PIMPRI CHINCHWAD COLLEGE OF ENGINEERING
NIGDI , PUNE -411044

DEPARTMENT OF COMPUTER ENGINEERING

Semester – 3
Academic year 2024-25

Under the Guidance of

Prof. Namrata Gawande

Submitted by

Pratik Gaikwad (124B2B028)

Krishna Shinde(124B2B035)

Mayank Udapurkar(124B2B036)

INDEX

List of Figures & Tables

Introduction

a) Background :

Overview of IPL Auctions: Provide a brief description of the IPL auction process, explaining the significance of player selection, team strategies, and financial investment in players.

Objective of the Project: Your goal is to analyze the 2023-2024 IPL auction data to uncover insights such as the spending trends of teams, distribution of player prices, the influence of nationality on pricing, and which types of players (Type) fetched the highest prices.

b) Problem Statement:

The Indian Premier League (IPL) auction is a crucial event in professional cricket, where franchises bid for players to build their teams for the season. The prices at which players are sold vary significantly, and understanding the factors influencing these prices can provide valuable insights into team strategies, player valuation, and market trends.

c) Project Idea/Objectives

The main objectives of this project are:

- **Price Distribution Analysis**: Investigate the distribution of player prices to understand the auction dynamics and identify trends (e.g., how many players were bought at higher or lower prices).
- **Team Spending Behavior**: Compare the total spending by each team, identifying which franchises invested the most and how their spending correlates with the types of players they purchased.
- **Player Type Analysis**: Examine the correlation between player roles (batsman, bowler, all-rounder) and their auction prices, and determine if certain roles are valued higher than others.
- **Nationality Trends**: Analyze the distribution of players based on nationality (Indian vs. foreign players) and their price differences.

d) Scope :

This project aims to provide a comprehensive analysis of the IPL 2023-2024 auction, helping teams and analysts understand the factors that drive player prices and spending patterns. The insights derived from this project could contribute to more informed decision-making in future IPL auctions, particularly in optimizing player acquisition strategies based on player roles, age, and nationality.

2: Data Collection

Dataset Information

The dataset contains auction data for the IPL 2023-2024 season, detailing the players sold during the auction. It includes key fields such as Name (player's name), Team (the IPL franchise that bought the player), Price (the auction price in crores), Nationality (the player's country of origin), and Type (the player's role, such as batsman, bowler, or all-rounder). This dataset allows for analysis of player pricing trends, team spending behavior, the impact of nationality and player roles on pricing, and the distribution of player acquisition across various teams and types. The data helps to identify patterns in the IPL auction and understand how franchises allocate their resources.

Dataset Overview:

The IPL 2023-2024 Auction Dataset provides detailed information about the players who were sold during the IPL 2023-2024 auction, including their auction prices, teams, and roles. This dataset includes the following key columns:

- season: The IPL season (2023-2024).
- Name: The name of the player bought at the auction.
- Nationality: The country of origin of the player (e.g., India, Australia, South Africa).
- Type: The player's role or type, such as Batsman, Bowler, or All-rounder.
- Team: The IPL franchise that purchased the player (e.g., Mumbai Indians, Chennai Super Kings).
- Price: The auction price of the player, generally recorded in crores (1 crore = ₹10 million).

This dataset enables an in-depth analysis of player acquisition strategies by IPL teams, including trends in player pricing, team spending behavior, the influence of nationality and player roles on pricing, and the distribution of players by team and type. By analyzing this data, we can gain valuable insights into how teams prioritize their bids and what factors contribute most to a player’s auction price.

Dataset Attributes:

1. season

- **Description**: This field indicates the IPL season for which the auction data is relevant. In your case, it will primarily contain the value **2023-2024** to specify that the data pertains to the IPL 2023-2024 auction.
- **Type**: Categorical/String (e.g., '2023-2024')
- **Purpose**: Helps to filter the dataset by season, especially if you have data for multiple seasons.

---

2. Name

- **Description**: The name of the player who was sold in the IPL 2023-2024 auction.
- **Type**: String (e.g., 'Shubman Gill', 'Kieron Pollard')
- **Purpose**: Identifies the individual player who was bought during the auction. It can be used to search specific players or compare the prices of multiple players.

---

3. Nationality

- **Description**: The country of origin of the player (e.g., India, Australia, South Africa, etc.).
- **Type**: Categorical/String (e.g., 'India', 'Australia', 'South Africa')
- **Purpose**: Useful for analyzing the nationality distribution of players and determining how nationality affects pricing trends.

---

4. Type

- **Description**: This field represents the type of player. It categorizes players based on their roles in cricket:
  - **Batsman**: A player whose primary role is batting.
  - **Bowler**: A player whose primary role is bowling.
  - **All-rounder**: A player who can both bat and bowl effectively.
- **Type**: Categorical/String (e.g., 'Batsman', 'Bowler', 'All-rounder')
- **Purpose**: To understand how player roles influence auction prices and the distribution of players across teams.

---

5. `Team`

- **Description**: The IPL franchise that bought the player during the auction (e.g., 'Mumbai Indians',
- 'Chennai Super Kings', 'Delhi Capitals').
- **Type**: Categorical/String (e.g., 'Mumbai Indians', 'Kolkata Knight Riders')
- **Purpose**: To analyze spending patterns of different teams, team-specific strategies, and compare how much
- each team invested in player acquisition.

---

6. `Price`

- **Description**: The auction price of the player, typically recorded in **Indian Rupees (₹)** and often shown in
- **Crores** (1 Crore = ₹10 million).
- **Type**: Numeric (e.g., 8.0, 12.5, 4.5)
- **Purpose**: The main variable for analysis, as it is the most significant indicator of the player's value in the
- auction. This will be used for comparisons, visualizations, and price distribution analysis.

- **Data Source**

The dataset was sourced from www.kaggle.com.

This data is publicly accessible and provides a comprehensive view of factors that may affect sleep quality and health, making it valuable for analysis in health and wellness research.

3. Exploratory Data Analysis (EDA)

- **Data Preprocessing:**Data preprocessing is an important step in the data mining process. It refers to the cleaning, transforming, and integrating of data in order to make it ready for analysis. The goal of data preprocessing is to improve the quality of the data and to make it more suitable for the specific data mining task.

Steps to clean and prepare the dataset (handling missing values, removing outliers, etc.).

PROGRAM:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

- **`pandas` (pd)**:

Used for data manipulation and analysis. It provides data structures like DataFrames, which allow you to efficiently store and work with data in table format.

- **`numpy` (np)**:

Provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.

- **`matplotlib.pyplot` (plt)**:

A plotting library for creating static, interactive, and animated visualizations. particularly useful for creating basic charts like histograms, bar charts, and scatter plots.

- **`seaborn` (sns)**:

Built on top of `matplotlib`, `seaborn` is a statistical data visualization library that makes it easier to generate more complex plots and improve the aesthetics of your visualizations.

PROGRAM:

```
dataset = pd.read_csv("F:\PCCOE\Sem 3\DEVL\IPL_2023-22_Sold_Players.csv")

dataset
```

	Season	Name	Nationality	Type	Team	Price
0	2023	Ajinkya Rahane	Indian	Batter	Chennai Super Kings	50,00,000
1	2023	Bhagath Varma	Indian	All-Rounder	Chennai Super Kings	20,00,000
2	2023	Kyle Jamieson	Overseas	Bowler	Chennai Super Kings	1,00,00,000
3	2023	Ajay Mandal	Indian	All-Rounder	Chennai Super Kings	20,00,000
4	2023	Nishant Sindhu	Indian	All-Rounder	Chennai Super Kings	60,00,000
...	...	...	...	...	...	...
279	2022	Fazalhaq Farooqi	Overseas	Bowler	Sunrisers Hyderabad	50,00,000
280	2022	Sean Abbott	Overseas	Bowler	Sunrisers Hyderabad	2,40,00,000
281	2022	R Samarth	Indian	Batsman	Sunrisers Hyderabad	20,00,000
282	2022	Shashank Singh	Indian	All-Rounder	Sunrisers Hyderabad	20,00,000
283	2022	Saurabh Dubey	Indian	Bowler	Sunrisers Hyderabad	20,00,000

284 rows × 6 columns

Data Preprocessing :

print(dataset.head())

```
     Season              Name Nationality          Type  \
279    2022  Fazalhaq Farooqi    Overseas        Bowler
280    2022       Sean Abbott    Overseas        Bowler
281    2022         R Samarth      Indian       Batsman
282    2022    Shashank Singh      Indian   All-Rounder
283    2022     Saurabh Dubey      Indian        Bowler

                    Team         Price
279   Sunrisers Hyderabad    50,00,000
280   Sunrisers Hyderabad  2,40,00,000
281   Sunrisers Hyderabad    20,00,000
282   Sunrisers Hyderabad    20,00,000
283   Sunrisers Hyderabad    20,00,000
```

print(dataset.tail())

```
     Season              Name Nationality          Type  \
279    2022  Fazalhaq Farooqi    Overseas        Bowler
280    2022       Sean Abbott    Overseas        Bowler
281    2022         R Samarth      Indian       Batsman
282    2022    Shashank Singh      Indian   All-Rounder
283    2022     Saurabh Dubey      Indian        Bowler

                    Team         Price
279   Sunrisers Hyderabad    50,00,000
280   Sunrisers Hyderabad  2,40,00,000
281   Sunrisers Hyderabad    20,00,000
282   Sunrisers Hyderabad    20,00,000
283   Sunrisers Hyderabad    20,00,000
```

print(dataset.isnull().sum())

```
Season         0
Name           0
Nationality    0
Type           0
Team           0
Price          0
dtype: int64
```

dataset.dtypes

```
Season          int64
Name           object
Nationality    object
Type           object
Team           object
Price          object
dtype: object
```

dataset.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284 entries, 0 to 283
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Season       284 non-null    int64
 1   Name         284 non-null    object
 2   Nationality  284 non-null    object
 3   Type         284 non-null    object
 4   Team         284 non-null    object
 5   Price        284 non-null    object
dtypes: int64(1), object(5)
memory usage: 13.4+ KB
```

print(dataset['Price'].unique())

