")
display(rm_season_stats.nlargest(5, 'total_minutes')[['player_name',␣
↪'position', 'goals', 'assists', 'avg_rating', 'total_minutes']])
```

```
SAMPLE PLAYER DATA
==================================================

Manchester City Top 5 Players (by minutes):
```

	player_name	position	goals	assists	avg_rating	total_minutes
5	İlkay Gündoğan	MF	5	7	7.5	3276
1	Rodri	MF	9	11	7.6	3234
19	Ederson	GK	0	1	7.5	3011
0	Erling Haaland	FW	12	5	7.6	2990
6	Mateo Kovačić	MF	5	4	7.5	2979

```
Real Madrid Top 5 Players (by minutes):

     player_name position  goals  assists  avg_rating  total_minutes
0   Andriy Lunin       GK      0        0         7.1           2465
1  Dani Carvajal       DF      0        2         6.9           2163
2    David Alaba       DF      1        4         7.0           2122
3  Ferland Mendy       DF      1        4         7.1           2058
4   Éder Militão       DF      4        2         7.0           2053
```

1.4 Team Performance Comparison

[5]:
```python
# Calculate team-level statistics
def calculate_team_stats(season_stats, matches, team_name):
    """Calculate comprehensive team statistics."""

    # Basic team stats
    total_goals = season_stats['goals'].sum()
    total_assists = season_stats['assists'].sum()
    avg_rating = season_stats['avg_rating'].mean()
    total_minutes = season_stats['total_minutes'].sum()

    # Match statistics
    total_matches = len(matches)
    wins = len(matches[matches['result'] == 'Win'])
    draws = len(matches[matches['result'] == 'Draw'])
    losses = len(matches[matches['result'] == 'Loss'])
    win_rate = (wins / total_matches) * 100 if total_matches > 0 else 0

    # Goals scored and conceded
    goals_for = matches['manchester_city_score'].sum()  # Both teams use this␣
 ↪column name
    goals_against = matches['opponent_score'].sum()
    goal_difference = goals_for - goals_against

    return {
        'team': team_name,
        'total_goals': total_goals,
        'total_assists': total_assists,
        'avg_rating': round(avg_rating, 2),
        'total_matches': total_matches,
        'wins': wins,
        'draws': draws,
        'losses': losses,
        'win_rate': round(win_rate, 1),
        'goals_for': goals_for,
        'goals_against': goals_against,
        'goal_difference': goal_difference,
```

```
        'avg_goals_per_match': round(goals_for / total_matches, 2),
        'avg_goals_conceded': round(goals_against / total_matches, 2)
    }

# Calculate stats for both teams
mc_stats = calculate_team_stats(mc_season_stats, mc_matches, 'Manchester City')
rm_stats = calculate_team_stats(rm_season_stats, rm_matches, 'Real Madrid')

# Create comparison dataframe
team_comparison = pd.DataFrame([mc_stats, rm_stats])
team_comparison = team_comparison.set_index('team')

print(" TEAM PERFORMANCE COMPARISON")
print("=" * 60)
display(team_comparison)
```

```
 TEAM PERFORMANCE COMPARISON
============================================================

                 total_goals  total_assists  avg_rating  total_matches  wins  \
team
Manchester City           79             83        7.57             57    37
Real Madrid               63             71        7.38             46    28

                 draws  losses  win_rate  goals_for  goals_against  \
team
Manchester City     10      10      64.9        103             51
Real Madrid          8      10      60.9         90             55

                 goal_difference  avg_goals_per_match  avg_goals_conceded
team
Manchester City               52                 1.81                0.89
Real Madrid                   35                 1.96                1.20
```

[ ]:
```
# Create comprehensive team comparison visualization
fig = make_subplots(
    rows=2, cols=3,
    subplot_titles=('Win Rate Comparison', 'Goals Scored vs Conceded', 'Average␣
 ↪Team Rating',
                   'Match Results Distribution', 'Goal Difference', 'Goals per␣
 ↪Match'),
    specs=[[{"type": "bar"}, {"type": "scatter"}, {"type": "bar"}],
           [{"type": "pie"}, {"type": "bar"}, {"type": "bar"}]]
)

# Win Rate Comparison
fig.add_trace(
    go.Bar(x=team_comparison.index, y=team_comparison['win_rate'],
```

```
            name='Win Rate %', marker_color=['#6CABDD', '#FEBE10']),
    row=1, col=1
)

# Goals Scored vs Conceded
fig.add_trace(
    go.Scatter(x=team_comparison['goals_for'],␣
 ↪y=team_comparison['goals_against'],
               mode='markers+text', text=team_comparison.index,
               textposition="top center", marker_size=15,
               marker_color=['#6CABDD', '#FEBE10']),
    row=1, col=2
)

# Average Team Rating
fig.add_trace(
    go.Bar(x=team_comparison.index, y=team_comparison['avg_rating'],
           name='Avg Rating', marker_color=['#6CABDD', '#FEBE10']),
    row=1, col=3
)

# Match Results Distribution (Manchester City)
fig.add_trace(
    go.Pie(labels=['Wins', 'Draws', 'Losses'],
           values=[mc_stats['wins'], mc_stats['draws'], mc_stats['losses']],
           name="Manchester City", title="Manchester City"),
    row=2, col=1
)

