# PLAYER PERFORMANCE ANALYSIS FOR CHARLTON ATHLETIC

- ARJUN SRIDHAR

SEPTEMBER 12, 2024

# PROJECT OVERVIEW & OBJECTIVES

**Primary Objective** 🎯

- Recommend 3 standout players for Charlton Athletic's first-team squad based on in-depth performance analysis.

**Sub-objectives**

- **Data Cleaning & Preprocessing**

  Handle missing values, standardize metrics, and normalize data.

- **Performance Analysis** 📊

  Assess players' attacking, defensive, and transitional abilities.

- **Player Comparison** 🔍

  Identify top performers for each position based on impact per 90 minutes.

- **Player Recommendation** 🏆

  Deliver data-backed recommendations for the top 3 players.

**Key Metrics Defined** 📍

- **Play Duration**

  Total seconds a player has spent on the pitch, reflecting their experience and contribution.

- **Match Share** 📈

  Proportion of match time participated in, indicating consistency and reliability.

**Deliverables** 📋

- **Comprehensive Data Analysis**

  Detailed review of player performance across positions and leagues.

- **Data-Driven Visualizations**

  Insights into playtime, ultimate scores, and top players.

- **Recommendations**

  Selection of 3 key players with detailed justification and supporting visuals.

## DATASET BREAKDOWN

**Key Features of the Dataset 📊**

➢ **Play Duration**

Total time a player has spent on the pitch.

**Importance:** Reflects experience and contribution during matches.

➢ **Match Share 📈**

Percentage of total available match time played by the player.

**Importance:** Measures reliability and consistency in team selection.

**Additional Features 🔍**

➢ **Position Categories ⚽**

Various roles (e.g., Central Midfield, Goalkeeper, Winger) for role-specific comparison.

➢ **Performance Scores 🏅**

Metrics like AI Score, Weighted Score, Z-Score to objectively rank players.

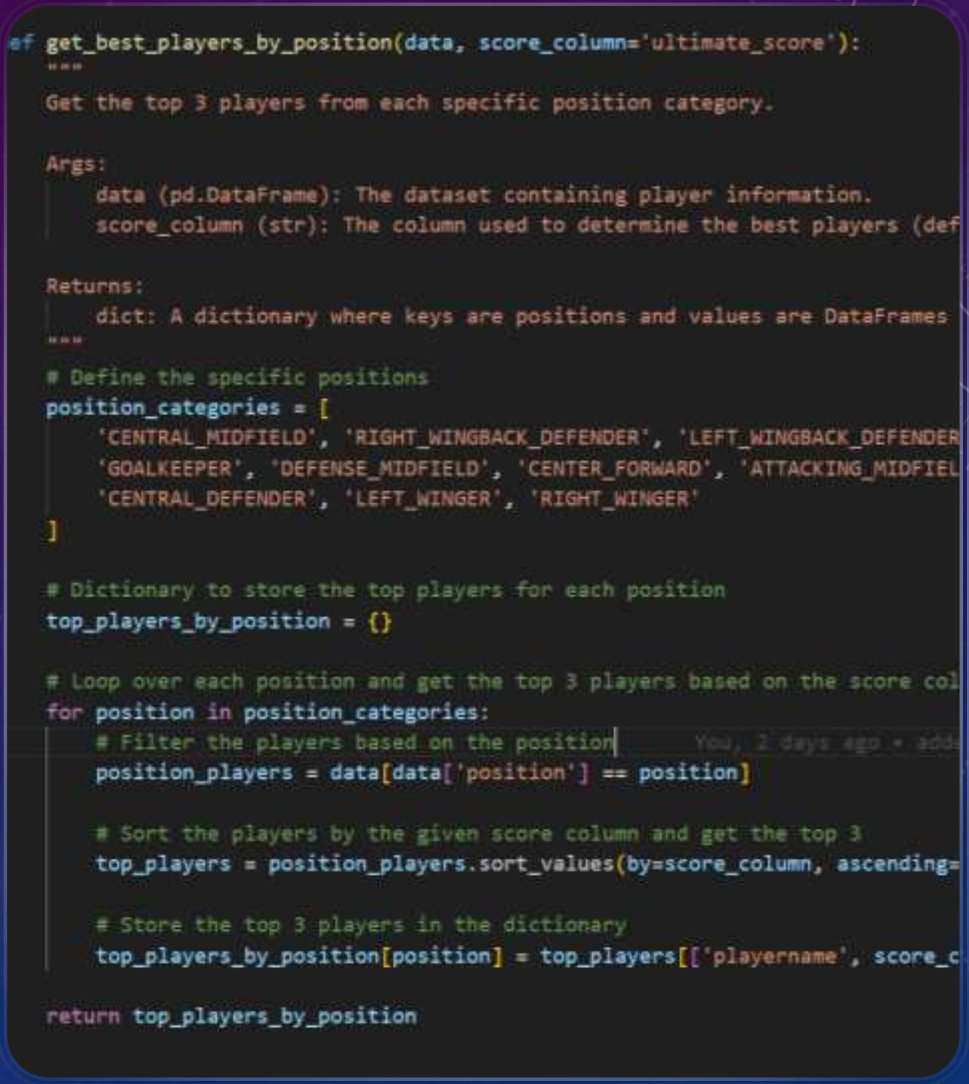
**Challenges 🚧**

➢ **Missing Data ❓**

Incomplete records addressed via imputation or removal.