```
['50,00,000 ' '20,00,000 ' '1,00,00,000 ' '60,00,000 ' '16,25,00,000 '
 '2,00,00,000 ' '4,60,00,000 ' '2,40,00,000 ' '5,50,00,000 '
 '6,00,00,000 ' '1,20,00,000 ' '4,40,00,000 ' '1,50,00,000 ' '90,00,000 '
 '16,00,00,000 ' '75,00,000 ' '45,00,000 ' '17,50,00,000 ' '40,00,000 '
 '18,50,00,000 ' '5,75,00,000 ' '30,00,000 ' '3,20,00,000 ' '1,90,00,000 '
 '70,00,000 ' '13,25,00,000 ' '5,25,00,000 ' '2,60,00,000 ' '25,00,000 '
 '8,25,00,000 ' '1,80,00,000 ' '6,75,00,000 ' '14,00,00,000 '
 '4,00,00,000 ' '3,60,00,000 ' '6,25,00,000 ' '6,50,00,000 '
 '10,75,00,000 ' '1,10,00,000 ' '4,20,00,000 ' '65,00,000 ' '2,80,00,000 '
 '3,00,00,000 ' '10,00,00,000 ' '9,00,00,000 ' '1,40,00,000 '
 '1,70,00,000 ' '7,25,00,000 ' '12,25,00,000 ' '8,00,00,000 ' '55,00,000 '
 '7,50,00,000 ' '8,75,00,000 ' '15,25,00,000 ' '1,60,00,000 '
 '1,30,00,000 ' '9,25,00,000 ' '3,80,00,000 ' '11,50,00,000 '
 '5,00,00,000 ' '8,50,00,000 ' '7,75,00,000 ' '7,00,00,000 '
 '3,40,00,000 ' '95,00,000 ' '80,00,000 ']
```

dataset['Price'] = pd.to_numeric(dataset['Price'], errors='coerce')

dataset

	Season	Name	Nationality	Type	Team	Price
0	2023	Ajinkya Rahane	Indian	Batter	Chennai Super Kings	50,00,000
1	2023	Bhagath Varma	Indian	All-Rounder	Chennai Super Kings	20,00,000
2	2023	Kyle Jamieson	Overseas	Bowler	Chennai Super Kings	1,00,00,000
3	2023	Ajay Mandal	Indian	All-Rounder	Chennai Super Kings	20,00,000
4	2023	Nishant Sindhu	Indian	All-Rounder	Chennai Super Kings	60,00,000
...	...	...	...	...	...	...
279	2022	Fazalhaq Farooqi	Overseas	Bowler	Sunrisers Hyderabad	50,00,000
280	2022	Sean Abbott	Overseas	Bowler	Sunrisers Hyderabad	2,40,00,000
281	2022	R Samarth	Indian	Batsman	Sunrisers Hyderabad	20,00,000
282	2022	Shashank Singh	Indian	All-Rounder	Sunrisers Hyderabad	20,00,000
283	2022	Saurabh Dubey	Indian	Bowler	Sunrisers Hyderabad	20,00,000

284 rows × 6 columns

---

dataset.dtypes

```
Season          int64
Name           object
Nationality    object
Type           object
Team           object
Price          object
dtype: object
```

```
print(dataset['Type'].unique())
```

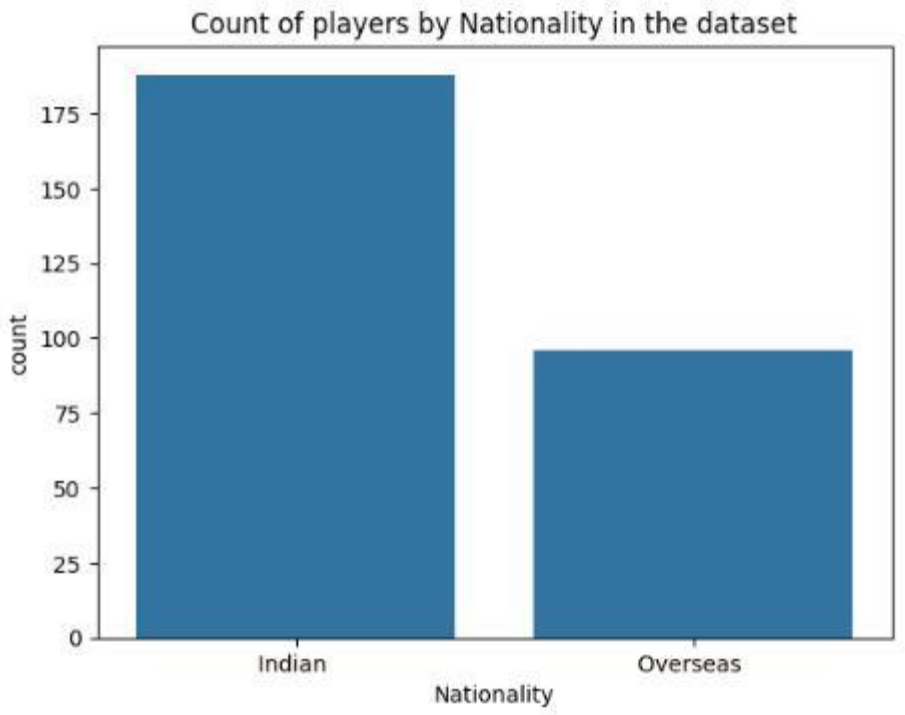
```
['Batter ' 'All-Rounder ' 'Bowler ' 'Wicket-Keeper ' 'Batsman '
 'Wicket Keeper ']
```

```
dataset['Type'].replace({'Batter ':'Batsman ','Wicket Keeper ':'Wicket-Keeper '},inplace=True)
```

```
print(dataset['Type'].unique())
```

```
['Batsman ' 'All-Rounder ' 'Bowler ' 'Wicket-Keeper ']
```

sns.countplot(x='Nationality',data=dataset)
plt.title("Count of players by Nationality in the dataset")
plt.show()

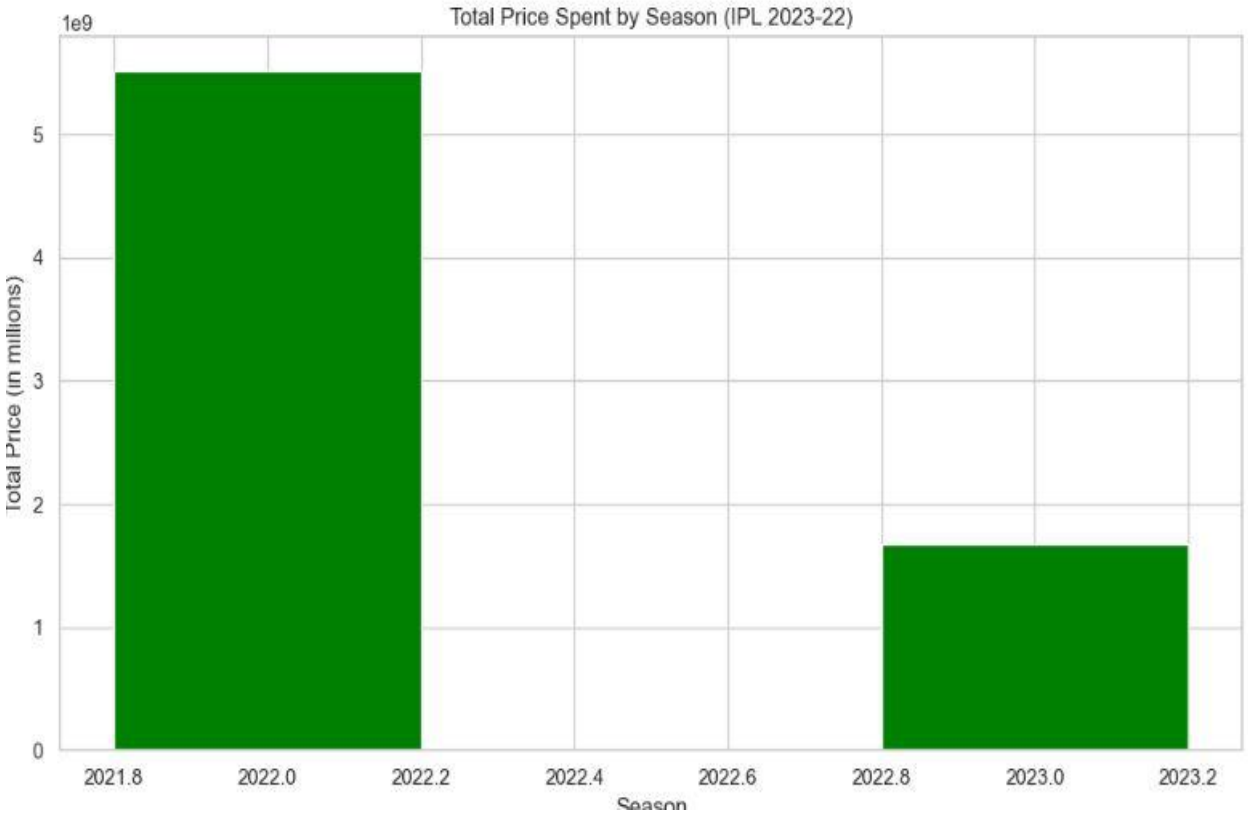


Explanation:

- `sns.countplot(x='Nationality', data=dataset)`:
  - `sns.countplot` is used to create a bar plot where each bar represents the count of occurrences of
  - each unique value in the `Nationality` column.
  - `x='Nationality'` tells `seaborn` to plot the counts for each nationality.
  - `data=dataset` specifies the DataFrame you're using.
- `plt.title("Count of players by Nationality in the dataset")`:
  - This sets the title of the plot, making it clear that you're visualizing the count of players based on their nationality.
- `plt.show()`:
  - This displays the plot on the screen.

```
total_price_by_season = dataset.groupby('Season')['Price'].sum()

# Creating the figure
fig = plt.figure(figsize=(10, 6))

# Creating the bar plot
plt.bar(total_price_by_season.index, total_price_by_season.values, color='green', width=0.4)

# Adding labels and title
plt.xlabel("Season")
plt.ylabel("Total Price (in millions)")
plt.title("Total Price Spent by Season (IPL 2023-22)")

# Displaying the plot
```

```
plt.tight_layout()  # Adjust layout to prevent clipping of tick-labels
plt.show()
```



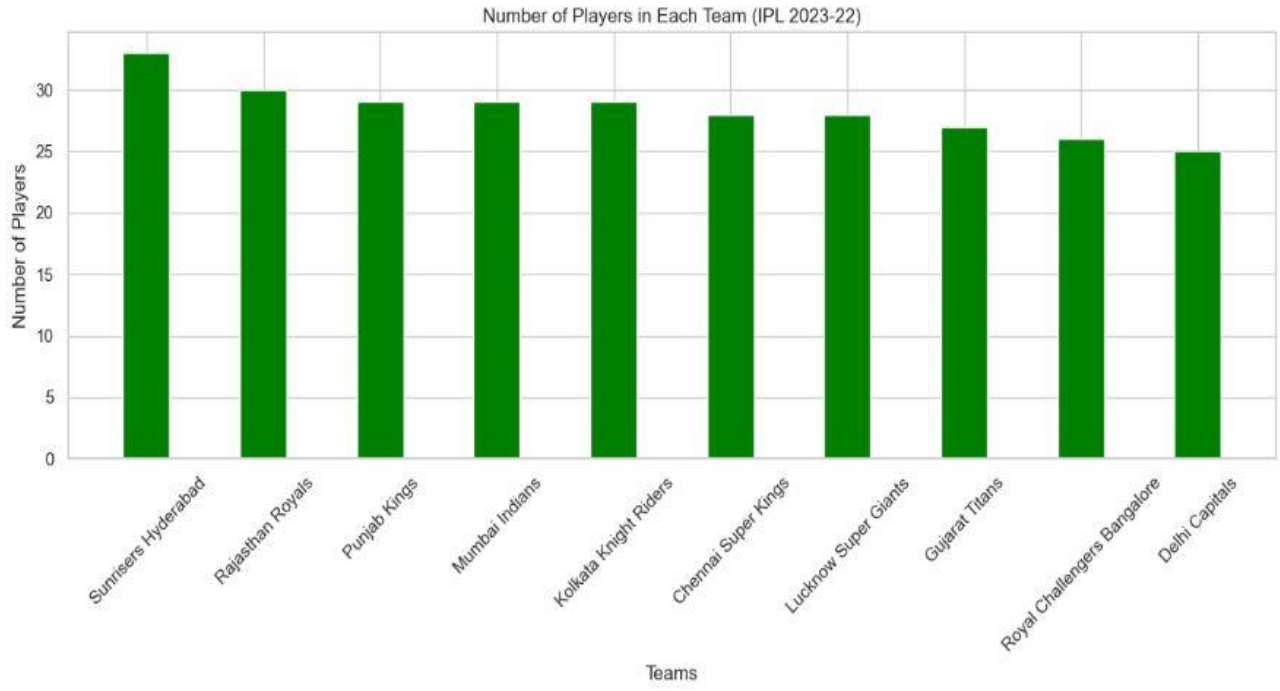
Explanation :

- `dataset.groupby('Season')`: This groups your data by the `Season` column (assuming the season is in the dataset).
- `['Price'].sum()`: For each season, it calculates the sum of the `Price` column, which represents the total money spent on players in that season.

- `total_price_by_season.index`: This is the x-axis (season).
- `total_price_by_season.values`: This is the y-axis (total price spent).
- `color='green'`: Sets the color of the bars to green.
- `width=0.4`: Adjusts the width of the bars to make them thinner for better spacing.