# Goal Difference
fig.add_trace(
    go.Bar(x=team_comparison.index, y=team_comparison['goal_difference'],
           name='Goal Difference', marker_color=['#6CABDD', '#FEBE10']),
    row=2, col=2
)

# Goals per Match
fig.add_trace(
    go.Bar(x=team_comparison.index, y=team_comparison['avg_goals_per_match'],
           name='Goals/Match', marker_color=['#6CABDD', '#FEBE10']),
    row=2, col=3
)

fig.update_layout(height=800, showlegend=False, title_text= "Manchester City vs␣
 ↪Real Madrid - Team Performance Analysis")
fig.show()
```

1.5 Player Performance Analysis

[7]:
```python
# Combine both teams' data for comparison
combined_stats = pd.concat([mc_season_stats, rm_season_stats],␣
↪ignore_index=True)

# Top performers analysis
print("TOP PERFORMERS ANALYSIS")
print("=" * 50)

# Top scorers
top_scorers = combined_stats.nlargest(10, 'goals')[['player_name', 'team',␣
↪'goals', 'goals_per_90', 'position']]
print("\nTop 10 Goal Scorers:")
display(top_scorers)

# Top assisters
top_assisters = combined_stats.nlargest(10, 'assists')[['player_name', 'team',␣
↪'assists', 'assists_per_90', 'position']]
print("\nTop 10 Assist Providers:")
display(top_assisters)

# Highest rated players
top_rated = combined_stats.nlargest(10, 'avg_rating')[['player_name', 'team',␣
↪'avg_rating', 'position']]
print("\nTop 10 Highest Rated Players:")
display(top_rated)
```

```
TOP PERFORMERS ANALYSIS
==================================================
```

Top 10 Goal Scorers:

	player_name	team	goals	goals_per_90	position
0	Erling Haaland	Manchester City	12	0.36	FW
1	Rodri	Manchester City	9	0.25	MF
2	Bernardo Silva	Manchester City	8	0.25	MF,FW
42	Vinícius Jr.	Real Madrid	8	0.39	FW
44	Rodrygo	Real Madrid	8	0.40	FW
54	Mariano Díaz	Real Madrid	8	0.52	FW
3	Phil Foden	Manchester City	7	0.28	MF,FW
4	Joško Gvardiol	Manchester City	6	0.21	DF
45	Federico Valverde	Real Madrid	6	0.30	MF
5	İlkay Gündoğan	Manchester City	5	0.14	MF

Top 10 Assist Providers:

	player_name	team	assists	assists_per_90	position

```
1                Rodri  Manchester City       11          0.31        MF
2       Bernardo Silva  Manchester City        9          0.28     MF,FW
3           Phil Foden  Manchester City        8          0.32     MF,FW
5       İlkay Gündoğan  Manchester City        7          0.19        MF
8      Kevin De Bruyne  Manchester City        6          0.31     MF,FW
0       Erling Haaland  Manchester City        5          0.15        FW
36  Kepa Arrizabalaga       Real Madrid        5          0.22        GK
40       Karim Benzema      Real Madrid        5          0.24        FW
42        Vinícius Jr.      Real Madrid        5          0.24        FW
45  Federico Valverde       Real Madrid        5          0.25        MF

Top 10 Highest Rated Players:
            player_name             team  avg_rating position
22   Abdukodir Khusanov  Manchester City         8.5       DF
25         Jacob Wright  Manchester City         8.1       MF
29         Scott Carson  Manchester City         8.0       GK
42         Vinícius Jr.      Real Madrid         8.0       FW
20          Issa Kaboré  Manchester City         7.9    DF,MF
44              Rodrygo      Real Madrid         7.9       FW
27   Josh Wilson-Esbrand  Manchester City         7.8       DF
37          Eden Hazard      Real Madrid         7.8       FW
11        Manuel Akanji  Manchester City         7.7       DF
24         Divin Mubama  Manchester City         7.7       FW
```

[8]:
```python
# Player performance visualizations
fig = make_subplots(
    rows=2, cols=2,
    subplot_titles=('Goals vs Assists (Top Players)', 'Goals per 90 by␣
↪Position',
                    'Player Ratings Distribution', 'Minutes Played␣
↪Distribution'),
    specs=[[{"type": "scatter"}, {"type": "box"}],
           [{"type": "histogram"}, {"type": "histogram"}]]
)

# Goals vs Assists scatter plot
for team in ['Manchester City', 'Real Madrid']:
    team_data = combined_stats[combined_stats['team'] == team]
    color = '#6CABDD' if team == 'Manchester City' else '#FEBE10'

    fig.add_trace(
        go.Scatter(x=team_data['goals'], y=team_data['assists'],
                   mode='markers', name=team, marker_color=color,
                   text=team_data['player_name'],␣
↪hovertemplate='%{text}<br>Goals: %{x}<br>Assists: %{y}'),
        row=1, col=1
```

```
    )

# Goals per 90 by position
for team in ['Manchester City', 'Real Madrid']:
    team_data = combined_stats[combined_stats['team'] == team]

    fig.add_trace(
        go.Box(y=team_data['goals_per_90'], x=team_data['position'],
               name=f'{team}', boxpoints='all'),
        row=1, col=2
    )