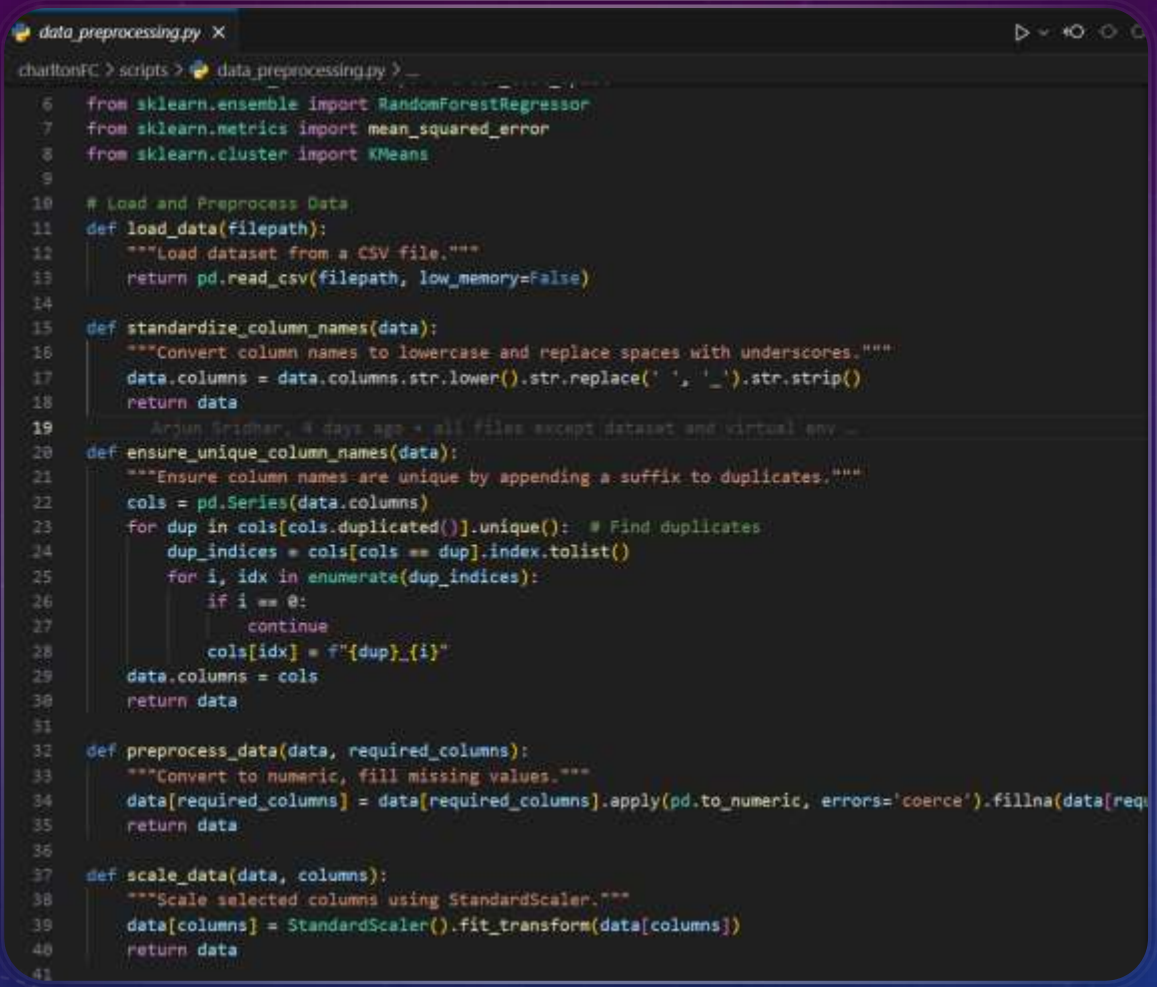
➢ **Data Normalization 📏**

Applied scaling across leagues for fair player comparison.



```
ef get_best_players_by_position(data, score_column='ultimate_score'):
    """
    Get the top 3 players from each specific position category.

    Args:
        data (pd.DataFrame): The dataset containing player information.
        score_column (str): The column used to determine the best players (def

    Returns:
        dict: A dictionary where keys are positions and values are DataFrames
    """
    # Define the specific positions
    position_categories = [
        'CENTRAL_MIDFIELD', 'RIGHT_WINGBACK_DEFENDER', 'LEFT_WINGBACK_DEFENDER
        'GOALKEEPER', 'DEFENSE_MIDFIELD', 'CENTER_FORWARD', 'ATTACKING_MIDFIEL
        'CENTRAL_DEFENDER', 'LEFT_WINGER', 'RIGHT_WINGER'
    ]

    # Dictionary to store the top players for each position
    top_players_by_position = {}

    # Loop over each position and get the top 3 players based on the score col
    for position in position_categories:
        # Filter the players based on the position      You, 2 days ago • adde
        position_players = data[data['position'] == position]

        # Sort the players by the given score column and get the top 3
        top_players = position_players.sort_values(by=score_column, ascending=

        # Store the top 3 players in the dictionary
        top_players_by_position[position] = top_players[['playername', score_c

    return top_players_by_position
```

# DATA CLEANING & PREPROCESSING


data_preprocessing.py

charltonFC > scripts > data_preprocessing.py > ...

```python
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.cluster import KMeans

# Load and Preprocess Data
def load_data(filepath):
    """Load dataset from a CSV file."""
    return pd.read_csv(filepath, low_memory=False)

def standardize_column_names(data):
    """Convert column names to lowercase and replace spaces with underscores."""
    data.columns = data.columns.str.lower().str.replace(' ', '_').str.strip()
    return data
        Arjun Sridhar, 4 days ago • all files except dataset and virtual env ...
def ensure_unique_column_names(data):
    """Ensure column names are unique by appending a suffix to duplicates."""
    cols = pd.Series(data.columns)
    for dup in cols[cols.duplicated()].unique():  # Find duplicates
        dup_indices = cols[cols == dup].index.tolist()
        for i, idx in enumerate(dup_indices):
            if i == 0:
                continue
            cols[idx] = f"{dup}_{i}"
    data.columns = cols
    return data

def preprocess_data(data, required_columns):
    """Convert to numeric, fill missing values."""
    data[required_columns] = data[required_columns].apply(pd.to_numeric, errors='coerce').fillna(data[requ
    return data

def scale_data(data, columns):
    """Scale selected columns using StandardScaler."""
    data[columns] = StandardScaler().fit_transform(data[columns])
    return data
```

**Steps Taken**

- **Standardized Column Names**

  Ensured uniformity and clarity in data labels.

- **Handled Missing Values** ✓

  Addressed incomplete data through imputation or removal.

- **Scaled Important Features**

  Normalized Play Duration and Match Share for consistent comparison.

**Outcome**

- **Data Consistency Achieved**

  Dataset is now uniform and ready for insightful analysis.