```
team_counts = dataset['Team'].value_counts()

# Creating the figure
fig = plt.figure(figsize=(12, 6))

# Creating the bar plot
plt.bar(team_counts.index, team_counts.values, color='green', width=0.4)

# Adding labels and title
plt.xlabel("Teams")
plt.ylabel("Number of Players")
plt.title("Number of Players in Each Team (IPL 2023-22)")
```

```
# Rotate team names for better visibility
plt.xticks(rotation=45)

# Displaying the plot
plt.tight_layout()  # Adjust layout to prevent clipping of tick-labels
plt.show()
```



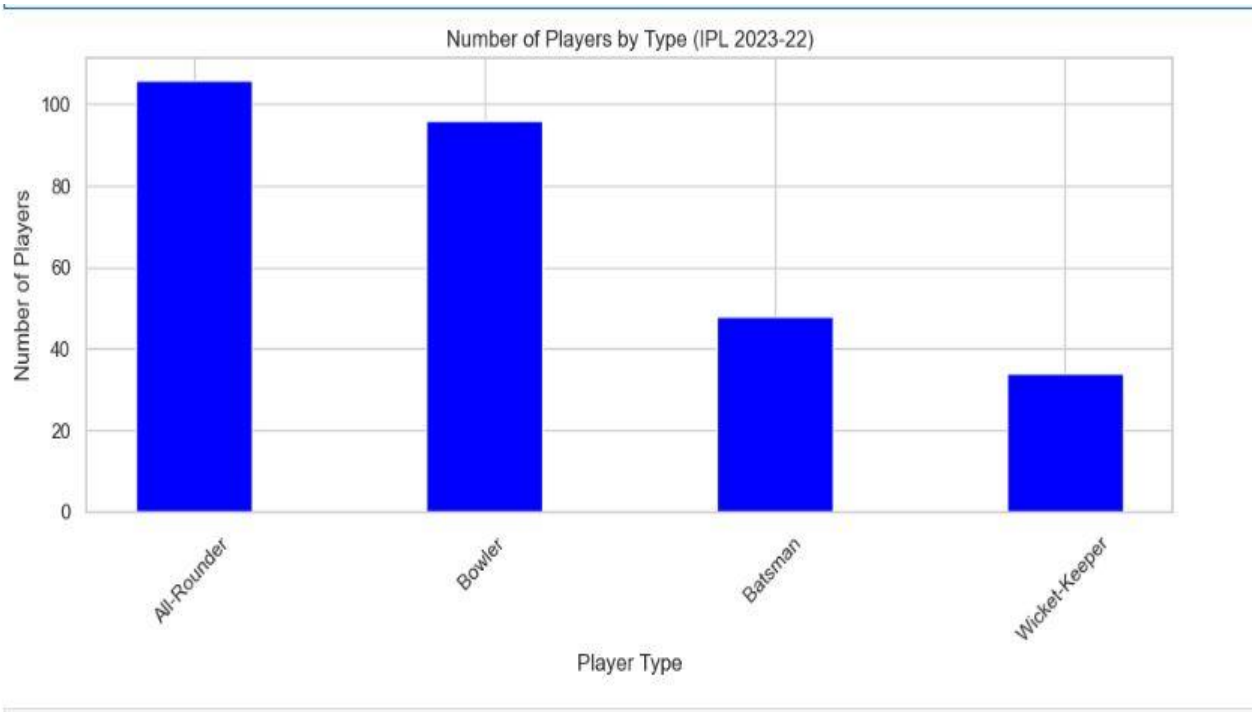
Explanation:

`dataset['Team'].value_counts()` counts the number of occurrences of each unique value in the `Team` column, which gives you the number of players bought by each team.

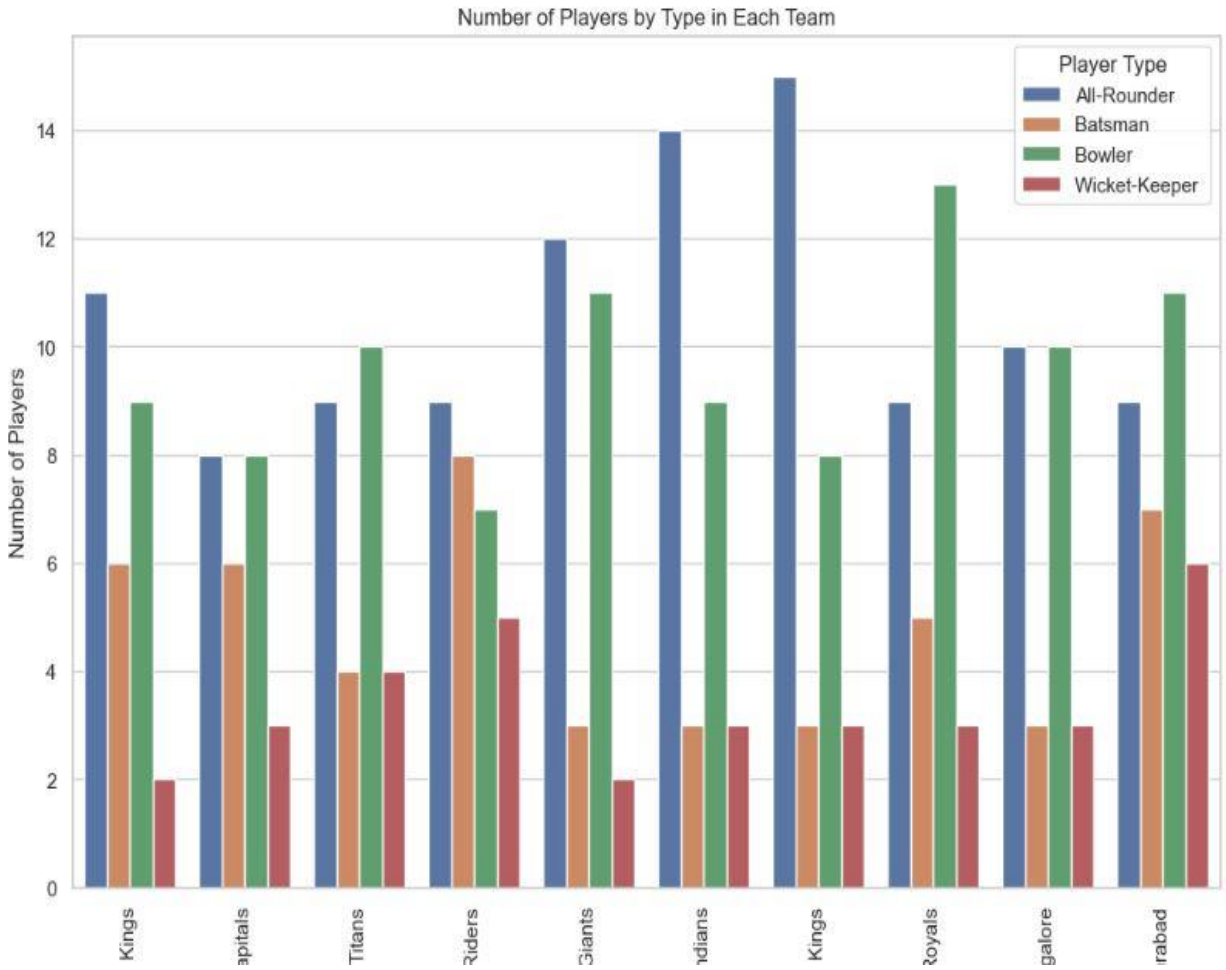
- `team_counts.index`: The x-axis will represent the names of the IPL teams.
- `team_counts.values`: The y-axis will show the number of players each team has bought.
- `color='green'`: This sets the color of the bars to green.
- `width=0.4`: This adjusts the width of the bars to make them thinner and avoid overlap.

```
type_counts = dataset['Type'].value_counts()

# Creating the figure
fig = plt.figure(figsize=(10, 5))

# Creating the bar plot
plt.bar(type_counts.index, type_counts.values, color='blue', width=0.4)

# Adding labels and title
plt.xlabel("Player Type")
plt.ylabel("Number of Players")
plt.title("Number of Players by Type (IPL 2023-22)")