# Player ratings distribution
fig.add_trace(
    go.Histogram(x=mc_season_stats['avg_rating'], name='Manchester City',
                 opacity=0.7, marker_color='#6CABDD'),
    row=2, col=1
)
fig.add_trace(
    go.Histogram(x=rm_season_stats['avg_rating'], name='Real Madrid',
                 opacity=0.7, marker_color='#FEBE10'),
    row=2, col=1
)

# Minutes played distribution
fig.add_trace(
    go.Histogram(x=mc_season_stats['total_minutes'], name='Manchester City',
                 opacity=0.7, marker_color='#6CABDD'),
    row=2, col=2
)
fig.add_trace(
    go.Histogram(x=rm_season_stats['total_minutes'], name='Real Madrid',
                 opacity=0.7, marker_color='#FEBE10'),
    row=2, col=2
)

fig.update_layout(height=800, title_text="  Player Performance Analysis")
fig.show()
```

1.6 Position-Based Analysis

[9]:
```
# Position-based performance analysis
def analyze_by_position(data, team_name):
    """Analyze performance metrics by position."""
    position_stats = data.groupby('position').agg({
        'goals': ['sum', 'mean'],
        'assists': ['sum', 'mean'],
```

```
        'avg_rating': 'mean',
        'goals_per_90': 'mean',
        'assists_per_90': 'mean',
        'total_minutes': 'sum',
        'player_name': 'count'
    }).round(2)

    position_stats.columns = ['Total_Goals', 'Avg_Goals', 'Total_Assists',␣
 ↪'Avg_Assists',
                             'Avg_Rating', 'Goals_per_90', 'Assists_per_90',␣
 ↪'Total_Minutes', 'Player_Count']
    position_stats['Team'] = team_name
    return position_stats

# Analyze both teams
mc_position_stats = analyze_by_position(mc_season_stats, 'Manchester City')
rm_position_stats = analyze_by_position(rm_season_stats, 'Real Madrid')

print("POSITION-BASED PERFORMANCE ANALYSIS")
print("=" * 60)
print("\n Manchester City by Position:")
display(mc_position_stats)

print("\n Real Madrid by Position:")
display(rm_position_stats)
```

```
POSITION-BASED PERFORMANCE ANALYSIS
============================================================

 Manchester City by Position:

          Total_Goals  Avg_Goals  Total_Assists  Avg_Assists  Avg_Rating  \
position
DF                 12       1.50             10         1.25        7.65
DF,MF               4       1.00              8         2.00        7.62
FW                 14       3.50              8         2.00        7.62
FW,MF              11       3.67             10         3.33        7.37
GK                  0       0.00              1         0.33        7.73
MF                 19       3.80             22         4.40        7.46
MF,FW              19       4.75             24         6.00        7.50

          Goals_per_90  Assists_per_90  Total_Minutes  Player_Count  \
position
DF                0.05            0.05          12781             8
DF,MF             0.05            0.12           6360             4
FW                0.12            0.08           5003             4
FW,MF             0.15            0.14           6499             3
GK                0.00            0.01           3984             3
```

```
MF                0.11            0.12          9749             5
MF,FW             0.18            0.29          7266             4

                     Team
position
DF         Manchester City
DF,MF      Manchester City
FW         Manchester City
FW,MF      Manchester City
GK         Manchester City
MF         Manchester City
MF,FW      Manchester City


 Real Madrid by Position:
          Total_Goals  Avg_Goals  Total_Assists  Avg_Assists  Avg_Rating  \
position
DF                 13       1.62             20         2.50        7.09
FW                 33       5.50             21         3.50        7.75
GK                  0       0.00              6         2.00        7.30
MF                 17       2.43             24         3.43        7.41

          Goals_per_90  Assists_per_90  Total_Minutes  Player_Count  \
position
DF                0.08            0.12          15219             8
FW                0.29            0.18          10824             6
GK                0.00            0.09           6265             3
MF                0.12            0.18          12150             7

                 Team
position
DF        Real Madrid
FW        Real Madrid
GK        Real Madrid
MF        Real Madrid
```

[10]:

```python
# Position comparison visualization
combined_position_stats = pd.concat([mc_position_stats, rm_position_stats])

fig = make_subplots(
    rows=2, cols=2,
    subplot_titles=('Goals per 90 by Position', 'Assists per 90 by Position',
                   'Average Rating by Position', 'Player Count by Position')
)

positions = combined_position_stats.index.unique()
teams = ['Manchester City', 'Real Madrid']
```

```python
colors = ['#6CABDD', '#FEBE10']

# Goals per 90 by position
for i, team in enumerate(teams):
    team_data = combined_position_stats[combined_position_stats['Team'] == team]
    fig.add_trace(
        go.Bar(x=team_data.index, y=team_data['Goals_per_90'],
               name=f'{team} Goals/90', marker_color=colors[i], opacity=0.8),
        row=1, col=1
    )

# Assists per 90 by position
for i, team in enumerate(teams):
    team_data = combined_position_stats[combined_position_stats['Team'] == team]
    fig.add_trace(
        go.Bar(x=team_data.index, y=team_data['Assists_per_90'],
               name=f'{team} Assists/90', marker_color=colors[i], opacity=0.8),
        row=1, col=2
    )

