

```
import kagglehub

# Download latest version
path = kagglehub.dataset_download("stefanoleone992/fifa-23-complete-player-dataset")

print("Path to dataset files:", path)

Downloading from https://www.kaggle.com/api/v1/datasets/download/stefanoleone992/fifa-23-complete-player-dataset?dataset_ver
100%|██████████| 1.58G/1.58G [00:58<00:00, 29.1MB/s]Extracting files...

Path to dataset files: /root/.cache/kagglehub/datasets/stefanoleone992/fifa-23-complete-player-dataset/versions/1

# Re-import necessary modules after crash
import pandas as pd
import os

# Redefine dataset path
dataset_path = "/root/.cache/kagglehub/datasets/stefanoleone992/fifa-23-complete-player-dataset/versions/1"
```

✓ Feasibility Study – Predicting Soccer Player Position from Attributes (FIFA 23)

✓ Define Project

Project Link

[Kaggle Dataset – FIFA 23 Complete Player Dataset](#)

Project Summary

This project builds a supervised machine learning model to predict a soccer player's **primary position group** (Forward, Midfielder, Defender, or Goalkeeper) based on technical and physical attributes in the FIFA 23 dataset. This can help scouts and coaches make better data-informed decisions about lineup and player development.

Data Description

The dataset includes over 19,000 players with 100+ features ranging from numerical stats (pace, dribbling, defending, etc.) to categorical info (team, nationality, preferred foot). We will clean, transform, and group this data for modeling.

Machine Learning Task

This is a **Supervised Multiclass Classification** problem with 4 target classes: FWD, MID, DEF, GK.

```
print("Path to dataset files:", path)




Path to dataset files: /root/.cache/kagglehub/datasets/stefanoleone992/fifa-23-complete-player-dataset/versions/1

dataset_path = "/root/.cache/kagglehub/datasets/stefanoleone992/fifa-23-complete-player-dataset/versions/1"
os.listdir(dataset_path)

['male_players (legacy).csv',
 'female_players (legacy).csv',
 'female_teams.csv',
 'male_coaches.csv',
 'male_teams.csv',
 'male_players.csv',
 'female_coaches.csv',
 'female_players.csv']
```

Dataset Scope and Justification

While the full FIFA 23 dataset includes both male and female players, we are using **male_players.csv** as our primary dataset for this feasibility study. This decision is based on the following reasons:

-  **Consistency in Features:** The male player dataset includes the most complete set of technical and physical attributes needed for position classification.
-  **Sample Size:** It contains significantly more rows (player entries), which helps improve model training and generalization.
-  **Reduced Noise:** Mixing male and female datasets could introduce structural or statistical differences (e.g., league, competition level) that may complicate training a clean model without substantial preprocessing or normalization.

- 🔍 **Clearer Labeling:** Position data in the male dataset is more standardized and aligned with mainstream position groupings (FWD, MID, DEF, GK).

Female player data or coaches/team files may be used in future expansions, but for the initial model development and feasibility testing, focusing on `male_players.csv` ensures better feature quality and less data fragmentation.

▼ Data Loading and Initial Look

To begin assessing the data quality, we first load the FIFA 23 dataset and check the number of data points (rows) and features (columns).

```
# Load just the first 1000 rows to avoid RAM crash
df_sample = pd.read_csv(dataset_path + "/male_players.csv", nrows=1000)

# Show shape
print("Sample shape (rows, columns):", df_sample.shape)

df_sample.head()
```


↗ Sample shape (rows, columns): (1000, 110)

	player_id	player_url	fifa_version	fifa_update	fifa_update_date	short_name	long_name	player_positions	overall
0	158023	/player/158023/lionel-messi/230009	23	9	2023-01-13	L. Messi	Lionel Andrés Messi Cuccittini	RW	91
1	165153	/player/165153/karim-benzema/230009	23	9	2023-01-13	K. Benzema	Karim Benzema	CF, ST	91
2	188545	/player/188545/robert-lewandowski/230009	23	9	2023-01-13	R. Lewandowski	Robert Lewandowski	ST	91
3	192985	/player/192985/kevin-de-bruyne/230009	23	9	2023-01-13	K. De Bruyne	Kevin De Bruyne	CM, CAM	91
4	231747	/player/231747/kylian-mbappe/230009	23	9	2023-01-13	K. Mbappé	Kylian Mbappé Lottin	ST, LW	91