# Rotate type names for better visibility
plt.xticks(rotation=45)

# Displaying the plot
```

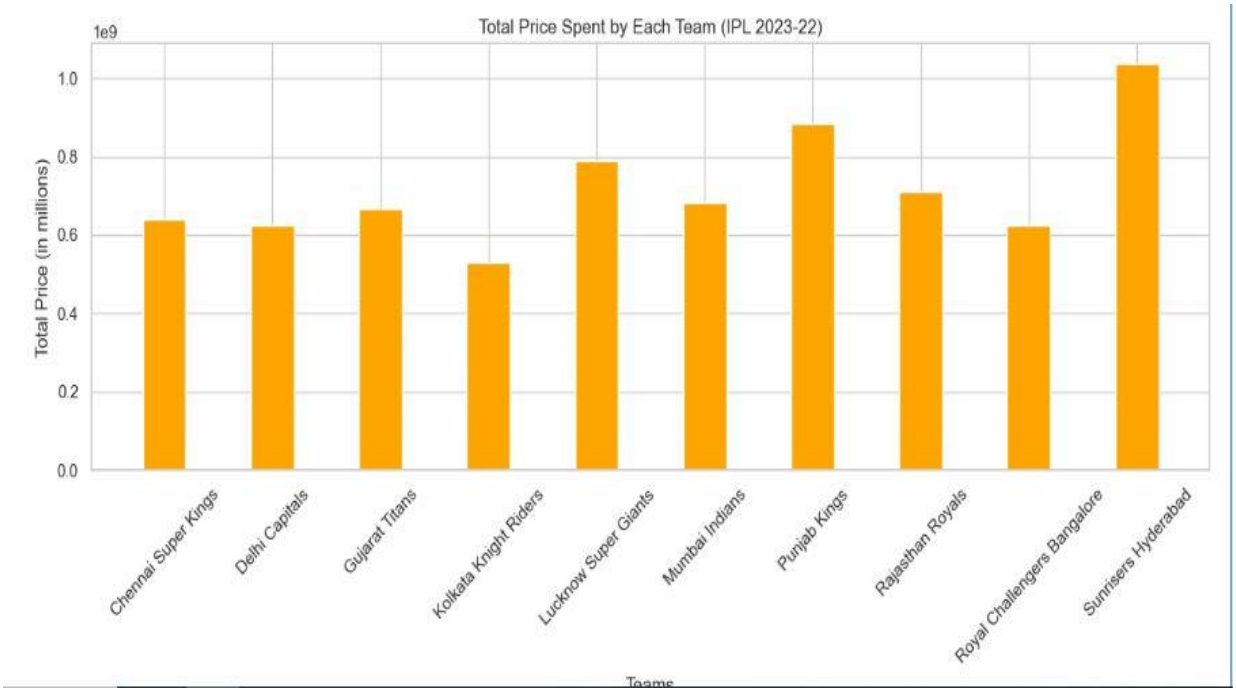
```
plt.tight_layout()  # Adjust layout to prevent clipping of tick-labels
plt.show()
```



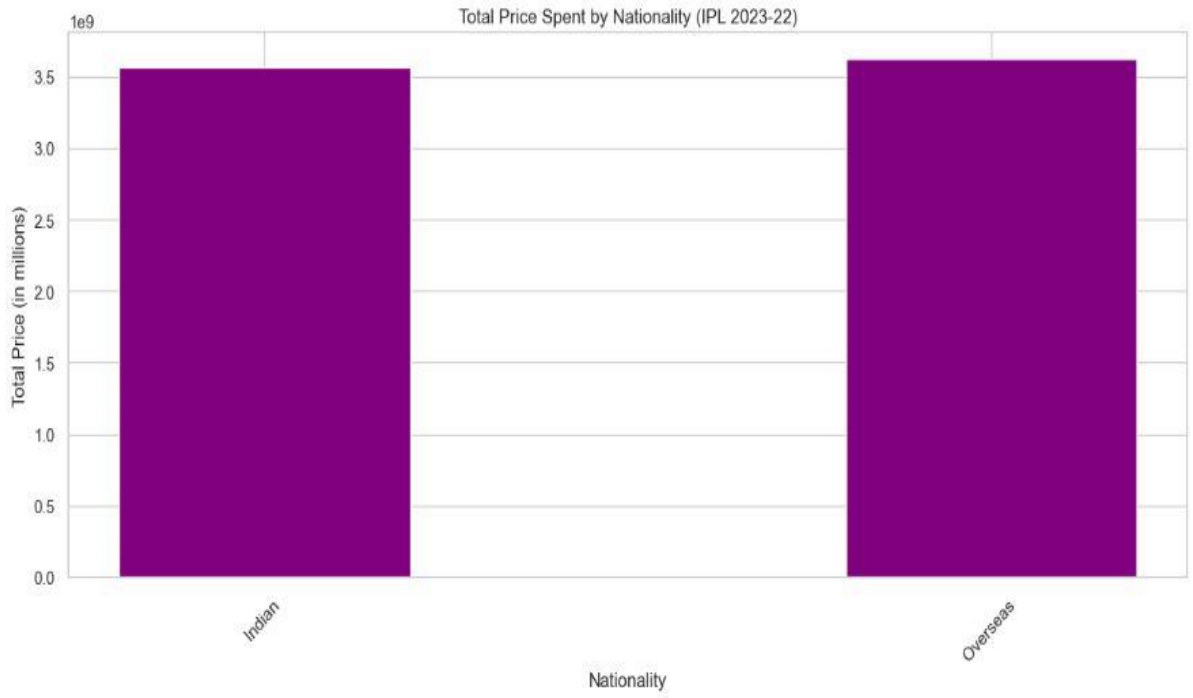

```
player_count = dataset.groupby(['Team', 'Type']).size().reset_index(name='Player_Count')

sns.set(style="whitegrid")

plt.figure(figsize=(12, 8))
sns.barplot(data=player_count, x='Team', y='Player_Count', hue='Type', errorbar=None)

plt.title('Number of Players by Type in Each Team')
plt.xlabel('Team Name')
plt.ylabel('Number of Players')
plt.xticks(rotation=90)
plt.legend(title='Player Type')


plt.show()
```



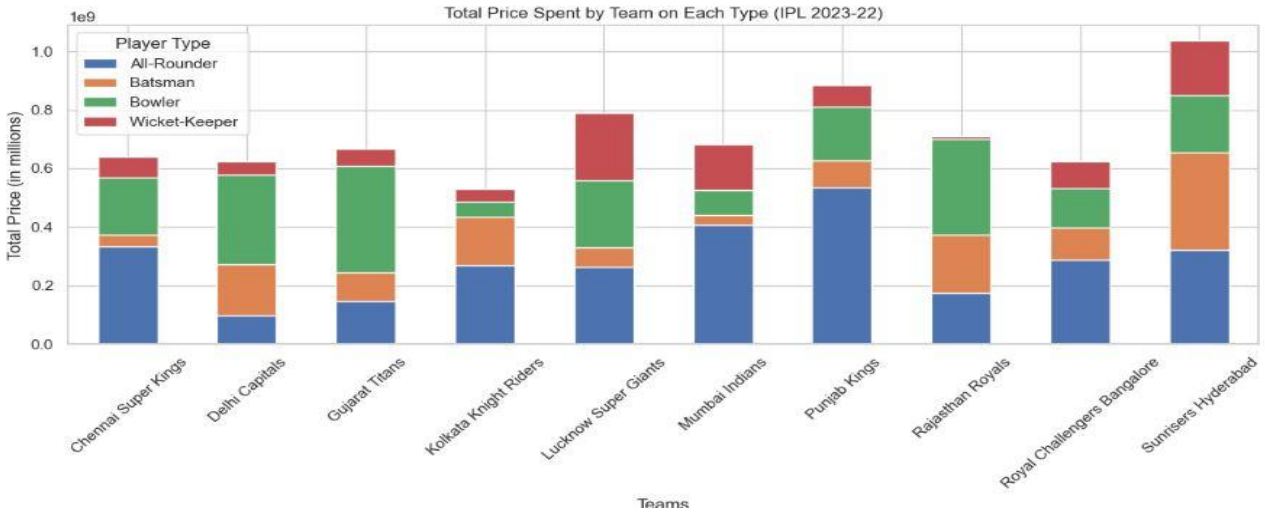

```
total_price_by_team = dataset.groupby('Team')['Price'].sum()

# Creating the figure
fig = plt.figure(figsize=(12, 6))

# Creating the bar plot
plt.bar(total_price_by_team.index, total_price_by_team.values, color='orange', width=0.4)

# Adding labels and title
plt.xlabel("Teams")
plt.ylabel("Total Price (in millions)")
plt.title("Total Price Spent by Each Team (IPL 2023-22)")

# Rotate team names for better visibility
plt.xticks(rotation=45)

# Displaying the plot
plt.tight_layout()  # Adjust layout to prevent clipping of tick-labels
    plt.show()
```



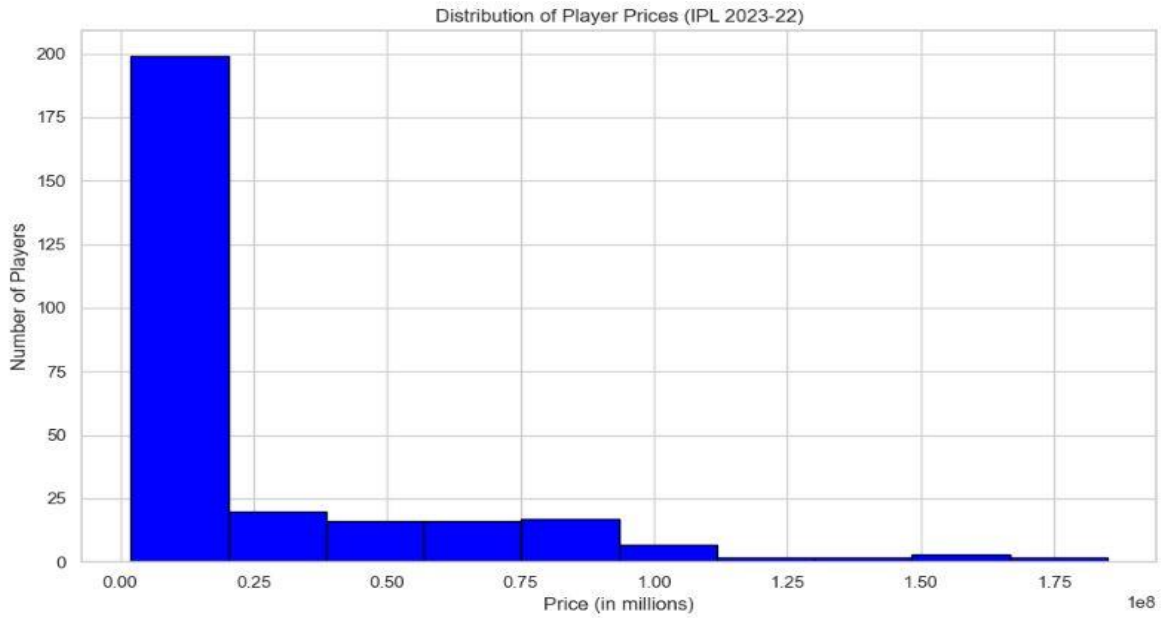

```
total_price_by_nationality = dataset.groupby('Nationality')['Price'].sum()

# Creating the figure
fig = plt.figure(figsize=(12, 6))

# Creating the bar plot
plt.bar(total_price_by_nationality.index, total_price_by_nationality.values, color='purple', width=0.4)

# Adding labels and title
plt.xlabel("Nationality")
plt.ylabel("Total Price (in millions)")
plt.title("Total Price Spent by Nationality (IPL 2023-22)")

# Rotate nationality names for better visibility
plt.xticks(rotation=45)

# Displaying the plot
plt.tight_layout()  # Adjust layout to prevent clipping of tick-labels
    plt.show()
```



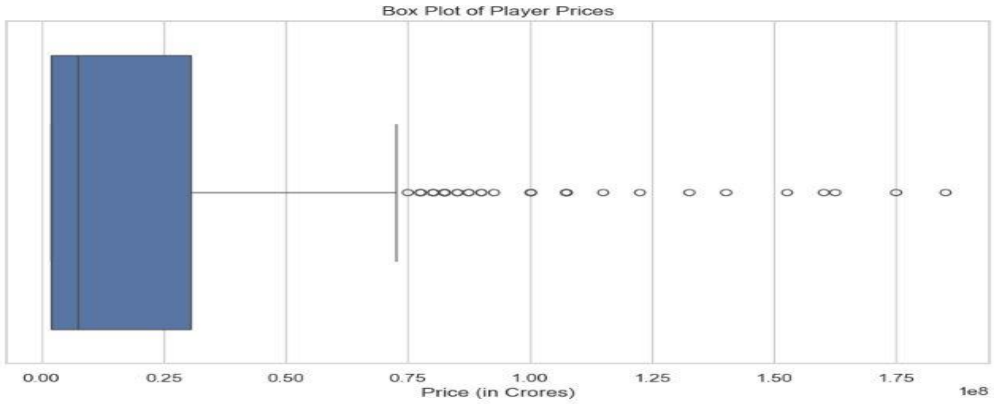

```
price_by_team_and_type = dataset.groupby(['Team', 'Type'])['Price'].sum().unstack()

# Creating the figure
fig = plt.figure(figsize=(12, 6))

# Creating the bar plot
price_by_team_and_type.plot(kind='bar', stacked=True, ax=fig.add_subplot(111))

# Adding labels and title
plt.xlabel("Teams")
plt.ylabel("Total Price (in millions)")
plt.title("Total Price Spent by Team on Each Type (IPL 2023-22)")