# Average rating by position
for i, team in enumerate(teams):
    team_data = combined_position_stats[combined_position_stats['Team'] == team]
    fig.add_trace(
        go.Bar(x=team_data.index, y=team_data['Avg_Rating'],
               name=f'{team} Rating', marker_color=colors[i], opacity=0.8),
        row=2, col=1
    )

# Player count by position
for i, team in enumerate(teams):
    team_data = combined_position_stats[combined_position_stats['Team'] == team]
    fig.add_trace(
        go.Bar(x=team_data.index, y=team_data['Player_Count'],
               name=f'{team} Players', marker_color=colors[i], opacity=0.8),
        row=2, col=2
    )

fig.update_layout(height=800, title_text=" Position-Based Performance␣
 ↪Comparison")
fig.show()
```

1.7 Competition Performance Analysis

[11]:
```python
# Competition performance comparison
print("COMPETITION PERFORMANCE COMPARISON")
print("=" * 60)

# Manchester City competition performance
print("\nManchester City Competition Summary:")
display(mc_competition)

# Real Madrid competition performance
print("\nReal Madrid Competition Summary:")
display(rm_competition)

# Calculate competition efficiency metrics
def calculate_competition_efficiency(matches_df: pd.DataFrame) -> pd.DataFrame:
    """Calculate efficiency metrics by competition."""
    comp_stats: pd.DataFrame = matches_df.groupby('competition').agg({
        'result': lambda x: (x == 'Win').sum(),  # Wins
        'manchester_city_score': ['sum', 'mean'],  # Goals for
        'opponent_score': ['sum', 'mean'],  # Goals against
        'possession_percentage': 'mean',
        'shots_total': 'mean',
        'pass_accuracy': 'mean'
    })
    comp_stats = comp_stats.round(2)

    comp_stats.columns = ['Wins', 'Total_Goals', 'Avg_Goals', 'Total_Conceded',
                          'Avg_Conceded', 'Avg_Possession', 'Avg_Shots',␣
 ↪'Avg_Pass_Accuracy']

    # Calculate win rate
    match_counts: pd.Series = matches_df['competition'].value_counts()
    comp_stats['Matches'] = match_counts
    comp_stats['Win_Rate'] = (comp_stats['Wins'] / comp_stats['Matches'] * 100).
 ↪round(1)

    return comp_stats

mc_comp_efficiency = calculate_competition_efficiency(mc_matches)
rm_comp_efficiency = calculate_competition_efficiency(rm_matches)

print("\nManchester City Competition Efficiency:")
display(mc_comp_efficiency)

print("\nReal Madrid Competition Efficiency:")
display(rm_comp_efficiency)
```

COMPETITION PERFORMANCE COMPARISON

==

Manchester City Competition Summary:

	competition	matches_played	wins	draws	losses	goals_for
0	Premier League	38	26	7	5	76
1	Champions League	10	6	2	2	13
2	FA Cup	6	4	0	2	12
3	EFL Cup	3	1	1	1	2

	goals_against	avg_possession	avg_shots	avg_pass_accuracy	highest_score
0	32	66.0	20.3	84.9	5
1	6	64.7	17.6	84.0	3
2	11	63.0	21.5	86.3	4
3	2	67.7	18.0	81.3	2

	lowest_score	avg_goals_scored	avg_goals_conceded	win_percentage
0	0	2.00	0.84	68.4
1	0	1.30	0.60	60.0
2	0	2.00	1.83	66.7
3	0	0.67	0.67	33.3

	goal_difference
0	44
1	7
2	1
3	0

Real Madrid Competition Summary:

	competition	matches_played	wins	draws	losses	goals_for
0	La Liga	38	25	6	7	78
1	Champions League	5	2	2	1	9
2	Copa del Rey	3	1	0	2	3

	goals_against	avg_possession	avg_shots	avg_pass_accuracy	highest_score
0	44	65.0	15.0	87.0	78
1	6	65.0	15.0	87.0	78
2	5	65.0	15.0	87.0	78

	lowest_score	avg_goals_scored	avg_goals_conceded	win_percentage
0	3	2.052632	1.157895	65.8
1	3	1.800000	1.200000	40.0
2	3	1.000000	1.666667	33.3

	goal_difference
0	34

```
1               3
2              -2

Manchester City Competition Efficiency:
                 Wins  Total_Goals  Avg_Goals  Total_Conceded  Avg_Conceded  \
competition
Champions League    6           13       1.30               6          0.60
EFL Cup             1            2       0.67               2          0.67
FA Cup              4           12       2.00              11          1.83
Premier League     26           76       2.00              32          0.84

                 Avg_Possession  Avg_Shots  Avg_Pass_Accuracy  Matches  \
competition
Champions League          64.70      17.60              84.00       10
EFL Cup                   67.67      18.00              81.33        3
FA Cup                    63.00      21.50              86.33        6
Premier League            66.03      20.34              84.89       38

                 Win_Rate
competition
Champions League     60.0
EFL Cup              33.3
FA Cup               66.7
Premier League       68.4

Real Madrid Competition Efficiency:
                 Wins  Total_Goals  Avg_Goals  Total_Conceded  Avg_Conceded  \
competition
Champions League    2            9       1.80               6          1.20
Copa del Rey        1            3       1.00               5          1.67
La Liga            25           78       2.05              44          1.16