# SCORING AND RANKING PLAYERS

**Scores Applied** 📊

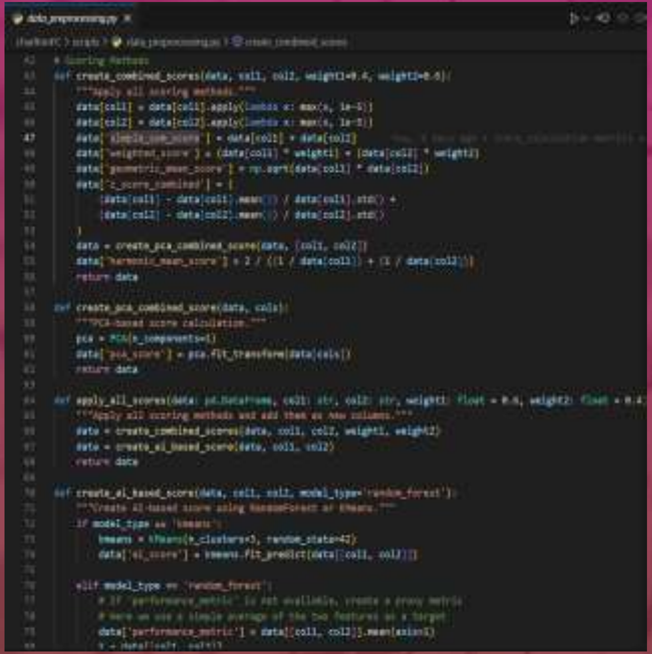
- ➢ **AI Score** : Generated using the rainforest model for unlabeled data.
- ➢ **Weighted Score** : Balanced based on key metrics.
- ➢ **Z-Score** 📈: Standardized performance measure.
- ➢ **PCA Score** 🔍: Principal Component Analysis for dimensionality reduction.
- ➢ **Geometric Mean Score** 📐: Average score using multiplicative factors.
- ➢ **Harmonic Mean Score** ⚖: Average emphasizing lower values.
- ➢ **Simple Sum Score** ➕: Aggregated sum of key metrics.

**Final Score** 🏆**:**

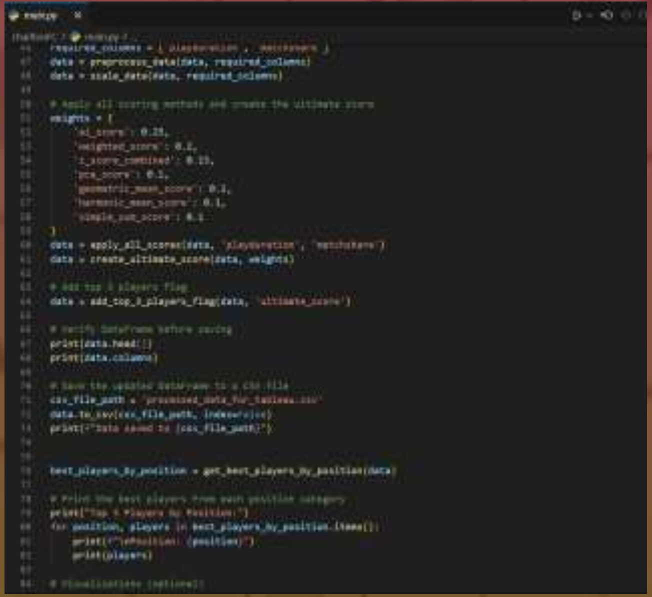
- ➢ **The Ultimate Score** ★: A weighted combination of all scores for effective player ranking.

**Weights** ⚖**:**

- ➢ **AI Score:** 25%
- ➢ **Weighted Score:** 20%
- ➢ **Z-Score:** 15%
- ➢ **PCA Score**: 10%
- ➢ **Geometric Mean Score**: 10%
- ➢ **Harmonic Mean Score**: 10%
- ➢ **Simple Sum Score**: 10%


data_preprocessing.py
theRookIC > scripts > data_preprocessing.py > create_combined_scores

```python
# Scoring Methods
def create_combined_scores(data, col1, col2, weight1=0.4, weight2=0.6):
    """Apply all scoring methods."""
    data[col1] = data[col1].apply(lambda x: max(x, 1e-5))
    data[col2] = data[col2].apply(lambda x: max(x, 1e-5))
    data['simple_sum_score'] = data[col1] + data[col2]        [illegible]
    data['weighted_score'] = (data[col1] * weight1) + (data[col2] * weight2)
    data['geometric_mean_score'] = np.sqrt(data[col1] * data[col2])
    data['z_score_combined'] = (
        (data[col1] - data[col1].mean()) / data[col1].std() +
        (data[col2] - data[col2].mean()) / data[col2].std()
    )
    data = create_pca_combined_score(data, [col1, col2])
    data['harmonic_mean_score'] = 2 / ((1 / data[col1]) + (1 / data[col2]))
    return data

def create_pca_combined_score(data, cols):
    """PCA-based score calculation."""
    pca = PCA(n_components=1)
    data['pca_score'] = pca.fit_transform(data[cols])
    return data

def apply_all_scores(data: pd.DataFrame, col1: str, col2: str, weight1: float = 0.6, weight2: float = 0.4
    """Apply all scoring methods and add them as new columns."""
    data = create_combined_scores(data, col1, col2, weight1, weight2)
    data = create_ai_based_score(data, col1, col2)
    return data

def create_ai_based_score(data, col1, col2, model_type='random_forest'):
    """Create AI-based score using RandomForest or KMeans."""
    if model_type == 'kmeans':
        kmeans = KMeans(n_clusters=3, random_state=42)
        data['ai_score'] = kmeans.fit_predict(data[[col1, col2]])

    elif model_type == 'random_forest':
        # If 'performance_metric' is not available, create a proxy metric
        # Here we use a simple average of the two features as a target
        data['performance_metric'] = data[[col1, col2]].mean(axis=1)
        X = data[[col1, col2]]
```



```python
required_columns = ['playduration', 'matchstars']
data = preprocess_data(data, required_columns)
data = scale_data(data, required_columns)

# Apply all scoring methods and create the ultimate score
weights = {
    'ai_score': 0.25,
    'weighted_score': 0.2,
    'z_score_combined': 0.15,
    'pca_score': 0.1,
    'geometric_mean_score': 0.1,
    'harmonic_mean_score': 0.1,
    'simple_sum_score': 0.1
}
data = apply_all_scores(data, 'playduration', 'matchstars')
data = create_ultimate_score(data, weights)

# Add top 3 players flag
data = add_top_3_players_flag(data, 'ultimate_score')

# Verify DataFrame before saving
print(data.head())
print(data.columns)

# Save the updated DataFrame to a CSV file
csv_file_path = 'processed_data_for_tableau.csv'
data.to_csv(csv_file_path, index=False)
print(f"Data saved to {csv_file_path}")


best_players_by_position = get_best_players_by_position(data)

# Print the best players from each position category
print("Top 5 Players by Position:")
for position, players in best_players_by_position.items():
    print(f"\nPosition: {position}")
    print(players)

# Visualizations (optional)
```