5 rows x 110 columns

▼ Missing Values?

```
df_sample.isnull().sum()[df_sample.isnull().sum() > 0]
```




	0
value_eur	8
wage_eur	6
league_id	6
league_name	6
league_level	6
club_team_id	6
club_name	6
club_position	6
club_jersey_number	6
club_loaned_from	947
club_joined_date	59
club_contract_valid_until_year	6
nation_team_id	674
nation_position	674
nation_jersey_number	674
release_clause_eur	61
player_tags	690
player_traits	78
pace	120
shooting	120
passing	120
dribbling	120
defending	120
physic	120
goalkeeping_speed	880

dtype: int64

```
categorical_cols = df_sample.select_dtypes(include=['object']).columns.tolist()
numerical_cols = df_sample.select_dtypes(include=['number']).columns.tolist()

print("Categorical features:", len(categorical_cols))
print("Numerical features:", len(numerical_cols))
```

 Categorical features: 47
Numerical features: 63

Feature Summary Table

Summary table that includes:

- Feature type (categorical or numerical)
- Value ranges or categories
- Missing value count
- Outlier check (for numerical columns)

```
# Helper function to detect outliers using IQR
def count_outliers(series):
    if pd.api.types.is_numeric_dtype(series):
        Q1 = series.quantile(0.25)
        Q3 = series.quantile(0.75)
        IQR = Q3 - Q1
        lower = Q1 - 1.5 * IQR
        upper = Q3 + 1.5 * IQR
        return ((series < lower) | (series > upper)).sum()
```

```

    else:
        return np.nan # not applicable

# Build feature summary table
summary = []

for col in df_sample.columns:
    col_type = 'categorical' if col in categorical_cols else 'numerical'
    unique_vals = df_sample[col].unique()
    if col_type == 'categorical':
        val_summary = ", ".join(map(str, unique_vals[:5])) + ("..." if len(unique_vals) > 5 else "")
    else:
        val_summary = f"{df_sample[col].min()} to {df_sample[col].max()}"

    summary.append({
        "Feature": col,
        "Type": col_type,
        "Values": val_summary,
        "Missing": df_sample[col].isnull().sum(),
        "Outliers": count_outliers(df_sample[col])
    })

feature_summary_df = pd.DataFrame(summary)
```


The summary table below gives us a quick overview of each feature in the sample dataset. It helps us detect data quality issues and guides how we treat different feature types in preprocessing.



```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

import seaborn as sns
import warnings
warnings.filterwarnings("ignore")

# Display the summary table (first 20 rows)
feature_summary_df.head(20)
```



	Feature	Type	Values	Missing	Outliers	
0	player_id	numerical	1179 to 264240	0	15.0	
1	player_url	categorical	/player/158023/lionel-messi/230009, /player/16...	0	NaN	
2	fifa_version	numerical	23 to 23	0	0.0	
3	fifa_update	numerical	9 to 9	0	0.0	
4	fifa_update_date	categorical	2023-01-13	0	NaN	
5	short_name	categorical	L. Messi, K. Benzema, R. Lewandowski, K. De Br...	0	NaN	
6	long_name	categorical	Lionel Andrés Messi Cuccittini, Karim Benzema,...	0	NaN	
7	player_positions	categorical	RW, CF, ST, ST, CM, CAM, ST, LW...	0	NaN	
8	overall	numerical	77 to 91	0	18.0	
9	potential	numerical	77 to 95	0	8.0	
10	value_eur	numerical	1200000.0 to 190500000.0	8	64.0	
11	wage_eur	numerical	850.0 to 450000.0	6	71.0	
12	age	numerical	17 to 44	0	13.0	
13	dob	categorical	1987-06-24, 1987-12-19, 1988-08-21, 1991-06-28...	0	NaN	
14	height_cm	numerical	158 to 201	0	1.0	
15	weight_kg	numerical	56 to 103	0	4.0	
16	league_id	numerical	1.0 to 2012.0	6	81.0	
17	league_name	categorical	Ligue 1, La Liga, Premier League, Bundesliga, ...	6	NaN	
18	league_level	numerical	1.0 to 2.0	6	4.0	
19	club_team_id	numerical	1.0 to 116361.0	6	159.0	

