Data Science Project Protocol – NBA Playoff games predictions

Author(s):
Itay Avioz

Table of Contents

Introduction

The NBA playoffs represent the pinnacle of competition in professional basketball, where the intensity and stakes are significantly higher than in the regular season. Unlike regular-season games, playoff games are characterized by heightened pressure, maximum effort from teams, and a strategic emphasis on each game due to the elimination format. The atmosphere in playoff games is electric, with sold-out arenas and a palpable home-court advantage that can significantly impact the outcome.

Since 2003, the series format has been best-of-seven games, providing a robust dataset for analysis. The primary goal of this project is to predict the outcome of NBA playoff games, focusing on who will win each game. This involves analyzing a vast amount of historical data and statistics that the NBA provides, leveraging this data to create a predictive model.

Key questions the project aims to answer include:

- Which factors most significantly influence the outcome of NBA playoff games?
- Can we identify patterns or trends from historical playoff data that predict future game outcomes?

Known factors influencing the outcome of NBA playoff games include team performance metrics, player statistics, home-court advantage, and historical performance in playoff contexts. Additionally, advancements in data science and machine learning now enable more sophisticated analysis and more accurate predictions than ever before.

Moreover, the law of large numbers versus the law of small numbers plays a crucial role in this analysis. In basketball, characterized by frequent scoring opportunities and a larger sample size of events (such as positions and shots), the theoretical probability tends to align closely with observed outcomes over time. In contrast, sports like football, with fewer scoring opportunities and lower scores per game, are more susceptible to "surprise" outcomes.

Previous research has explored various aspects of game prediction, but this project aims to build on that foundation by applying cutting-edge techniques to a comprehensive dataset. The potential applications of accurate game predictions are vast, including enhancing fan engagement, informing team strategies, and aiding sports betting.

By leveraging extensive NBA data and sophisticated modeling techniques, the project seeks to provide accurate predictions of game outcomes, offering new perspectives and enhancing the understanding of what drives success in the NBA playoffs.

Methodology (Project Design)

Data

1. **Data Sources**:
    - **Primary Data Sources**:

    The primary data sources provide a comprehensive dataset comprising 3256 rows, encapsulating data from 1763 individual playoff games. Each game is examined from the perspectives of both teams involved, denoted as Team X and Team Y. This dataset encompasses a rich array of features, ~ 200 variables, ranging from basic game statistics to advanced performance metrics.

        - NBA official statistics from the NBA Stats API provide comprehensive and up-to-date player and team statistics, game logs, and advanced metrics. Examples of primary data include points scored, blocks, steals, win/loss records, home/away game information, free throws, matchup history between the teams and more.
        - Basketball Reference is another key source, offering extensive historical data, including team stats, game results, offensive and defensive statistics, regular season summaries, rankings, player ages, crowd averages, starters' stats, bench stats, and more.
    - **External Data Sources**:
        - Player injuries and availability from sources like ESPN or the NBA injury report.
        - Player-specific statistics, such as player efficiency ratings (PER), win shares, and injury history.
    - **Data for External Validation**:
        - 20% of the data will be set aside for external validation.
        - Data from the 2023-2024 NBA playoff games will be used to validate the model's performance on new and unseen data.
2. **Time Frames**:
    - **Training Time-Frame**:
        - 60% of the data, spanning all available years from the 2002-2003 to the 2022-2023 season, used for training the model.
    - **Testing Time-Frame**:
        - 20% of the data, covering the same period (2002-2003 to 2022-2023), used for testing the model's performance.
    - **Model Evaluation**:
        - An additional 20% of the data from the same time frame (2002-2003 to 2022-2023) will be reserved for model evaluation.

Decision Not to Use a Time Series Model:

The deliberate choice to avoid employing a time series model was rooted in the intrinsic characteristics of the data. With the data aggregated on a yearly basis, encompassing both individual game data and seasonal summaries, this aggregation effectively "hides" the intricate temporal details within each season, the adoption of a time series approach appeared restrictive. , blurring the continuity between seasons and franchise history, thereby potentially concealing valuable insights crucial for enhancing predictive accuracy.

Moreover, solely relying on annual information poses the risk of masking the latest temporal trends, such as shifts in scoring patterns or changes in strategic approaches like increased three-point attempts. Capturing these recent developments is pivotal for understanding current team dynamics and performance trends. Neglecting this aspect could undermine the model's ability to detect subtle patterns and fluctuations, thereby diminishing its predictive capabilities.

By opting out of a model centered on time seasons, the project endeavors to embrace a broader spectrum of modeling techniques. This approach facilitates the creation of novel features that not only connect game-to-game performance within a season but also establish links across seasons and franchises. By enriching the analysis in this manner, the project aims to enhance predictive capabilities while promoting a more comprehensive understanding of NBA playoff game outcomes. Additionally, circumventing a rigid time-based framework helps mitigate biases inherent in such models, fostering a nuanced and holistic perspective on playoff dynamics.

3. **Subjects**:
    - **Inclusion Criteria**:
        - All teams and games participating in NBA playoff games from the 2002-2003 (best-of-seven series format) season onward.
        - All teams and games in regular season summary from the 2002-2003 (best-of-seven series format) season onward.
        - Franchise history.
    - **Exclusion Criteria**:
        - Playiers data – personal player data.
        - Injuers data.
4. **Outcome Variable**:
    - The primary outcome variable is win/loss (1/0) result of each NBA playoff game – classification model.
5. **Confounder Variables**:
    - The complexity of playoff game data necessitates careful consideration of confounding variables, which encompass various factors influencing game outcomes. These variables span multiple dimensions, including the stage of the playoffs (reflecting different difficulty levels), the duration of rest days between series, the separation between seasons, and the distinction between home and away games. Moreover, meaningful

analysis requires comparison against regular season data, as well as juxtaposition of Team X's offensive metrics with those of Team Y.

Effectively delineating these confounder variables is essential for contextualizing playoff game data and unraveling causal relationships. Without such comparisons, the significance of individual game statistics may be obscured, impeding the ability to draw accurate conclusions regarding game outcomes..

6. **Bias**:

Potential sources of bias include selection bias (e.g., focusing only on more recent games with more complete data) and survivorship bias (e.g., teams that perform better are more frequently included). To address these biases, we will ensure a representative sample across seasons and teams, and apply techniques like stratified sampling.

To further avoid bias in the information, the following measures were taken:

- **Consistent Series Format**: Data was collected starting from the 2002-2003 season, when the playoff series format changed to best-of-seven games, ensuring consistency in the format being analyzed.
- **Home Game Variable**: A home game variable was created and updated according to the division structures that existed until 2012-2013 and from 2013-2014 onwards. This adjustment ensures that the home-court advantage is accurately represented across different eras.
- **Uniform Data Availability**: Information was sourced uniformly across all teams and years included in the study, ensuring comprehensive coverage and avoiding the exclusion of any team or season.
- **Special Year Feature**: Years where the season was not played in full, such as the COVID-19 season with playoffs on neutral courts and seasons affected by player strikes, were included. A feature was created to describe these "special years" to account for the unique conditions affecting those seasons.
- **Conference and Ranking Variables**: Data includes variables for East/West conferences and team rankings, encompassing both overall and subdivisional rankings. This helps account for the different competitive contexts teams operate in.
- **Regular Season Momentum**: Regular season data post-All-Star break was incorporated to identify team momentum and performance trends, particularly for lower-ranked teams making a late-season push. This helps in understanding how late-season performance might influence playoff outcomes.
- **Additional Features for Data Balance**: More features were created to balance the data and present a comprehensive picture, ensuring that various aspects of team performance and context are adequately captured and represented.

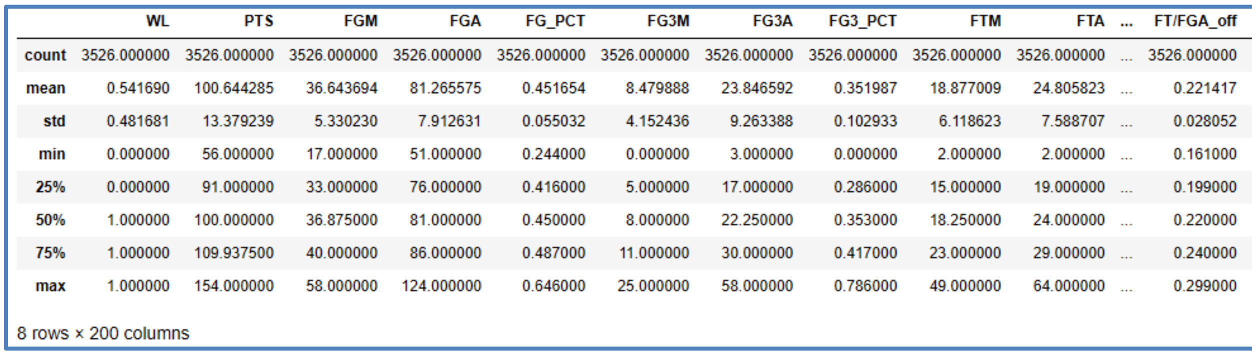
By implementing these strategies, the project aims to mitigate biases, ensuring a fair and comprehensive analysis of NBA playoff game outcomes. These measures help to contextualize the data accurately, facilitating the development of robust predictive models.

7. **Data Exploration**:
    - Initial data analysis will include summary statistics, visualizations of key variables (e.g., distributions of points scored, win/loss rates), and correlation analysis to identify significant predictors.
    - Initial data exploration involved a comprehensive analysis of the dataset to understand its structure, identify key patterns, and prepare it for modeling. The steps included:
    - Division into Groups:

      Divided variables into continuous (normal/abnormal) and categorical (Boolean, ordinal, nominal) groups.

    - Statistical Analysis of Continuous Variables:

      Analyzed center indices (mean, median) and dispersion indices (standard deviation, interquartile range).

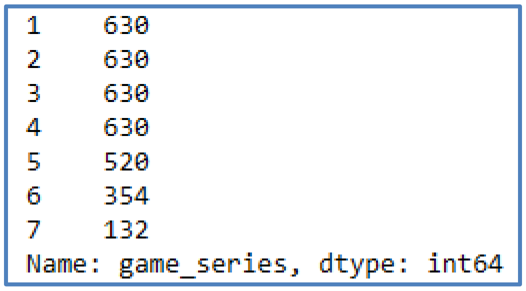


	WL	PTS	FGM	FGA	FG_PCT	FG3M	FG3A	FG3_PCT	FTM	FTA	...	FT/FGA_off
count	3526.000000	3526.000000	3526.000000	3526.000000	3526.000000	3526.000000	3526.000000	3526.000000	3526.000000	3526.000000	...	3526.000000
mean	0.541690	100.644285	36.643694	81.265575	0.451654	8.479888	23.846592	0.351987	18.877009	24.805823	...	0.221417
std	0.481681	13.379239	5.330230	7.912631	0.055032	4.152436	9.263388	0.102933	6.118623	7.588707	...	0.028052
min	0.000000	56.000000	17.000000	51.000000	0.244000	0.000000	3.000000	0.000000	2.000000	2.000000	...	0.161000
25%	0.000000	91.000000	33.000000	76.000000	0.416000	5.000000	17.000000	0.286000	15.000000	19.000000	...	0.199000
50%	1.000000	100.000000	36.875000	81.000000	0.450000	8.000000	22.250000	0.353000	18.250000	24.000000	...	0.220000
75%	1.000000	109.937500	40.000000	86.000000	0.487000	11.000000	30.000000	0.417000	23.000000	29.000000	...	0.240000
max	1.000000	154.000000	58.000000	124.000000	0.646000	25.000000	58.000000	0.786000	49.000000	64.000000	...	0.299000

8 rows × 200 columns

    - Statistical Analysis of Categorical Variables:

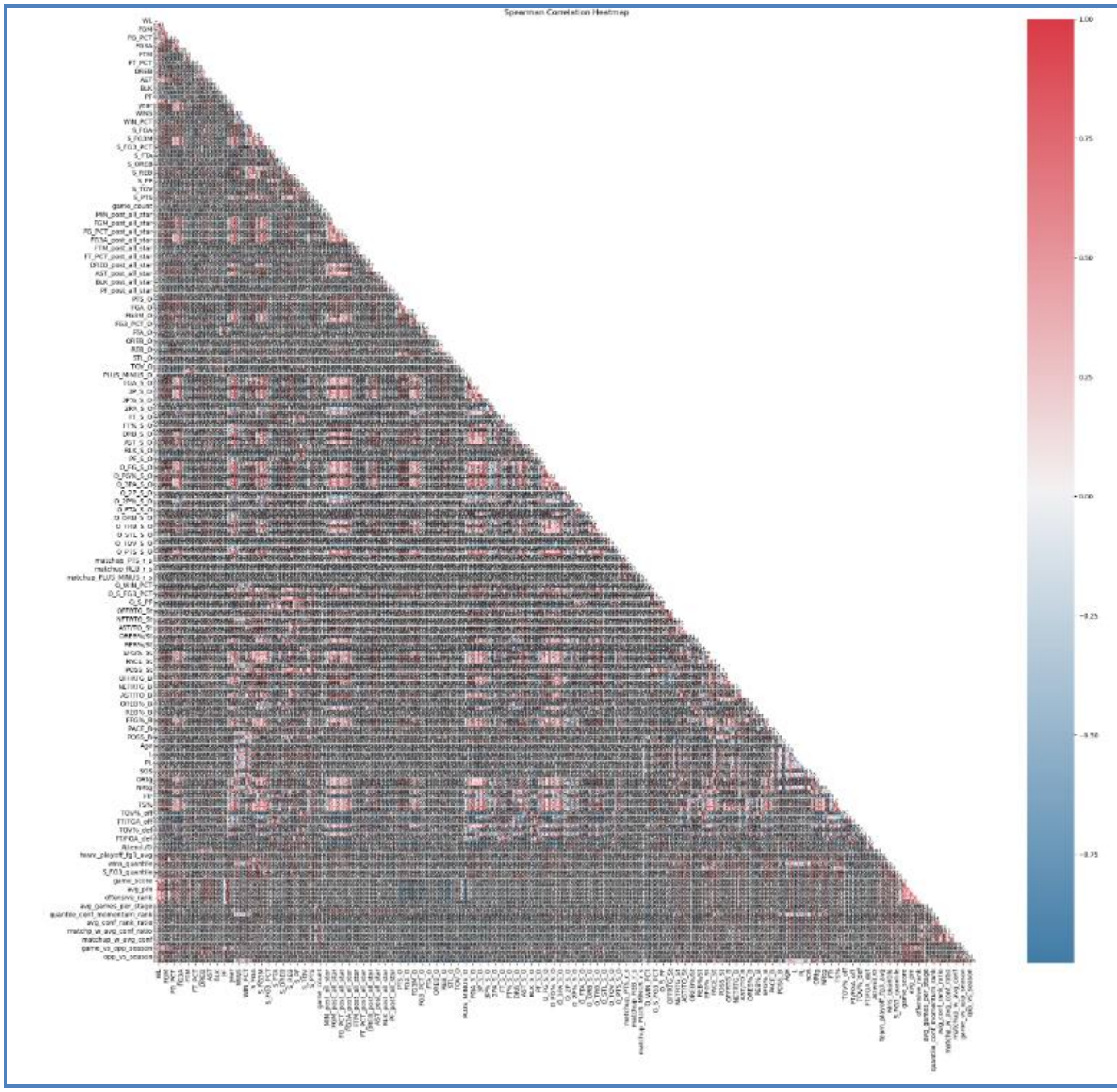
      Assessed the values of each categorical variable and the frequency of each value.



```
1    630
2    630
3    630
4    630
5    520
6    354
7    132
Name: game_series, dtype: int64
```

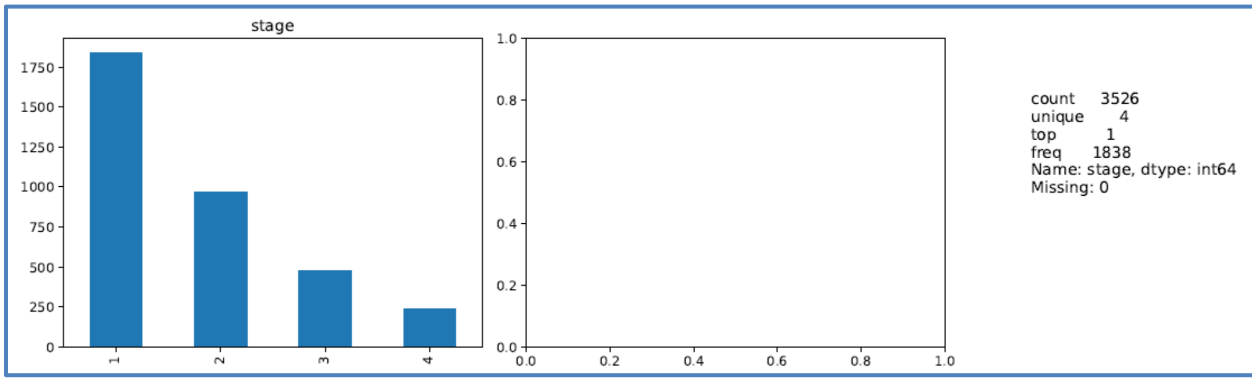
- <u>Correlation Analysis</u>:

  Identified correlations between all variables to uncover significant relationships.