# Rotate team names for better visibility
plt.xticks(rotation=45)

# Displaying the plot
plt.tight_layout()  # Adjust layout to prevent clipping of tick-labels
plt.legend(title='Player Type')
    plt.show()
```



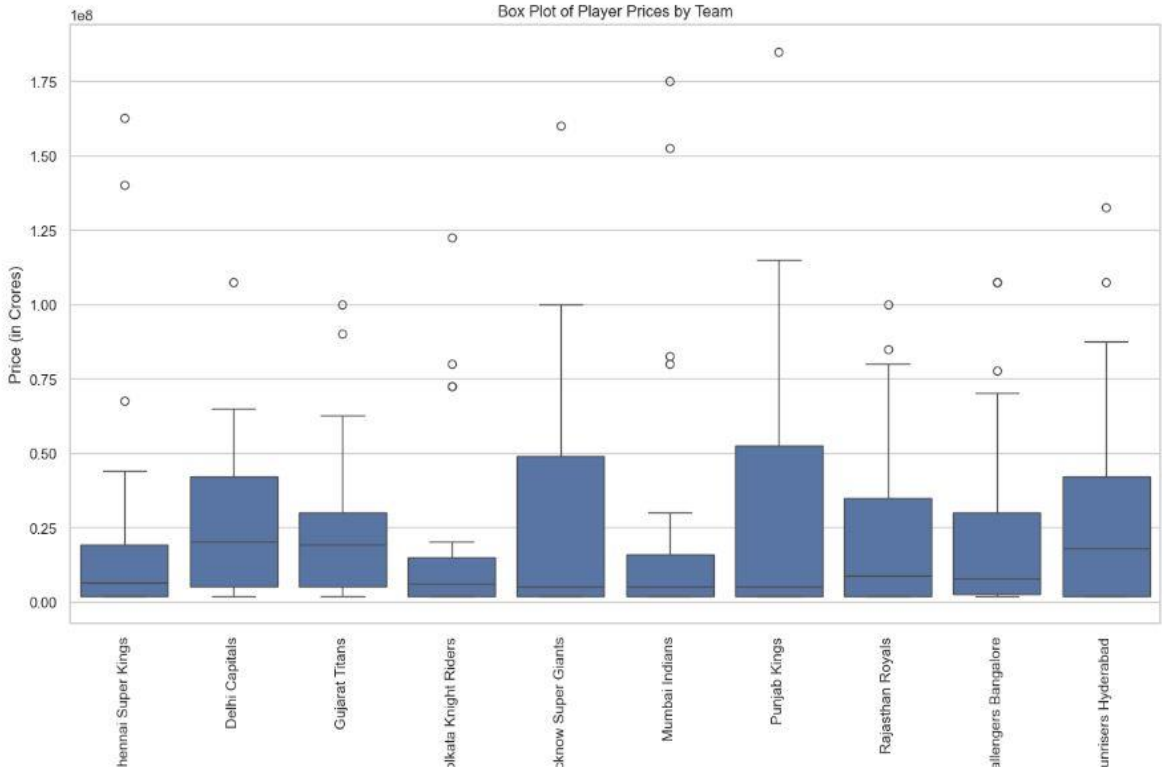

```
fig = plt.figure(figsize=(10, 6))

# Creating the histogram
plt.hist(dataset['Price'], bins=10, color='blue', edgecolor='black')

# Adding labels and title
plt.xlabel("Price (in millions)")
plt.ylabel("Number of Players")
plt.title("Distribution of Player Prices (IPL 2023-22)")

# Displaying the plot
plt.tight_layout()  # Adjust layout to prevent clipping of tick-labels
    plt.show()
```



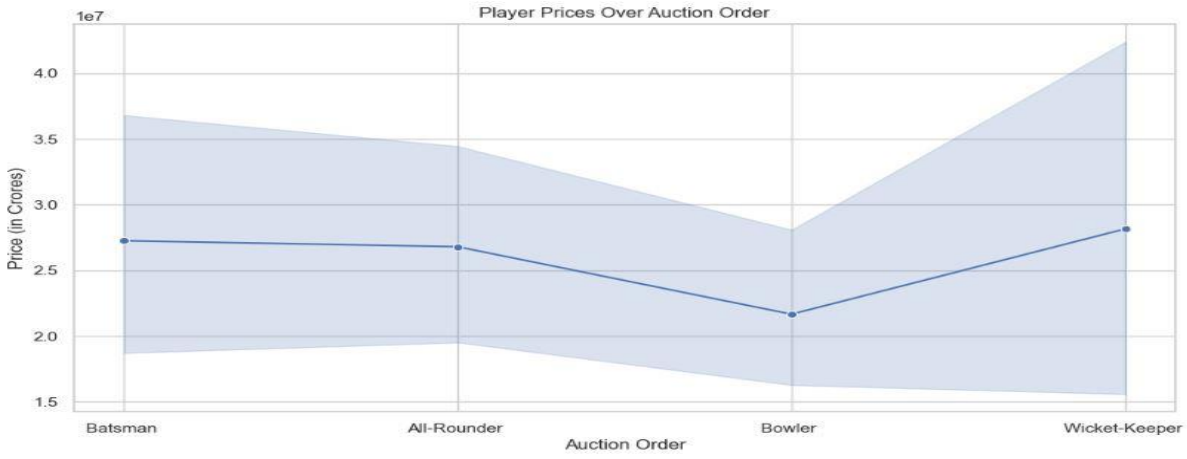
```
plt.figure(figsize=(10,6))
sns.boxplot(x=dataset['Price'])  # Adjust 'price' with the actual column name if needed
plt.title('Box Plot of Player Prices')
plt.xlabel('Price (in Crores)')
    plt.show()
```



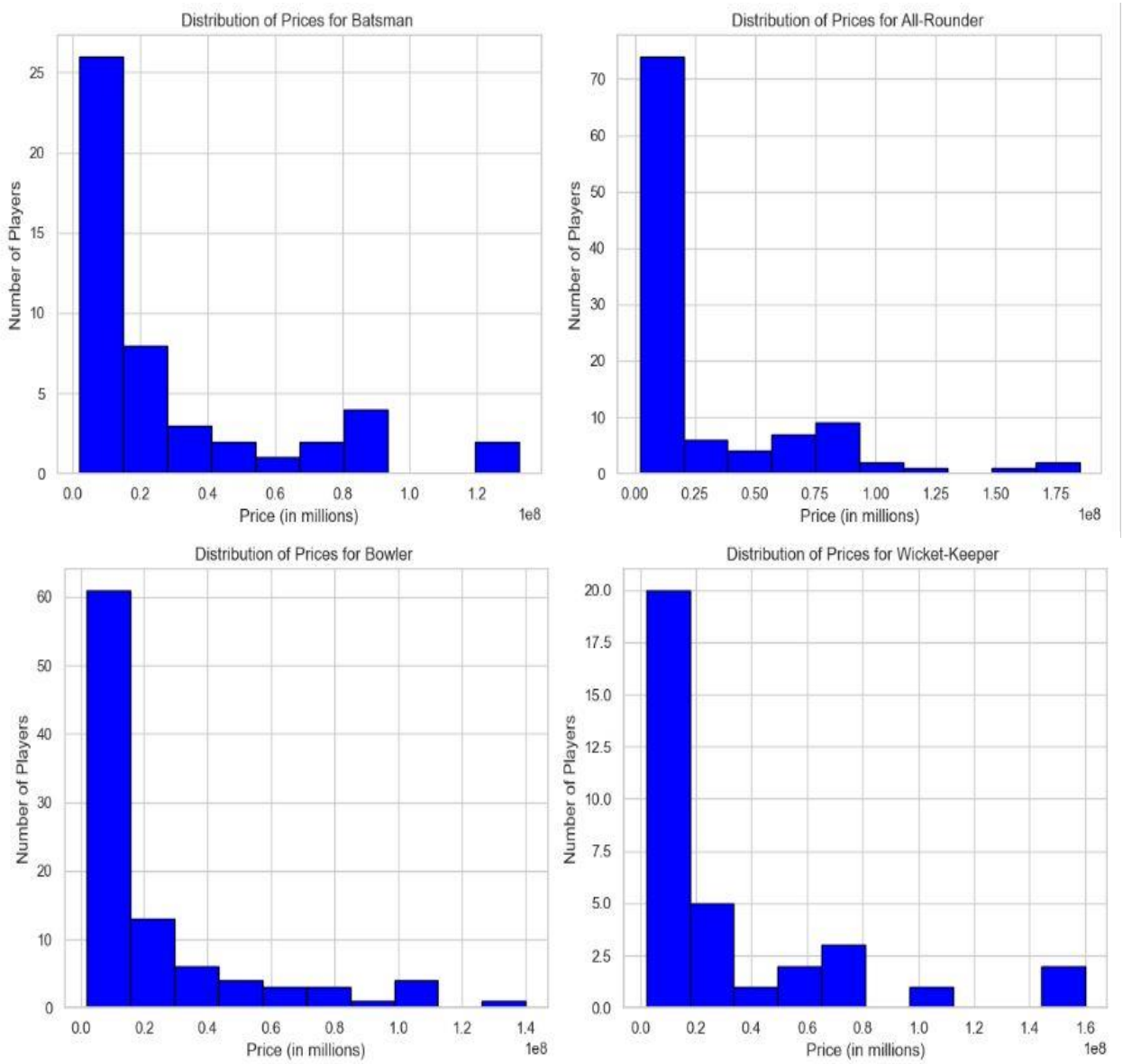

```
# Create a box plot of Player Prices by Team
plt.figure(figsize=(15, 8))  # Increase the figure size if necessary
sns.boxplot(x='Team', y='Price', data=dataset)

plt.title('Box Plot of Player Prices by Team')
plt.xlabel('Team')
plt.ylabel('Price (in Crores)')

plt.xticks(rotation=90)  # Rotate team names if they overlap
    plt.show()
```



```
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming 'auction_order' is the column representing the order in which players were sold
# and 'Price' is the column for player prices. Adjust if needed.
plt.figure(figsize=(12, 6))

sns.lineplot(x=dataset['Type'], y=dataset['Price'], marker='o')  # Adjust column names if necessary

# Title and labels
plt.title('Player Prices Over Auction Order')
plt.xlabel('Auction Order')
plt.ylabel('Price (in Crores)')