                 Avg_Possession  Avg_Shots  Avg_Pass_Accuracy  Matches  \
competition
Champions League          67.40      14.20              85.60        5
Copa del Rey              59.67      12.33              88.67        3
La Liga                   63.58      13.08              86.58       38

                 Win_Rate
competition
Champions League     40.0
Copa del Rey         33.3
La Liga              65.8
```

[12]:
```
# Competition performance visualization
fig = make_subplots(
    rows=2, cols=2,
    subplot_titles=('Win Rate by Competition', 'Goals Scored by Competition',
                   'Possession % by Competition', 'Pass Accuracy by␣
↪Competition')
)

# Get common competitions
common_competitions = set(mc_comp_efficiency.index) & set(rm_comp_efficiency.
↪index)

# Win Rate by Competition
mc_win_rates = [mc_comp_efficiency.loc[comp, 'Win_Rate'] if comp in␣
↪mc_comp_efficiency.index else 0
                for comp in common_competitions]
rm_win_rates = [rm_comp_efficiency.loc[comp, 'Win_Rate'] if comp in␣
↪rm_comp_efficiency.index else 0
                for comp in common_competitions]

fig.add_trace(
    go.Bar(x=list(common_competitions), y=mc_win_rates,
           name='Manchester City', marker_color='#6CABDD'),
    row=1, col=1
)
fig.add_trace(
    go.Bar(x=list(common_competitions), y=rm_win_rates,
           name='Real Madrid', marker_color='#FEBE10'),
    row=1, col=1
)

# Goals by Competition
mc_goals = [mc_comp_efficiency.loc[comp, 'Avg_Goals'] if comp in␣
↪mc_comp_efficiency.index else 0
            for comp in common_competitions]
rm_goals = [rm_comp_efficiency.loc[comp, 'Avg_Goals'] if comp in␣
↪rm_comp_efficiency.index else 0
            for comp in common_competitions]

fig.add_trace(
    go.Bar(x=list(common_competitions), y=mc_goals,
           name='Manchester City', marker_color='#6CABDD', showlegend=False),
    row=1, col=2
)
fig.add_trace(
    go.Bar(x=list(common_competitions), y=rm_goals,
           name='Real Madrid', marker_color='#FEBE10', showlegend=False),
```

```python
    row=1, col=2
)

# Possession by Competition
mc_possession = [mc_comp_efficiency.loc[comp, 'Avg_Possession'] if comp in␣
 ↪mc_comp_efficiency.index else 0
                 for comp in common_competitions]
rm_possession = [rm_comp_efficiency.loc[comp, 'Avg_Possession'] if comp in␣
 ↪rm_comp_efficiency.index else 0
                 for comp in common_competitions]

fig.add_trace(
    go.Bar(x=list(common_competitions), y=mc_possession,
           name='Manchester City', marker_color='#6CABDD', showlegend=False),
    row=2, col=1
)
fig.add_trace(
    go.Bar(x=list(common_competitions), y=rm_possession,
           name='Real Madrid', marker_color='#FEBE10', showlegend=False),
    row=2, col=1
)

# Pass Accuracy by Competition
mc_pass_acc = [mc_comp_efficiency.loc[comp, 'Avg_Pass_Accuracy'] if comp in␣
 ↪mc_comp_efficiency.index else 0
               for comp in common_competitions]
rm_pass_acc = [rm_comp_efficiency.loc[comp, 'Avg_Pass_Accuracy'] if comp in␣
 ↪rm_comp_efficiency.index else 0
               for comp in common_competitions]

fig.add_trace(
    go.Bar(x=list(common_competitions), y=mc_pass_acc,
           name='Manchester City', marker_color='#6CABDD', showlegend=False),
    row=2, col=2
)
fig.add_trace(
    go.Bar(x=list(common_competitions), y=rm_pass_acc,
           name='Real Madrid', marker_color='#FEBE10', showlegend=False),
    row=2, col=2
)

fig.update_layout(height=800, title_text=" Competition Performance Analysis")
fig.show()
```

1.8 Advanced Statistical Analysis

[13]:
```
# Advanced statistical analysis
from scipy import stats
import numpy as np

print(" ADVANCED STATISTICAL ANALYSIS")
print("=" * 60)

# Statistical tests comparing teams
def perform_statistical_tests(mc_data, rm_data, metric_name):
    """Perform statistical tests between teams."""
    # Remove any NaN values
    mc_clean = mc_data.dropna()
    rm_clean = rm_data.dropna()

    if len(mc_clean) == 0 or len(rm_clean) == 0:
        return None

    # T-test
    t_stat, t_p_value = stats.ttest_ind(mc_clean, rm_clean)

    # Mann-Whitney U test (non-parametric)
    u_stat, u_p_value = stats.mannwhitneyu(mc_clean, rm_clean,␣
↪alternative='two-sided')

    return {
        'metric': metric_name,
        'mc_mean': mc_clean.mean(),
        'rm_mean': rm_clean.mean(),
        'mc_std': mc_clean.std(),
        'rm_std': rm_clean.std(),
        't_statistic': t_stat,
        't_p_value': t_p_value,
        'u_statistic': u_stat,
        'u_p_value': u_p_value,
        'significant': t_p_value < 0.05
    }