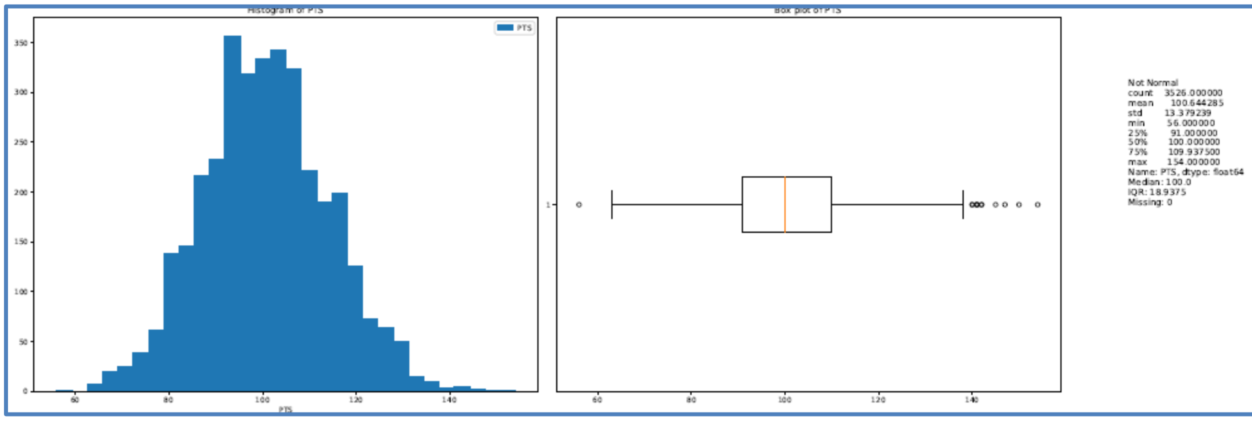


- Graphical Analysis:

  Created histograms, boxplots, and barplots to understand the distribution of data and identify missing and outliers.



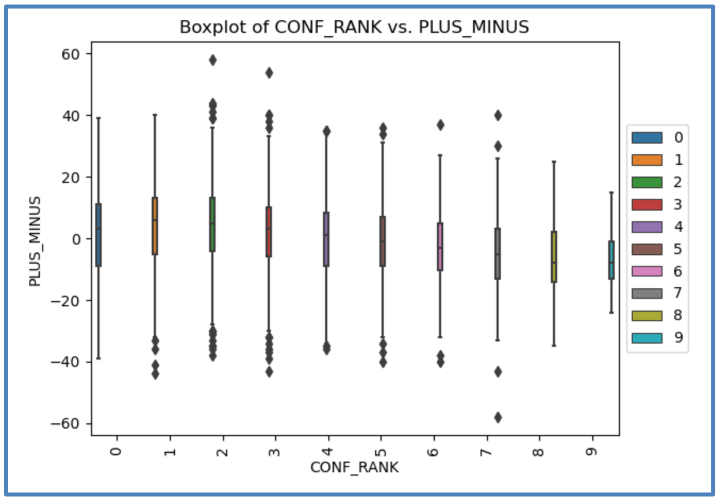
```
count     3526
unique       4
top          1
freq      1838
Name: stage, dtype: int64
Missing: 0
```

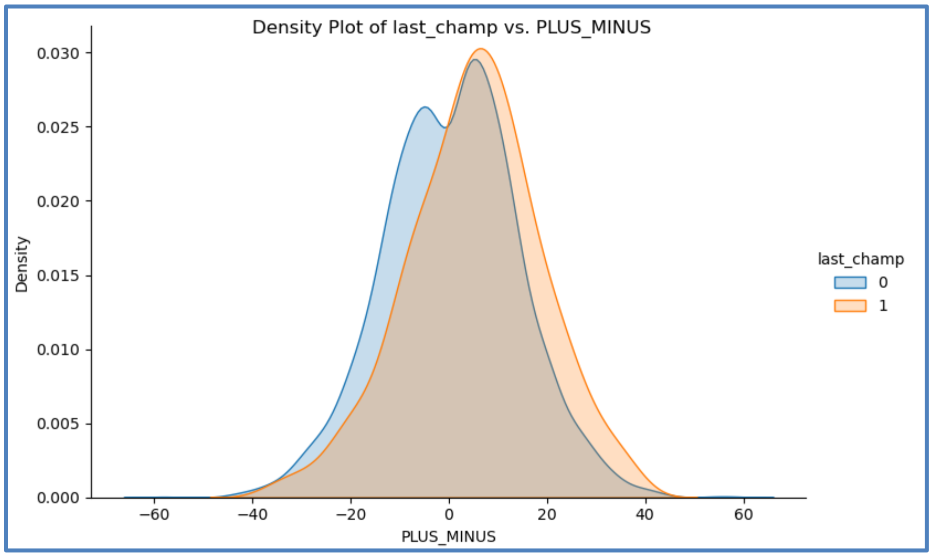


```
Not Normal
count    3526.000000
mean      100.644285
std        13.379239
min        56.000000
25%        91.000000
50%       100.000000
75%       109.937500
max       154.000000
Name: PTS, dtype: float64
Median: 100.0
IQR: 18.9375
Missing: 0
```

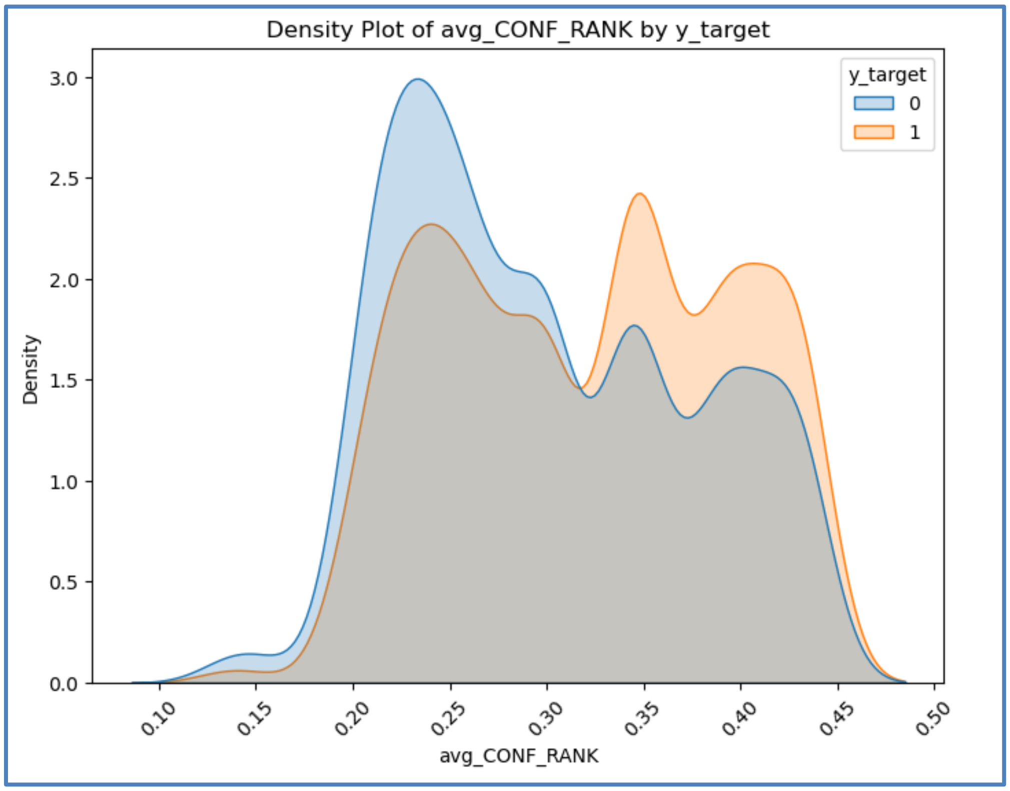
- Analysis of Categorical Variables:

  Examined differences between subcategories in a categorical variable.



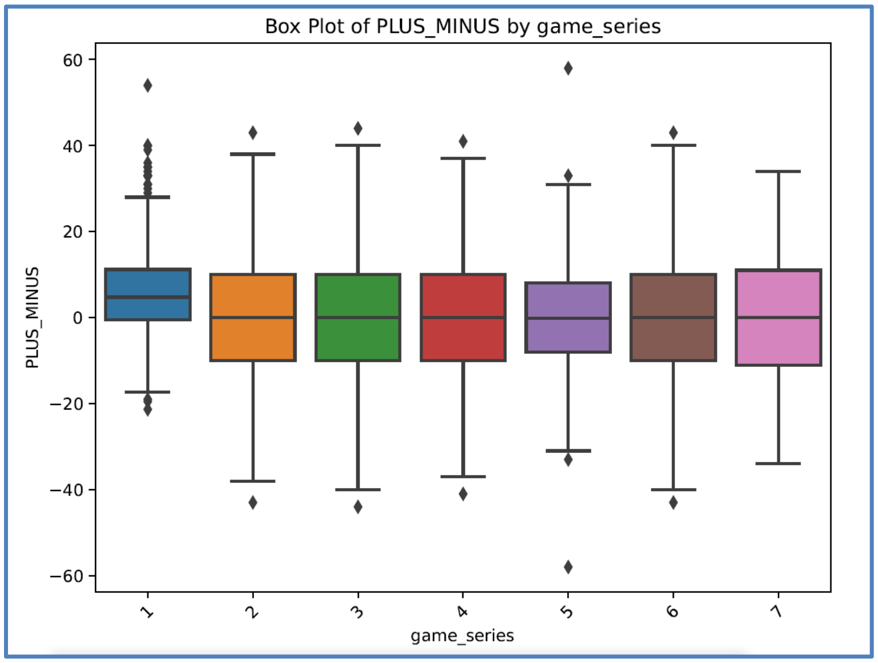


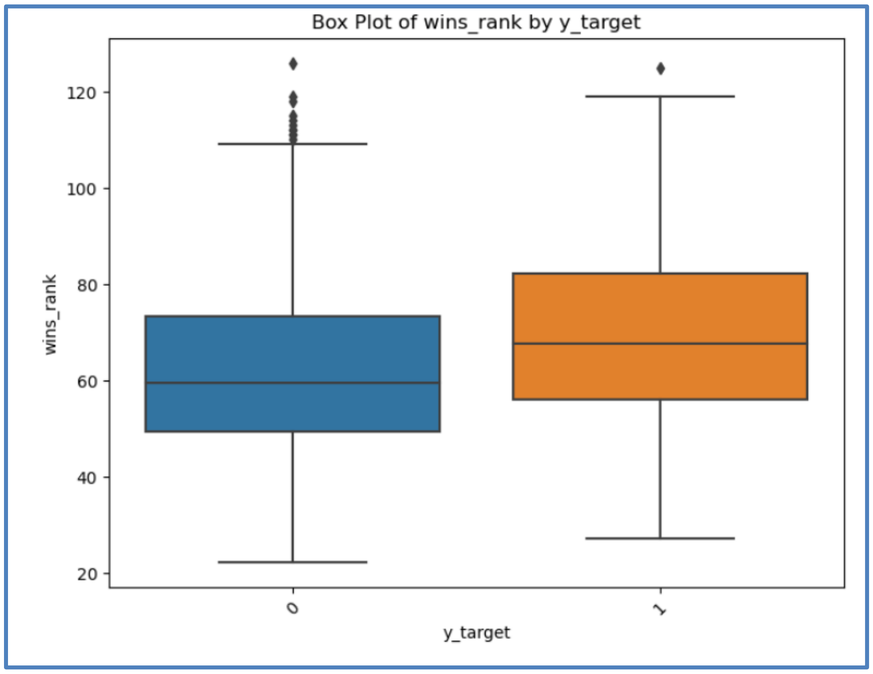
- Investigated differences among categorical variables themselves.

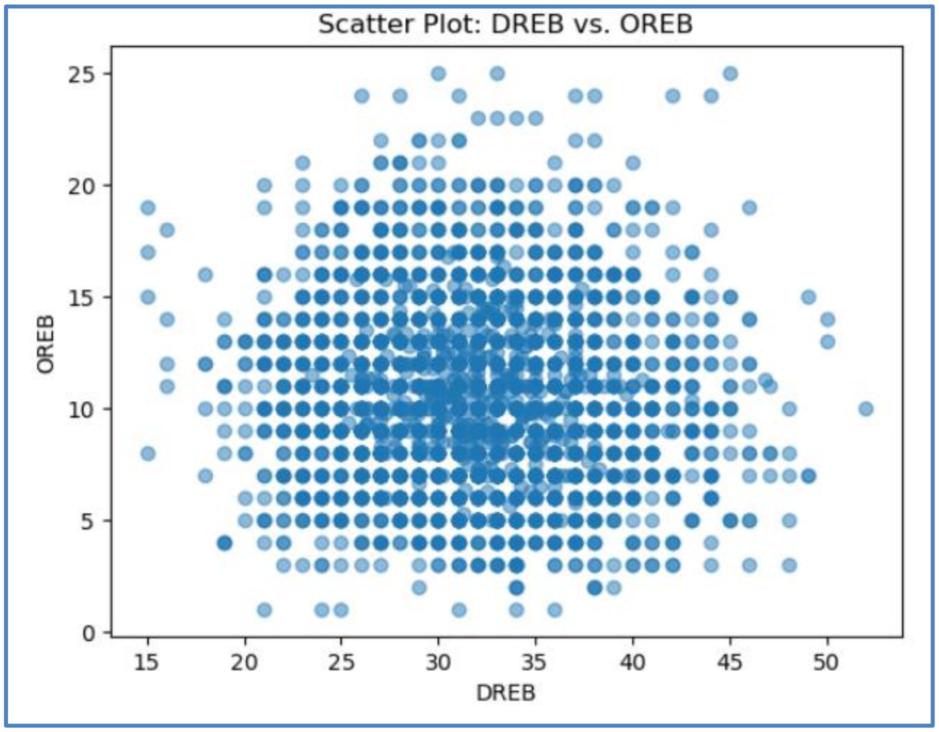


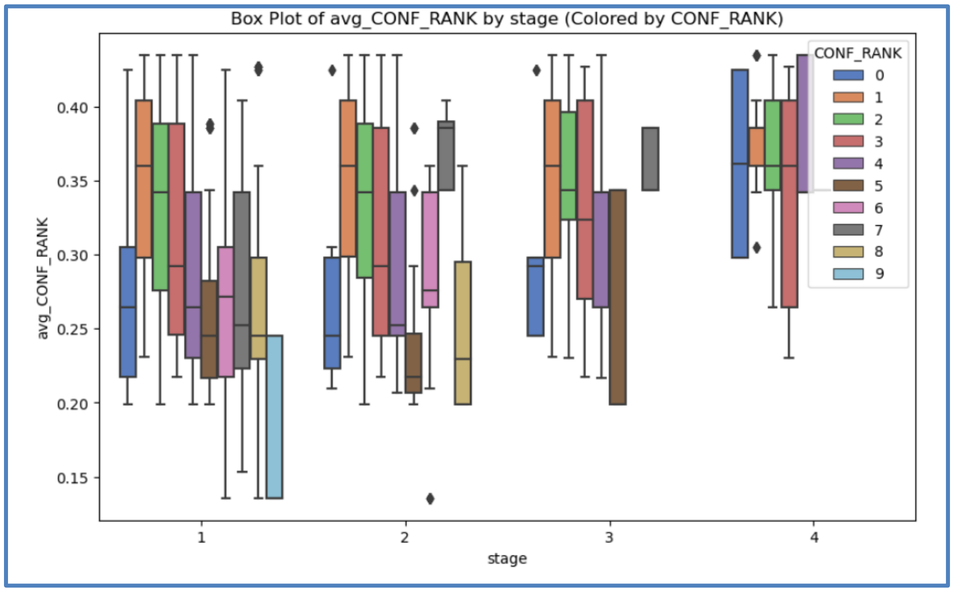
- Multivariable Graphs:

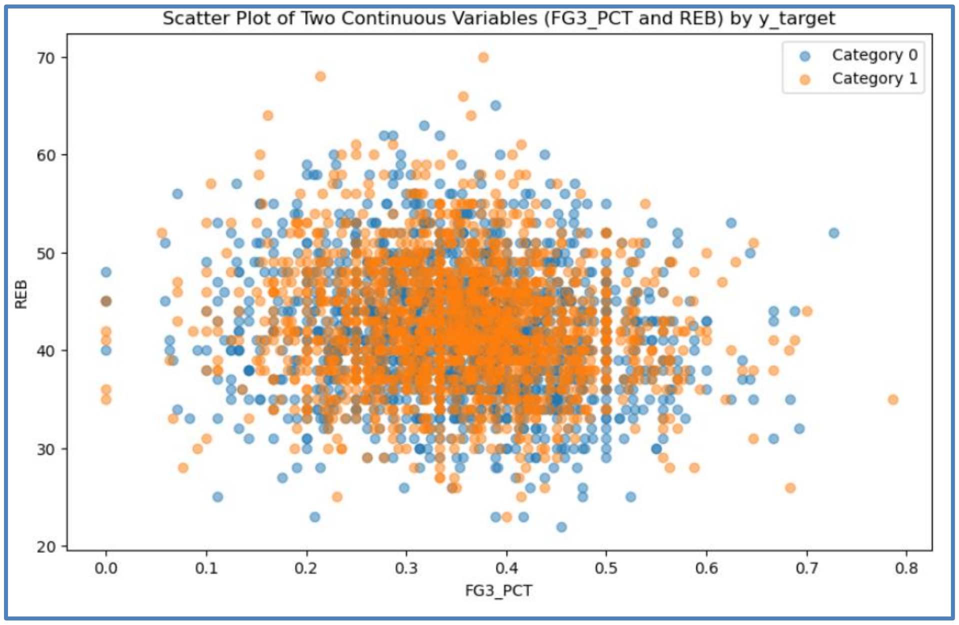
  Created graphs for sets of continuous and categorical variables to examine relationships, behaviors, and patterns involving two or three variables.