Next steps:

[Generate code with feature_summary_df](#)

[View recommended plots](#)

[New interactive sheet](#)

✓ Comparing Feature Distributions Across Classes

Visualizing how each feature varies across player positions (target classes) to identify which features might be useful for classification.

```
# Define target column
target_col = 'player_positions'

# Select numerical features only
numerical_cols = feature_summary_df[feature_summary_df['Type'] == 'numerical']['Feature'].tolist()

# Plot histograms grouped by class

for col in numerical_cols:
    plt.figure(figsize=(8, 4))
    sns.histplot(data=df_sample, x=col, hue=target_col, element='step', stat='density', common_norm=False)
    plt.title(f'{col} by Player Position')
    plt.xlabel(col)
    plt.ylabel("Density")
    plt.tight_layout()
    plt.show()
```

 [Show hidden output](#)

✓ Grouping Player Positions into Broader Categories


The original dataset contains many unique values in the `player_positions` column, such as "ST", "CAM", "RB", etc., and combinations like "ST, CF". To simplify our classification task and make our feature analysis more interpretable, we are grouping players into broader categories: Goalkeeper, Defender, Midfielder, and Forward. This makes our visualizations and later modeling steps clearer and more meaningful.

```
# Define simplified position mapping
position_map = {
    'GK': 'Goalkeeper',
    'CB': 'Defender', 'LB': 'Defender', 'RB': 'Defender', 'LWB': 'Defender', 'RWB': 'Defender',
    'CDM': 'Midfielder', 'CM': 'Midfielder', 'CAM': 'Midfielder', 'LM': 'Midfielder', 'RM': 'Midfielder',
    'CF': 'Forward', 'ST': 'Forward', 'LW': 'Forward', 'RW': 'Forward'
}

# Helper function to extract primary position and map it
def map_position(pos_str):
    if pd.isna(pos_str):
        return 'Unknown'
    primary = pos_str.split(',')[0].strip()
    return position_map.get(primary, 'Other')

df_sample['position_group'] = df_sample['player_positions'].apply(map_position)

# Show value counts for sanity check
df_sample['position_group'].value_counts()
```