**Method** ⍰

- **Ranking Players** 📊

  Used **Ultimate Score** to rank players within each position.

- **Top 3 Flag** 🎯

  Created a flag for the top 3 players in each position category.

**Positions Covered** 📍

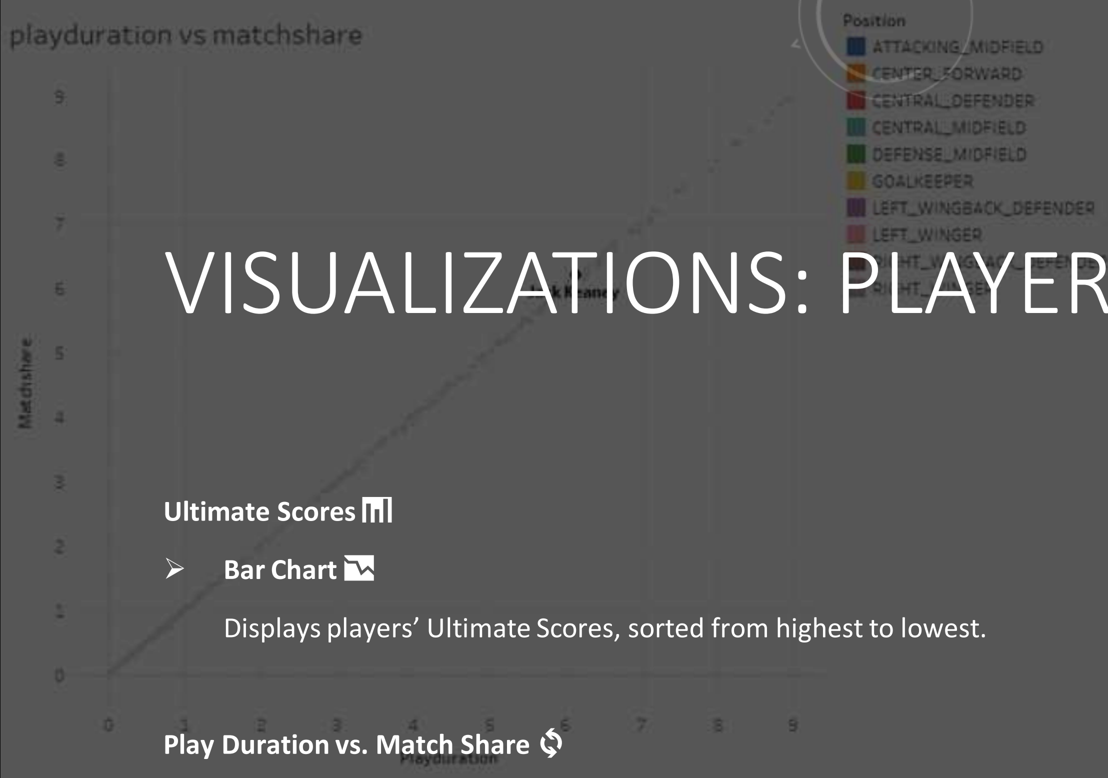
- **Central Midfield**
- **Goalkeeper**
- **Center Forward**
- **(and others)**