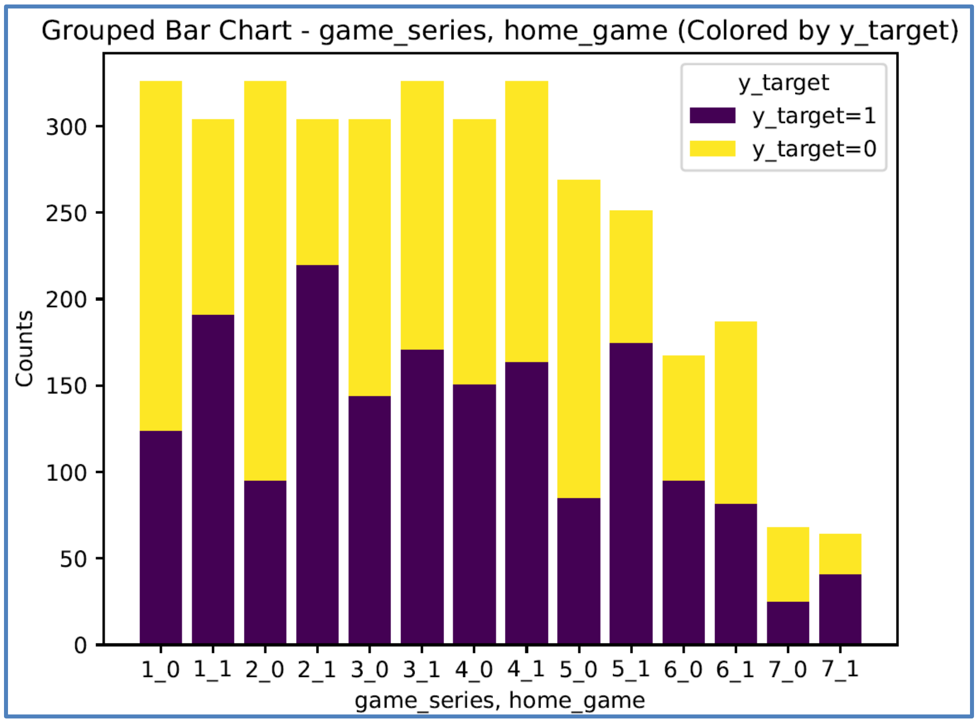


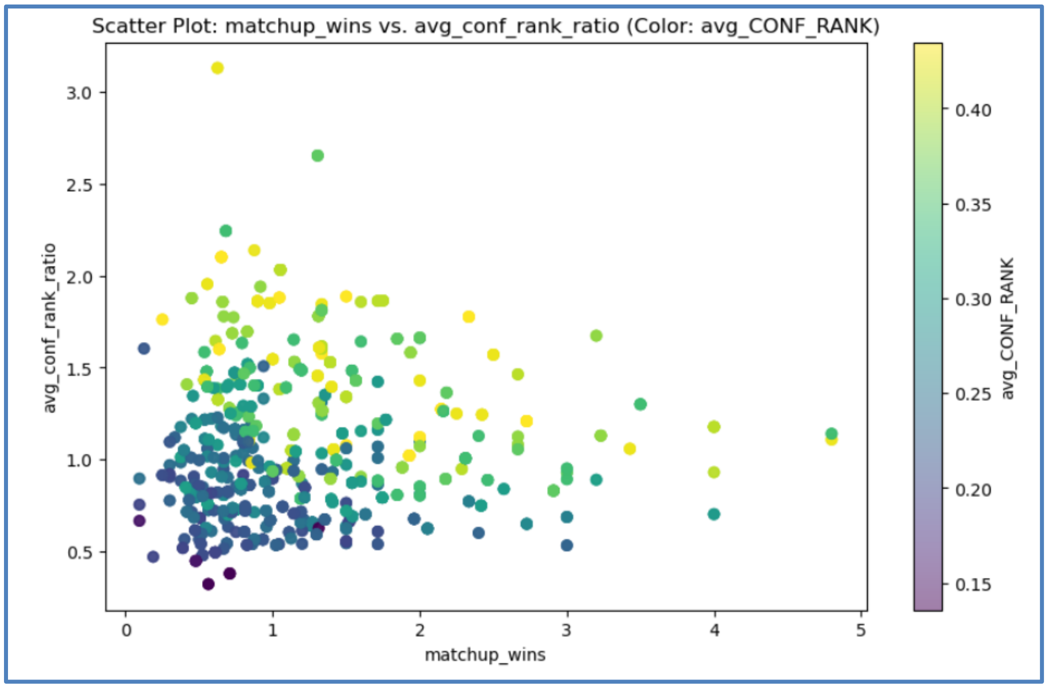



Scatter Plot: DREB vs. OREB
OREB
DREB
0
5
10
15
20
25
15
20
25
30
35
40
45
50


Box Plot of avg_CONF_RANK by stage (Colored by CONF_RANK)
avg_CONF_RANK
stage
0.15
0.20
0.25
0.30
0.35
0.40
1
2
3
4
CONF_RANK
0
1
2
3
4
5
6
7
8
9

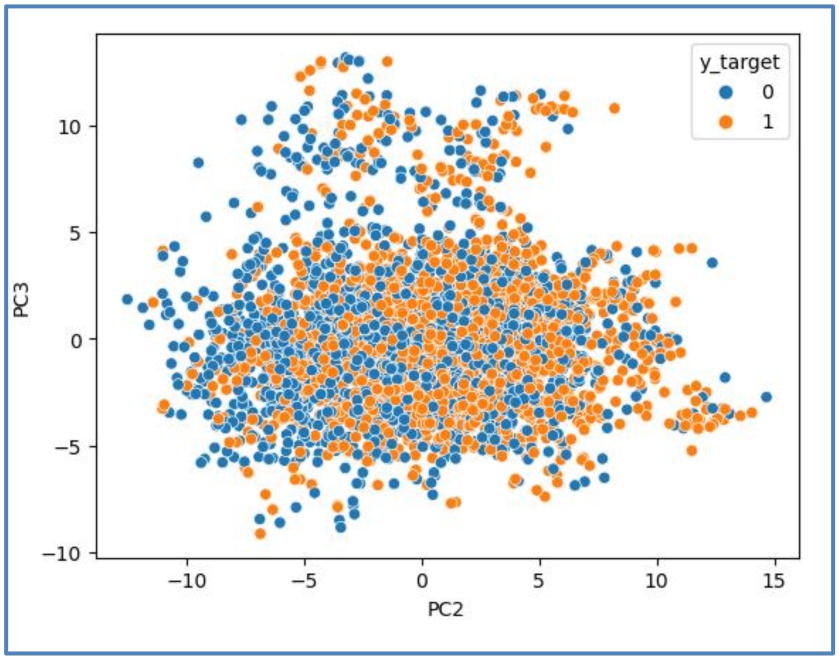

Scatter Plot of Two Continuous Variables (FG3_PCT and REB) by y_target
Category 0
Category 1
REB
FG3_PCT
70
60
50
40
30
20
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

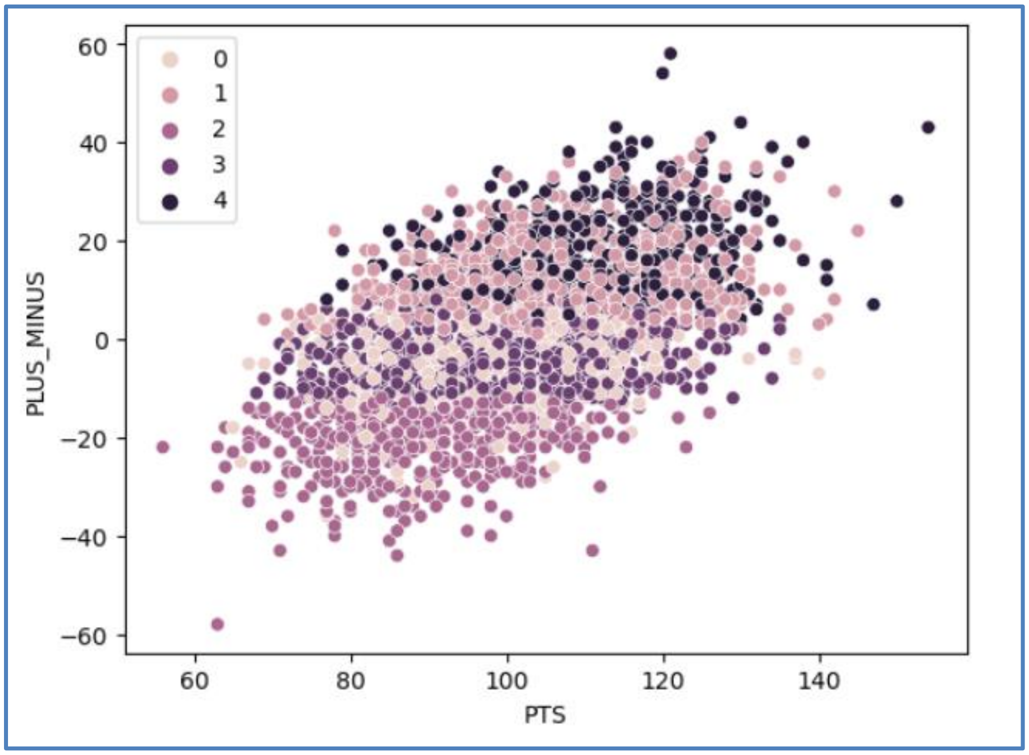

Grouped Bar Chart - game_series, home_game (Colored by y_target)
y_target
y_target=1
y_target=0
Counts
game_series, home_game
1_0
1_1
2_0
2_1
3_0
3_1
4_0
4_1
5_0
5_1
6_0
6_1
7_0
7_1
0
50
100
150
200
250
300



- <u>Unsupervised Learning</u>:

  Applied distance-based models and unsupervised models to discover hidden patterns within the data (K-means, PCA, MCA models).





- <u>Domain knowledge</u>:

  Investigating logically related variables, opposing variables that influence each other, variables that describe the same phenomenon, etc.

- By following these steps, the project achieved a thorough understanding of the dataset, identified potential issues, and uncovered key insights that informed the modeling process.

8. **Outliers**:

- Given that the data represents real game and season statistics, outliers values were not removed. The goal was to preserve the integrity of the data with minimal statistical manipulation. The percentage of outliers values was assessed as follows:
  - Lower limit: Q1−1.5×IQR
  - Upper limit: Q3+1.5×IQR
  - Interquartile Range (IQR): Q3−Q1

  Outliers values in the continuous variables ranged up to 5%, with most variables having up to 1.5% outliers values. The observed outliers values were not abnormal within NBA standards. Therefore, we proceeded with two approaches:

1. No handling of outliers values.
2. Treatment of outliers values, which was performed as follows:

    - For variables with total outliers values ≤ 0.5%, the minimum outliers values were set to the lower limit $Q1-1.5\times IQR$ and the maximum outliers values were set to the upper limit $Q3+1.5\times IQR$.
    - For variables with total outliers values > 0.5%, logarithmic (Log) and square root (Sqrt) transformations were applied. The transformation that reduced the number of outliers values the most was selected for each variable.
    - If neither Log nor Sqrt transformations reduced the outliers values:
        - For total outliers values ≤ 1.5%, the minimum and maximum extreme values were set as described above.
        - For total extreme values > 1.5%, the variable was converted to a categorical variable. Subcategories were divided based on quantiles that best described the original continuous variable.
- The model with continuous variables without any treatment of outliers values produced better results by 1%, therfore we continue without outliers treatment, as well as the understanding that tree-based models are less sensitive to outliers

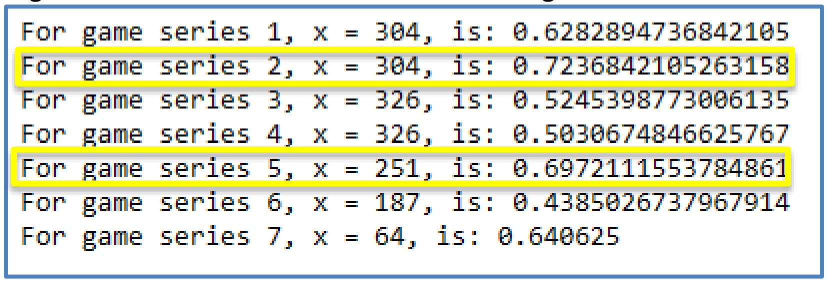
9. **Missing Values**:

- There were no missing values in the data set.

Data Enrichment

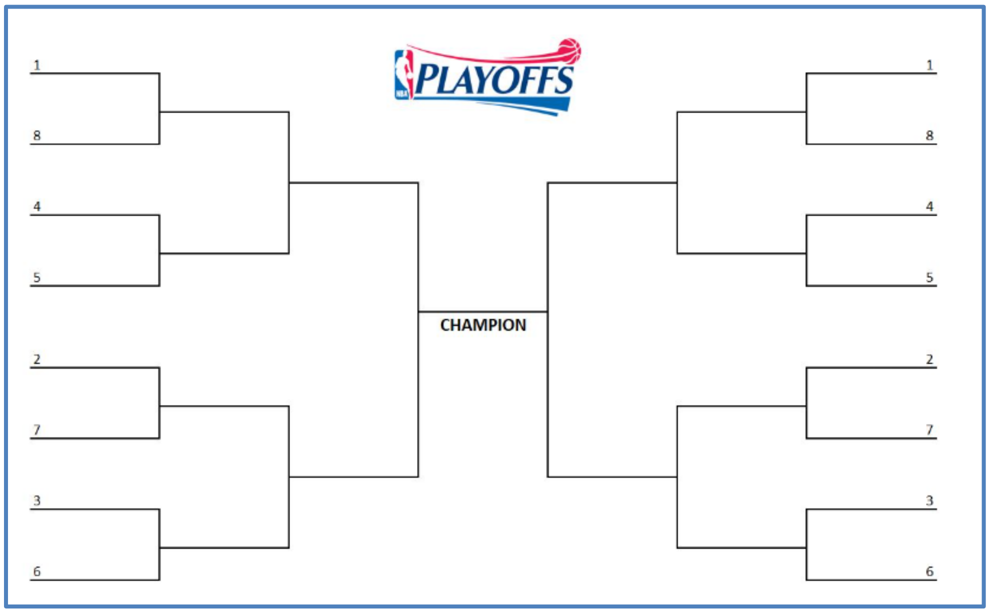
Conclusions:

- **The presence of a large volume of data does not necessarily equate to a wealth of information**. Many variables essentially encapsulate the same core underlying variables, resulting in multiple manipulations of the same data that convey identical information. This redundancy highlights the importance of identifying and focusing on the fundamental variables that truly contribute unique insights to the analysis.
- 60% of the wins are in home games (home advantage is of the team that finished higher in the regular season, reflecting their greater number of victories).
- Lower ranked teams have lower wins % in home games.
- Higher ranked teams with 70% wins in home games series #2 and #5.



```
For game series 1, x = 304, is: 0.6282894736842105
For game series 2, x = 304, is: 0.7236842105263158
For game series 3, x = 326, is: 0.5245398773006135
For game series 4, x = 326, is: 0.5030674846625767
For game series 5, x = 251, is: 0.6972111553784861
For game series 6, x = 187, is: 0.4385026737967914
For game series 7, x = 64, is: 0.640625
```

- A team which reached the NBA Finals the previous season has an average of 72% home wins in the playoffs the following season.
- The team that won the NBA championship in the previous season and finished the regular season after that in places 1-3 has a 64-73% win rate, if it finished in 4th place or lower the win % drops drastically.
- NBA Playoff - Rank:



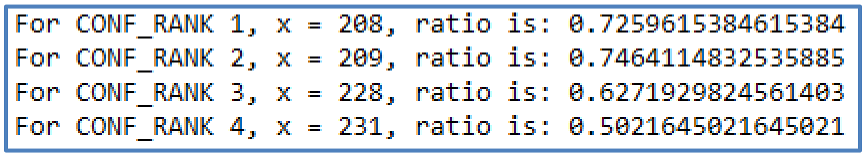
Round 1:

Rank 1 Vs 8 – Clear advantage for Rank 1 (wins %).