# Perform tests on key metrics
metrics_to_test = [
    ('goals_per_90', 'Goals per 90'),
    ('assists_per_90', 'Assists per 90'),
    ('avg_rating', 'Average Rating'),
    ('shots_per_90', 'Shots per 90'),
    ('passes_per_90', 'Passes per 90')
]
```

```
statistical_results = []
for metric_col, metric_name in metrics_to_test:
    if metric_col in mc_season_stats.columns and metric_col in rm_season_stats.
 ↪columns:
        result = perform_statistical_tests(
            mc_season_stats[metric_col],
            rm_season_stats[metric_col],
            metric_name
        )
        if result:
            statistical_results.append(result)

# Display results
stats_df = pd.DataFrame(statistical_results)
if not stats_df.empty:
    stats_df = stats_df.round(4)
    print("\n Statistical Test Results:")
    display(stats_df[['metric', 'mc_mean', 'rm_mean', 't_p_value',␣
 ↪'significant']])
else:
    print("\n No statistical tests could be performed due to data limitations.")
```

```
 ADVANCED STATISTICAL ANALYSIS
============================================================

 Statistical Test Results:
```

	metric	mc_mean	rm_mean	t_p_value	significant
0	Goals per 90	0.0906	0.1338	0.1998	False
1	Assists per 90	0.1097	0.1483	0.1369	False
2	Average Rating	7.5742	7.3750	0.0229	True
3	Shots per 90	2.0603	3.1075	0.0050	True
4	Passes per 90	54.9613	74.8375	0.0000	True

1.9 Correlation Analysis

[14]:
```
# Correlation analysis
print(" CORRELATION ANALYSIS")
print("=" * 50)

# Select numeric columns for correlation
numeric_cols = ['goals', 'assists', 'avg_rating', 'goals_per_90',␣
 ↪'assists_per_90',
                'shots_per_90', 'passes_per_90', 'tackles_per_90',␣
 ↪'total_minutes']

# Filter columns that exist in both datasets
```

```
available_cols = [col for col in numeric_cols if col in mc_season_stats.columns and col in rm_season_stats.columns]

if available_cols:
    # Calculate correlations for both teams
    mc_corr = mc_season_stats[available_cols].corr()
    rm_corr = rm_season_stats[available_cols].corr()

    # Create correlation heatmaps
    fig = make_subplots(
        rows=1, cols=2,
        subplot_titles=('Manchester City Correlations', 'Real Madrid Correlations'),
        specs=[[{"type": "heatmap"}, {"type": "heatmap"}]]
    )

    # Manchester City correlation heatmap
    fig.add_trace(
        go.Heatmap(z=mc_corr.values, x=mc_corr.columns, y=mc_corr.columns,
                   colorscale='RdBu', zmid=0, showscale=True),
        row=1, col=1
    )

    # Real Madrid correlation heatmap
    fig.add_trace(
        go.Heatmap(z=rm_corr.values, x=rm_corr.columns, y=rm_corr.columns,
                   colorscale='RdBu', zmid=0, showscale=False),
        row=1, col=2
    )

    fig.update_layout(height=600, title_text="  Performance Metrics Correlation Analysis")
    fig.show()

    # Display strongest correlations
    print("\n Strongest Correlations (Manchester City):")
    mc_corr_flat = mc_corr.unstack().sort_values(ascending=False)
    mc_strong_corr = mc_corr_flat[(mc_corr_flat < 0.99) & (mc_corr_flat > 0.7)].head(5)
    for idx, corr in mc_strong_corr.items():
        print(f"   {idx[0]}   {idx[1]}: {corr:.3f}")

    print("\n Strongest Correlations (Real Madrid):")
    rm_corr_flat = rm_corr.unstack().sort_values(ascending=False)
    rm_strong_corr = rm_corr_flat[(rm_corr_flat < 0.99) & (rm_corr_flat > 0.7)].head(5)
    for idx, corr in rm_strong_corr.items():
```

```
        print(f"   {idx[0]}   {idx[1]}: {corr:.3f}")
else:
    print("\n Insufficient numeric columns for correlation analysis.")
```

```
 CORRELATION ANALYSIS
==================================================


 Strongest Correlations (Manchester City):
   goals_per_90   goals: 0.974
   goals   goals_per_90: 0.974
   assists   assists_per_90: 0.871
   assists_per_90   assists: 0.871
   assists   goals: 0.792

 Strongest Correlations (Real Madrid):
   goals   goals_per_90: 0.985
   goals_per_90   goals: 0.985
   assists   assists_per_90: 0.960
   assists_per_90   assists: 0.960
   goals   shots_per_90: 0.707
```

1.10 Player Efficiency Metrics

[15]:

```
# Calculate advanced efficiency metrics
def calculate_efficiency_metrics(data):
    """Calculate advanced player efficiency metrics."""
    data = data.copy()

    # Goal contribution efficiency
    data['goal_contribution'] = data['goals'] + data['assists']
    data['goal_contribution_per_90'] = data['goals_per_90'] +␣
 ↪data['assists_per_90']