 **count**

position_group	
Midfielder	346
Defender	302
Forward	232
Goalkeeper	120

dtype: int64

✓ Re-Visualizing Feature Distributions by Broad Position Groups

To better understand how player attributes differ by role, we re-plot distributions using the `position_group` categories (Goalkeeper, Defender, Midfielder, Forward). This time, we focus on numeric features that are appropriate for density plots—specifically, those with enough variation and no known plotting issues.

```
# Define new target column
target_col = 'position_group'

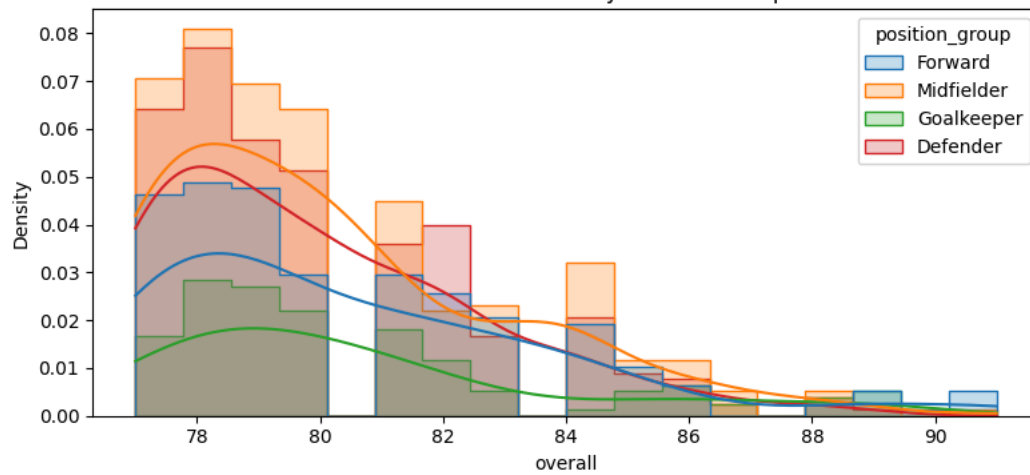
# Only keep useful numeric columns (remove problematic ones)
exclude_cols = ['player_id', 'fifa_version'] # Add more if needed

# Filter numeric columns that are not excluded and have enough unique values
plot_cols = [
    col for col in df_sample.columns
    if df_sample[col].dtype in ['float64', 'int64']
    and col != target_col
    and col not in exclude_cols
    and df_sample[col].nunique() > 10
]

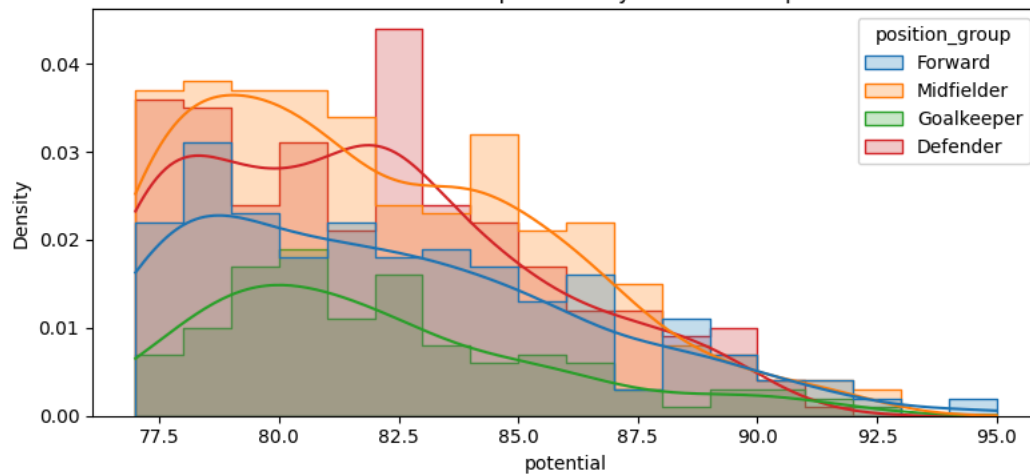
# Plot filtered histograms
for col in plot_cols:
    plt.figure(figsize=(8, 4))
    sns.histplot(data=df_sample, x=col, hue=target_col, kde=True, element='step', stat='density')
    plt.title(f'Distribution of {col} by Position Group')
    plt.xlabel(col)
    plt.ylabel('Density')
    plt.tight_layout()
    plt.show()
```



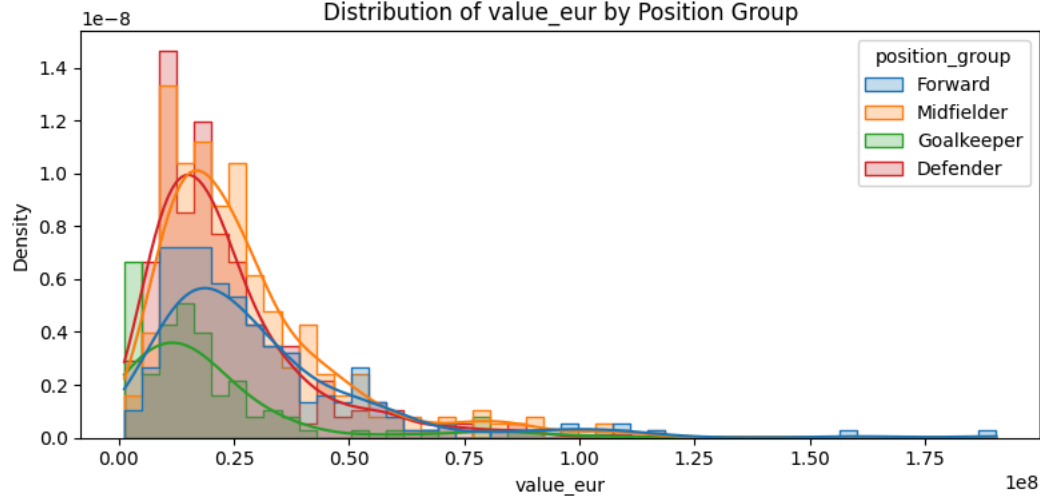
Distribution of overall by Position Group



Distribution of potential by Position Group



Distribution of value_eur by Position Group



Distribution of wage_eur by Position Group