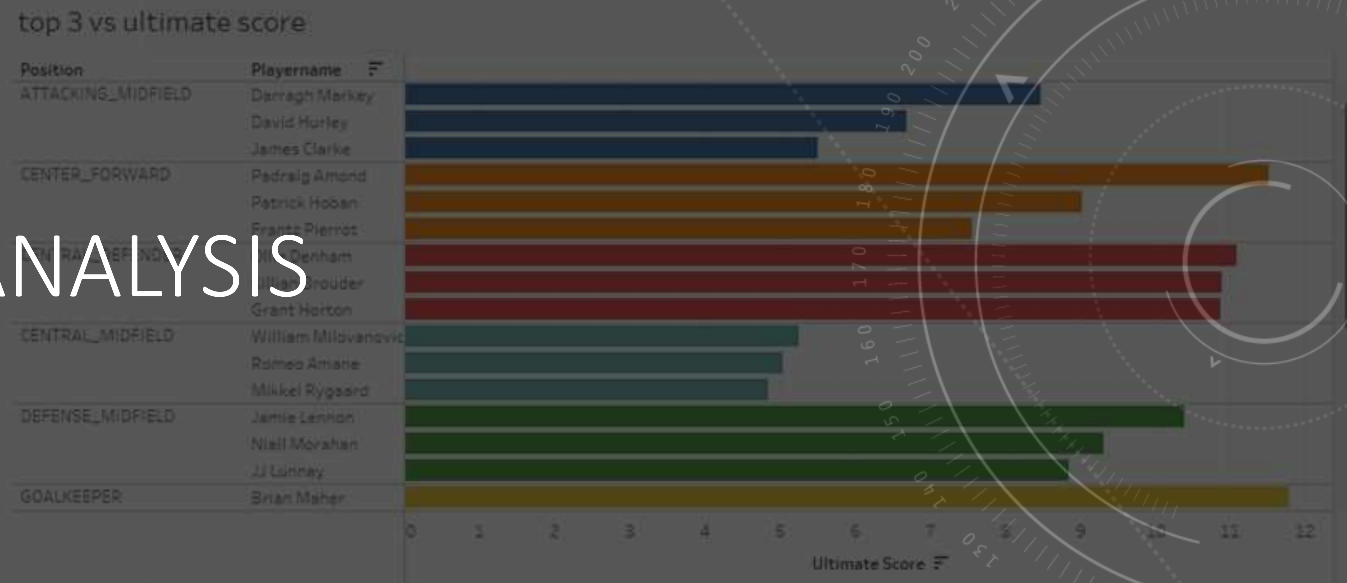
**New Column** 🆕

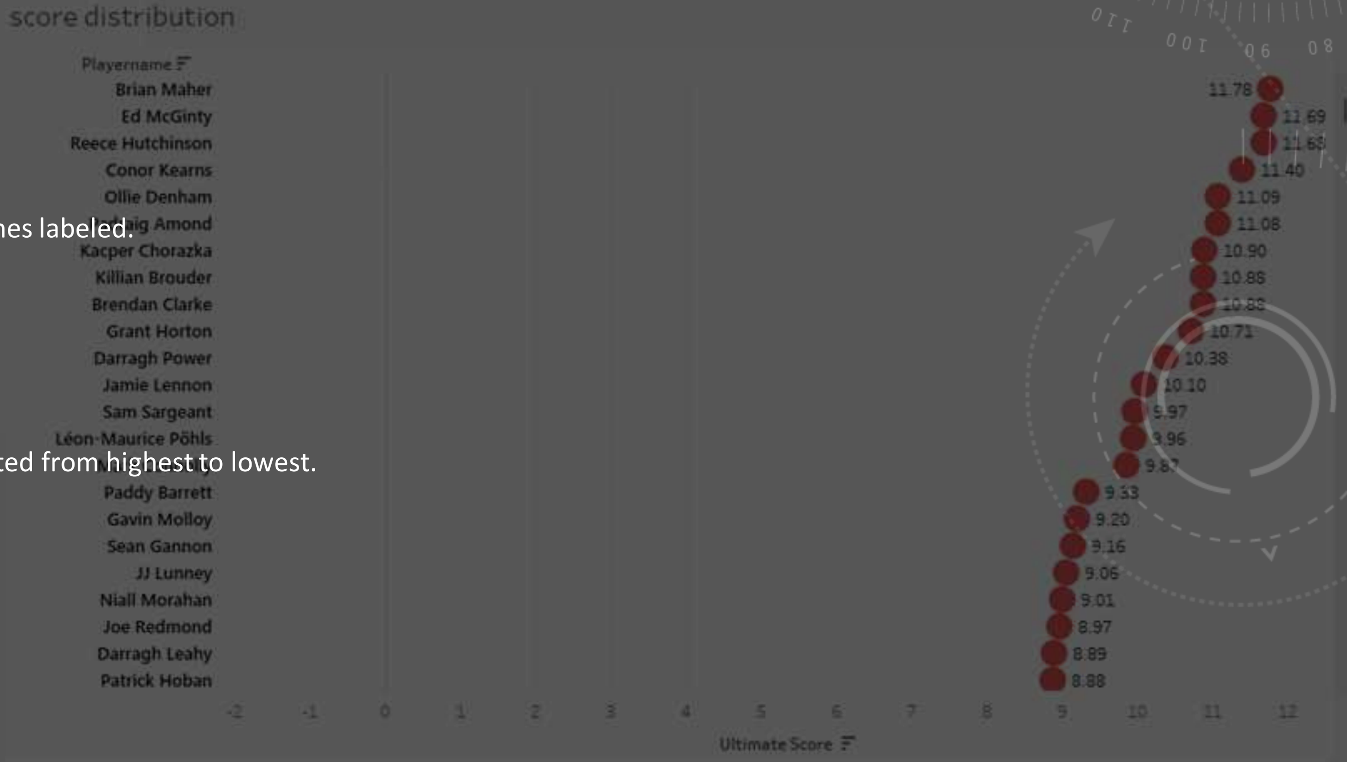
- `is_top_3_in_position` ✅

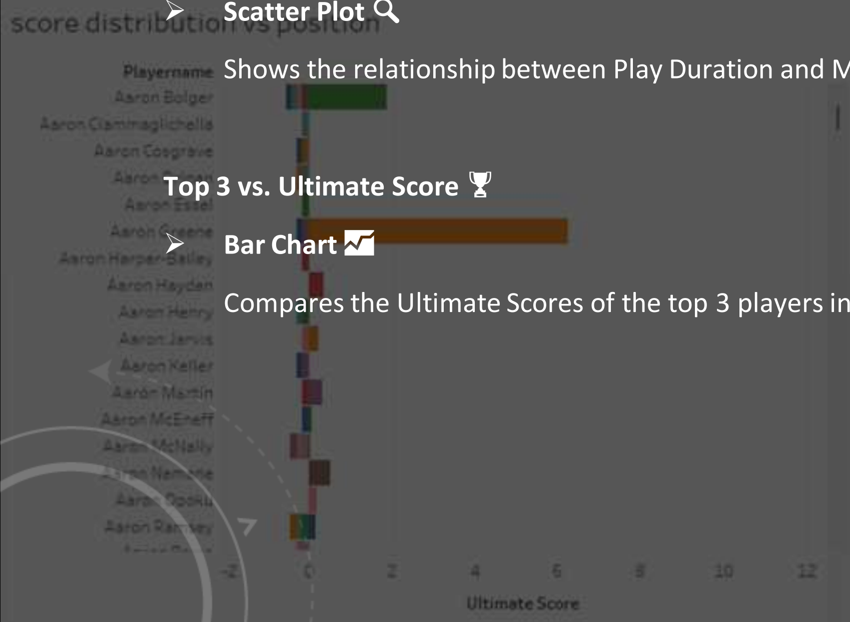
  Indicates whether a player is among the top 3 in their position.

# IDENTIFYING TOP PLAYERS BY POSITION









# VISUALIZATIONS: PLAYER ANALYSIS

**Ultimate Scores**

- **Bar Chart**

  Displays players' Ultimate Scores, sorted from highest to lowest.

