NBA Player Performance Prediction

Name: Shefali Khera
Roll no: 22csu426
Affiliation: The NorthCap University

INTRODUCTION

Background:

- The NBA is rich with player statistics, and analyzing these stats can help predict future performance.
- Machine learning models can play a critical role in generating predictions based on past performance, helping teams optimize their strategies.

Problem Statement:

- How can we use historical data from the 2023-24 season to predict future player performance metrics, like points per game, rebounds?

Objective:

- Build a machine learning model to predict key performance indicators (KPIs) like points, assists, and rebounds for NBA players based on their 2023-24 season stats that help teams make informed decisions about player utilization, game strategy, and player development

Dataset Description

Dataset Source:

- The dataset was sourced from Kaggle and contains comprehensive NBA player statistics for the 2023-24 season, including playoff data.

Data Overview:

- The dataset includes 68 columns and 52218 rows.
- The dataset includes categorical data like PLAYER_NAME, TEAM_NAME, LOCATION, and Conference, and numerical data such as FGM, MIN_x, REB, PTS, AST, 3PA, 3PM, FTA, and FTM, which quantify player performance.
- The dataset consists of regular-season and playoff data, with no missing values or duplicate rows.

Key Features:

- Player Stats: Minutes Played, Field Goals Made, Rebounds, Assists, etc.
- Team Stats: Team name, opponent stats such as points in the paint, and overall team rank.

Preprocessing and Data Cleaning

Data Cleaning:

- No missing values were detected, simplifying preprocessing.
- No duplicate rows were found (dups. sum() = 0)

Outlier Removal:

- The Interquartile Range (IQR) calculated the bounds, capping values outside this range.

Feature Engineering:

- New features like PER (Player Efficiency Rating) were created from existing statistics.
- Text vectorization on the Comments column to convert textual data into numerical features for machine learning analysis.

Data Transformation:

- Categorical variables such as Conference were prepared for machine learning models using one-hot encoding.
- Log transformation to reduces skewness by converting values into their logarithmic form

DUPLICATE VALUES

```
[ ] dups = df.duplicated()
    print('Number of duplicate rows = %d' % (dups.sum()))

    df[dups]
```

Number of duplicate rows = 0

PLAYER_NAME	PLAYER_ID	TEAM_NAME_x	LOCATION	MIN_x	FGM	FGA	FG_PCT	FG3M	FG3A	...	DEF_RATING_RANK	DREB_RANK	DREB_PCT_RANK	STL_RANK	BLK_RANK	OPP_PTS_OFF_TOV_RANK