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

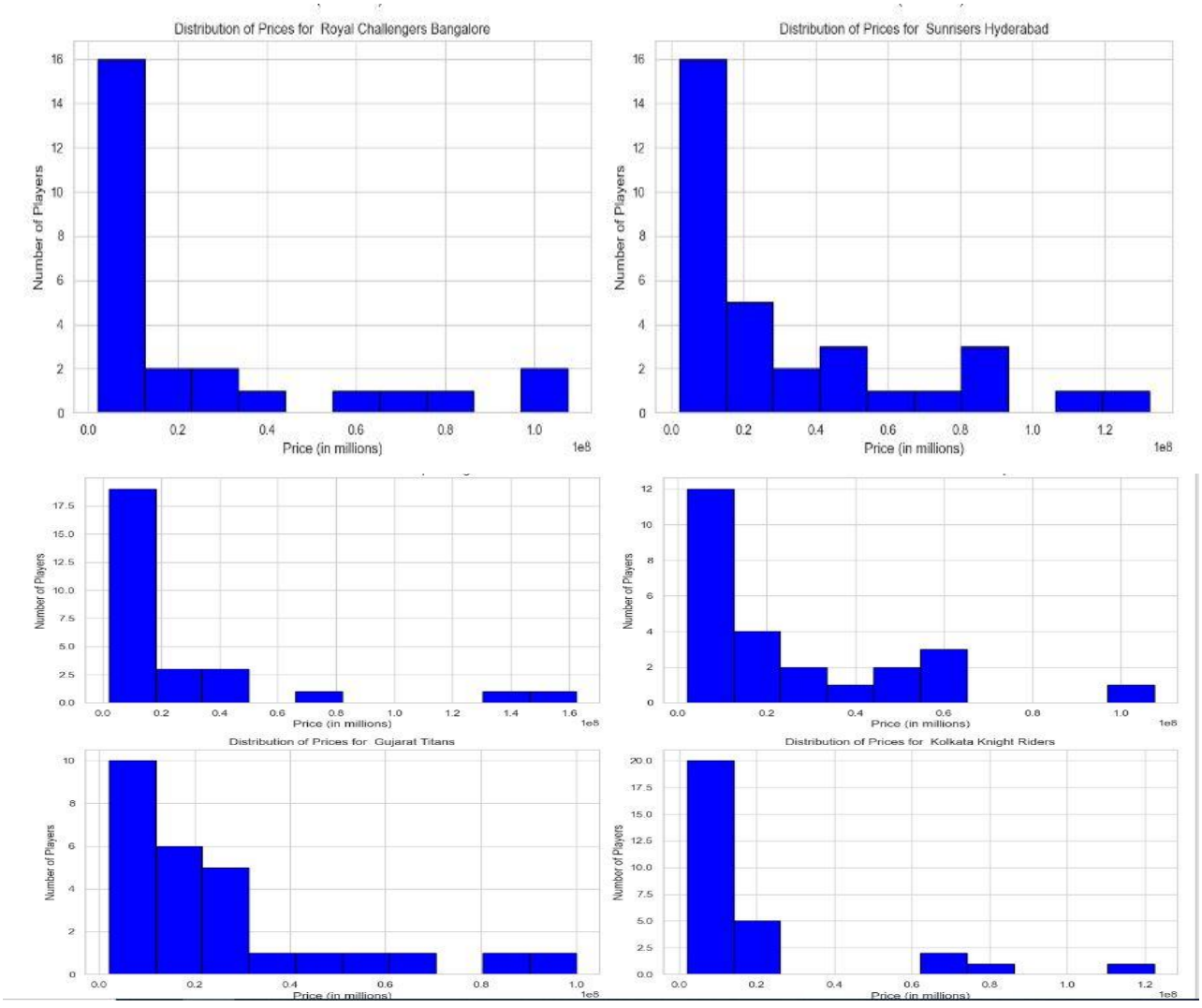


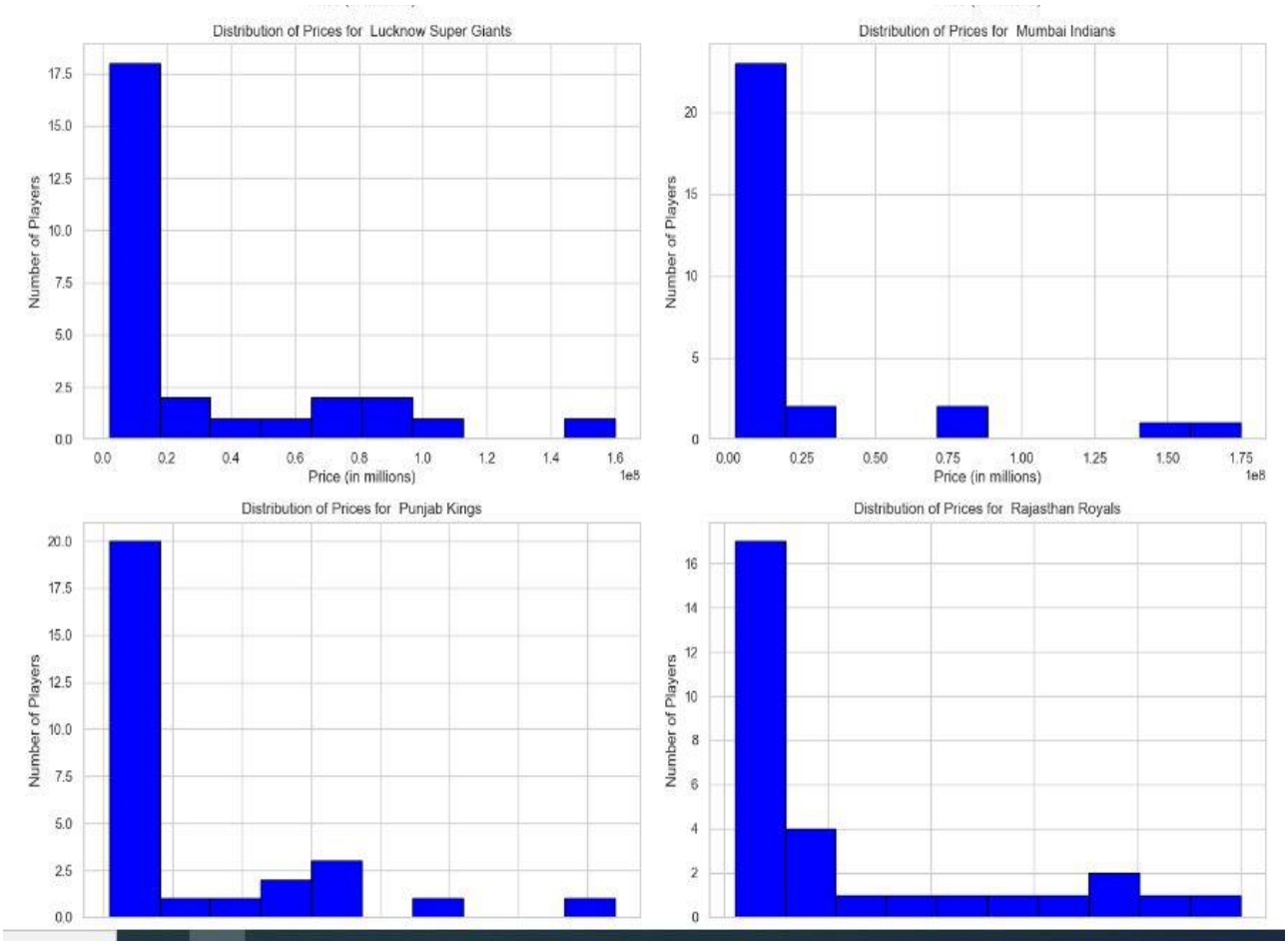

```
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(12, 10))
axes = axes.flatten()  # Flatten the 2D array of axes to 1D

# List of player types
player_types = dataset['Type'].unique()

# Plotting histograms for each player type
for i, player_type in enumerate(player_types):
    # Filter dataset for the current type
    prices = dataset[dataset['Type'] == player_type]['Price']

    # Creating the histogram
    axes[i].hist(prices, bins=10, color='blue', edgecolor='black')

    # Adding labels and title
    axes[i].set_xlabel("Price (in millions)")
    axes[i].set_ylabel("Number of Players")
    axes[i].set_title(f"Distribution of Prices for {player_type}")

# Adjust layout
plt.tight_layout()
    plt.show()
```



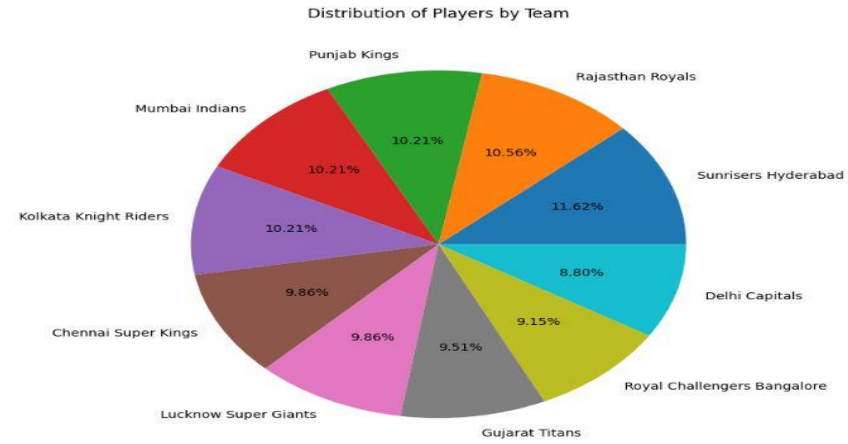

```
teams = dataset['Team'].unique()
num_teams = len(teams)

# Calculate the number of rows and columns needed for subplots
cols = 2
rows = np.ceil(num_teams / cols).astype(int)

# Creating the figure with dynamic subplots
fig, axes = plt.subplots(nrows=rows, ncols=cols, figsize=(15, 5 * rows))
axes = axes.flatten()  # Flatten the 2D array of axes to 1D

# Plotting histograms for each team
for i, team in enumerate(teams):
    # Filter dataset for the current team
    prices = dataset[dataset['Team'] == team]['Price']
```

```
    # Creating the histogram
    axes[i].hist(prices, bins=10, color='blue', edgecolor='black')

    # Adding labels and title
    axes[i].set_xlabel("Price (in millions)")
    axes[i].set_ylabel("Number of Players")
    axes[i].set_title(f"Distribution of Prices for {team}")

# Hide any unused subplots
for j in range(i + 1, len(axes)):
    axes[j].axis('off')

# Adjust layout
plt.tight_layout()
      plt.show()
```



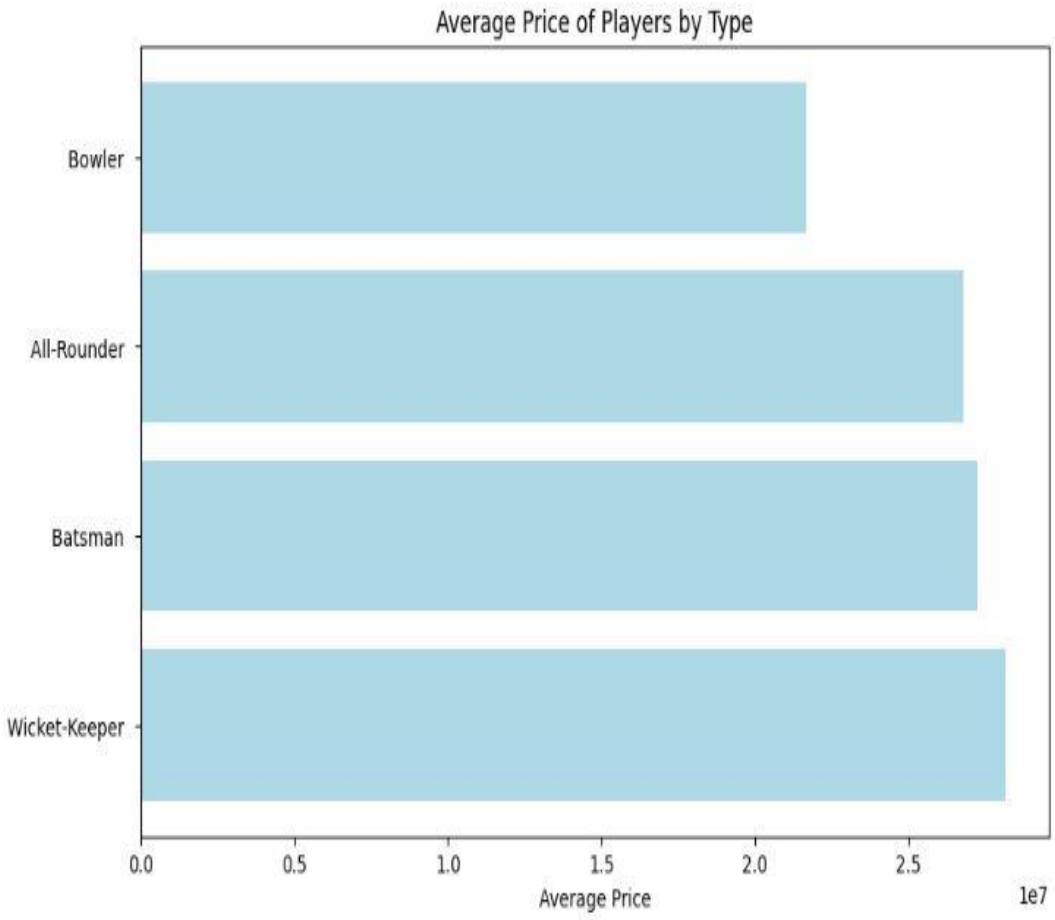



```
team_counts = dataset['Team'].value_counts()

# Plot the pie chart
plt.figure(figsize=(10, 7))
plt.pie(team_counts, labels=team_counts.index, autopct='%1.2f%%', startangle=0)
plt.title('Distribution of Players by Team\n\n')

    plt.axis('equal')
```