**Play Duration vs. Match Share**

- **Scatter Plot**

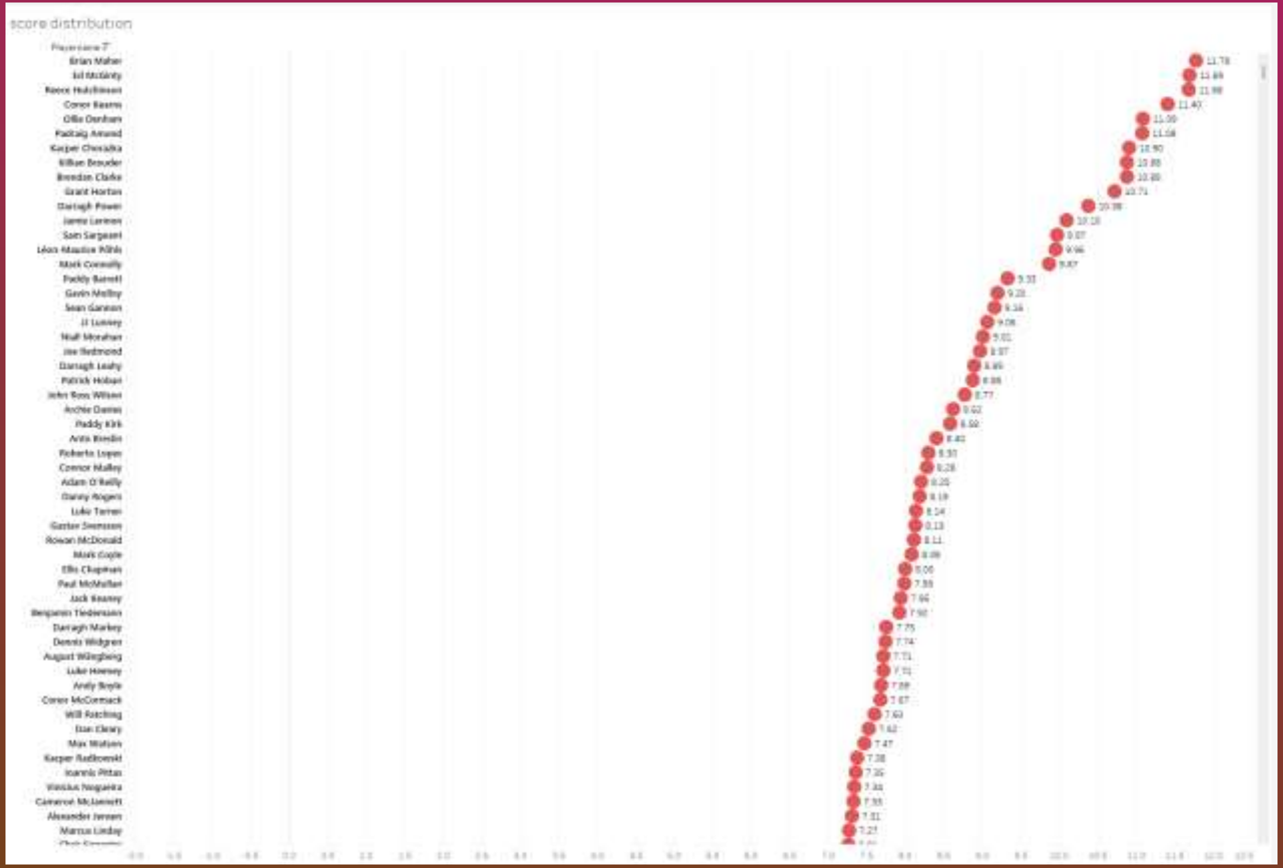
  Shows the relationship between Play Duration and Match Share, with player names labeled.

**Top 3 vs. Ultimate Score**

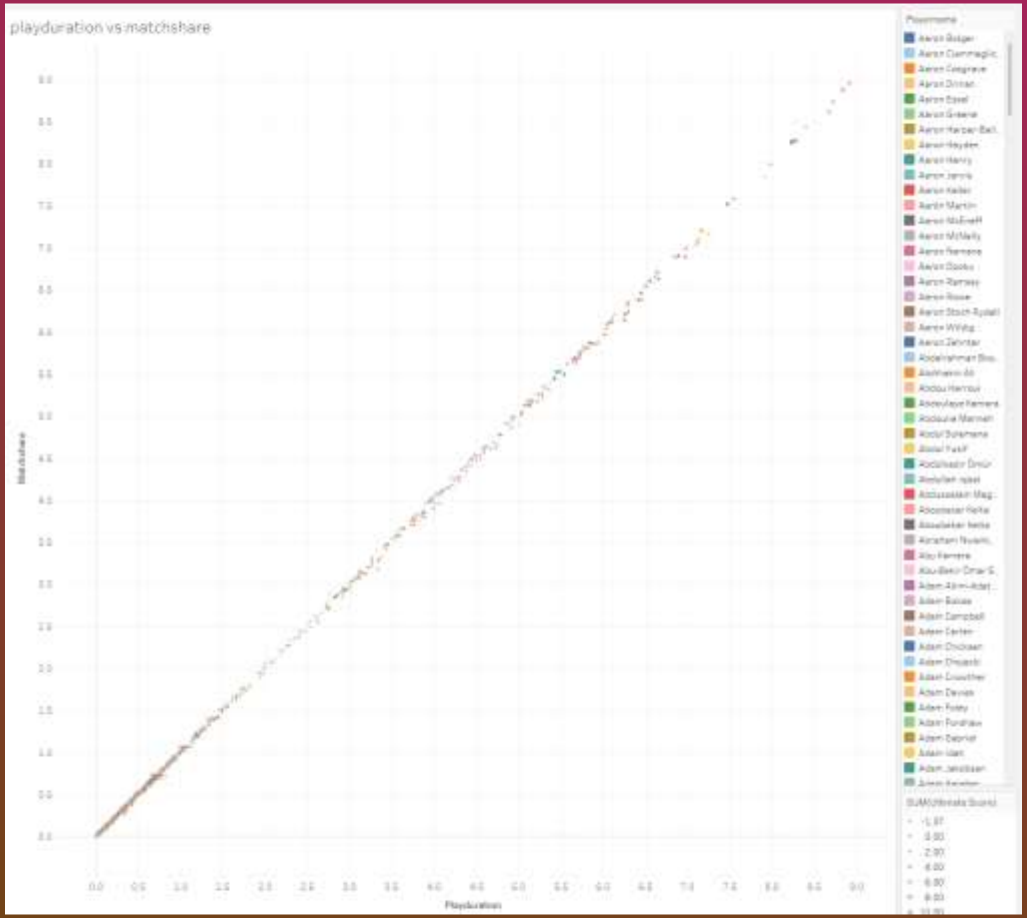
- **Bar Chart**

  Compares the Ultimate Scores of the top 3 players in each position category, sorted from highest to lowest.