Rank 2 Vs 7 - Clear advantage for Rank 2 (wins %).

Rank 3 Vs 6 - Clear advantage for Rank 3 (wins %), not as Rank 1 and 2.

Rank 4 Vs 5 - There is no distinct advantage or any advantage at all for any of the Ranks.

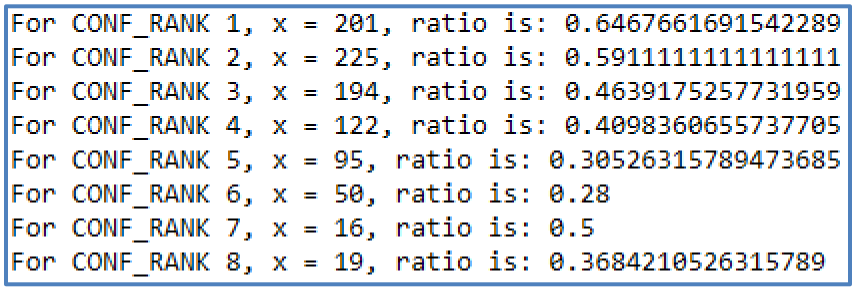


```
For CONF_RANK 1, x = 208, ratio is: 0.7259615384615384
For CONF_RANK 2, x = 209, ratio is: 0.7464114832535885
For CONF_RANK 3, x = 228, ratio is: 0.6271929824561403
For CONF_RANK 4, x = 231, ratio is: 0.5021645021645021
```

The wins % of 1 and 2 Rank in the 1st and 2nd round in games 5, 7 are higher than in games 1, 2.

<u>Round 2:</u>

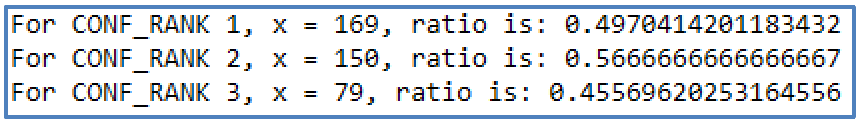
Advantage for Ranks 1 and 2, less pronounced than the 1st round



```
For CONF_RANK 1, x = 201, ratio is: 0.6467661691542289
For CONF_RANK 2, x = 225, ratio is: 0.5911111111111111
For CONF_RANK 3, x = 194, ratio is: 0.4639175257731959
For CONF_RANK 4, x = 122, ratio is: 0.4098360655737705
For CONF_RANK 5, x = 95, ratio is: 0.30526315789473685
For CONF_RANK 6, x = 50, ratio is: 0.28
For CONF_RANK 7, x = 16, ratio is: 0.5
For CONF_RANK 8, x = 19, ratio is: 0.3684210526315789
```

<u>Round 3:</u>

There is no clear advantage to either Rank, although the 2 Rank seems to have the best "chance of success" in the 3rd round.



```
For CONF_RANK 1, x = 169, ratio is: 0.4970414201183432
For CONF_RANK 2, x = 150, ratio is: 0.5666666666666667
For CONF_RANK 3, x = 79, ratio is: 0.45569620253164556
```

<u>East – West:</u>

The 1st Rank in the East in the 3rd round has a % of wins smaller by almost 50% compared to the equivalent in the West.

2nd Rank in the East in the 1st, 2nd, 3rd rounds has a winning % that is almost 10% greater than the equivalent in the West.

- <u>Most Informative Variables:</u>

  Rank

  Wins % (regular season)

  Stage (round)

  Game series

  3-point %

  Total %

  Rebounds (offensive, defensive, total)

  Fouls

These variables provide the most significant insights and should be prioritized in the analysis to avoid redundancy and ensure the extraction of meaningful information.

- The data by itself does not present the full picture and the rael meaning apart from the value of the data itself. The attributions of the data provide additional dimensions and prespctive, for example:

  Game Offensive Stats, Team X Vs Seasonal Offensive Stats Average, Team X.

  Game Defensive Stats, Team X Vs Seasonal Defensive Stats Average, Team X.

  Offensive stats Team X Vs Defense stats Team Y.

  Defensive stats, Team X: Game Offensive stats, Team Y.

  Team X opponents average stats Vs Team Y stat.

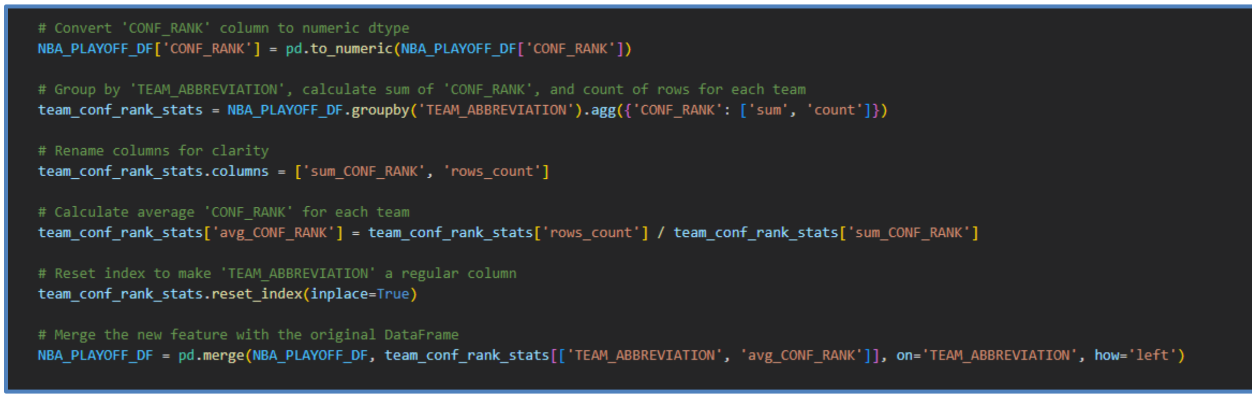
  By comparing offensive to defensive game stats and seasonal averages, we can gain a deeper understanding of team performance. This approach allows us to see how well a team's offensive capabilities match up against another team's defensive strengths, providing a more comprehensive view of their overall effectiveness.

Feature Engineering:

- <u>Transforming Categorical to Continuous Variables:</u>

To retain as much information as possible, we converted categorical variables with many subcategories into continuous variables. This approach allows for a more nuanced representation of the data, which can enhance the model's performance. Here are two specific examples:

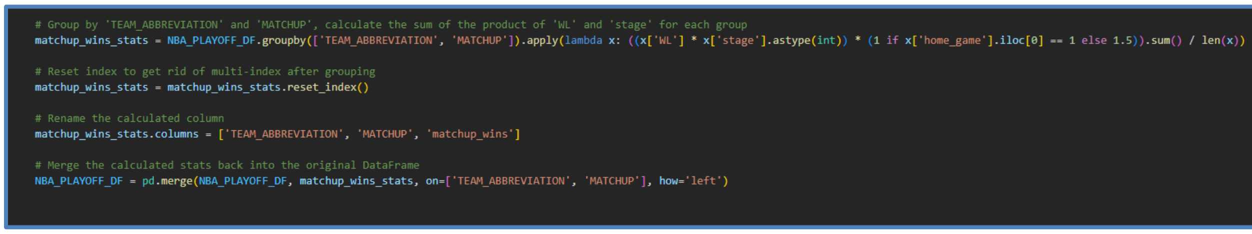
1. <u>Team:</u>
    - Original Categorical Variable: Team with multiple nominal subcategories representing different teams.
    - Transformation: We calculated the sum of Rank for each group across all seasons and divided it by the total number of playoff appearances for that group in all seasons.



```python
# Convert 'CONF_RANK' column to numeric dtype
NBA_PLAYOFF_DF['CONF_RANK'] = pd.to_numeric(NBA_PLAYOFF_DF['CONF_RANK'])

# Group by 'TEAM_ABBREVIATION', calculate sum of 'CONF_RANK', and count of rows for each team
team_conf_rank_stats = NBA_PLAYOFF_DF.groupby('TEAM_ABBREVIATION').agg({'CONF_RANK': ['sum', 'count']})

# Rename columns for clarity
team_conf_rank_stats.columns = ['sum_CONF_RANK', 'rows_count']

# Calculate average 'CONF_RANK' for each team
team_conf_rank_stats['avg_CONF_RANK'] = team_conf_rank_stats['rows_count'] / team_conf_rank_stats['sum_CONF_RANK']

# Reset index to make 'TEAM_ABBREVIATION' a regular column
team_conf_rank_stats.reset_index(inplace=True)

# Merge the new feature with the original DataFrame
NBA_PLAYOFF_DF = pd.merge(NBA_PLAYOFF_DF, team_conf_rank_stats[['TEAM_ABBREVIATION', 'avg_CONF_RANK']], on='TEAM_ABBREVIATION', how='left')
```

    - Rationale: This transformation provides a continuous variable that reflects the average performance of each group over time, capturing both their consistency in the playoffs.

2. <u>Match-up:</u>
    - Original Categorical Variable: Match-up with multiple nominal subcategories representing different Match-up's.
    - Transformation: We created a continuous variable by multiplying the binary playoff win outcome (0 for loss, 1 for win) by the stage of the playoffs (1 for the first round, 2 for the second round, 3 for the conference finals, 4 for the NBA Finals), and then adjusted for home (multiplied by 1) or away (multiplied by 1.5) games. Finally, we divided this by the total number of matches between the teams.



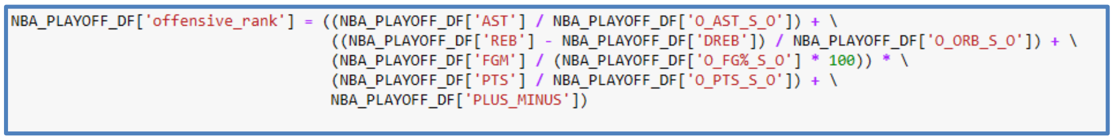
```python
# Group by 'TEAM_ABBREVIATION' and 'MATCHUP', calculate the sum of the product of 'WL' and 'stage' for each group
matchup_wins_stats = NBA_PLAYOFF_DF.groupby(['TEAM_ABBREVIATION', 'MATCHUP']).apply(lambda x: ((x['WL'] * x['stage'].astype(int)) * (1 if x['home_game'].iloc[0] == 1 else 1.5)).sum() / len(x))

# Reset index to get rid of multi-index after grouping
matchup_wins_stats = matchup_wins_stats.reset_index()

# Rename the calculated column
matchup_wins_stats.columns = ['TEAM_ABBREVIATION', 'MATCHUP', 'matchup_wins']

# Merge the calculated stats back into the original DataFrame
NBA_PLAYOFF_DF = pd.merge(NBA_PLAYOFF_DF, matchup_wins_stats, on=['TEAM_ABBREVIATION', 'MATCHUP'], how='left')
```

    - Rationale: This transformation creates a continuous variable that accounts for the importance of the stage, the advantage or disadvantage of playing at home or away, and the historical frequency of matches between the teams. It captures the overall competitive strength and situational performance of the teams.

By converting these categorical variables into continuous ones, we ensure that the data retains its richness and complexity, providing more detailed inputs for our predictive models.

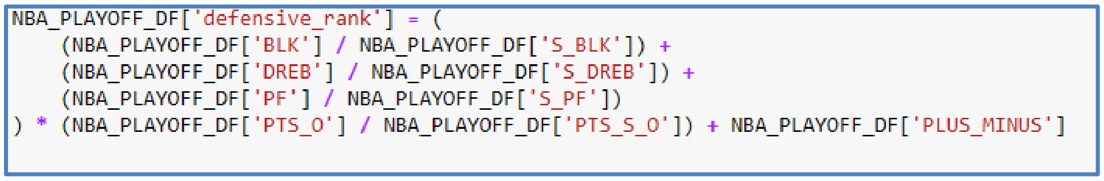
- Domain knowledge - Mathematical Operations + Logical Relationships:
  To enhance the dataset and capture complex relationships, we generated new variables through a combination of mathematical operations and logical relationships. This approach was applied to variables mentioned in the conclusions section, variables with high correlation values (greater than 0.8), and logically related variables, for example:

- **Offensive_rank**: Created by combining offensive parameters relative to the opponent's defense and seasonal averages.



```
NBA_PLAYOFF_DF['offensive_rank'] = ((NBA_PLAYOFF_DF['AST'] / NBA_PLAYOFF_DF['O_AST_S_O']) + \
                                   ((NBA_PLAYOFF_DF['REB'] - NBA_PLAYOFF_DF['DREB']) / NBA_PLAYOFF_DF['O_ORB_S_O']) + \
                                   (NBA_PLAYOFF_DF['FGM'] / (NBA_PLAYOFF_DF['O_FG%_S_O'] * 100)) * \
                                   (NBA_PLAYOFF_DF['PTS'] / NBA_PLAYOFF_DF['O_PTS_S_O']) + \
                                   NBA_PLAYOFF_DF['PLUS_MINUS'])
```

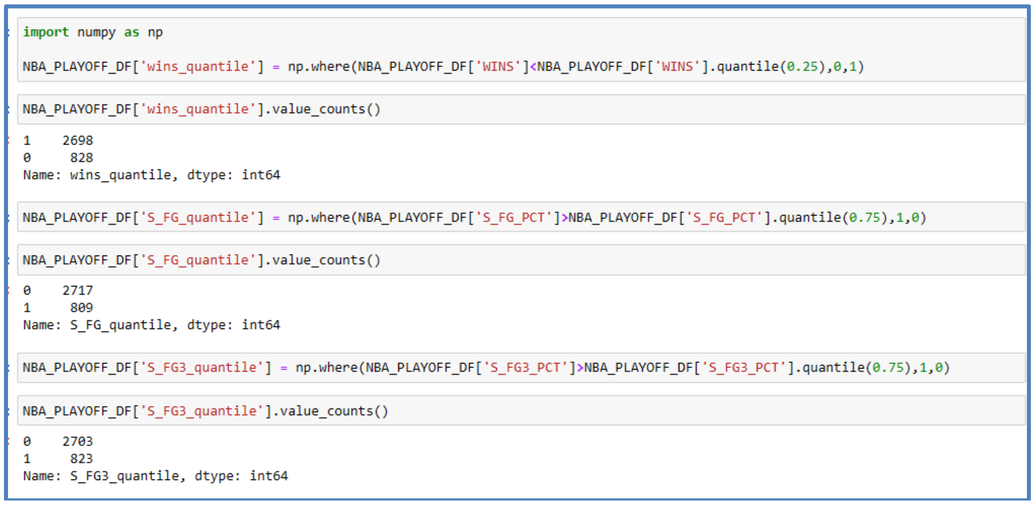
- **Defensive_rank**: Created by combining defensive parameters relative to the opponent's offense and seasonal averages.



```
NBA_PLAYOFF_DF['defensive_rank'] = (
    (NBA_PLAYOFF_DF['BLK'] / NBA_PLAYOFF_DF['S_BLK']) +
    (NBA_PLAYOFF_DF['DREB'] / NBA_PLAYOFF_DF['S_DREB']) +
    (NBA_PLAYOFF_DF['PF'] / NBA_PLAYOFF_DF['S_PF'])
) * (NBA_PLAYOFF_DF['PTS_O'] / NBA_PLAYOFF_DF['PTS_S_O']) + NBA_PLAYOFF_DF['PLUS_MINUS']
```

By deriving these new variables, we aimed to uncover deeper insights and enhance the predictive power of our models.