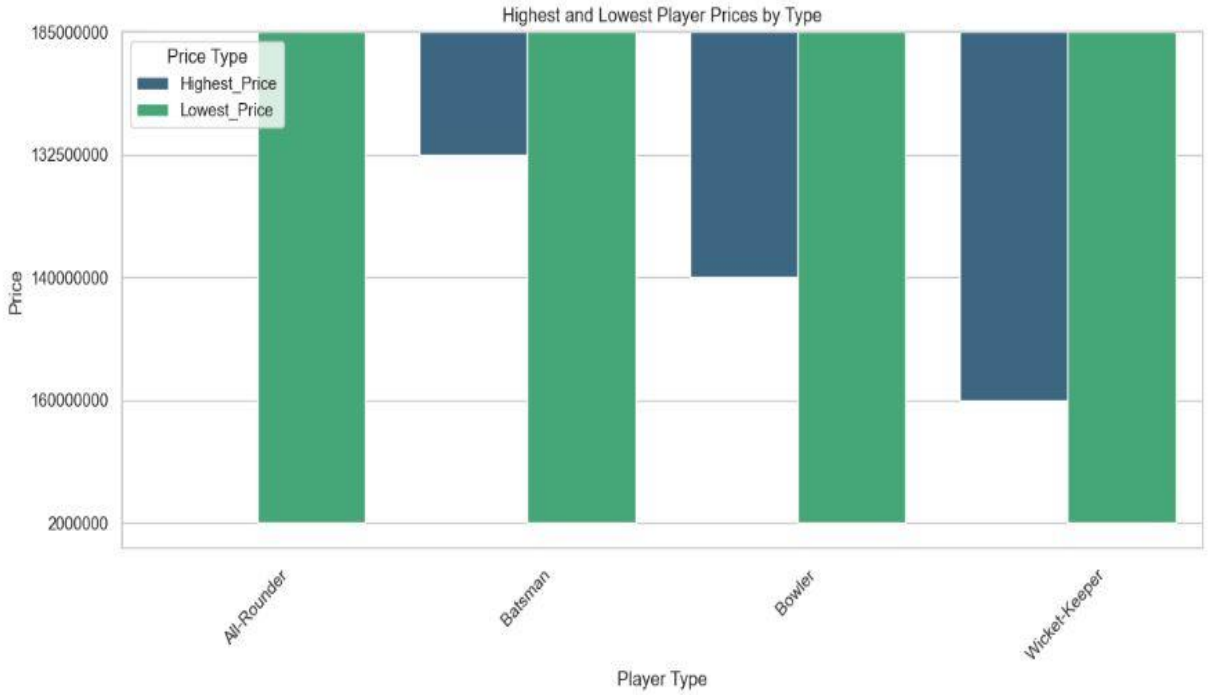


```
average_prices = dataset.groupby('Type')['Price'].mean().reset_index()

# Sort the results for better visualization
average_prices.sort_values(by='Price', ascending=False, inplace=True)

# Display the average prices
print("Average Prices by Player Type:\n", average_prices)

plt.figure(figsize=(10, 6))
plt.barh(average_prices['Type'], average_prices['Price'], color='lightblue')
plt.xlabel('Average Price')
plt.title('Average Price of Players by Type')
    plt.show()
```

```
Average Prices by Player Type:
          Type         Price
3  Wicket-Keeper  2.816176e+07
1        Batsman  2.726042e+07
0    All-Rounder  2.680189e+07
2         Bowler  2.166667e+07
```



```
price_stats = dataset.groupby('Type')['Price'].agg(['max', 'min']).reset_index()

# Rename the columns for clarity
price_stats.columns = ['Type', 'Highest_Price', 'Lowest_Price']

# Convert prices to string format with commas
price_stats['Highest_Price'] = price_stats['Highest_Price'].astype(str).str.replace('.', '').str.replace('e', '').str.replace(' ', '')
price_stats['Lowest_Price'] = price_stats['Lowest_Price'].astype(str).str.replace('.', '').str.replace('e', '').str.replace(' ', '')

# Melt the DataFrame for easier plotting
price_stats_melted = price_stats.melt(id_vars='Type', value_vars=['Highest_Price', 'Lowest_Price'],
                                      var_name='Price_Type', value_name='Price')

# Set the visual style of seaborn
sns.set(style="whitegrid")

# Create a bar plot
plt.figure(figsize=(12, 6))
sns.barplot(data=price_stats_melted, x='Type', y='Price', hue='Price_Type', palette='viridis')

# Customize the plot
plt.title('Highest and Lowest Player Prices by Type')
plt.xlabel('Player Type')
plt.ylabel('Price')

plt.xticks(rotation=45)
plt.legend(title='Price Type')
plt.tight_layout()

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



pip install scikit-learn

Once the installation is complete, you can verify that `scikit-learn` has been installed correctly by running the following command in a Python shell:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import numpy as np

# read_csv() is an import pandas function
data=pd.read_csv('F:\PCCOE\Sem 3\DEVL\IPL_2023-22_Sold_Players.csv')
    print(data)
```

```
     Season              Name Nationality          Type  \
0      2023    Ajinkya Rahane      Indian        Batter
1      2023     Bhagath Varma      Indian   All-Rounder
2      2023     Kyle Jamieson    Overseas        Bowler
3      2023       Ajay Mandal      Indian   All-Rounder
4      2023    Nishant Sindhu      Indian   All-Rounder
..      ...               ...         ...           ...
279    2022  Fazalhaq Farooqi    Overseas        Bowler
280    2022       Sean Abbott    Overseas        Bowler
281    2022         R Samarth      Indian       Batsman
282    2022    Shashank Singh      Indian   All-Rounder
283    2022     Saurabh Dubey      Indian        Bowler

                    Team         Price
0     Chennai Super Kings    50,00,000
1     Chennai Super Kings    20,00,000
2     Chennai Super Kings  1,00,00,000
3     Chennai Super Kings    20,00,000
4     Chennai Super Kings    60,00,000
..                    ...           ...
279   Sunrisers Hyderabad    50,00,000
280   Sunrisers Hyderabad  2,40,00,000
281   Sunrisers Hyderabad    20,00,000
282   Sunrisers Hyderabad    20,00,000
283   Sunrisers Hyderabad    20,00,000

[284 rows x 6 columns]
```

```
data.fillna(data.mean(), inplace=True)
    print(data)
```

```
     Season  Name_Abdul P A   Name_Abhijeet Tomar   Name_Abhinav Sadarangani   \
0      2023            False                 False                      False
1      2023            False                 False                      False
2      2023            False                 False                      False
3      2023            False                 False                      False
4      2023            False                 False                      False
..      ...              ...                   ...                        ...
279    2022            False                 False                      False
280    2022            False                 False                      False
281    2022            False                 False                      False
282    2022            False                 False                      False
283    2022            False                 False                      False

     Name_Abhishek Sharma   Name_Adam Milne   Name_Adam Zampa   \
0                   False             False             False
1                   False             False             False
2                   False             False             False
3                   False             False             False
4                   False             False             False
..                    ...               ...               ...
279                 False             False             False
280                 False             False             False
281                 False             False             False
282                 False             False             False
283                 False             False             False

     Name_Adil Rashid   Name_Aiden Markram   Name_Ajay Mandal   ...  \
0               False                False              False  ...
1               False                False              False  ...
2               False                False              False  ...
3               False                False               True  ...
4               False                False              False  ...
..                ...                  ...                ...  ...
279             False                False              False  ...
280             False                False              False  ...
281             False                False              False  ...
282             False                False              False  ...
283             False                False              False  ...
```

```
data = pd.get_dummies(data)
```

Explanation:

`pd.get_dummies(data)`**: This function automatically identifies categorical columns in the DataFrame (`data`) and converts them into numerical columns, with each category represented as a binary (0 or 1) column.**

- For each unique category in a categorical column, a new binary column is created.
- If the original value matches the category, the column will have a `1`; otherwise, it will have a `0`.

This is a common step in preparing data for machine learning models, as most models expect numerical input categorical data.

```
import pandas as pd
from sklearn.model_selection import train_test_split

# Load your dataset (adjust path if needed)
dataset = pd.read_csv("F:/PCCOE/Sem 3/DEVL/IPL_2023-22_Sold_Players.csv")

# Drop any non-numeric columns if needed, or choose specific features
# You can use 'Price' or any other numerical columns to create the split
# If you want to use all features to train a model, make sure to separate X (features) and y (target)

# Example: Splitting on features and target variable
X = dataset.drop(columns=['Price'])  # Features (everything except 'Price')
    y = dataset['Price']  # Target variable (Price)
```

```
# Split the entire dataset into training and testing
train_data, test_data = train_test_split(dataset, test_size=0.2, random_state=42)

# Check the shape of the resulting datasets
print(f"Training data size: {train_data.shape[0]} rows")
    print(f"Testing data size: {test_data.shape[0]} rows")
```

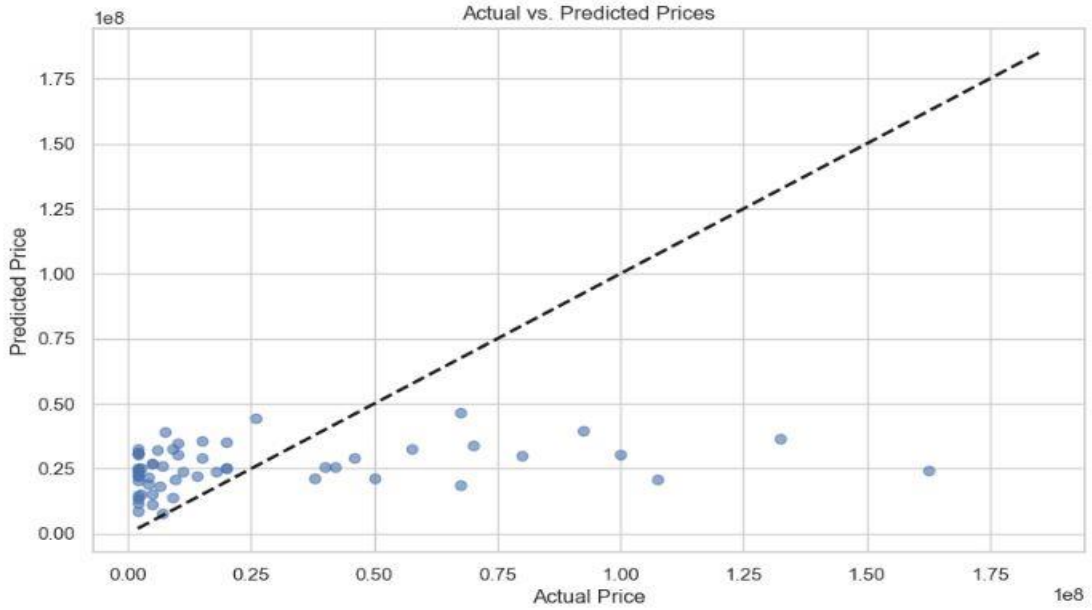
Explanation:

`train_test_split()` splits the entire dataset into training and testing subsets.