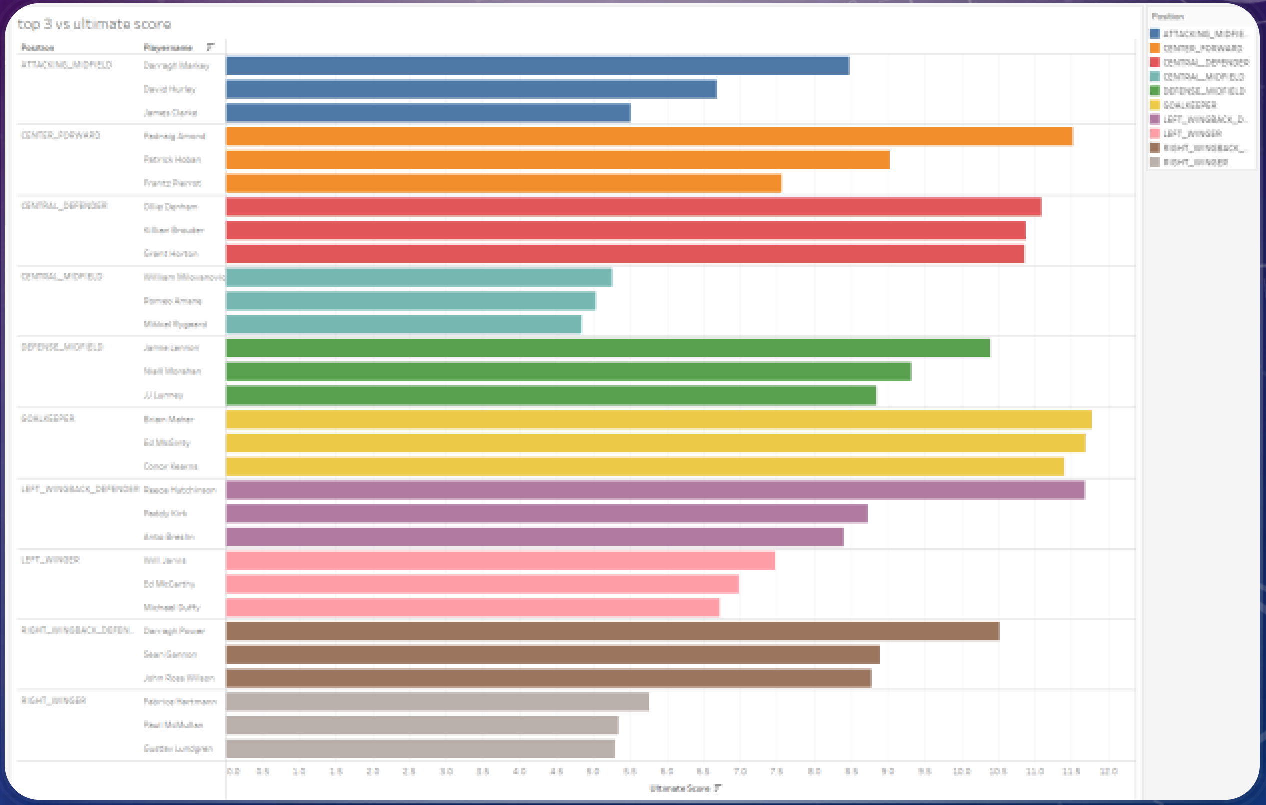
# ULTIMATE SCORES

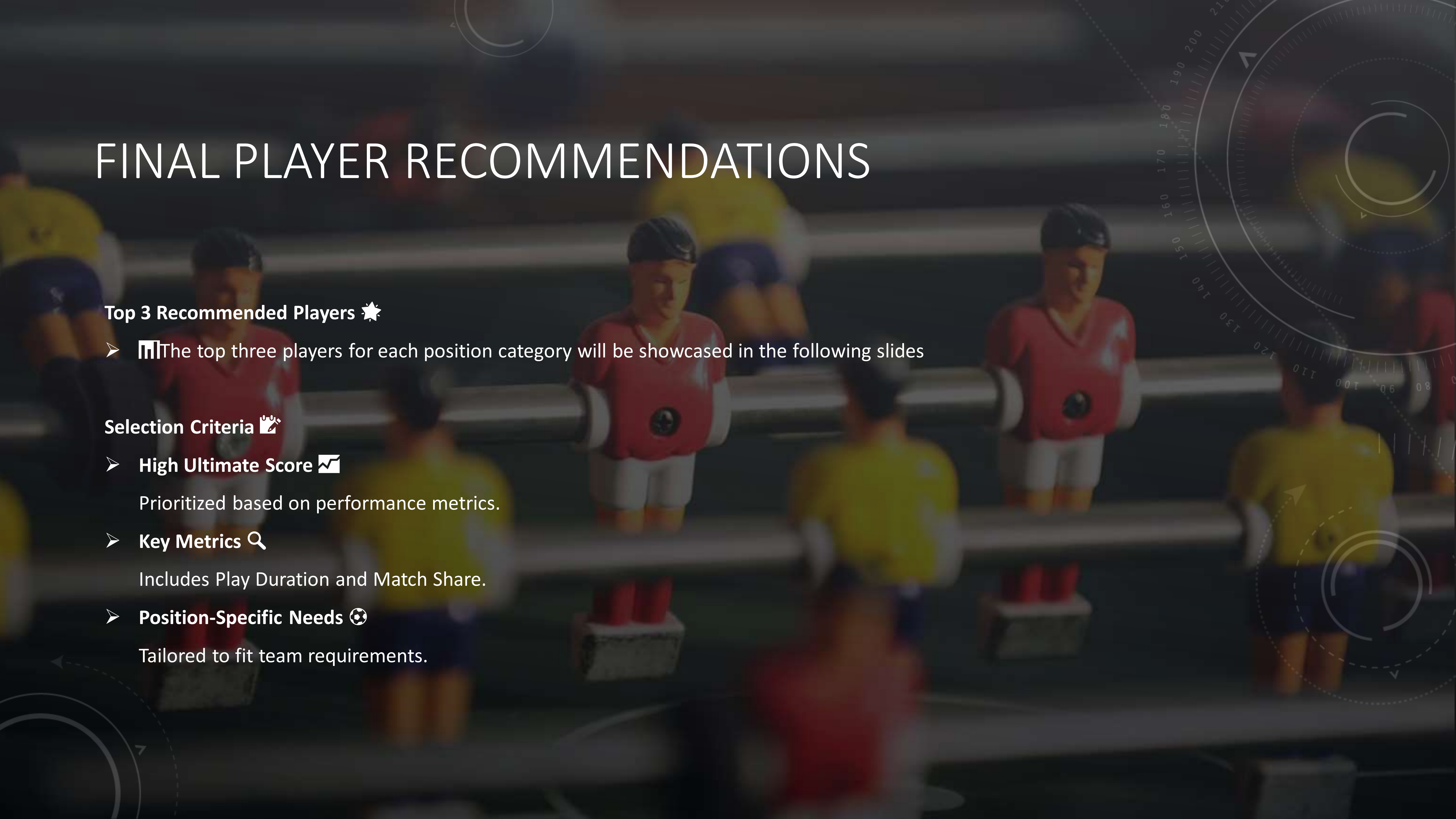


# PLAY DURATION VS MATCH SHARE



# TOP THREE VS ULTIMATE SCORE





## FINAL PLAYER RECOMMENDATIONS

**Top 3 Recommended Players 🌟**

- 📊The top three players for each position category will be showcased in the following slides

**Selection Criteria 📝**

- **High Ultimate Score 📈**

  Prioritized based on performance metrics.
- **Key Metrics 🔍**

  Includes Play Duration and Match Share.
- **Position-Specific Needs ⚽**

  Tailored to fit team requirements.

## 🏆 GOALKEEPERS

**1. Brian Maher**

- ✨ **Ultimate Score**: 11.78

**2. Ed McGinty**

- 🔝 **Ultimate Score**: 11.69

**3. Conor Kearns**

- 🏅 **Ultimate Score**: 11.40

## CENTRAL MIDFIELDERS ⚽□

1. **William Milovanovic**
   - 🌟 **Ultimate Score**: 5.26
2. **Romeo Amane**
   - 💫 **Ultimate Score**: 5.04
3. **Mikkel Rygaard**
   - 🏅 **Ultimate Score**: 4.84

# RIGHT WINGBACK DEFENDERS

**1. Darragh Power**

- ⚡**Ultimate Score**: 10.52

**2. Sean Gannon**

- 🔝 **Ultimate Score**: 8.89

**3. John Ross Wilson**

- 🏆 **Ultimate Score**: 8.77

## LEFT WINGBACK DEFENDERS ⍰

1. **Reece Hutchinson**
   - ⚡**Ultimate Score**: 11.68
2. **Paddy Kirk**
   - 🔝 **Ultimate Score**: 8.73
3. **Anto Breslin**
   - 🏆 **Ultimate Score**: 8.40

## DEFENSE MIDFIELDERS ⍰

1. **Jamie Lennon**
   - ⚡**Ultimate Score**: 10.39
2. **Niall Morahan**
   - 🔝 **Ultimate Score**: 9.32
3. **JJ Lunney**
   - 🏆 **Ultimate Score**: 8.84