- Statistical Data (Continuous Variables):



```
import numpy as np

NBA_PLAYOFF_DF['wins_quantile'] = np.where(NBA_PLAYOFF_DF['WINS']<NBA_PLAYOFF_DF['WINS'].quantile(0.25),0,1)
```

```
NBA_PLAYOFF_DF['wins_quantile'].value_counts()
```

```
1    2698
0     828
Name: wins_quantile, dtype: int64
```

```
NBA_PLAYOFF_DF['S_FG_quantile'] = np.where(NBA_PLAYOFF_DF['S_FG_PCT']>NBA_PLAYOFF_DF['S_FG_PCT'].quantile(0.75),1,0)
```

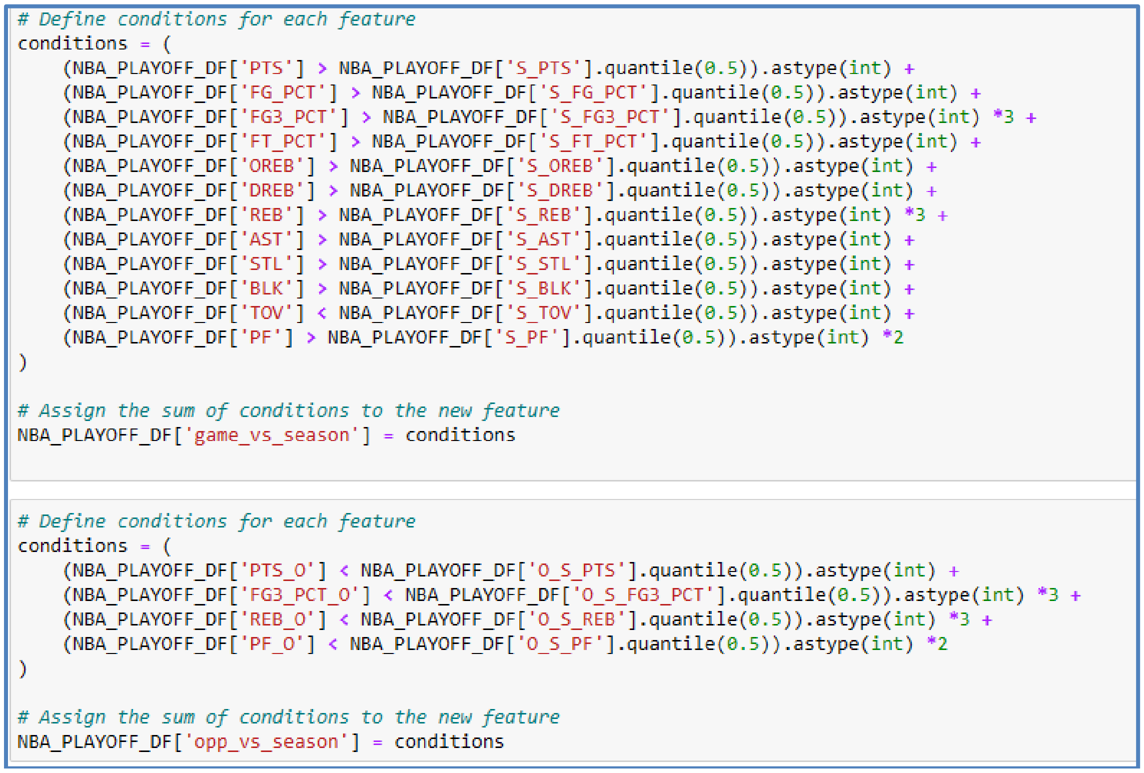
```
NBA_PLAYOFF_DF['S_FG_quantile'].value_counts()
```

```
0    2717
1     809
Name: S_FG_quantile, dtype: int64
```

```
NBA_PLAYOFF_DF['S_FG3_quantile'] = np.where(NBA_PLAYOFF_DF['S_FG3_PCT']>NBA_PLAYOFF_DF['S_FG3_PCT'].quantile(0.75),1,0)
```

```
NBA_PLAYOFF_DF['S_FG3_quantile'].value_counts()
```

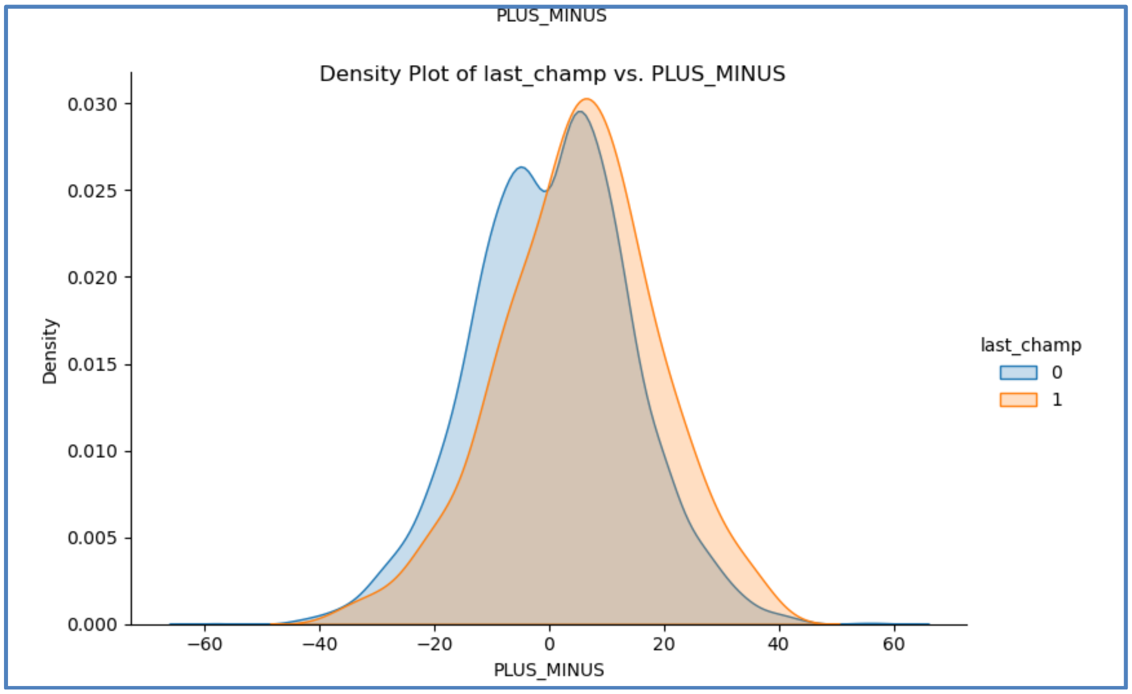
```
0    2703
1     823
Name: S_FG3_quantile, dtype: int64
```

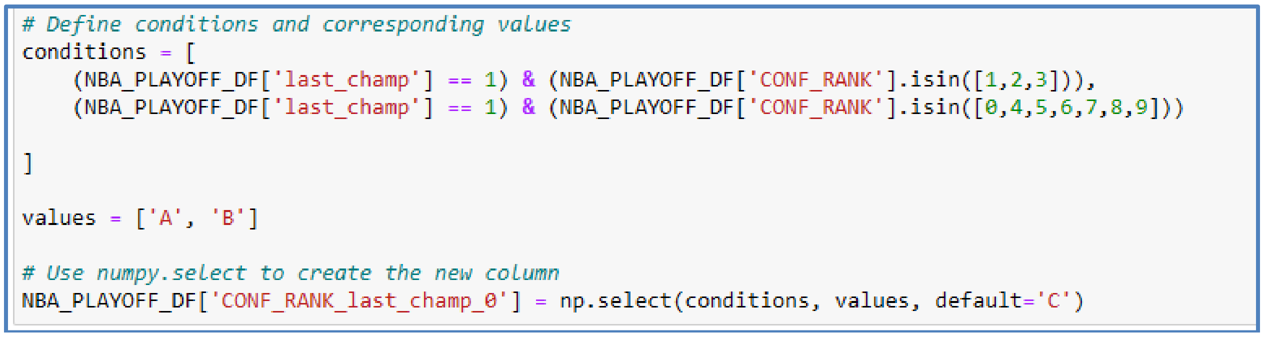


```
# Define conditions for each feature
conditions = (
    (NBA_PLAYOFF_DF['PTS'] > NBA_PLAYOFF_DF['S_PTS'].quantile(0.5)).astype(int) +
    (NBA_PLAYOFF_DF['FG_PCT'] > NBA_PLAYOFF_DF['S_FG_PCT'].quantile(0.5)).astype(int) +
    (NBA_PLAYOFF_DF['FG3_PCT'] > NBA_PLAYOFF_DF['S_FG3_PCT'].quantile(0.5)).astype(int) *3 +
    (NBA_PLAYOFF_DF['FT_PCT'] > NBA_PLAYOFF_DF['S_FT_PCT'].quantile(0.5)).astype(int) +
    (NBA_PLAYOFF_DF['OREB'] > NBA_PLAYOFF_DF['S_OREB'].quantile(0.5)).astype(int) +
    (NBA_PLAYOFF_DF['DREB'] > NBA_PLAYOFF_DF['S_DREB'].quantile(0.5)).astype(int) +
    (NBA_PLAYOFF_DF['REB'] > NBA_PLAYOFF_DF['S_REB'].quantile(0.5)).astype(int) *3 +
    (NBA_PLAYOFF_DF['AST'] > NBA_PLAYOFF_DF['S_AST'].quantile(0.5)).astype(int) +
    (NBA_PLAYOFF_DF['STL'] > NBA_PLAYOFF_DF['S_STL'].quantile(0.5)).astype(int) +
    (NBA_PLAYOFF_DF['BLK'] > NBA_PLAYOFF_DF['S_BLK'].quantile(0.5)).astype(int) +
    (NBA_PLAYOFF_DF['TOV'] < NBA_PLAYOFF_DF['S_TOV'].quantile(0.5)).astype(int) +
    (NBA_PLAYOFF_DF['PF'] > NBA_PLAYOFF_DF['S_PF'].quantile(0.5)).astype(int) *2
)

# Assign the sum of conditions to the new feature
NBA_PLAYOFF_DF['game_vs_season'] = conditions
```

```
# Define conditions for each feature
conditions = (
    (NBA_PLAYOFF_DF['PTS_O'] < NBA_PLAYOFF_DF['O_S_PTS'].quantile(0.5)).astype(int) +
    (NBA_PLAYOFF_DF['FG3_PCT_O'] < NBA_PLAYOFF_DF['O_S_FG3_PCT'].quantile(0.5)).astype(int) *3 +
    (NBA_PLAYOFF_DF['REB_O'] < NBA_PLAYOFF_DF['O_S_REB'].quantile(0.5)).astype(int) *3 +
    (NBA_PLAYOFF_DF['PF_O'] < NBA_PLAYOFF_DF['O_S_PF'].quantile(0.5)).astype(int) *2
)

# Assign the sum of conditions to the new feature
NBA_PLAYOFF_DF['opp_vs_season'] = conditions
```

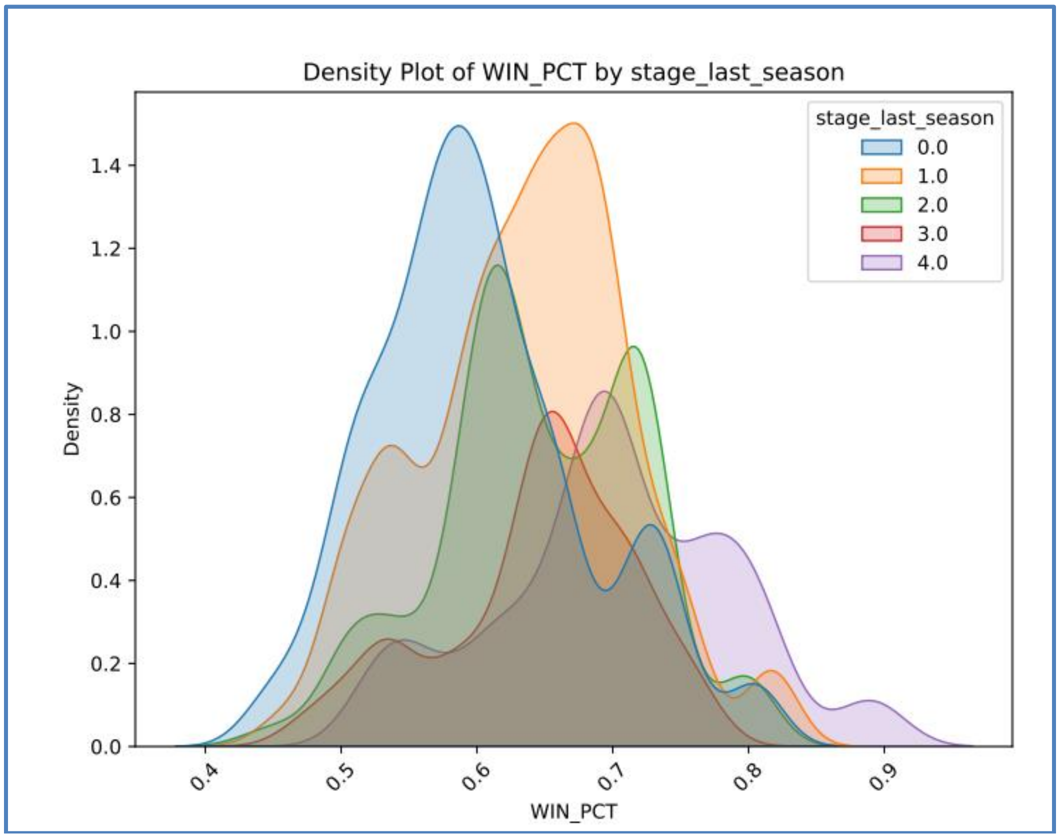
- <u>Differences Between Categorical Variables:</u>