    # Minutes per goal contribution
    data['minutes_per_goal_contribution'] = np.where(
        data['goal_contribution'] > 0,
        data['total_minutes'] / data['goal_contribution'],
        np.inf
    )

    # Shot efficiency (goals per shot)
    if 'shots' in data.columns:
        data['shot_efficiency'] = np.where(
            data['shots'] > 0,
            data['goals'] / data['shots'],
            0
        )
```

```python
    # Performance consistency (based on rating)
    data['performance_tier'] = pd.cut(
        data['avg_rating'],
        bins=[0, 6.5, 7.5, 8.5, 10],
        labels=['Below Average', 'Good', 'Very Good', 'Excellent']
    )

    return data

# Calculate efficiency metrics for both teams
mc_efficiency = calculate_efficiency_metrics(mc_season_stats)
rm_efficiency = calculate_efficiency_metrics(rm_season_stats)

print("  PLAYER EFFICIENCY ANALYSIS")
print("=" * 50)

# Top efficient players (goal contribution per 90)
combined_efficiency = pd.concat([mc_efficiency, rm_efficiency],␣
 ↪ignore_index=True)
top_efficient = combined_efficiency.nlargest(10, 'goal_contribution_per_90')

print("\n Most Efficient Players (Goal Contribution per 90):")
display(top_efficient[['player_name', 'team', 'position',␣
 ↪'goal_contribution_per_90',
                      'goals_per_90', 'assists_per_90', 'avg_rating']].round(3))

# Performance tier distribution
print("\n Performance Tier Distribution:")
tier_comparison = pd.crosstab(combined_efficiency['team'],␣
 ↪combined_efficiency['performance_tier'])
display(tier_comparison)
```

```
 PLAYER EFFICIENCY ANALYSIS
==================================================

 Most Efficient Players (Goal Contribution per 90):

          player_name             team position  goal_contribution_per_90  \
54       Mariano Díaz      Real Madrid       FW                      0.84
42       Vinícius Jr.      Real Madrid       FW                      0.63
3          Phil Foden  Manchester City    MF,FW                      0.60
1               Rodri  Manchester City       MF                      0.56
45  Federico Valverde      Real Madrid       MF                      0.55
2      Bernardo Silva  Manchester City    MF,FW                      0.53
0      Erling Haaland  Manchester City       FW                      0.51
8     Kevin De Bruyne  Manchester City    MF,FW                      0.51
44            Rodrygo      Real Madrid       FW                      0.50
```

```
41      Marco Asensio      Real Madrid       FW                         0.39

    goals_per_90  assists_per_90  avg_rating
54          0.52            0.32         7.7
42          0.39            0.24         8.0
3           0.28            0.32         7.5
1           0.25            0.31         7.6
45          0.30            0.25         7.5
2           0.25            0.28         7.5
0           0.36            0.15         7.6
8           0.20            0.31         7.6
44          0.40            0.10         7.9
41          0.24            0.15         7.6
```

```
 Performance Tier Distribution:
performance_tier  Good  Very Good
team
Manchester City     15         16
Real Madrid         18          6
```

[16]:

```
# Efficiency visualization
fig = make_subplots(
    rows=2, cols=2,
    subplot_titles=('Goal Contribution per 90', 'Performance Tier Distribution',
                   'Minutes per Goal Contribution', 'Shot Efficiency (if␣
 ↪available)')
)

# Goal contribution per 90 comparison
fig.add_trace(
    go.Box(y=mc_efficiency['goal_contribution_per_90'], name='Manchester City',
           marker_color='#6CABDD'),
    row=1, col=1
)
fig.add_trace(
    go.Box(y=rm_efficiency['goal_contribution_per_90'], name='Real Madrid',
           marker_color='#FEBE10'),
    row=1, col=1
)

# Performance tier distribution
tier_counts_mc = mc_efficiency['performance_tier'].value_counts()
tier_counts_rm = rm_efficiency['performance_tier'].value_counts()

fig.add_trace(
    go.Bar(x=tier_counts_mc.index, y=tier_counts_mc.values,
```

```
            name='Manchester City', marker_color='#6CABDD'),
    row=1, col=2
)
fig.add_trace(
    go.Bar(x=tier_counts_rm.index, y=tier_counts_rm.values,
           name='Real Madrid', marker_color='#FEBE10'),
    row=1, col=2
)

# Minutes per goal contribution (filter out infinite values)
mc_minutes_filtered =␣
 ↪mc_efficiency[mc_efficiency['minutes_per_goal_contribution'] != np.
 ↪inf]['minutes_per_goal_contribution']
rm_minutes_filtered =␣
 ↪rm_efficiency[rm_efficiency['minutes_per_goal_contribution'] != np.
 ↪inf]['minutes_per_goal_contribution']

fig.add_trace(
    go.Box(y=mc_minutes_filtered, name='Manchester City',
           marker_color='#6CABDD', showlegend=False),
    row=2, col=1
)
fig.add_trace(
    go.Box(y=rm_minutes_filtered, name='Real Madrid',
           marker_color='#FEBE10', showlegend=False),
    row=2, col=1
)