# CENTER FORWARDS ⚽

1. **Padraig Amond**
   - ➢ 🌟 **Ultimate Score:** 11.51
2. **Patrick Hoban**
   - ➢ 💫 **Ultimate Score:** 9.02
3. **Frantz Pierrot**
   - ➢ 🏅 **Ultimate Score:** 7.57

# ATTACKING MIDFIELDERS 🎯

1. **Darragh Markey**
   - 🌟 **Ultimate Score:** 8.48
2. **David Hurley**
   - 💫 **Ultimate Score:** 6.68
3. **James Clarke**
   - 🏅 **Ultimate Score:** 5.51

# CENTRAL DEFENDERS ⍰

1. **Ollie Denham**
   - ⚡ **Ultimate Score**: 11.09
2. **Killian Brouder**
   - 🔝 **Ultimate Score**: 10.88
3. **Grant Horton**
   - 🏆 **Ultimate Score**: 10.86

## LEFT WINGERS 🌟

1. **Will Jarvis**
   - ⚡**Ultimate Score**: 7.48
2. **Ed McCarthy**
   - 🔝 **Ultimate Score**: 6.98
3. **Michael Duffy**
   - 🏆 **Ultimate Score**: 6.73

## RIGHT WINGERS 🌟

1. **Fabrice Hartmann**
   - ⚡**Ultimate Score**: 5.76
2. **Paul McMullan**
   - 🔝 **Ultimate Score**: 5.35
3. **Gustav Lundgren**
   - 🏆 **Ultimate Score**: 5.30

## CONCLUSION & NEXT STEPS

**Summary**:

- Data-driven approach to identify top talent.
- Key insights from the analysis.

**Next Steps**:

- Further validation with scouting.
- Possible additional analysis based on more data.

# THANK YOU! 🙏

**Questions? ❓**

- Feel free to ask!

**Contact Info: 📧**

- arjunsridhar445@gmail.com
- https://arjunsridhar.journoportfolio.com
- https://www.linkedin.com/in/arjun-sridhar-6466751b7
- https://github.com/smooth-glitch

**Project repository: 📂**

- https://github.com/smooth-glitch/charltonFC

**Looking forward to discussing my recommendations further!** ☺