0 rows × 88 columns

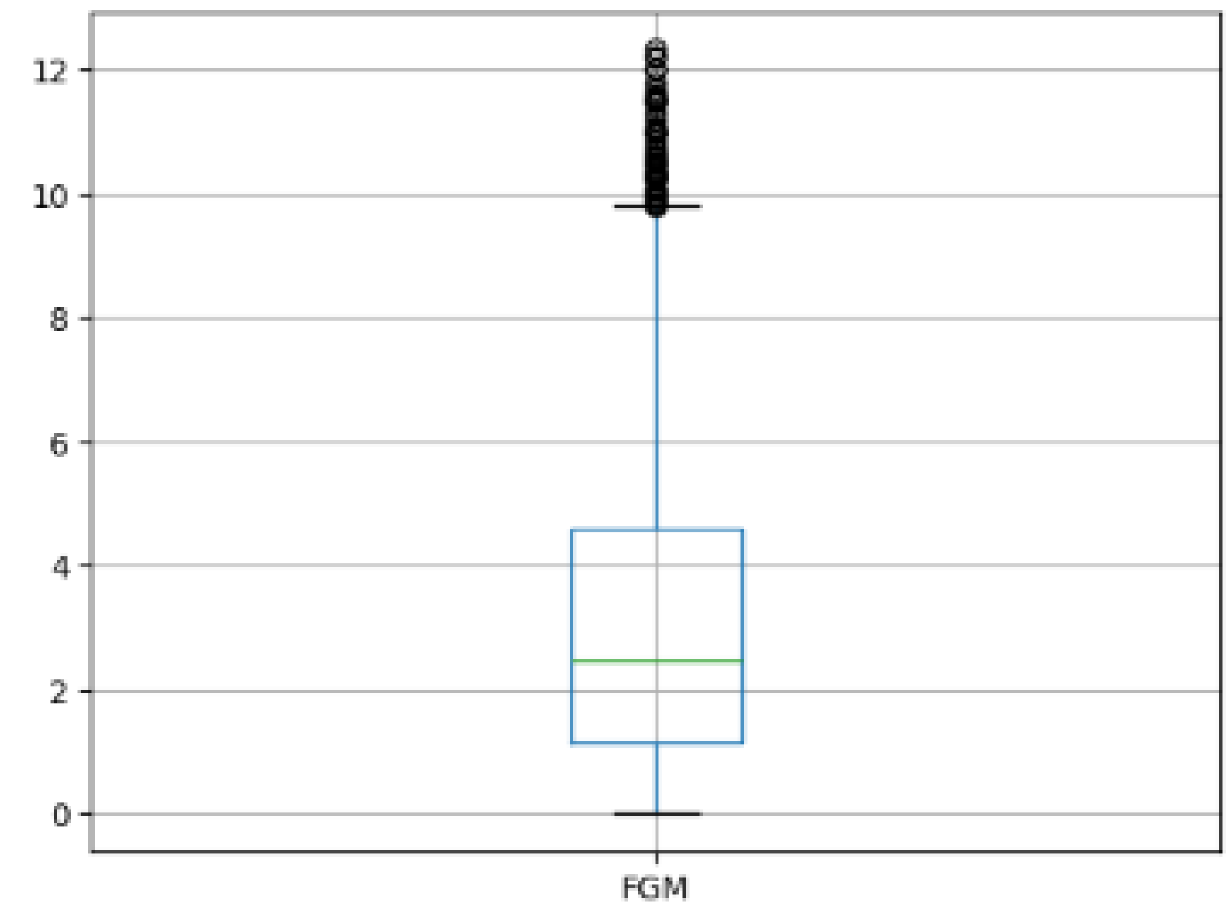
NULL VALUES

```
df.isnull().sum()
```

	0
PLAYER_NAME	0
PLAYER_ID	0
TEAM_NAME_x	0
LOCATION	0
MIN_x	0
...	...
OPP_PTS_OFF_TOV_RANK	0
OPP_PTS_2ND_CHANCE_RANK	0

OUTLIER DETECTION

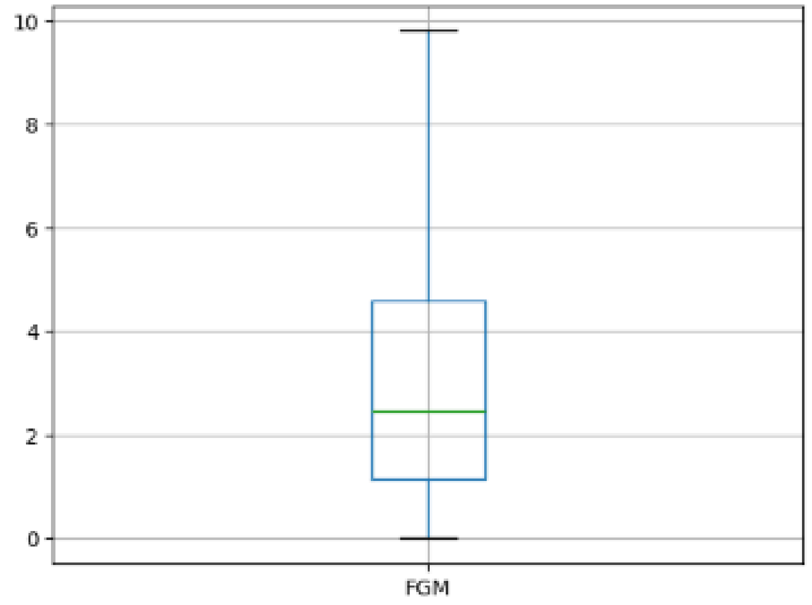
```
[ ] df.boxplot(column=['FGM'])
    plt.show()
```




```
[ ] def remove_outlier(col):
        sorted(col)
        Q1,Q3=col.quantile([0.25,0.75])
        IQR=Q3-Q1
        lower_range= Q1-(1.5 * IQR)
        upper_range= Q3+(1.5 * IQR)
        return lower_range, upper_range
```

```
[ ] lrincome,urincome=remove_outlier(df['FGM'])
    df['FGM']=np.where(df['FGM']>urincome,urincome,df['FGM'])
    df['FGM']=np.where(df['FGM']<lrincome,lrincome,df['FGM'])
```

```
df.boxplot(column=['FGM'])
plt.show()
```



ONE-HOT ENCODING

```python
from sklearn.preprocessing import OneHotEncoder
encoder = OneHotEncoder(handle_unknown='ignore', sparse_output=False)
X_encoded = encoder.fit_transform(df[['Conference']])

print("Shape of X_encoded:", X_encoded.shape)
print("First few rows of X_encoded:\n", X_encoded[:5])
```

```
Shape of X_encoded: (2418, 2)
First few rows of X_encoded:
 [[1. 0.]
 [1. 0.]
 [1. 0.]
 [1. 0.]
 [0. 1.]]
```

LOG TRANSFORMATION

```python
import numpy as np
df['log_PTS'] = np.log(df['PTS'] + 1)

print(df[['PTS', 'log_PTS']])
```

```
        PTS   log_PTS
0      4.11  1.631199
1      4.86  1.768150
2      2.75  1.321756
3      2.17  1.153732
4      3.63  1.532557
...     ...       ...
2413  29.83  3.428488
2414   7.57  2.148268
2415   1.17  0.774727
2416   1.50  0.916291
2417   1.67  0.982078

[2418 rows x 2 columns]
```

- **OneHotEncoder** transforms categorical data (like 'Conference') into numerical format using binary vectors
- **Log transformation** `np.log(df['PTS'] + 1)` calculates the natural logarithm of the 'PTS' column, adding 1 to avoid log(0).

Exploratory Data Analysis (EDA)

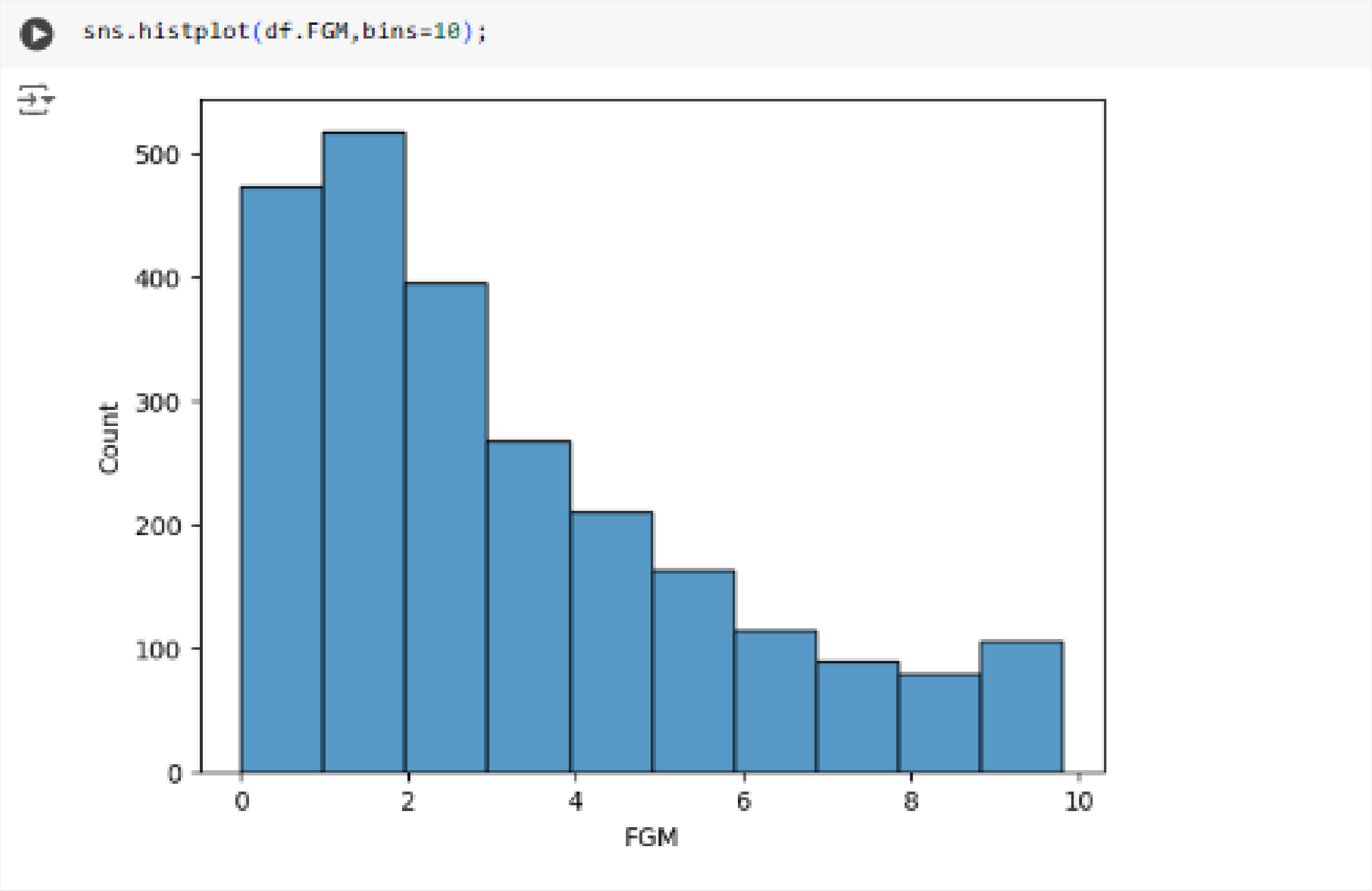
Visualizations

- A correlation matrix was generated using numeric columns.
- A heatmap visually represented the strength of these correlations.
- A count plot showed the distribution of players across different conferences.
- Radar charts to compare player statistics across different dimensions
- Create word clouds to visualize the most common words used like player comments.

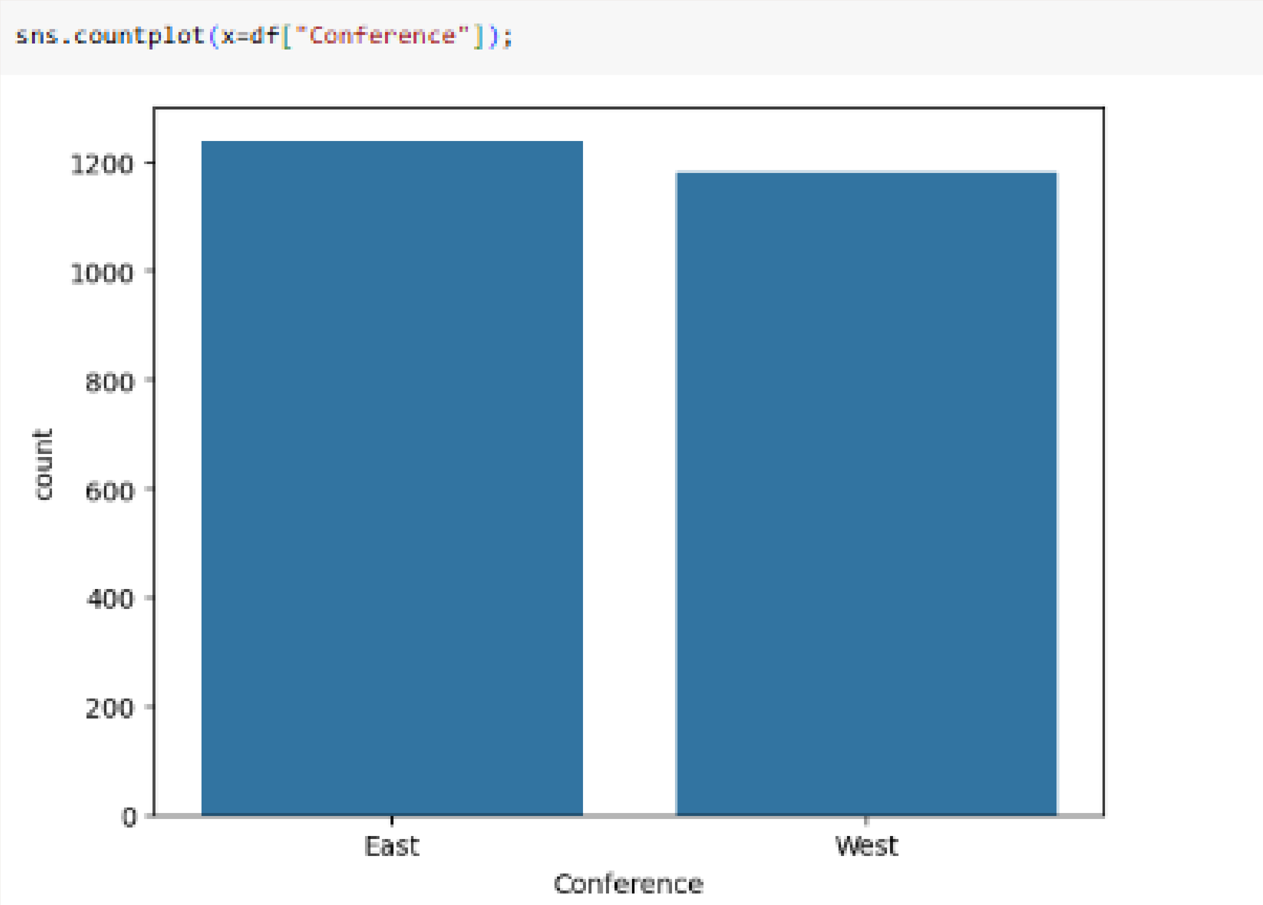
Key Insights

- The correlation heatmap revealed a strong positive correlation between assists (AST) and points (PTS). Players who assist more tend to score more points.
- A significant correlation was found between rebounds (REB) and minutes played (MIN_x), indicating that players on the court longer secure more rebounds.
- These insights can guide strategic decisions in team management, player development, and understanding player dynamics.

COUNT PLOT

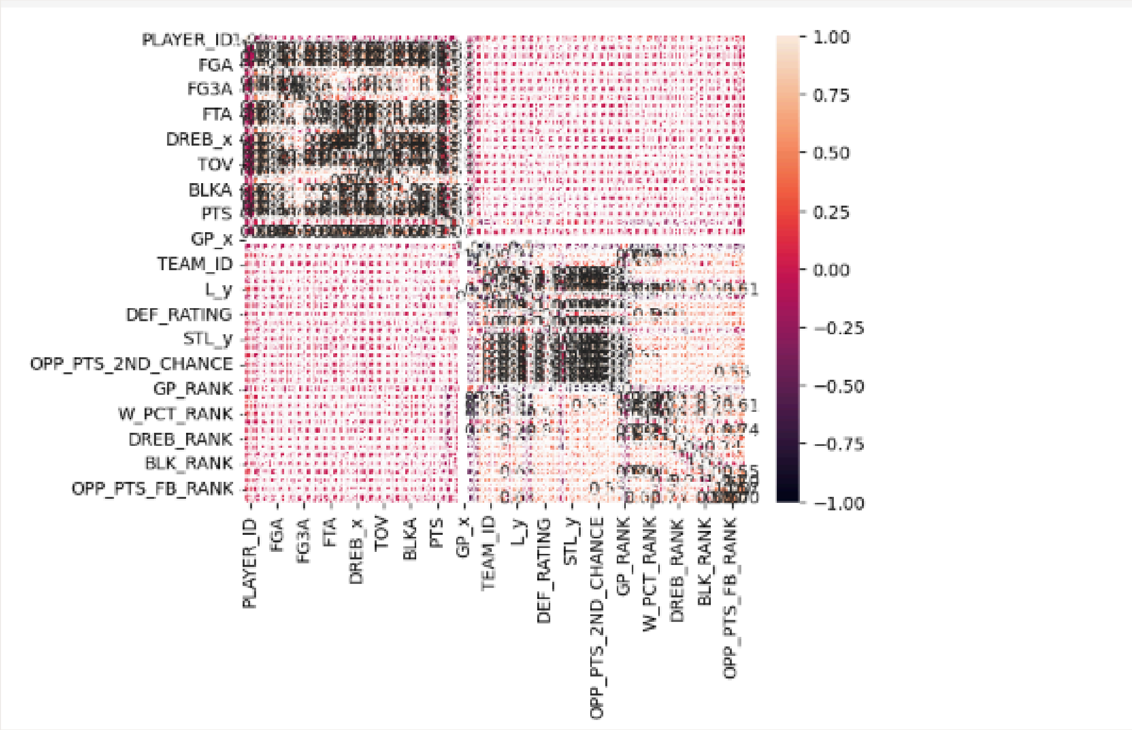


HIST PLOT

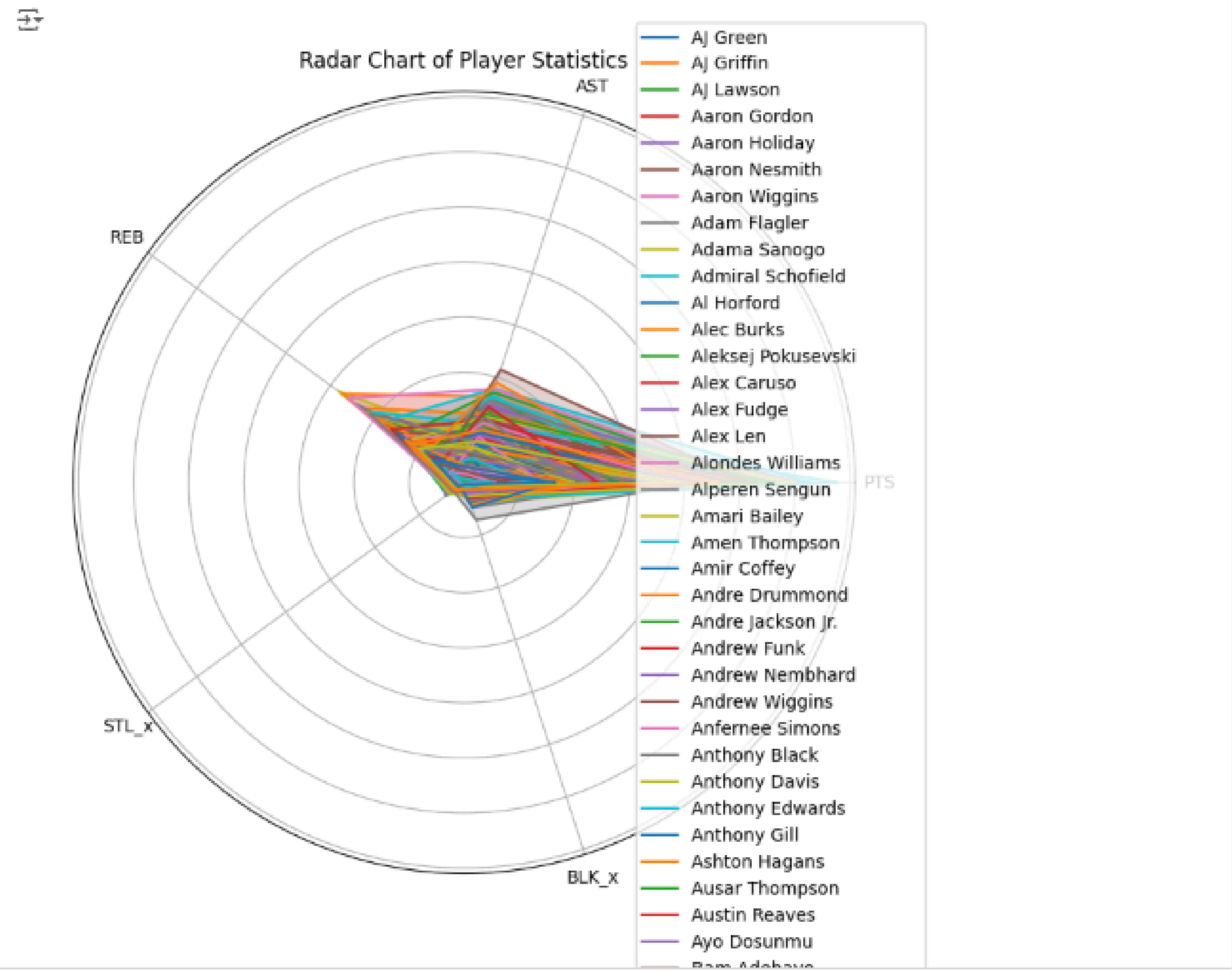


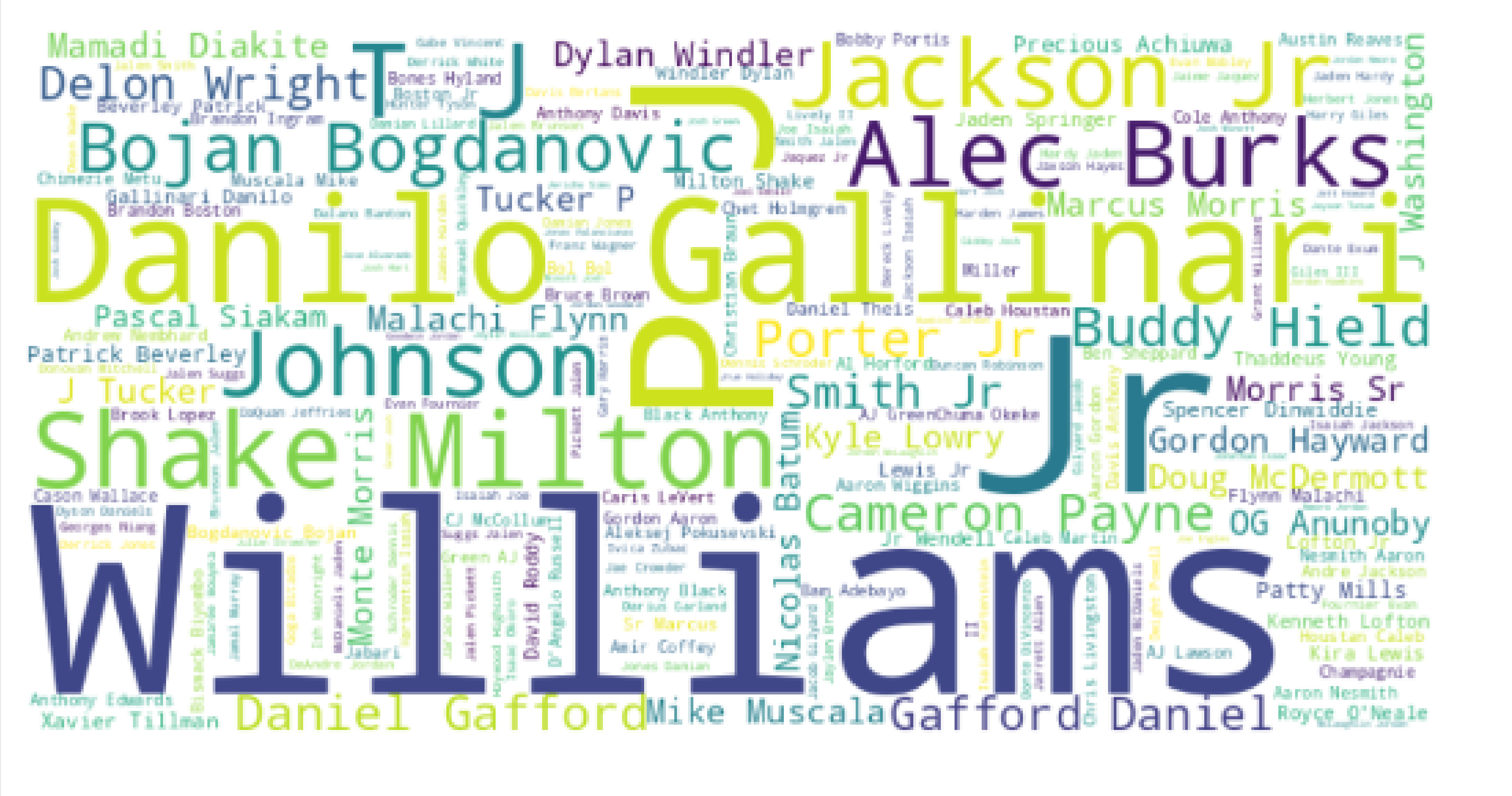
- The **correlation matrix** shows the relationships between numerical variables, while the **count plot** displays the frequency of each conference and the **histogram** shows the distribution of field goal made.

CORRELATION MATRIX



RADAR CHART

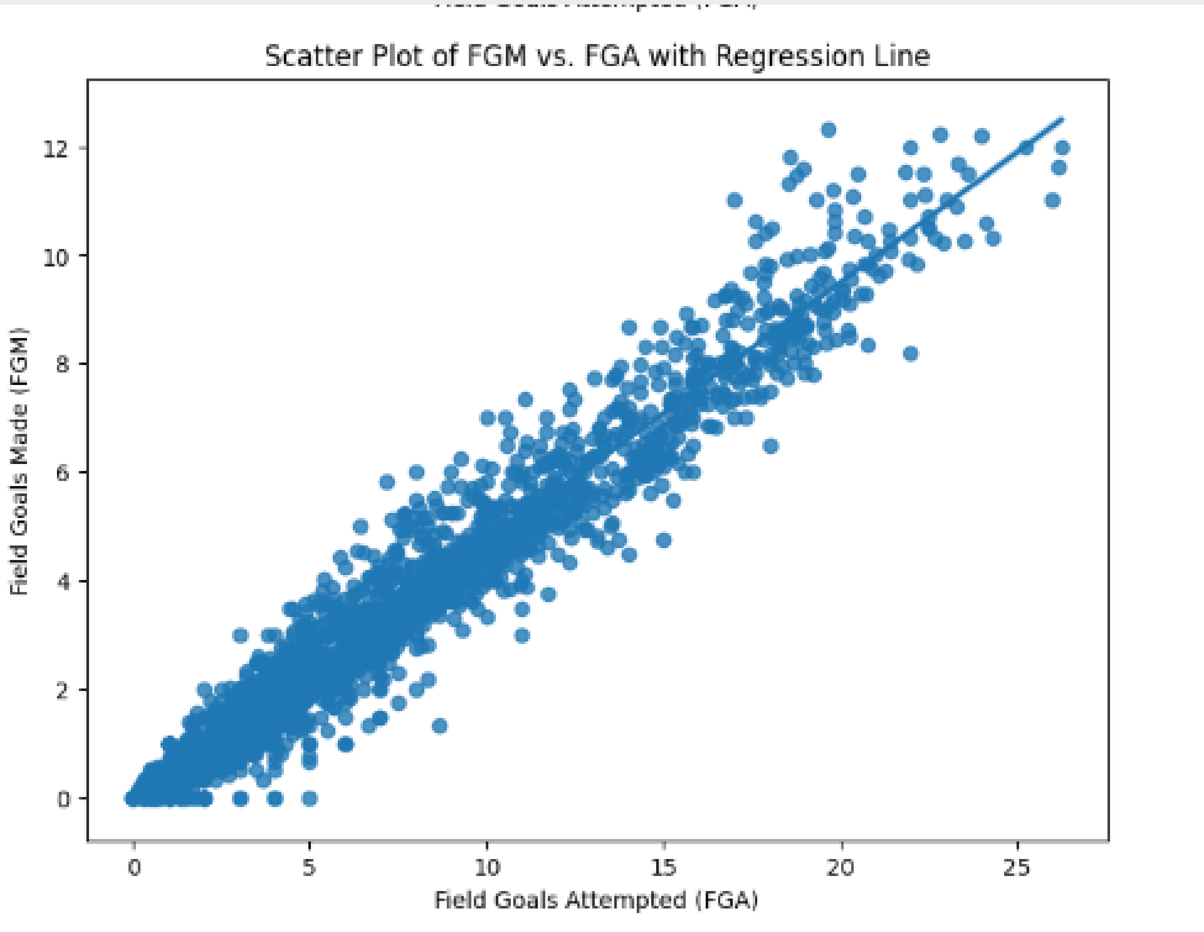




- The **radar chart** visualizes the average statistics (PTS, AST, REB, STL, BLK) of several players. Each player's stats are represented by a polygon, allowing for a comparison of their performance across various metrics. It helps in understanding the strengths and weaknesses of individual players relative to others.

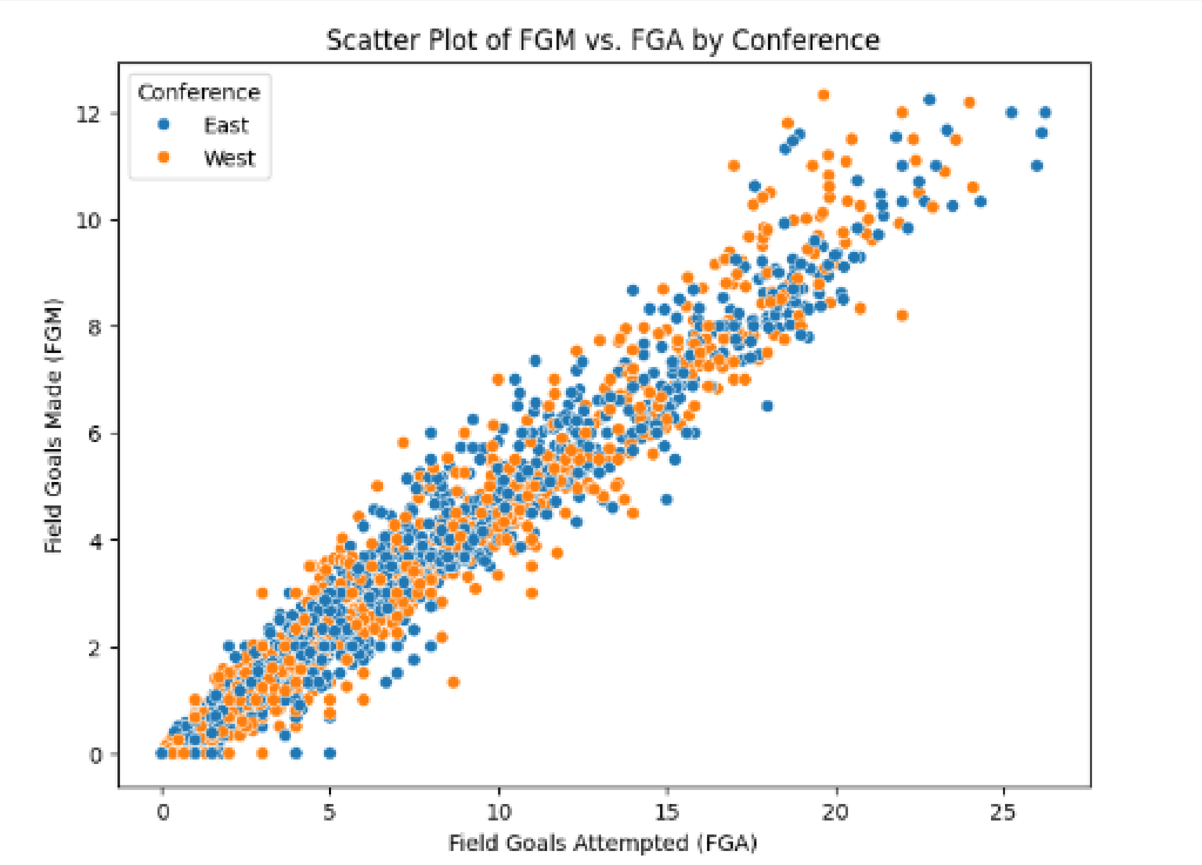
SCATTER PLOT

SCATTER PLOT

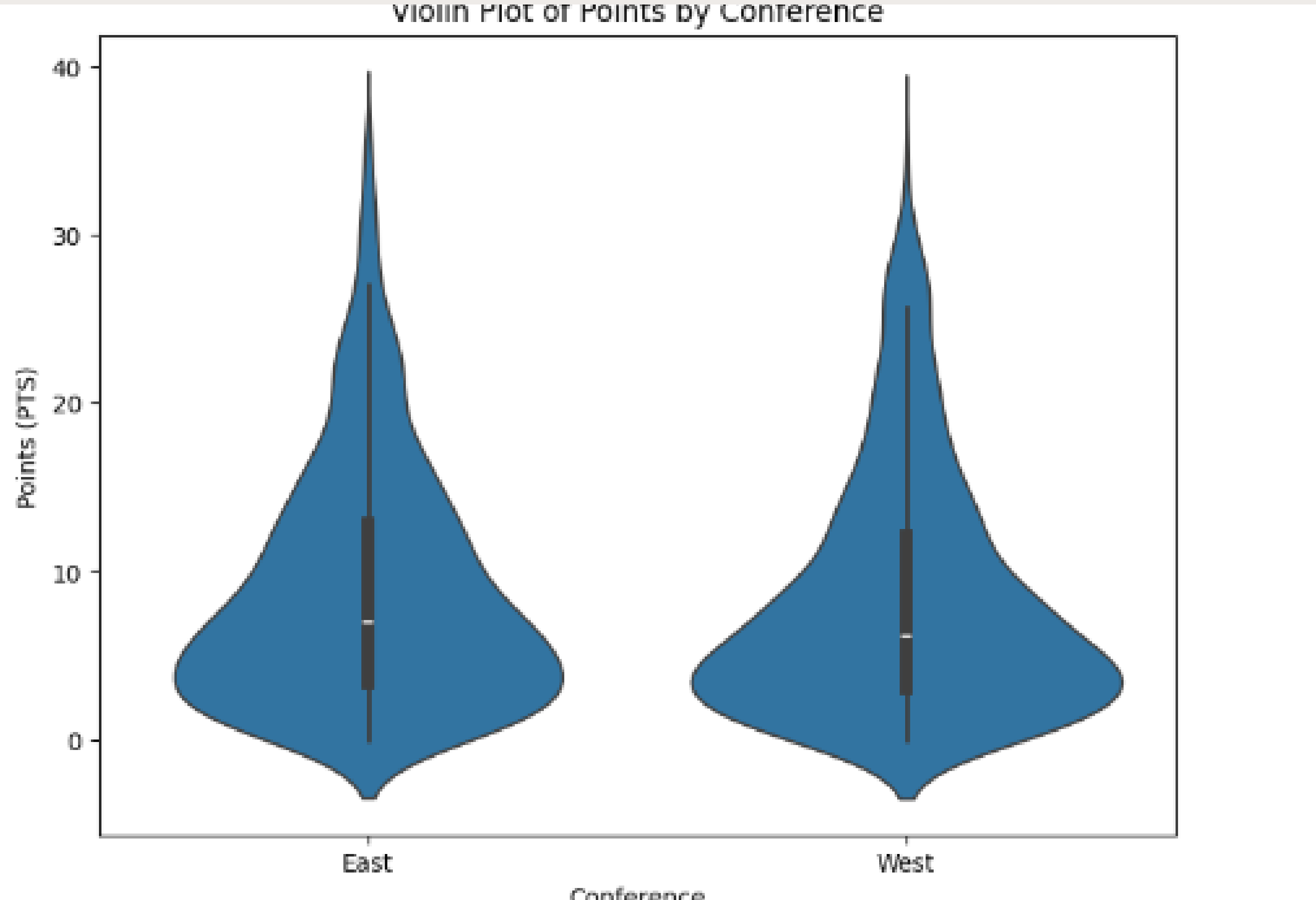


- **Scatter Plot:** Shows the relationship between two continuous variables (e.g., FGA and FGM).
- **Box Plot:** Displays the distribution of a continuous variable (e.g., PTS) across different categories (e.g., conferences). It provides information on the median, quartiles, and outliers.
- **Violin Plot**: Similar to box plot, but also shows the probability density of the data at different values, providing a more complete view of the distribution.

SCATTER PLOT



VIOLIN PLOT



Machine Learning Models

Regression Models

- Linear Regression:

Application: Predict continuous outcomes such as points scored (PTS) based on features like assists (AST), rebounds (REB), and field goals made (FGM).

- XGBoost (for Regression):

Application: Predict continuous performance metrics like Player Efficiency Rating (PER) using player statistics.

Classification Models

- Random Forest:

Application: Classify players based on performance categories, such as whether a player will score above a certain threshold in a game.

- XGBoost (for Classification):

Application: Predict categorical outcomes, such as whether a player will be an All-Star based on their performance metrics.

- Neural Networks :

Application: Classify player performance into categories like "high scorer" or "role player" based on their statistics.

PREDICTION FOR CLASSIFICATION

PREDICTION

```
[ ] sample_data = [[30, 8, 2, 5, 3, 6]]

[ ] sample_df = pd.DataFrame(sample_data, columns=['MIN_x', 'FGM', 'FG3M', 'FTM', 'OREB', 'DREB_x'])

[ ] predicted_conference = nb_model.predict(sample_df)
    print("Predicted Conference:", predicted_conference[0])

    Predicted Conference: East
```

- This code uses a classification model with features (`MIN_x`, `FGN`, `FG3M`, `FTM`, `OREB`, `DREB_x`) to predict the **conference** (East or West) and prints the predicted result.

PREDICTION FOR REGRESSION

PREDICTION

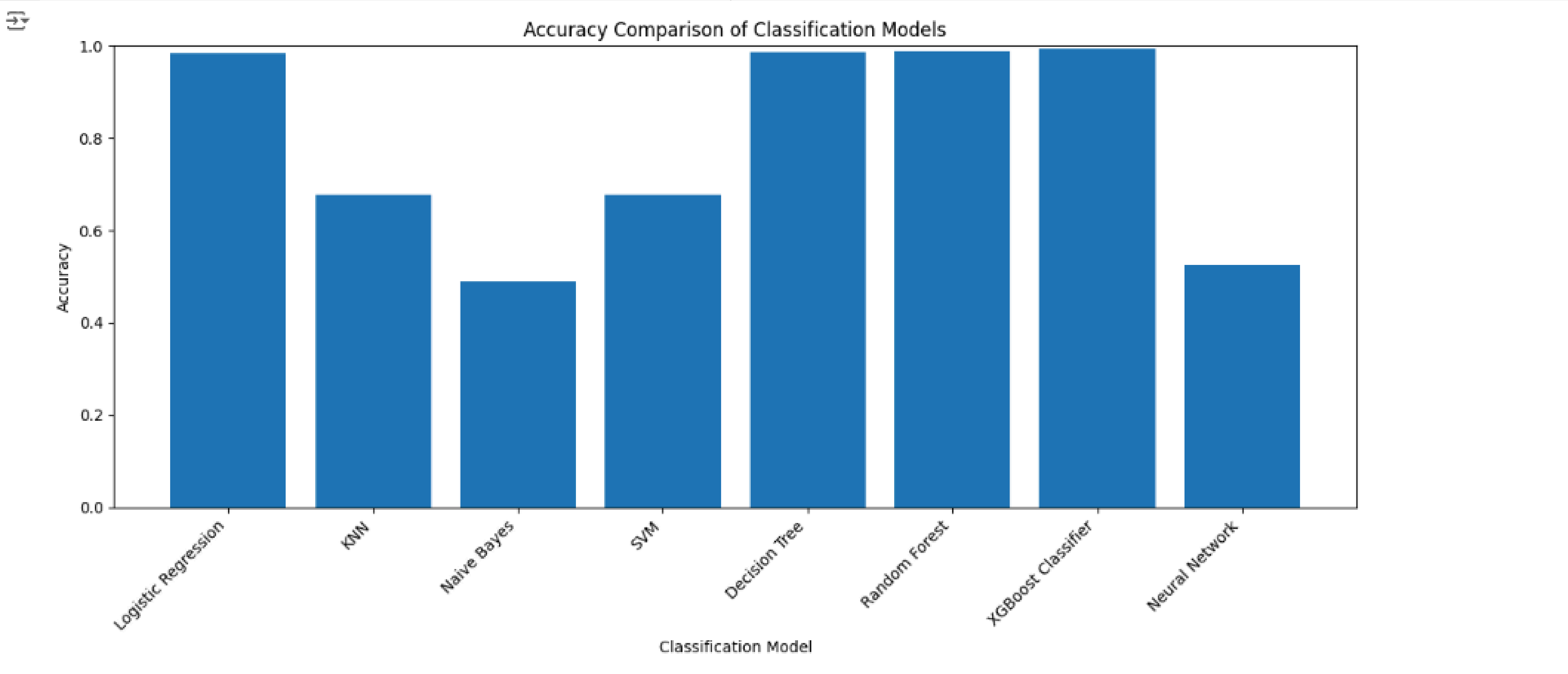
```
[76] sample_data = pd.DataFrame({
         'MIN_x': [200],
         'FGM': [0.5],
         'FG3M': [2],
         'FTM': [5],
         'OREB': [8],
         'DREB_x': [20]
     })
     predicted_pts = regressor.predict(sample_data)

     print("Predicted PTS:", predicted_pts[0])

     Predicted PTS: 8.000669102757023
```

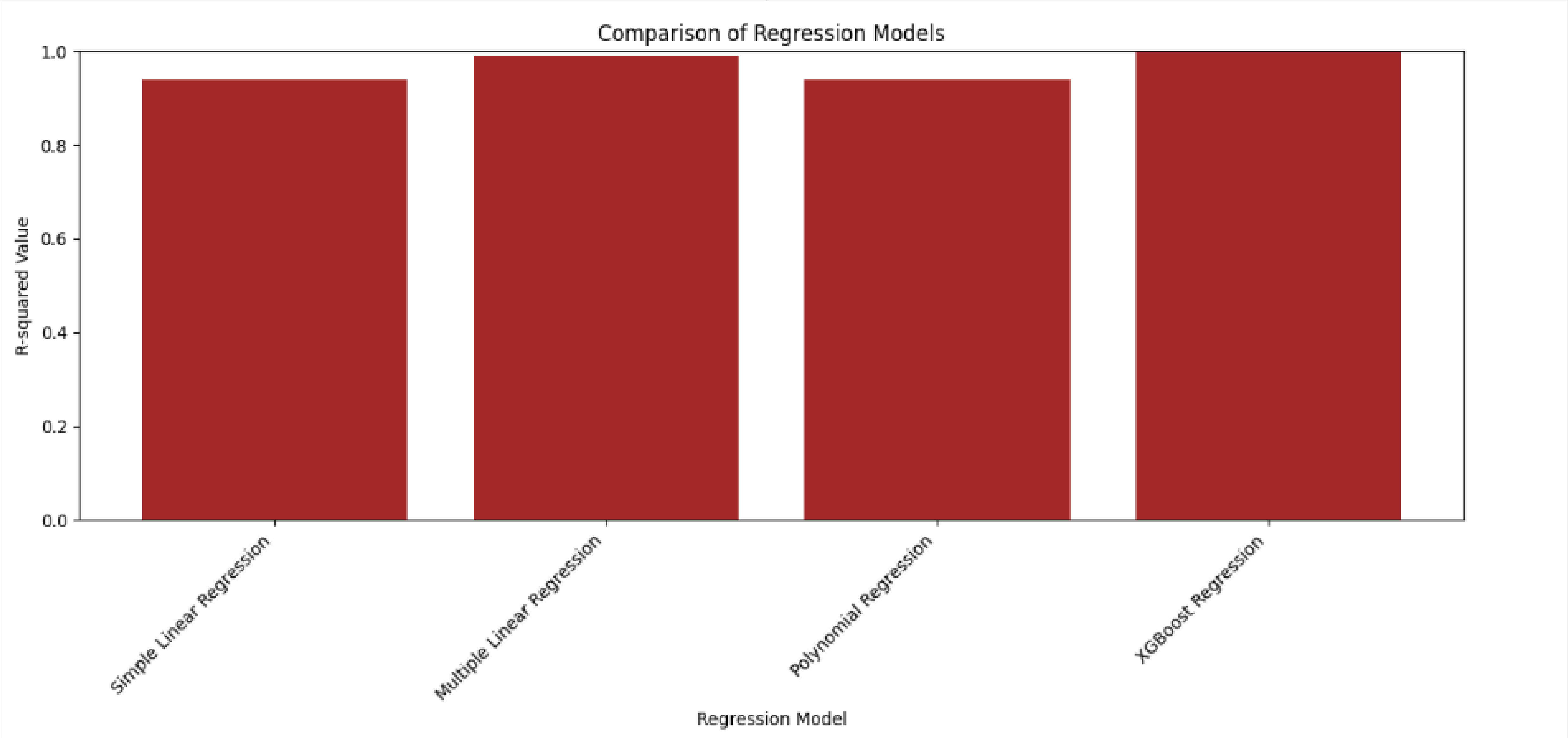
- This code creates sample data with features (`MIN_x`, `FGN`, `FG3M`, `FTM`, `OREB`, `DREB_x`) and uses a trained model (`regressor`) to predict **points** (`PTS`), then prints the predicted result.

Classification Model Analysis



- **XGBoost Classifier** and **Random Forest** have the highest accuracy, indicating that they are the best-performing models for predicting the 'Conference' variable. Neural Network and Naive Bayes show lower accuracy, suggesting they may not be the most suitable choices for this task.

Regression Model Analysis



- **XGBoost Regression** demonstrates the highest R-squared value. This implies that it explains the variance in the 'PTS' variable (the target for regression) the best. Multiple Linear Regression also shows a high R-squared, meaning that it fits the data well.

Feature Selection

Feature selection using `**SelectKBest**` with `**f_classif**` (ANOVA F-value) offers several benefits:

1. **Improved Model Performance**: Selecting the most relevant features reduces noise and irrelevant information that can negatively impact model performance. This often leads to higher accuracy, precision, recall, and other evaluation metrics.

2. **Reduced Overfitting**: Using fewer features can help prevent overfitting, which occurs when a model performs well on training data but poorly on unseen data. By focusing on the most informative features, we create a simpler and more generalizable model.

3. **Faster Training:** With fewer features, the model training process is faster and requires fewer computational resources.

4. **Enhanced Interpretability**: A model with a smaller set of features can be easier to interpret and understand. You can gain insights into the underlying relationships between the selected features and the target variable.

- We are using feature selection to identify the **top 10** features that are most relevant to predicting the `Conference` variable.
- The **`f_classif`** function is used for feature selection in classification problems.
- **`SelectKBest`** selects the top k features based on their statistical significance.

```
XGBoost Classifier Accuracy (Without Feature Selection): 0.993801652892562
/usr/local/lib/python3.10/dist-packages/sklearn/feature_selection/_univariate_selection.py:112: UserWarning: Features [23]
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
/usr/local/lib/python3.10/dist-packages/sklearn/feature_selection/_univariate_selection.py:113: RuntimeWarning: invalid va
  f = msb / msw
XGBoost Classifier Accuracy (With Feature Selection): 1.0
Difference in Accuracy: 0.006198347107438051
```

- The XGBoost classifier in the code achieves an accuracy of approximately 0.9938. This signifies a very high level of accuracy, indicating that the model is performing exceptionally well in predicting the 'Conference' variable. The model's performance is further enhanced by feature selection, resulting in potentially a slightly higher accuracy.

THANK YOU