- `dataset`: The DataFrame to be split.
- `test_size=0.2`: 20% of the data will be used for testing, and 80% will be used for training.
- `random_state=42`: This ensures reproducibility. If you run the code multiple times, you'll always get the same random split.
- `train_data.shape[0]` gives the number of rows (samples) in the training dataset.
- `test_data.shape[0]` gives the number of rows (samples) in the testing dataset.

```
Training data size: 227 rows
Testing data size: 57 rows
```

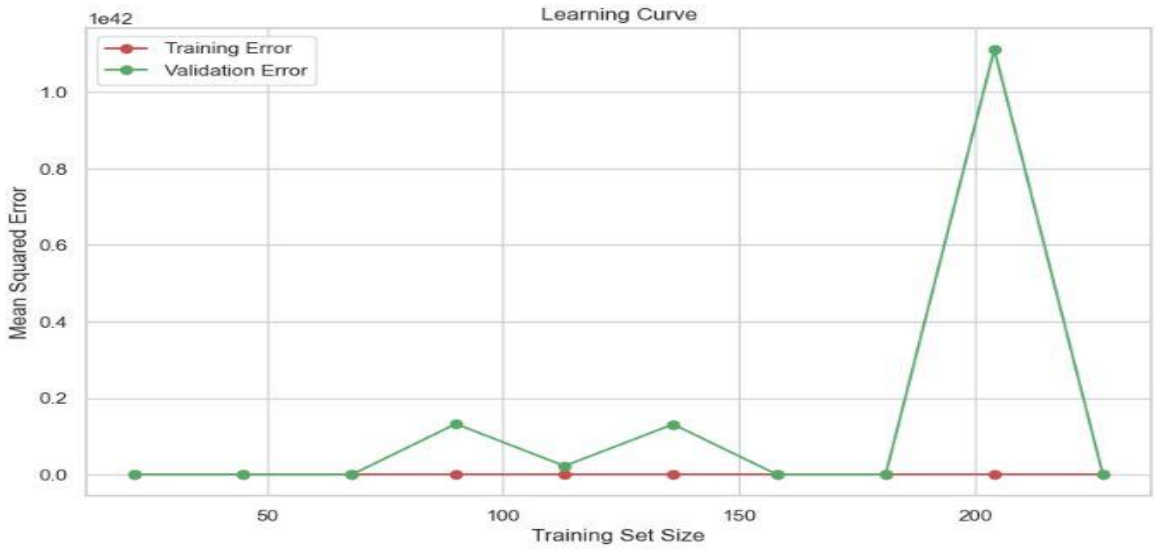
```
import matplotlib.pyplot as plt

# Plot actual vs predicted prices
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.6, color='b')
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw=2)  # 45-degree line for reference
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual vs. Predicted Prices")
    plt.show()
```




```
from sklearn.model_selection import learning_curve
```

```
import numpy as np

train_sizes, train_scores, test_scores = learning_curve(
    model, X, y, cv=5, scoring='neg_mean_squared_error', n_jobs=-1,
    train_sizes=np.linspace(0.1, 1.0, 10)
)

# Calculate average and std deviation of training and test scores
train_scores_mean = -train_scores.mean(axis=1)
test_scores_mean = -test_scores.mean(axis=1)

# Plot learning curve
plt.figure(figsize=(10, 6))
plt.plot(train_sizes, train_scores_mean, 'o-', color='r', label='Training Error')
plt.plot(train_sizes, test_scores_mean, 'o-', color='g', label='Validation Error')
plt.xlabel("Training Set Size")
plt.ylabel("Mean Squared Error")
plt.title("Learning Curve")
plt.legend(loc="best")
      plt.show()
```

dataset = pd.get_dummies(dataset, columns=['Nationality', 'Type', 'Team']) dataset = pd.get_dummies(dataset, columns=['Nationality', 'Type', 'Team'])



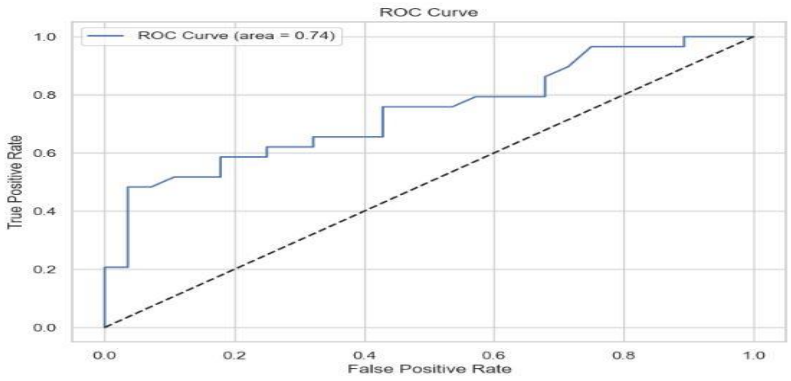
dataset = pd.get_dummies(dataset, columns=['Nationality', 'Type', 'Team'])

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, roc_auc_score, classification_report
from sklearn.preprocessing import StandardScaler

# Step 1: Create binary target variablemedian_price = dataset['Price'].median()
dataset['High_Price'] = (dataset['Price'] > median_price).astype(int)

# Step 2: Prepare feature matrix X and target vector y
X = dataset.drop(columns=['Price', 'High_Price', 'Name']) # Drop target columns from features
y = dataset['High_Price']

# Optional: Standardize features (can help logistic regression)
```

```
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Step 3: Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 4: Train the logistic regression model
logreg = LogisticRegression()
logreg.fit(X_train, y_train)

# Step 5: Make predictions
y_pred = logreg.predict(X_test)
y_pred_proba = logreg.predict_proba(X_test)[:, 1]  # Probabilities for ROC-AUC

# Step 6: Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred_proba)

print(f"Accuracy: {accuracy}")
print("Confusion Matrix:")
print(conf_matrix)
print(f"ROC-AUC Score: {roc_auc}")
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.6842105263157895
Confusion Matrix:
[[24  4]
 [14 15]]
ROC-AUC Score: 0.7395320197044335
Classification Report:
              precision    recall  f1-score   support

           0       0.63      0.86      0.73        28
           1       0.79      0.52      0.62        29

    accuracy                           0.68        57
   macro avg       0.71      0.69      0.68        57
weighted avg       0.71      0.68      0.68        57
```

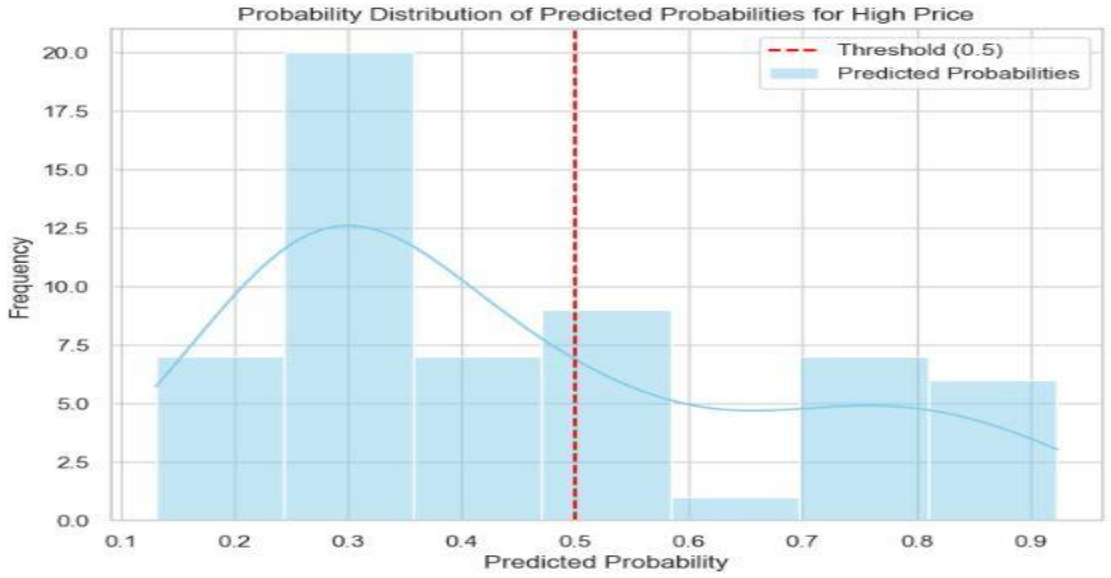
```
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve

fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f'ROC Curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.show()
```




```
import seaborn as sns
```

```
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

# Get the confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plot the confusion matrix using a heatmap
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Low Price', 'High Price'], yticklabels=['Low Price', 'High Price'])
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```


```
# Plot predicted probabilities for the positive class (High Price)
plt.figure(figsize=(8, 6))
sns.histplot(y_pred_proba, kde=True, color='skyblue', label='Predicted Probabilities')
plt.axvline(x=0.5, color='red', linestyle='--', label='Threshold (0.5)')
plt.title('Probability Distribution of Predicted Probabilities for High Price')
plt.xlabel('Predicted Probability')
plt.ylabel('Frequency')
plt.legend()
plt.show()
```



- **Conclusion**

In conclusion, the analysis of the sold players during the IPL 2023-24 auction provides valuable insights into the evolving dynamics of player valuation, team strategies, and the broader trends within the IPL ecosystem. The detailed examination of the auction data, we have been able to identify key patterns in player performance, pricing, and team composition, offering a clearer picture of how franchises are adapting to changing market conditions.

- # References

1. Datasets:

- IPL 2023-22 Players sold Dataset (used in our code):
  - Source: www.kaggle.com

2. Machine Learning Algorithms and Techniques:

- Random Forest Classifier:

  "Breiman, L. (2001). Random Forests." *Machine Learning*, 45(1), 5-32. https://doi.org/10.1023/A:1010933404324

- Label Encoding:

  Scikit-learn documentation on LabelEncoder: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html

- Feature Importance:

  "Scikit-learn documentation on Random Forest: Feature importance": https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

- StandardScaler:

  Scikit-learn documentation on StandardScaler: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

3. Visualization:

- Matplotlib Documentation:
  - "Matplotlib: Plotting with Python." https://matplotlib.org/stable/contents.html
- Visualization of Feature Importance:
  - "Visualizing Feature Importance in Random Forests." *Data Science Handbook* (online resource):

https://www.datacamp.com/community/tutorials/random-forests-classifier-python

4. General Resources:

- Scikit-learn Documentation:

  Official documentation for machine learning in Python:
  https://scikit-learn.org/stable/

- Pandas Documentation:

  Official pandas documentation for data manipulation:
  https://pandas.pydata.org/pandas-docs/stable/