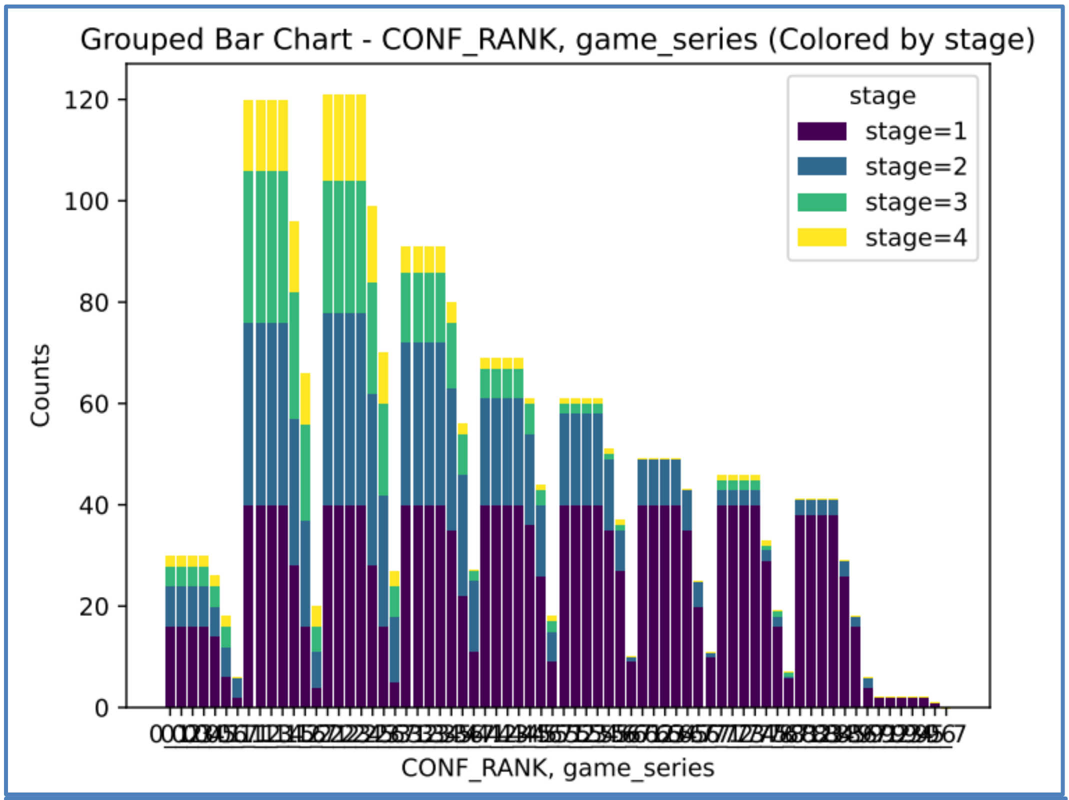


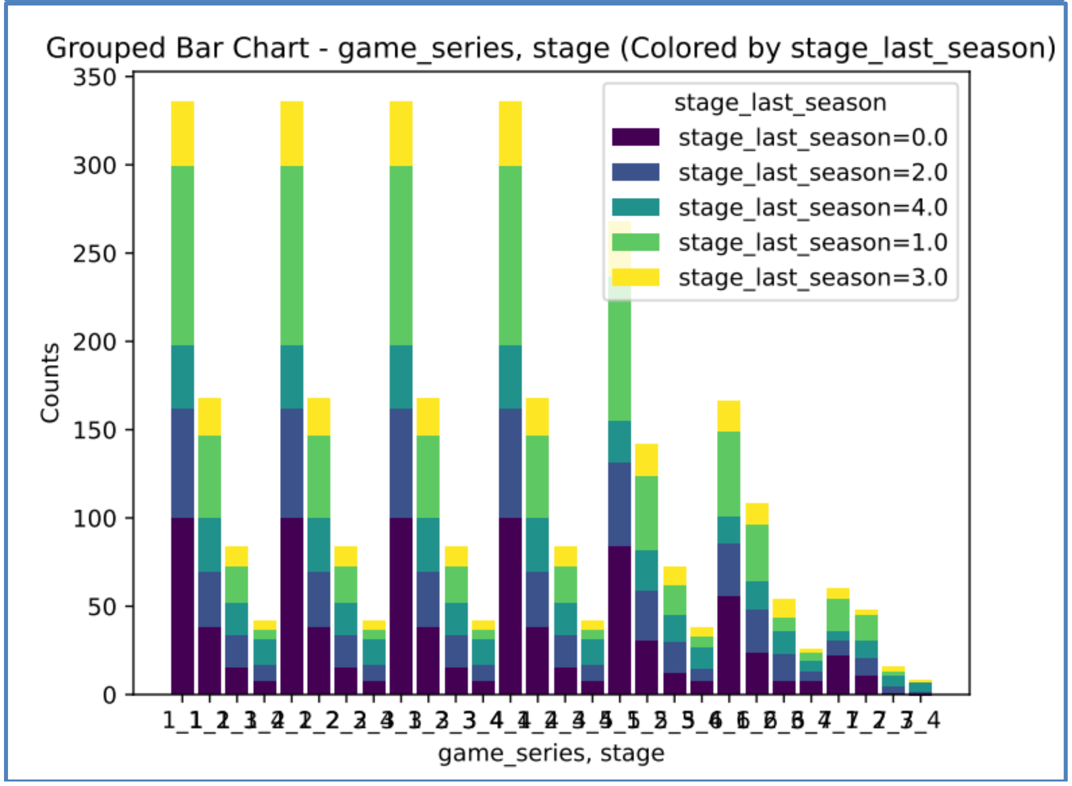


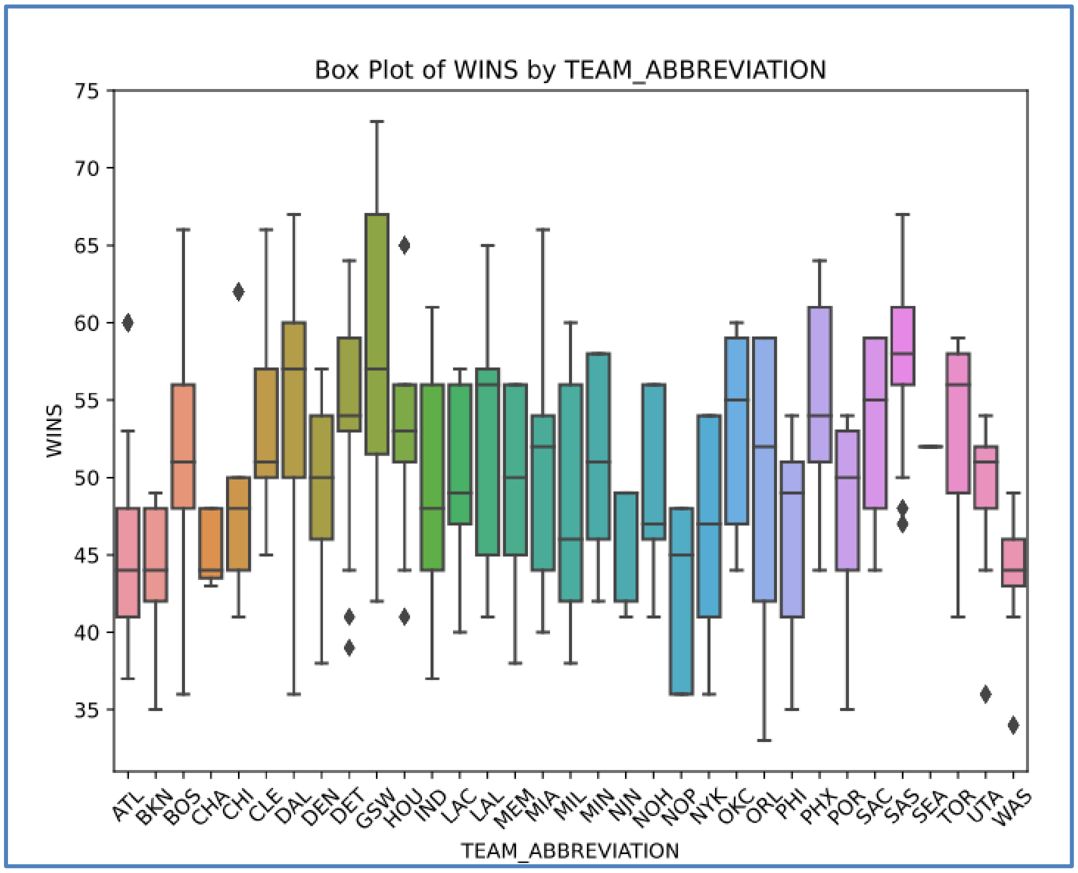
```
# Define conditions and corresponding values
conditions = [
    (NBA_PLAYOFF_DF['last_champ'] == 1) & (NBA_PLAYOFF_DF['CONF_RANK'].isin([1,2,3])),
    (NBA_PLAYOFF_DF['last_champ'] == 1) & (NBA_PLAYOFF_DF['CONF_RANK'].isin([0,4,5,6,7,8,9]))

]

values = ['A', 'B']

# Use numpy.select to create the new column
NBA_PLAYOFF_DF['CONF_RANK_last_champ_0'] = np.select(conditions, values, default='C')
```

- <u>Relationships Between 2-3 (categorical and continus) Variables Using Graphical Analysis:</u>

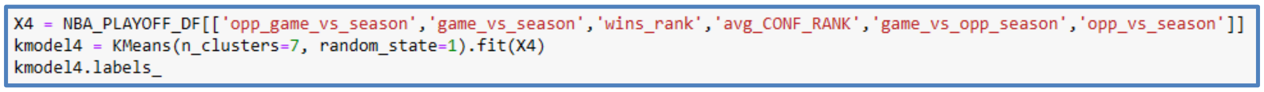


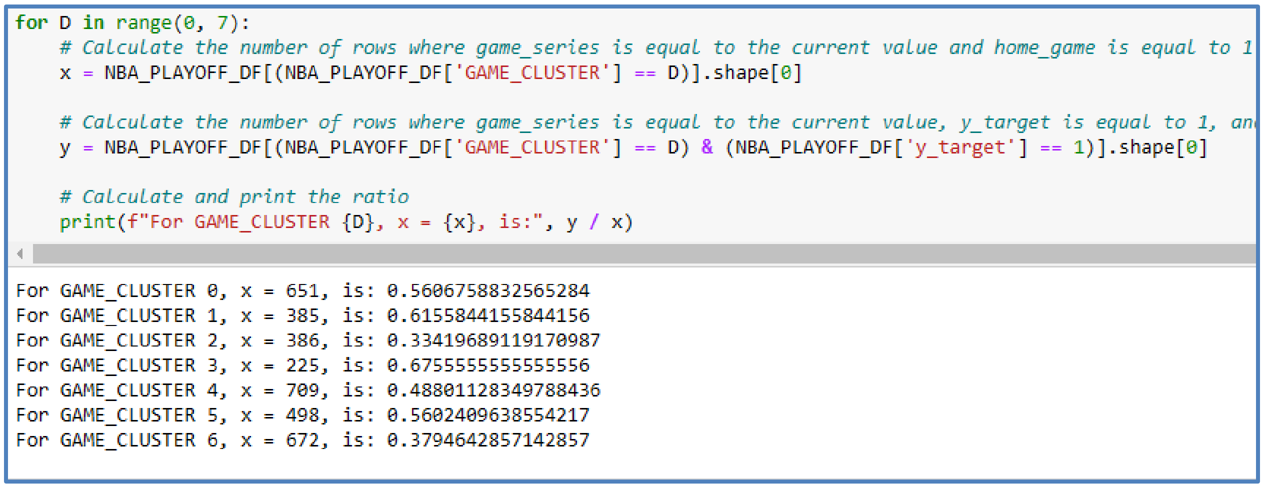

Grouped Bar Chart - CONF_RANK, game_series (Colored by stage)
stage
stage=1
stage=2
stage=3
stage=4
Counts
CONF_RANK, game_series


Grouped Bar Chart - game_series, stage (Colored by stage_last_season)
stage_last_season
stage_last_season=0.0
stage_last_season=2.0
stage_last_season=4.0
stage_last_season=1.0
stage_last_season=3.0
Counts
game_series, stage



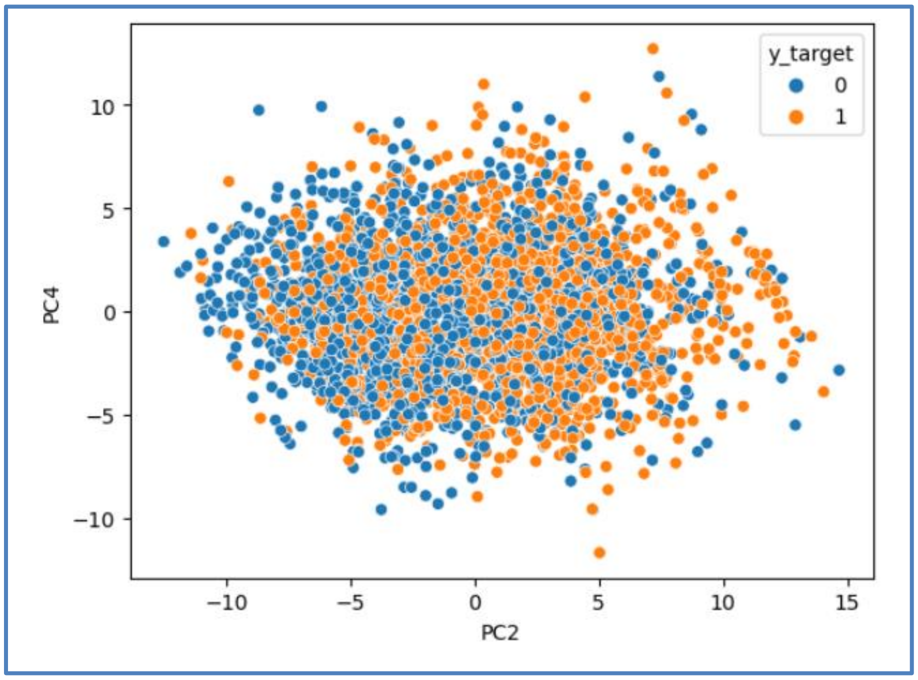
- <u>Distance based models:</u>
    1. Creating variables by searching for based model patterns - using variables with similar correlation values/
    2. Creating variables by searching for based model patterns - using variables that are related to each other in terms of their logical meaning.


```
X4 = NBA_PLAYOFF_DF[['opp_game_vs_season','game_vs_season','wins_rank','avg_CONF_RANK','game_vs_opp_season','opp_vs_season']]
kmodel4 = KMeans(n_clusters=7, random_state=1).fit(X4)
kmodel4.labels_
```

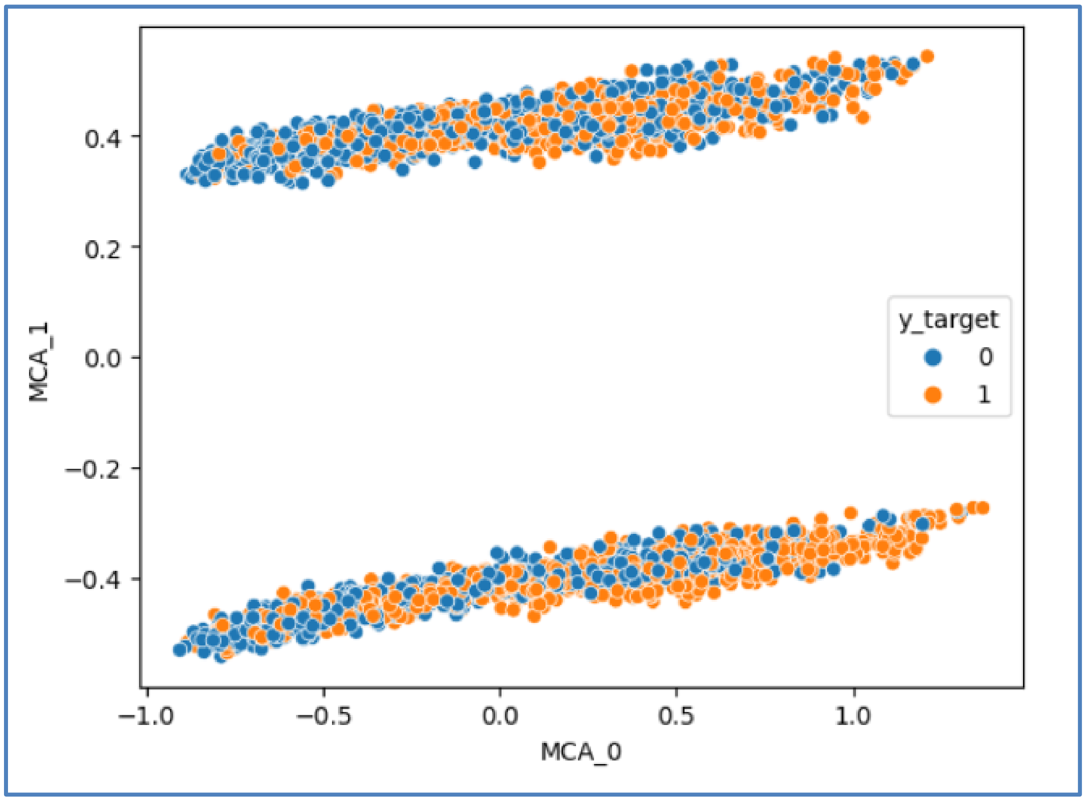

```
for D in range(0, 7):
    # Calculate the number of rows where game_series is equal to the current value and home_game is equal to 1
    x = NBA_PLAYOFF_DF[(NBA_PLAYOFF_DF['GAME_CLUSTER'] == D)].shape[0]

    # Calculate the number of rows where game_series is equal to the current value, y_target is equal to 1, an
    y = NBA_PLAYOFF_DF[(NBA_PLAYOFF_DF['GAME_CLUSTER'] == D) & (NBA_PLAYOFF_DF['y_target'] == 1)].shape[0]

    # Calculate and print the ratio
    print(f"For GAME_CLUSTER {D}, x = {x}, is:", y / x)
```

```
For GAME_CLUSTER 0, x = 651, is: 0.5606758832565284
For GAME_CLUSTER 1, x = 385, is: 0.6155844155844156
For GAME_CLUSTER 2, x = 386, is: 0.33419689119170987
For GAME_CLUSTER 3, x = 225, is: 0.6755555555555556
For GAME_CLUSTER 4, x = 709, is: 0.48801128349788436
For GAME_CLUSTER 5, x = 498, is: 0.5602409638554217
For GAME_CLUSTER 6, x = 672, is: 0.3794642857142857
```

- PCA - continuous variables:



- MCA - for categorical variables:



- **The guiding principles in creating new variables:**

1. Creating new variables that will contain as much information as possible from other variables with the understanding that most of the variables will be removed later when the feature is selected.
2. Creating new variables according to the conclusions we discovered and patterns we found.

Models

Feature Selection:

- Handling Categorical Nominal Variables:

  Converted categorical nominal variables into dummy variables to represent their categories numerically, making them suitable for modeling.

- Feature Selection Methods:

- **Two random noise variables were created** and added to the feature selection process to estimate the quality of the selected variables.

  1. **Univariable Analysis:** Examined each variable individually Vs target variable, using statistical tests - p-values to assess their predictive power.
  2. **Multivariable Analysis:** Analyzed combinations of variables to understand their joint predictive ability, utilized multiple models (LASSO, Ridge, ElsticNet, SVM, Random Forest, GBM, Decision Trees and ADABoost) to assess variable importance and relevance

- Voting and Summarization of Scores:

  Each model assigned a score of 0 or 1 to each variable:

  0: The variable does not contribute significantly to prediction according to the model.
  1: The variable contributes significantly to prediction according to the model.

  Univariate Analysis with P-values:

  Conducted statistical tests (such Sperman and chi-square tests) for each variable individually.
  If p-value < 0.05: Assigned a score of 1 (indicating the variable is statistically significant and contributes to prediction).
  Else: Assigned a score of 0 (indicating the variable is not statistically significant).

- Final Selection:

  Selecting the variables with the highest score in voting, domain knowledge and running a number of iterations to finding the combination of variables that achieve the best results.

  Identify a combination of 25 variables that consistently improve model performance across multiple iterations.

  Ensure the selected variables align with domain knowledge and contribute meaningfully to the model's predictive capability.

Standard Scaling:

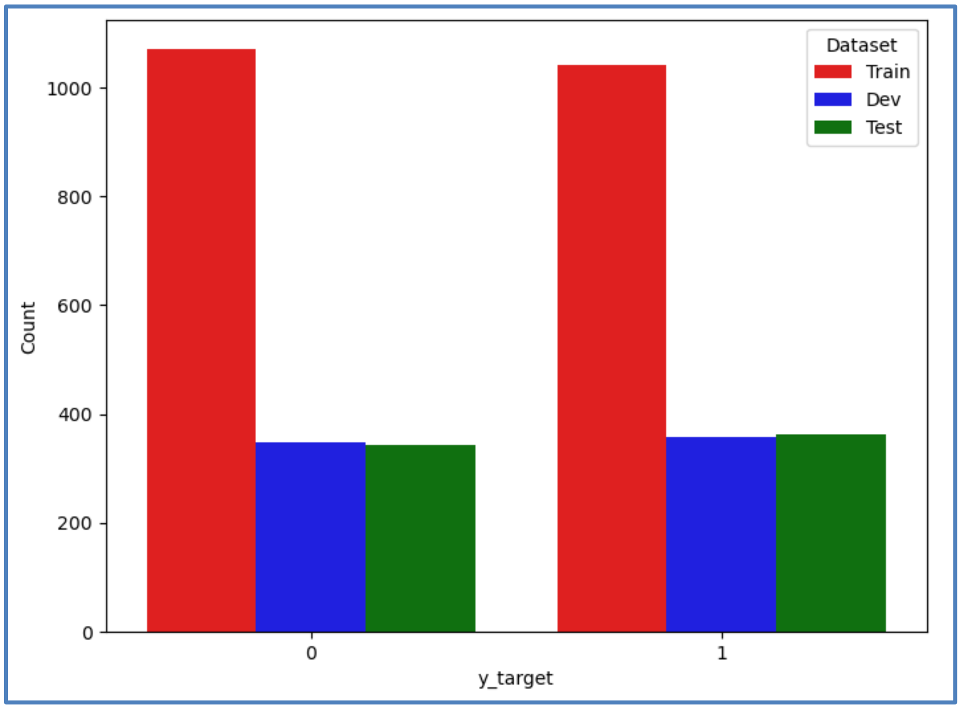
- Applying standard scalar and min-max scaling to normalize the continuous variables did not improve the model's performance and therefore not implemented, based on the project's goal to preserve the fidelity of the data with minimal manipulation, as well as the understanding that tree-based models are less sensitive to outliers and variable scales.

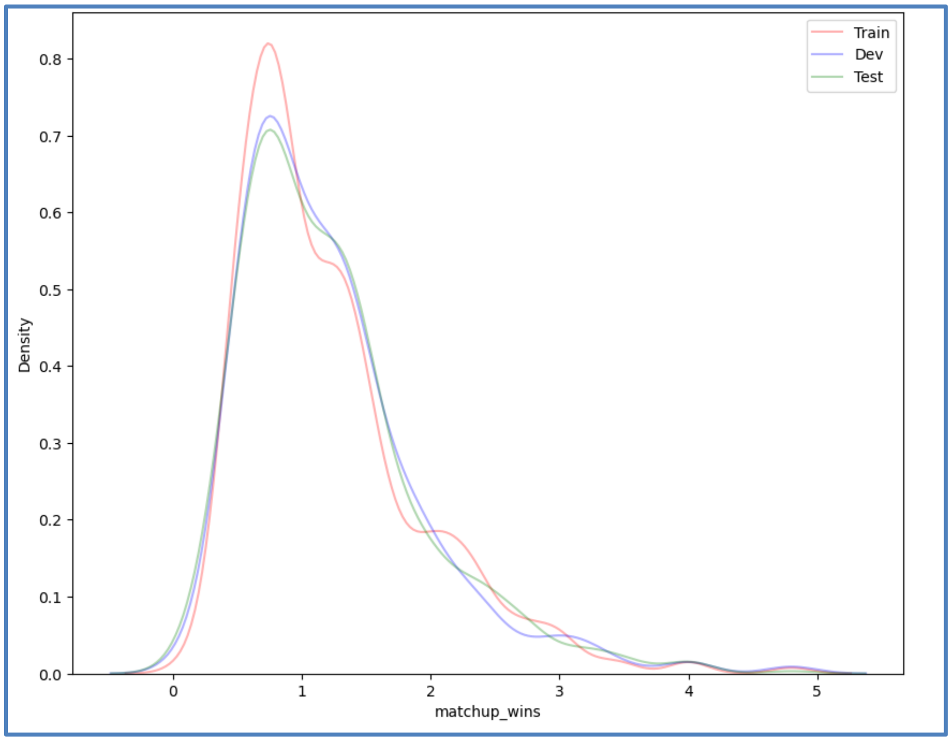
Data Division:

- Data Partitioning:
  1. Train Set: 60% of the data used for model training.
  2. Development Set: 20% of the data used for model validation and parameter tuning.
  3. Test Set: 20% of the data held out for final model evaluation.

- Distribution Comparison Test:
  1. Continuous Variables (Kolmogorov-Smirnov test.
  2. Categorical Variables (Chi-square Test).

- To ensure that the data is evenly distributed across the train, development, and test sets, we performed distribution comparison tests for all the variables selected for the model. Each variable was tested against itself across the different data sets (train, development, test) to confirm that its distribution remained consistent.
  1. Train vs. Development
  2. Train vs. Test
  3. Development vs. Test





Balancing Data:

- The data was balanced, there was no need to balance the data.

Model Evaluation Metrics + Model selection:

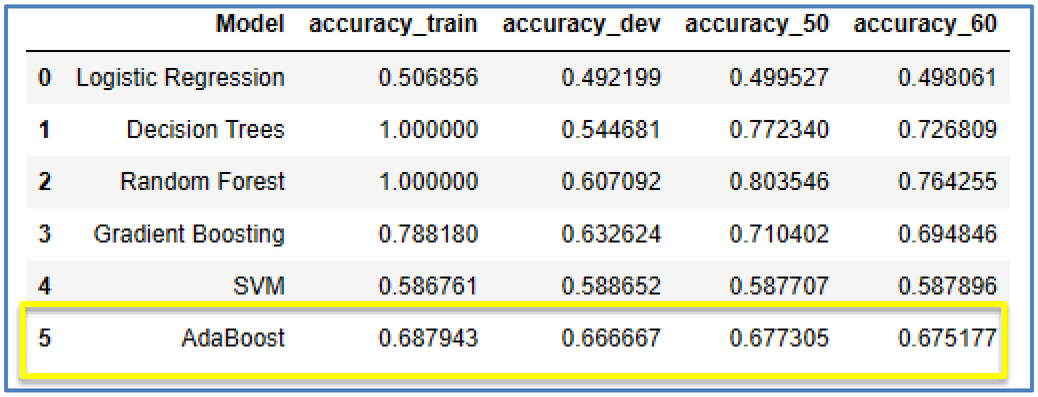
- **Target Variable:** Predicting a binary outcome, will the team win/lose the game, the results - win – 1 or a loss - 0.
- **Classification Models**: We used classification models because our target variable is binomial, meaning it has two possible outcomes (e.g., win or loss).
- **Evaluation Metric - Accuracy**: Accuracy was chosen as the primary metric for evaluating model performance. It measures the percentage of correct predictions out of all predictions made by the model.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

1. **Binary Nature of the Task**: Since the task is to predict win or loss, overall prediction accuracy is the most straightforward and relevant measure of success.
2. **Symmetry in Errors**: In our context, we are not more concerned about false positives (predicting a win when it is a loss) or false negatives (predicting a loss when it is a win). Both types of errors are considered equally significant, there is

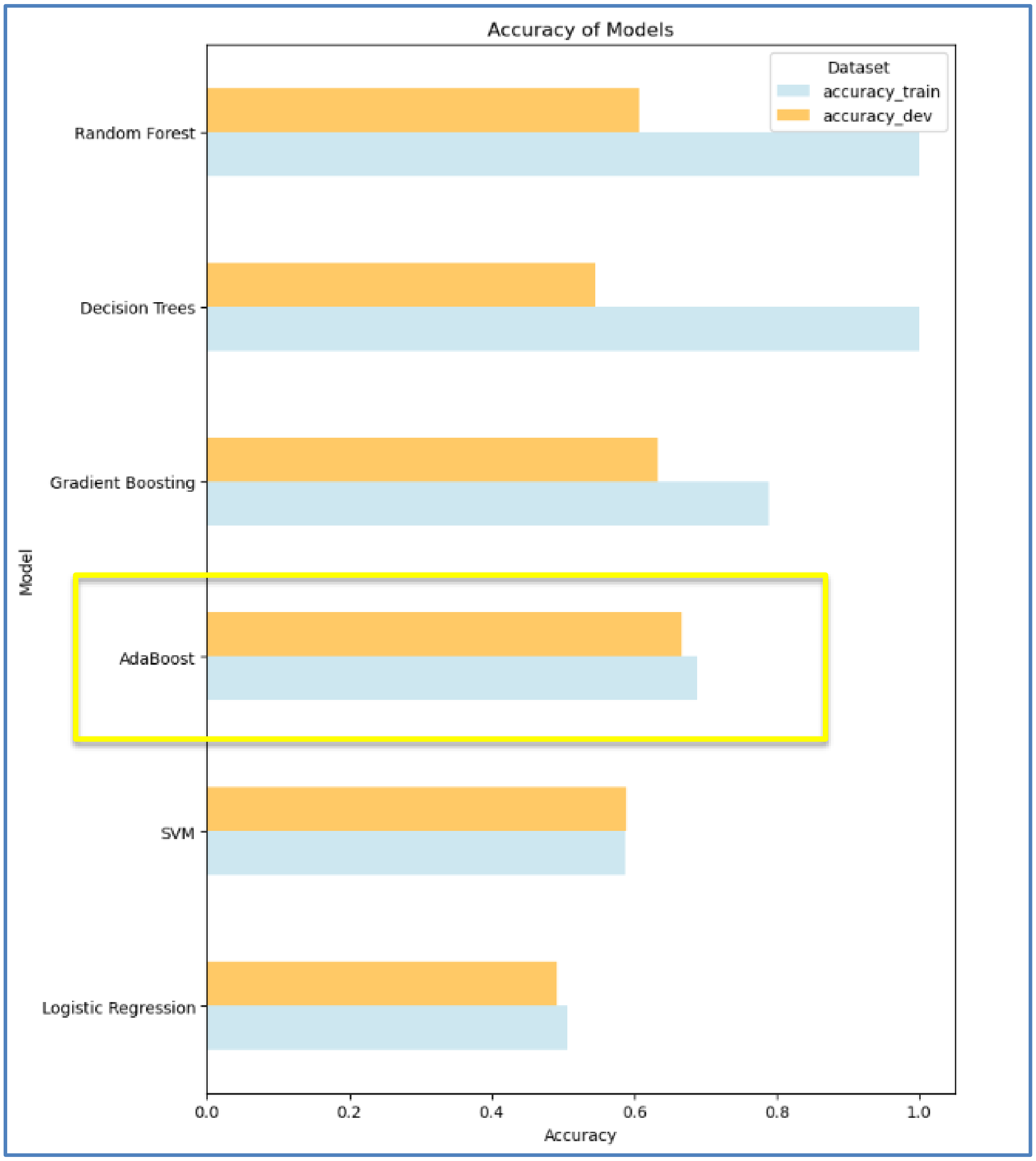
no impact in real life from incorrectly predicting a win as a loss or a loss as a win. Both errors are simply mistakes without consequences. Thus, minimizing overall errors (as measured by accuracy) is the primary goal.

3. **Model Sensitivity**: Our preliminary analysis indicated that our models are less sensitive to outliers and the scale of variable values. Hence, accuracy remains a reliable metric without being adversely affected by these factors.
4. **Maximizing Accuracy**: The primary criterion for choosing the best model was the maximum accuracy value. We sought to identify the model that provided the highest percentage of correct predictions.
5. **Consistency between Sets**: It was important to ensure that the accuracy was stable between the train and test sets. This consistency indicates that the model generalizes well and is not overfitting to the training data.
6. **Weighted Averages**: We also compared the weighted averages of accuracy to ensure there was no significant disparity, which helps in confirming the robustness of the model across different data splits.



	Model	accuracy_train	accuracy_dev	accuracy_50	accuracy_60
0	Logistic Regression	0.506856	0.492199	0.499527	0.498061
1	Decision Trees	1.000000	0.544681	0.772340	0.726809
2	Random Forest	1.000000	0.607092	0.803546	0.764255
3	Gradient Boosting	0.788180	0.632624	0.710402	0.694846
4	SVM	0.586761	0.588652	0.587707	0.587896
5	AdaBoost	0.687943	0.666667	0.677305	0.675177

Model accuracy dev - 66.6667%, during this process, additional data was collected from external sources, and many iterations were run to find the best combination of variables for optimal results. New data on starters, bench players, audience, team average age, etc., was collected. While data on starters and bench players slightly improved the model, data on audience, stadium, and average age were not relevant. Creating new features from the selected features and new insights, such as the number of wins needed to win the series and series advantage, led to overfitting.



- **Chosen Model: ADA Boost**

- Stability: Demonstrated stable performance between the training and development sets.
- Highest Accuracy: Achieved the highest accuracy score on the development set.

Final Decision: ADA Boost was selected as the final model for deployment due to its combination of stability and top performance, ensuring reliable and accurate predictions.

Model Improvement

- <u>Hyperparameter Tuning for Optimal Model Performance:</u>

  To enhance the performance of our ADA Boost model, we implemented hyperparameter tuning. This process involves running a function to find the optimal set of parameters that maximize the model's accuracy and stability.

- <u>Hyperparameter Tuning:</u>

- **Goal**: To find the set of parameters that optimize the ADA Boost model's performance on the development set.
- **Method**: Used techniques such as Grid Search Cross-Validation and Randomized Search Cross-Validation.
- **Randomized Search**: randomly samples a fixed number of hyperparameter combinations from the hyperparameter space.

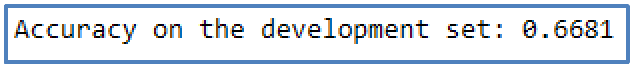
```
{'algorithm': 'SAMME.R',
 'base_estimator': RandomForestClassifier()
, 'learning_rate': 0.9997,
 'n_estimators': 20,
 'random_state': 45}
```

- **Manual Fine-Tuning**: After obtaining the best parameters from automated hyperparameter tuning, we conducted additional manual fine-tuning to further refine the model's performance.

```
Best Parameters Used in the AdaBoost Model:
algorithm: SAMME.R
learning_rate: 1
n_estimators: 51
```

- <u>Result:</u>

- **Minor improvement after the Hyperparameter Tuning – 0.1433%.**



```
Accuracy on the development set: 0.6681
```

- **Model accuracy dev - 66.81%**

Analysis of Model Results (train):

- **Each Game Considered Twice**: The model receives each game twice, once from the perspective of team X and once from the perspective of team Y.

o <u>Stability of the Model:</u>

High Stability: The model demonstrated consistent performance between the training and development stages across various factors such as game stages, years, rankings, home/away games, game serues and etc.

o <u>Lower Ratings (6, 7, 8, 9) vs. Model Accuracy:</u>

1. Accuracy for games won by lower ratings is lower compared to the overall model accuracy.
2. Games predicted as 1 (victory) align closely with the model's overall accuracy.
3. Games predicted as 0 (loss) have approximately 75% accuracy.

o <u>Games Seen from Both Teams' Perspectives</u>**:**

- **60% of Games**: In 60% of the games, where the model has input from both teams' perspectives. The accuracy for these games is consistent with the general model accuracy.
- For 67% of the games where the model has input from both teams' perspectives, the model predicts a win for team X and a loss for team Y (or vice versa). The accuracy for these predictions is approximately 77.5%.
- For 33% of Games, where the model has input from both teams' perspectives, the model predicts the same outcome (either win-win or loss-loss) for both teams. The accuracy for these predictions drop to 50%.

o <u>Games Seen from One Team's Perspective</u>**:**

- **40% of Games**: In 40% of the games, the model has seen the game from only one team perspective. The accuracy for these games is also consistent with the general model accuracy.

The average difference was tested under various scenarios: when the model was wrong for both teams in the same game, when it was right for both teams in the same game, when it predicted the same result for both teams in the same game, and for all the training data. Negligible differences were found in all these cases.

- Key Areas for Further Investigation and Model improvements:

1. **Games Won by Lower Ratings (6, 7, 8, 9) and Model Predictions**: Investigate cases where lower-rated teams win despite the model predicting a loss (prediction = 0). This discrepancy could provide insights into factors not captured by the model.
2. **The same predictions for both teams in the same game**: Explore games where the model predicts the same outcome for both teams. Understanding why these predictions are less accurate (50% accuracy) could highlight areas for model improvement or additional feature engineering.

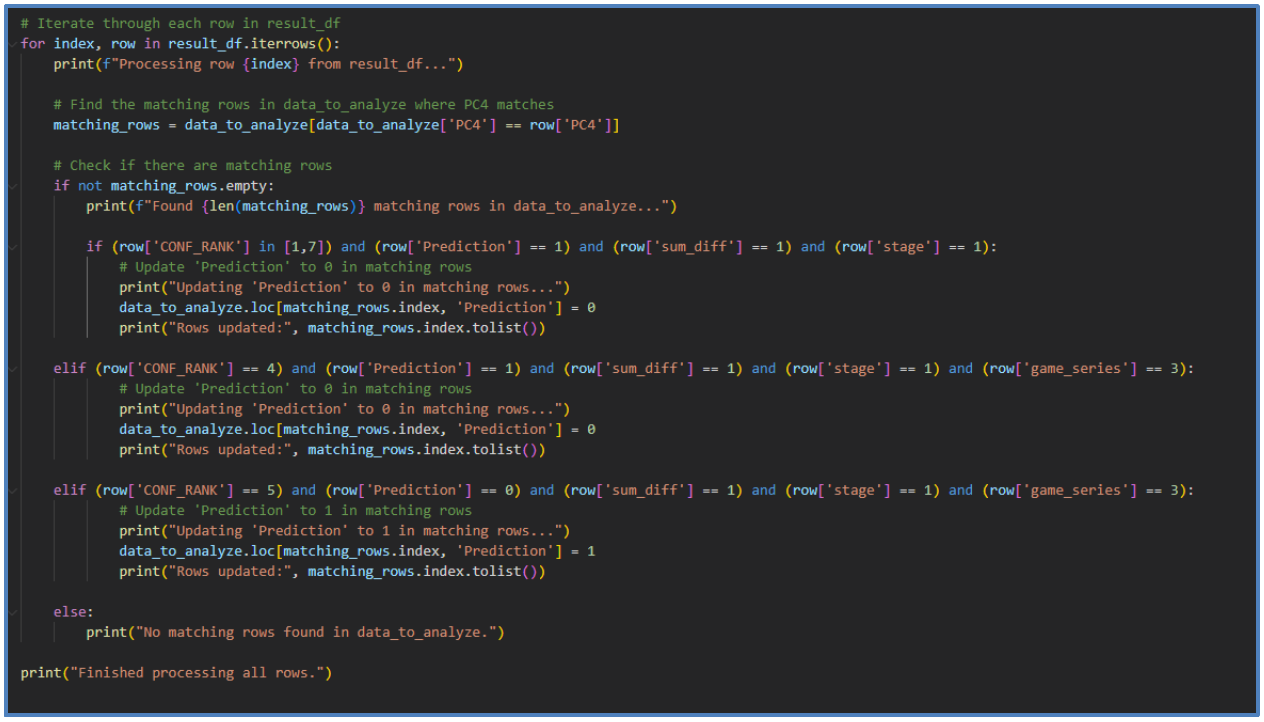
Model Improvments 2:

- First Stage Rankings 1 and 7:

- Consistent Prediction of Win (1) for Both Teams: When the model predicts a win (1) for both teams in the same game, and these teams are ranked 1 and 7, there is more than a 50% probability that both teams lose.

- First Stage Rankings 4 and 5, Third Game of the Series:

- Consistent Prediction of Win or Loss (1 or 0) for Both Teams: When the model predicts same result for both teams in the same game, and these teams are ranked 4 and 5 in the third game of the series, there is more than a 50% probability that the team ranked 4 loses.



```python
# Iterate through each row in result_df
for index, row in result_df.iterrows():
    print(f"Processing row {index} from result_df...")

    # Find the matching rows in data_to_analyze where PC4 matches
    matching_rows = data_to_analyze[data_to_analyze['PC4'] == row['PC4']]

    # Check if there are matching rows
    if not matching_rows.empty:
        print(f"Found {len(matching_rows)} matching rows in data_to_analyze...")

        if (row['CONF_RANK'] in [1,7]) and (row['Prediction'] == 1) and (row['sum_diff'] == 1) and (row['stage'] == 1):
            # Update 'Prediction' to 0 in matching rows
            print("Updating 'Prediction' to 0 in matching rows...")
            data_to_analyze.loc[matching_rows.index, 'Prediction'] = 0
            print("Rows updated:", matching_rows.index.tolist())

    elif (row['CONF_RANK'] == 4) and (row['Prediction'] == 1) and (row['sum_diff'] == 1) and (row['stage'] == 1) and (row['game_series'] == 3):
            # Update 'Prediction' to 0 in matching rows
            print("Updating 'Prediction' to 0 in matching rows...")
            data_to_analyze.loc[matching_rows.index, 'Prediction'] = 0
            print("Rows updated:", matching_rows.index.tolist())

    elif (row['CONF_RANK'] == 5) and (row['Prediction'] == 0) and (row['sum_diff'] == 1) and (row['stage'] == 1) and (row['game_series'] == 3):
            # Update 'Prediction' to 1 in matching rows
            print("Updating 'Prediction' to 1 in matching rows...")
            data_to_analyze.loc[matching_rows.index, 'Prediction'] = 1
            print("Rows updated:", matching_rows.index.tolist())

    else:
        print("No matching rows found in data_to_analyze.")

print("Finished processing all rows.")
```

- Calibration:

- **Context and Model Behavior - ADA Boost Model**: The chosen model is ADA Boost, which assigns weights to samples based on the model's ability to predict them correctly. This weighting can lead to probability estimates that are not fully calibrated and may not reflect the true likelihood of an outcome.

- **Problem Statement - Inaccuracy of Probabilities**: Due to the inherent weighting mechanism in ADA Boost, the predicted probabilities might not accurately represent the true probability of an event. This discrepancy is especially noted in games where the model provides the same prediction for both teams.

- **Purpose of Calibration:**

1. To adjust the predicted probabilities from the ADA Boost model so that they better reflect the true likelihood of the outcomes.
2. To ensure that the model's probability estimates are reliable and provide a clearer picture of the predictions.

- **Comparison of Calibration and Probability Results:**
  We compared the calibration results to the probability results in scenarios where the model predicted the same outcome for both teams in the same game. The results were divided into three categories:

Game	Team	Prediction	Probability_ Before_Calibration	Probability_ After_Calibration	y_target	Correct	Group	stage
XXX	X	1	0.88	0.904258678	1	0	A	1
XXX	Y	1	0.84	0.886871262	0	1	A	1
YYY	X	0	0.23	0.108694614	0	0	C	1
YYY	Y	0	0.12	0.111664865	1	-1	C	1
ZZZ	X	1	0.85	0.864994266	0	1	B	1
ZZZ	Y	1	0.77	0.684165985	1	0	B	1

1. Full Calibration Match – Group A:
   - Consistent trend where both teams have similar trends in calibration and probability values - same conclusion.
2. Partial Calibration Match – Group B:
   - One team shows calibration values greater than probability values, while the second team shows calibration values smaller than probability values, or vice versa.
   - Despite the difference in the trend (higher or lower) between calibration and probability, the conclusion remains consistent across both teams - same conclusion.
3. No Calibration Match – Group C:
   - The calibration and probability trends are inconsistent across both teams.

```
# Create an empty list to store the updated rows
updated_rows = []

# Iterate over the rows in pairs
for i in range(0, len(cal), 2):
    if i + 1 < len(cal):
        row0 = cal.iloc[i].copy()
        row1 = cal.iloc[i + 1].copy()

        # Check if row0 and row1 have the same trend before and after calibration
        same_trend_row0 = row0['Probability_Before_Calibration'] > row0['Probability_After_Calibration']
        same_trend_row1 = row1['Probability_Before_Calibration'] > row1['Probability_After_Calibration']

        if ((row0['Probability_Before_Calibration'] > row1['Probability_Before_Calibration']) and
            (row0['Probability_After_Calibration'] > row1['Probability_After_Calibration'])) or \
           ((row0['Probability_Before_Calibration'] < row1['Probability_Before_Calibration']) and
            (row0['Probability_After_Calibration'] < row1['Probability_After_Calibration'])):
                if (same_trend_row0 == same_trend_row1):
                    row0['Group'] = 'A'
                    row1['Group'] = 'A'
                else:
                    row0['Group'] = 'B'
                    row1['Group'] = 'B'
        else:
            row0['Group'] = 'C'
            row1['Group'] = 'C'

        # Append the updated rows to the list
        updated_rows.append(row0)
        updated_rows.append(row1)

# Create a new DataFrame with the updated rows
updated_cal = pd.DataFrame(updated_rows)

# Print the updated DataFrame
updated_cal
```

This categorization helps in understanding the relationship between calibration and probability values for the model's predictions across different games.

<u>Conclusions:</u>

- <u>Conditions for Group 'A' and 'B' in stage 2 and 3:</u>
    - The predictions are opposite to the trend.

- <u>Conditions for Group 'B' in stage 4:</u>
    - The predictions are opposite to the trend.

```python
# Create an empty list to store the updated rows
updated_rows = []
# Iterate over the rows in pairs
for i in range(0, len(updated_cal), 2):
    if i + 1 < len(updated_cal):
        row0 = updated_cal.iloc[i].copy()
        row1 = updated_cal.iloc[i + 1].copy()

        for index, row in updated_cal.iterrows():
            group = row['Group']
            stage = row['stage']

            if (group in ['A','B']) and (stage in [2, 3]):
                if row0['Probability_Before_Calibration'] > row1['Probability_Before_Calibration']:
                    row0['Prediction'] = 0
                    row1['Prediction'] = 1
                else:
                    row0['Prediction'] = 1
                    row1['Prediction'] = 0

            elif (group == 'B') and (stage == 4):
                if (row0['Probability_Before_Calibration'] > row1['Probability_Before_Calibration']):
                    row0['Prediction'] = 0
                    row1['Prediction'] = 1
                else:
                    row0['Prediction'] = 1
                    row1['Prediction'] = 0

        # Append the updated rows to the list
        updated_rows.append(row0)
        updated_rows.append(row1)
# Create a new DataFrame with the updated rows
updated_cal = pd.DataFrame(updated_rows)
# Print the updated DataFrame
updated_cal
```

- Results:
  - **Train accuracy improved by ~ 1.9%.**

When refining model predictions, it is crucial to have substantial data and clear conclusions from analysis and calibration to ensure accurate adjustments. Due to, insufficient amount of data, it was decided not to implement these adjustments on development and test datasets. This approach is just a preliminary direction for future work.

With sufficient data, it is possible to run specialized models on error patterns to improve the results of the general model. Specifically, if there were enough data, models could be run on predictions with similar outcomes for the same game to refine the accuracy of those predictions. This targeted approach could enhance the overall performance of the prediction system by addressing specific areas where the general model may fall short.

- Test data – result:

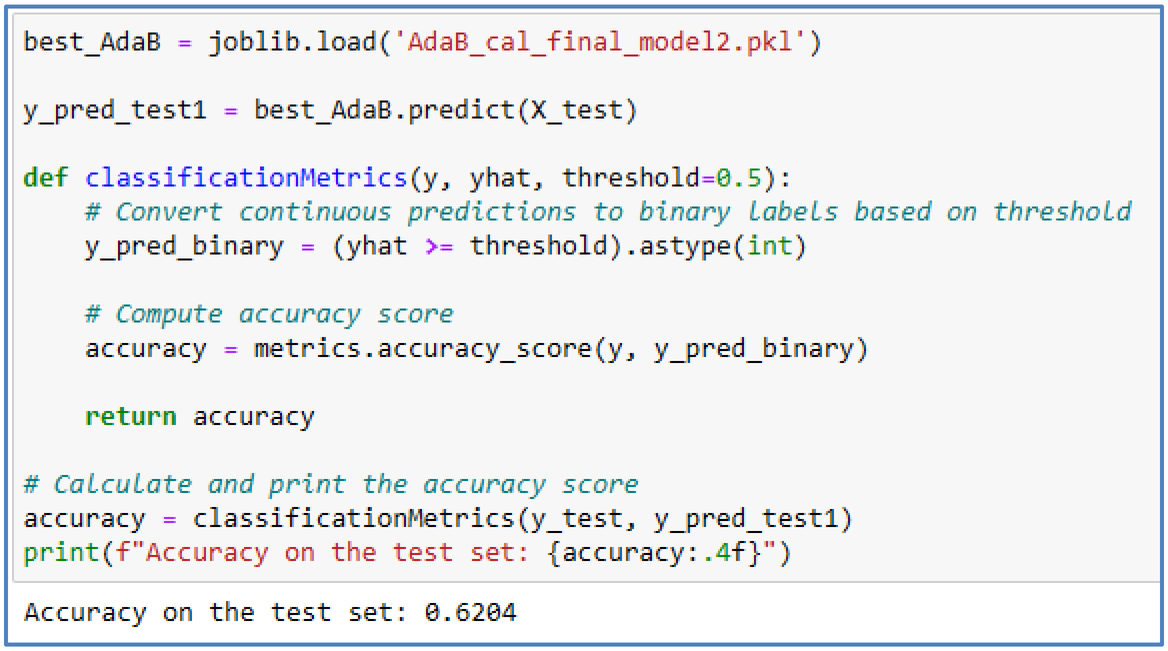
The test data was evaluated using the same model but with three different sets of parameters:

1. Best Dev Parameters: Parameters yielding the best results for the development data.
2. **Grid Search Parameters: Parameters determined by a comprehensive grid search.**
3. Manual Fine-Tuning: Parameters refined manually based on grid search results.

A voting mechanism was employed to determine the final prediction:

- If two or more models predict a 1, the final prediction is 1.
- Otherwise, the final prediction is 0.

The set of parameters providing balanced results across both development and test data was ultimately selected.



```
best_AdaB = joblib.load('AdaB_cal_final_model2.pkl')

y_pred_test1 = best_AdaB.predict(X_test)

def classificationMetrics(y, yhat, threshold=0.5):
    # Convert continuous predictions to binary labels based on threshold
    y_pred_binary = (yhat >= threshold).astype(int)

    # Compute accuracy score
    accuracy = metrics.accuracy_score(y, y_pred_binary)

    return accuracy

# Calculate and print the accuracy score
accuracy = classificationMetrics(y_test, y_pred_test1)
print(f"Accuracy on the test set: {accuracy:.4f}")
```

```
Accuracy on the test set: 0.6204
```

- **Model accuracy test - 62.04%**