# Shot efficiency (if available)
if 'shot_efficiency' in mc_efficiency.columns and 'shot_efficiency' in␣
 ↪rm_efficiency.columns:
    fig.add_trace(
        go.Box(y=mc_efficiency['shot_efficiency'], name='Manchester City',
               marker_color='#6CABDD', showlegend=False),
        row=2, col=2
    )
    fig.add_trace(
        go.Box(y=rm_efficiency['shot_efficiency'], name='Real Madrid',
               marker_color='#FEBE10', showlegend=False),
        row=2, col=2
    )

fig.update_layout(height=800, title_text=" Player Efficiency Analysis")
fig.show()
```

1.11 Key Insights and Conclusions

[17]:
```
# Generate key insights
print(" KEY INSIGHTS AND CONCLUSIONS")
print("=" * 60)

# Team comparison insights
print("\n TEAM PERFORMANCE INSIGHTS:")
print("-" * 40)

# Win rate comparison
mc_win_rate = team_comparison.loc['Manchester City', 'win_rate']
rm_win_rate = team_comparison.loc['Real Madrid', 'win_rate']
print(f"• Manchester City win rate: {mc_win_rate}%")
print(f"• Real Madrid win rate: {rm_win_rate}%")
if mc_win_rate > rm_win_rate:
    print(f"  → Manchester City has a {mc_win_rate - rm_win_rate:.1f}% higher␣
↪win rate")
else:
    print(f"  → Real Madrid has a {rm_win_rate - mc_win_rate:.1f}% higher win␣
↪rate")

# Goal scoring comparison
mc_goals_per_match = team_comparison.loc['Manchester City',␣
↪'avg_goals_per_match']
rm_goals_per_match = team_comparison.loc['Real Madrid', 'avg_goals_per_match']
print(f"\n• Manchester City goals per match: {mc_goals_per_match}")
print(f"• Real Madrid goals per match: {rm_goals_per_match}")
if mc_goals_per_match > rm_goals_per_match:
    print(f"  → Manchester City scores {mc_goals_per_match - rm_goals_per_match:
↪.2f} more goals per match")
else:
    print(f"  → Real Madrid scores {rm_goals_per_match - mc_goals_per_match:.
↪2f} more goals per match")

# Player insights
print("\n PLAYER PERFORMANCE INSIGHTS:")
print("-" * 40)

# Top scorers
mc_top_scorer = mc_season_stats.loc[mc_season_stats['goals'].idxmax()]
rm_top_scorer = rm_season_stats.loc[rm_season_stats['goals'].idxmax()]
print(f"• Manchester City top scorer: {mc_top_scorer['player_name']}␣
↪({mc_top_scorer['goals']} goals)")
print(f"• Real Madrid top scorer: {rm_top_scorer['player_name']}␣
↪({rm_top_scorer['goals']} goals)")
```

```python
# Squad depth
mc_squad_size = len(mc_season_stats)
rm_squad_size = len(rm_season_stats)
print(f"\n• Manchester City squad size: {mc_squad_size} players")
print(f"• Real Madrid squad size: {rm_squad_size} players")

# Average team rating
mc_avg_rating = mc_season_stats['avg_rating'].mean()
rm_avg_rating = rm_season_stats['avg_rating'].mean()
print(f"\n• Manchester City average team rating: {mc_avg_rating:.2f}")
print(f"• Real Madrid average team rating: {rm_avg_rating:.2f}")

print("\n TACTICAL INSIGHTS:")
print("-" * 40)
print("• Both teams show strong attacking capabilities")
print("• Squad rotation strategies differ between competitions")
print("• Performance consistency varies by position and player")

print("\n RECOMMENDATIONS FOR FURTHER ANALYSIS:")
print("-" * 40)
print("• Analyze match-by-match performance trends")
print("• Investigate home vs away performance differences")
print("• Study player performance in high-pressure matches")
print("• Examine tactical formations and their effectiveness")
print("• Apply PVOI framework for advanced player valuation")

print("\n" + "=" * 60)
print(" ANALYSIS COMPLETE - Ready for dashboard integration!")
print("=" * 60)
```

```
 KEY INSIGHTS AND CONCLUSIONS
============================================================

 TEAM PERFORMANCE INSIGHTS:
----------------------------------------
• Manchester City win rate: 64.9%
• Real Madrid win rate: 60.9%
  → Manchester City has a 4.0% higher win rate

• Manchester City goals per match: 1.81
• Real Madrid goals per match: 1.96
  → Real Madrid scores 0.15 more goals per match

 PLAYER PERFORMANCE INSIGHTS:
----------------------------------------
• Manchester City top scorer: Erling Haaland (12 goals)
• Real Madrid top scorer: Vinícius Jr. (8 goals)
```

```
• Manchester City squad size: 31 players
• Real Madrid squad size: 24 players

• Manchester City average team rating: 7.57
• Real Madrid average team rating: 7.38

 TACTICAL INSIGHTS:
----------------------------------------
• Both teams show strong attacking capabilities
• Squad rotation strategies differ between competitions
• Performance consistency varies by position and player

 RECOMMENDATIONS FOR FURTHER ANALYSIS:
----------------------------------------
• Analyze match-by-match performance trends
• Investigate home vs away performance differences
• Study player performance in high-pressure matches
• Examine tactical formations and their effectiveness
• Apply PVOI framework for advanced player valuation

============================================================
 ANALYSIS COMPLETE - Ready for dashboard integration!
============================================================
```

[ ]: