Analyzing Baseball Performance for University of Maryland Terrapins

Team 4

Isha Tyagi

Zidong Liu

Piyali Bedagkar

Gnapika Komaragiri

TABLE OF CONTENTS

INTRODUCTION

"Diamond Insights: Enhancing Performance in University of Maryland Terrapins Baseball" is a pioneering endeavor leveraging data-driven strategies to elevate collegiate baseball excellence. Through meticulous analysis of longitudinal impact reports and predictive modeling, our mission is to uncover correlations between game features and outcomes.

MISSION STATEMENT

- **Identifying Correlated Features**: The mission is to meticulously analyze historical game data to identify features that exhibit significant correlations with game outcomes.
- **Optimizing Features to Improve Impacts and Outcomes**: Our mission is to leverage the understanding gained from correlated features to optimize key aspects of game preparation and execution.
- **Understanding Future Game Improvement**: Our ultimate mission is to gain a comprehensive understanding of which features hold the most promise for improving future games.

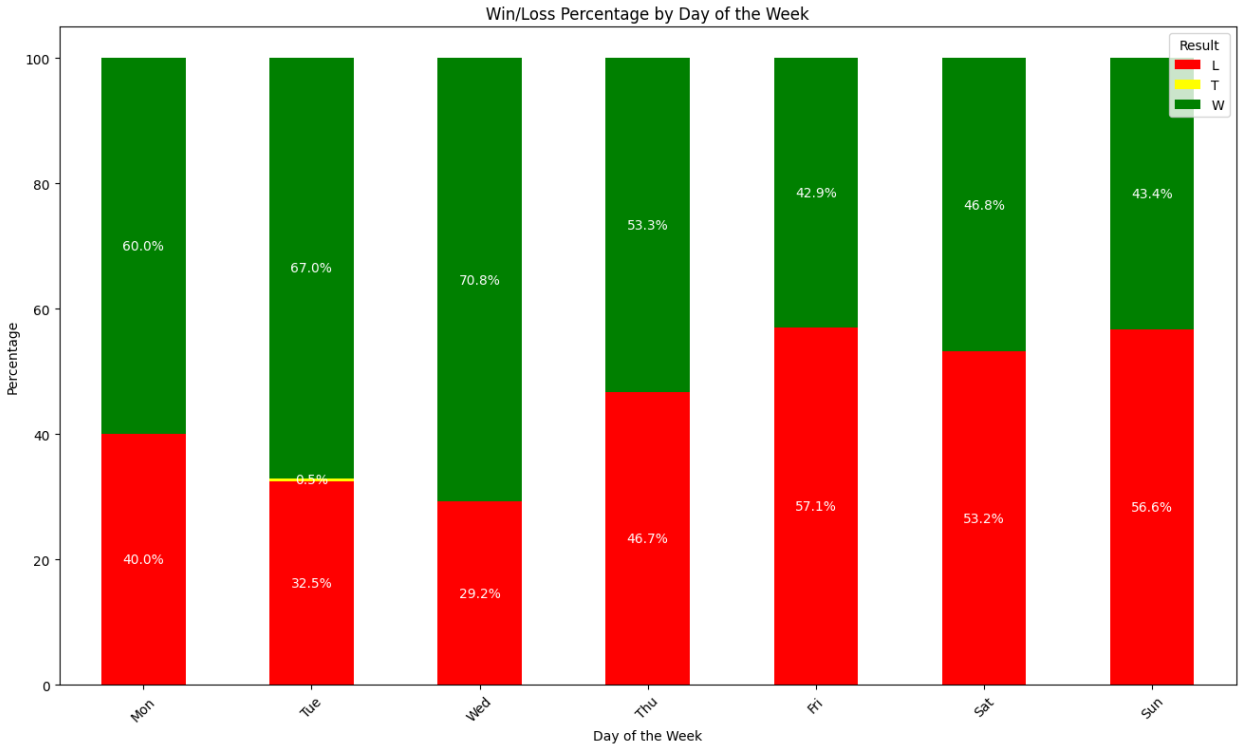
MISSION OBJECTIVES

- To identify patterns and correlations between game outcomes and various features.
- Conduct Exploratory Data Analysis (EDA) reports.
- Generate statistical analyses matching the Year worksheet.

➢ Develop analytical models with reports.

EXPLORATORY DATA ANALYSIS REPORT

1. Win/Loss Percentage by Day of the Week



Detailed Explanation of the Plot

Visual Representation: The graph generated provides a visual representation of the trends in the average scores of 'Terps' and their opponents throughout the week.

X-axis (Categorical):

- This axis represents the days of the week. It includes Monday, Tuesday, Wednesday, Thursday,Friday, Saturday, and Sunday.

Y-axis (Quantitative):

- This axis shows the percentage. It goes from 20% to 100%.

Plot Details:

- Title:
  - The title of the plot is "Win/Loss Percentage by Day of the Week)", indicating the content and time range of the data.
- Axis Labels:
  - The x-axis is labeled" Day of the week" to indicate the time dimension of the data.
  - The y-axis is labeled "Percentage" to indicate the value being measured.
- Bars:
  - Each bar on the X-axis represents a day of the week.
  - The height of each bar represents the win and loss percentage for that day.There are three colors in the graph: green, red, and yellow.
  - The legend at the top right corner of the graph indicates that green represents wins, red represents losses, and the yellow represents ties.

Data Representation:

- The graph shows the win percentages for each day of the week.

Insights:

- Wednesday has the highest win percentage at 70.8%.
- Sunday has the lowest win percentage at 29.2%.
- Win percentages are higher on weekdays than weekends.

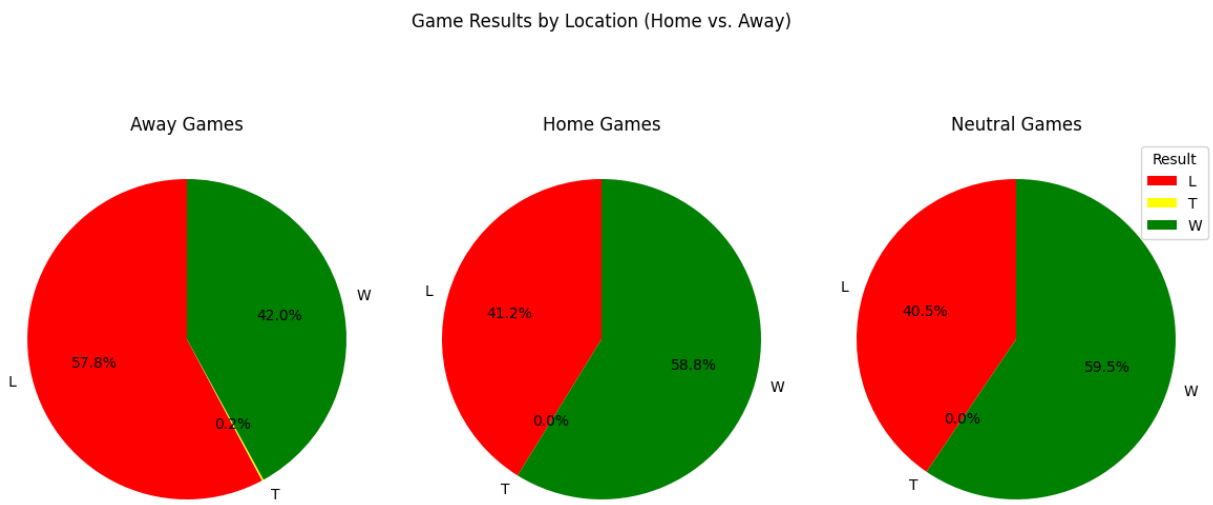
VISUALIZATION

This image displays the win/loss percentages of the University of Maryland Terrapins baseball team for each day of the week. Each bar represents a day, with different colors indicating different game outcomes:

- Green (W) represents the percentage of wins.
- Red (L) represents the percentage of losses.
- Yellow (T) represents the percentage of ties (there are no ties in this chart).

The chart is titled "Win/Loss Percentage by Day of the Week". The percentage labels on each bar show the specific win and loss proportions for that day. The y-axis represents the percentage, while the x-axis represents the days of the week, from Monday to Sunday.By this graph representation we can say that Win percentages are higher on weekdays than weekends.

2. Game results by the Location



Detailed Explanation of the Plot

Visual Representation: The graph generated provides a visual representation of the game results by location (Home vs. Away)

Plot Details:

Title:

- The title of the plot is" Game Results by Location (Home vs. Away)"

Slices:

- ➢ The pie chart is divided into three slices representing the three locations: Home, Away, and Neutral.
- ➢ The slices are colored Green for win , red for loss, and yellow for tie.

Data Representation:

- ➢ Each slice represents the percentage of games won/lost/tied in that location.
- ➢ Percentages are displayed next to each slice in white text.

Insights :

- ➢ There are higher number of losses in away games.
- ➢ The highest number of wins recorded for neutral games.

VISUALIZATION

The pie-chart above indicates that the University of Maryland Terrapins Baseball team performs better at neutral games compared to away or home locations, with the highest number of wins recorded for neutral games. The team also has a higher number of losses in away games compared to home and neutral games.Ties are very rare, with only one tie recorded in away games and none in home or neutral games.

STAT REPORT:

#1 Average Score Trend

Output Description: The statement **print(average_scores)** will display a dataframe in the console. This dataframe contains the average scores of the 'Terps' team and their opponents ('Oppnt') for each year present in the dataset.

Structure of the DataFrame:

- Index: The index of the data frame will be the years, which are extracted from the 'Date' column in the original dataframe. Each row corresponds to a different year.
- Columns: There will be two columns in the dataframe:
    - Terps: This column contains the average score of the 'Terps' team for each year.
    - Oppnt: This column contains the average score of the opponents for each year.

Example Output: The printed dataframe might look something like this:

Year	Terps	Oppnt
1999	7.041667	7.583333
2000	6.617021	6.808511
2001	6.520000	8.780000
2002	8.603774	6.962264
2003	5.358491	7.528302
2004	5.857143	6.946429
2005	5.701754	6.368421
2006	5.017857	6.321429
2007	5.767857	5.946429
2008	6.500000	6.214286
2009	6.166667	6.462963
2010	4.517857	8.053571
2011	4.375000	5.946429
2012	4.928571	3.803571
2013	5.563636	4.618182
2014	5.333333	4.142857
2015	5.969697	4.196970
2016	5.000000	4.543860
2017	6.114754	4.721311
2018	5.018519	5.796296
2019	5.620690	6.327586
2020	7.400000	4.733333
2021	6.604167	5.312500
2022	9.225806	5.596774
2023	9.174603	6.841270

This shows the average scores per year, where you can see the average score for both 'Terps' and their opponents for each specific year.

Detailed Explanation of the Plot

Visual Representation: The plot generated by the script provides a visual representation of the trends in the average scores of 'Terps' and their opponents from 1999 to 2023.

X-axis (Horizontal Axis):

- Represents the years from 1999 to 2023.
- Each tick on the x-axis corresponds to a different year within this range.

Y-axis (Vertical Axis):

- Represents the average scores.
- Each tick on the y-axis corresponds to a different score value.

Plot Details:

- Two Lines:
    - One line represents the 'Terps' average scores over the years, plotted with circle markers ('o').
    - Another line represents the opponents' average scores over the years, plotted with 'x' markers.
- Title:
    - The title of the plot is "Average Scores Trend (1999-2023)", indicating the content and time range of the data.
- Axis Labels:
    - The x-axis is labeled "Year" to indicate the time dimension of the data.
    - The y-axis is labeled "Average Score" to indicate the value being measured.
- Legend:
    - A legend is included to differentiate between the 'Terps Average Score' line and the 'Opponent Average Score' line. This helps the viewer understand which line corresponds to which team's scores.
- Grid:
    - A grid is added to the plot to enhance readability. The grid helps in tracing the values from the lines to the axes more easily.

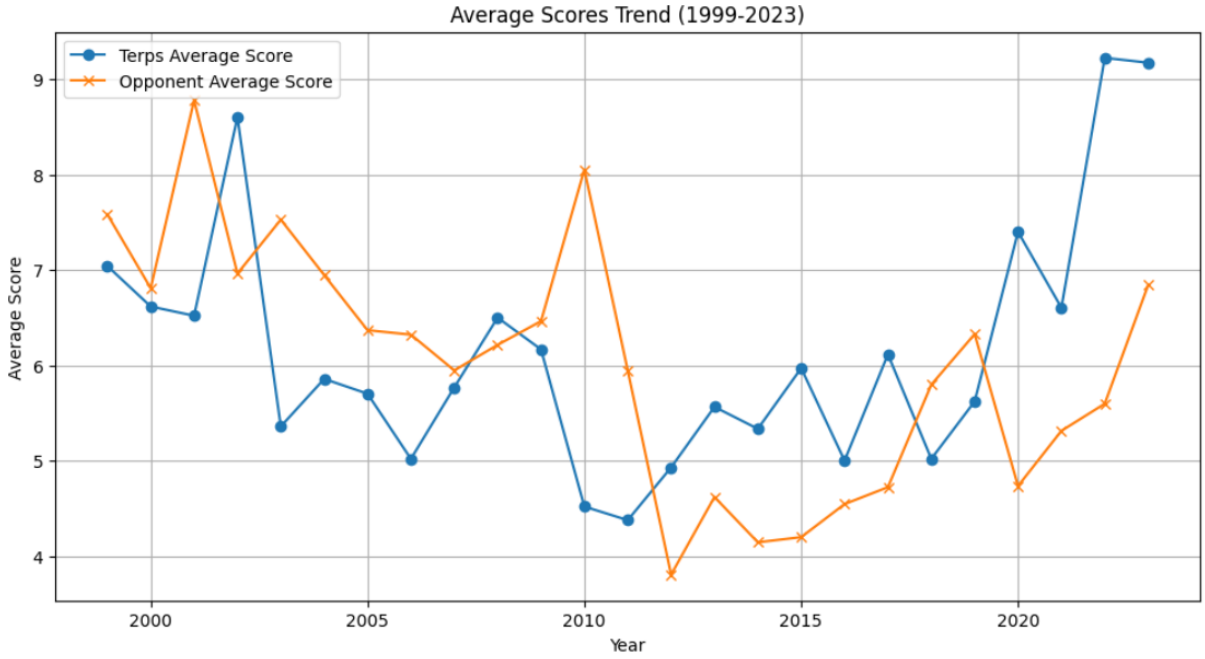
Overall Purpose of the Code

The overall purpose of this code is to analyze and visualize the trends in average scores for the 'Terps' team and their opponents over a specified period (1999-2023). By converting the dates, extracting years, calculating average scores, and plotting the data:

1. Analysis:
    - The code calculates and displays the average scores per year, which helps in understanding how the scores have varied annually.
2. Visualization:
    - The plot provides a clear visual representation of trends, making it easier to compare the performance of the 'Terps' and their opponents over time.
    - Trends such as improvements, declines, or stability in performance can be easily identified.

This kind of analysis and visualization is particularly useful for sports analysts, coaches, and fans who are interested in the performance trends of a team over multiple seasons. It provides insights into how a team's scoring performance has evolved and can help in making data-driven decisions or assessments.

The Graph looks like below :-



The line graph above represents the average scores of Terps' and the Opponents.The Terps' average score appears to be generally higher than their Opponents' average score throughout the time period. Both teams' average scores seem to fluctuate a bit from year to year, but there is no clear upward or downward trend over time. In 2023, the Terps' average score was 5 and their opponents' average score was 6..

#2 Win Rate Over Time

Data Processing Output

1. Printed DataFrame

The print(yearly_results[['Year', 'Win_Rate']]) statement outputs a DataFrame that looks like this

Result	Year	Win_Rate
0	1999	0.427
1	2000	0.426
2	2001	0.300
3	2002	0.604
4	2003	0.377
5	2004	0.393
6	2005	0.456
7	2006	0.464
8	2007	0.464
9	2008	0.536
10	2009	0.500
11	2010	0.304
12	2011	0.375
13	2012	0.571
14	2013	0.545
15	2014	0.635
16	2015	0.636
17	2016	0.526
18	2017	0.623
19	2018	0.444
20	2019	0.500
21	2020	0.667
22	2021	0.625
23	2022	0.774
24	2023	0.667

Here's what each column represents:

- ➢ Year: This column indicates the year in which the games were played.
- ➢ Win_Rate: This column represents the calculated win rate for each respective year.

Explanation of Win Rate Calculation:

- ➢ The win rate for each year is computed by adjusting the number of wins and ties (if any) and then dividing by the total number of games played.
- ➢ For example, if in a particular year the team won 20 games, lost 10, and tied 5, the adjusted wins would be 20 + 0.5 * 5 = 22.5. If the total number of games is 35, the win rate would be 22.5 / 35 = 0.643, rounded to three decimal places.

This DataFrame allows you to quickly see how the win rate has changed over the years.

Plotting Output

2. Win Rate Over Time Plot

The code produces a plot visualizing the win rate over the years from 1999 to 2023. Here are the components of the plot:

- ➢ Title: "Win Rate Over Time"
  - ○ This gives a clear indication of what the plot is about.
- ➢ X-axis: Labeled as "Year"
  - ○ Represents the years from 1999 to 2023.

- The years are shown horizontally along the bottom of the plot.

➢ Y-axis: Labeled as "Win Rate"

- Represents the win rate, ranging from 0 to 1 (or 0% to 100%).
- The win rates are shown vertically along the left side of the plot.

➢ Data Points:

- Each point on the plot represents the win rate for a specific year.
- Points are marked with circles (marker='o').

➢ Line:

- A line connects the data points (linestyle='-'), making it easier to observe trends over time.

➢ Grid:

- A grid is enabled (plt.grid(True)) to help read values more precisely.

➢ X-axis Labels Rotation:

- The x-axis labels are rotated by 45 degrees (plt.xticks(yearly_results['Year'], rotation=45)) to make them readable.

➢ Layout Adjustment:

- The layout is adjusted (plt.tight_layout()) to ensure everything fits well within the figure size.

Interpretation of the Plot

➢ Trend Observation: The plot allows you to observe how the win rate has evolved over the years. You can identify periods of improvement, consistency, or decline in the team's performance.

➢ Peaks and Troughs: Specific years might stand out as particularly good or bad seasons. These could correlate with changes in team composition, coaching staff, or other factors.

- General Performance: The overall trend might show whether the team's performance has been improving, declining, or remaining stable over the analyzed period.

Based on the provided outcome, we can analyze the trend of the win rates for the UMTerps Baseball team from 1999 to 2023.

Trend Analysis

Early Years (1999-2004)

- 1999-2000: The win rates are quite low, around 0.426-0.427, indicating a struggling period.
- 2001: There's a significant drop to 0.300, marking the lowest point in this period.
- 2002: A notable improvement to 0.604, suggesting a positive change in performance.
- 2003-2004: The win rate drops again to 0.377 and 0.393, respectively, indicating inconsistency in performance.

Mid Years (2005-2014)

- 2005-2007: The win rate shows slight improvement and stabilization around 0.456-0.464.
- 2008: Another improvement to 0.536, indicating a positive trend.
- 2009-2010: Fluctuations occur with win rates of 0.500 in 2009 and a drop to 0.304 in 2010.
- 2011: A slight increase to 0.375, still relatively low.

- ➢ 2012-2013: Significant improvement with win rates rising to 0.571 and 0.545, respectively.
- ➢ 2014-2015: The best performance period so far with win rates of 0.635 and 0.636, indicating strong and consistent performance.

Recent Years (2015-2023)

- ➢ 2016-2017: Continuation of good performance with win rates of 0.526 and 0.623.
- ➢ 2018: A drop to 0.444, showing some inconsistency.
- ➢ 2019-2020: Improvement with win rates at 0.500 and 0.667.
- ➢ 2021-2023: The strongest period overall:
  - 2021: A win rate of 0.625.
  - 2022: The highest win rate of 0.774.
  - 2023: Maintains a high win rate of 0.667.

Interpretation of the Trends

1. Early Struggles (1999-2004):
   - The team had fluctuating and generally low win rates, struggling to maintain consistent performance.
2. Gradual Improvement (2005-2014):
   - From 2005 onwards, there is a trend of gradual improvement, with occasional dips.
   - Significant improvements are observed around 2008 and again from 2012 onwards, leading to peak performances in 2014 and 2015.
3. Strong Recent Performance (2015-2023):
   - The period from 2015 to 2023 shows consistent and strong performance, with only minor fluctuations.

- The team achieved its highest win rates in recent years, peaking in 2022 with a win rate of 0.774.

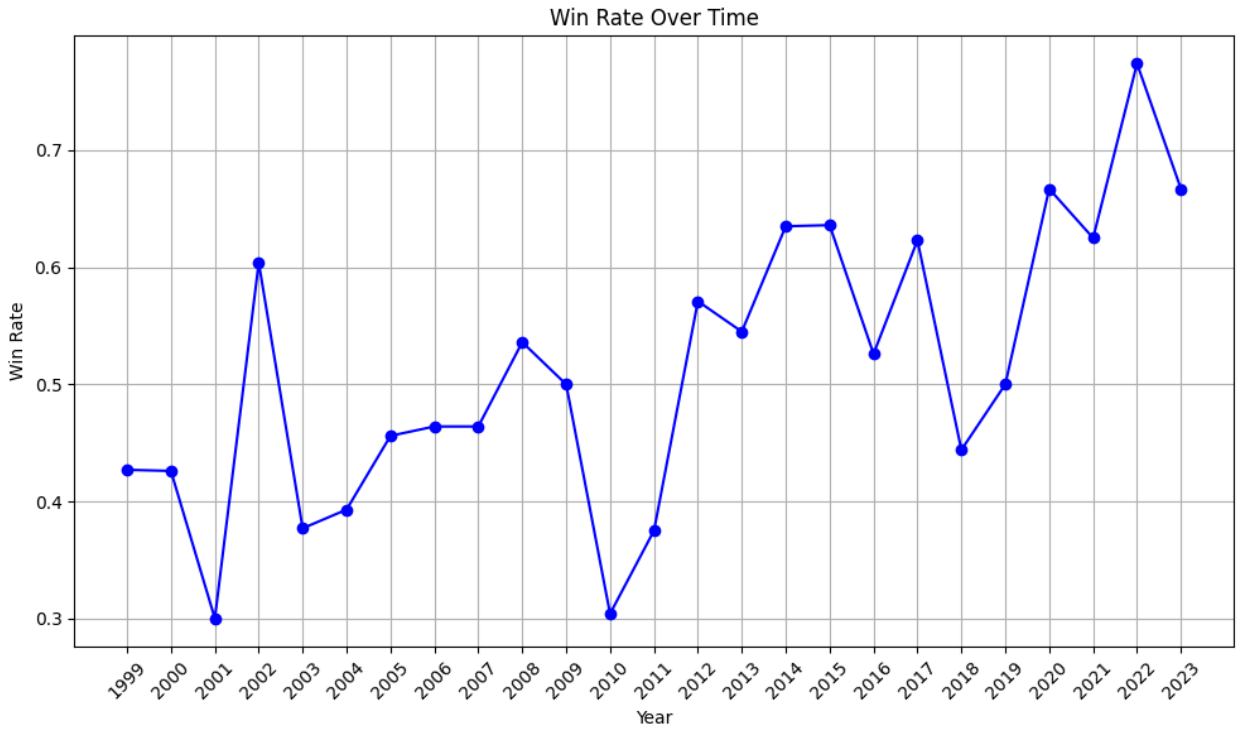
Conclusion

The UMTerps Baseball team's performance shows a clear trajectory of improvement over the 25-year span:

- ➢ Early years were challenging with lower win rates and inconsistency.
- ➢ Middle years marked gradual and significant improvements.
- ➢ Recent years demonstrate the team's peak performance, with high and consistent win rates indicating successful strategies, better team management, or stronger player development.

This detailed analysis helps stakeholders (like coaches, players, and analysts) understand performance trends, make data-driven decisions, and strategize for future seasons. The

combination of the DataFrame and the plot provides both a numerical and a visual summary of the team's performance over time.



The generated graph shows the win rates over time for UMD baseball matches. By analyzing the trends in these win rates, one can assess the team's performance in different environments over the past 25 years.

#3 Win Rate Over Time by Location

The code processes baseball game data for the University of Maryland Terrapins (UMTerps) spanning from 1999 to 2023. The outcome presented in the table compares two sets of data: game results from the 'Game' sheet and season summaries from the 'Year' sheet in an Excel file. The resulting DataFrame includes yearly performance metrics, verifying the accuracy and consistency between the two sources. Let's break down the columns and the information each conveys:

	Year	Overall	WLT	OverallP	Win_Rate	Same
0	1999	20-27-1	20-27-1	0.427	0.427	True
1	2000	20-27	20-27	0.426	0.426	True
2	2001	15-35	15-35	0.3	0.3	True
3	2002	32-21	32-21	0.604	0.604	True
4	2003	20-33	20-33	0.377	0.377	True
5	2004	22-34	22-34	0.393	0.393	True
6	2005	26-31	26-31	0.456	0.456	True
7	2006	26-30	26-30	0.464	0.464	True
8	2007	26-30	26-30	0.464	0.464	True
9	2008	30-26	30-26	0.536	0.536	True
10	2009	27-27	27-27	0.5	0.5	True
11	2010	17-39	17-39	0.304	0.304	True
12	2011	21-35	21-35	0.375	0.375	True
13	2012	32-24	32-24	0.571	0.571	True
14	2013	30-25	30-25	0.545	0.545	True
15	2014	40-23	40-23	0.635	0.635	True
16	2015	42-24	42-24	0.636	0.636	True
17	2016	30-27	30-27	0.526	0.526	True
18	2017	38-23	38-23	0.623	0.623	True
19	2018	24-30	24-30	0.444	0.444	True
20	2019	29-29	29-29	0.5	0.5	True
21	2020	10-5	10-5	0.667	0.667	True
22	2021	30-18	30-18	0.625	0.625	True
23	2022	48-14	48-14	0.774	0.774	True
24	2023	42-21	42-21	0.667	0.667	True

Columns Explained:

- Year:

- Represents the year of the baseball season.

➢ Overall:

- The overall record for the year, directly sourced from the 'Year' sheet.
- Format: "Wins-Losses" or "Wins-Losses-Ties".

➢ WLT:

- The combined record of wins, losses, and ties, calculated from individual game data in the 'Game' sheet.
- Format matches 'Overall', ensuring consistency in how data is presented.

➢ OverallP:

- The overall win percentage for the year from the 'Year' sheet.
- Calculated as wins divided by total games, often rounded to three decimal places.

➢ Win_Rate:

- The win rate calculated from the game data in the 'Game' sheet.
- Uses adjusted wins (wins plus half of ties) divided by total games, rounded to three decimal places.

➢ Same:

- A boolean column indicating whether the 'Overall' record matches 'WLT' and 'OverallP' matches 'Win_Rate'.
- A value of True confirms that the records and win percentages from both sheets are consistent.

Outcome Analysis:

Example Entries:

➢ 1999:

- Overall: "20-27-1" (20 wins, 27 losses, 1 tie)
- WLT: "20-27-1" (Calculated the same as 'Overall')

- OverallP: "0.427" (Winning percentage from 'Year' sheet)
- Win_Rate: "0.427" (Calculated from 'Game' sheet)
- Same: True (Both records and win percentages match)

➢ 2000:

- Overall: "20-27" (20 wins, 27 losses)
- WLT: "20-27" (Calculated the same as 'Overall')
- OverallP: "0.426" (Winning percentage from 'Year' sheet)
- Win_Rate: "0.426" (Calculated from 'Game' sheet)
- Same: True (Both records and win percentages match)

➢ 2023:

- Overall: "42-21" (42 wins, 21 losses)
- WLT: "42-21" (Calculated the same as 'Overall')
- OverallP: "0.667" (Winning percentage from 'Year' sheet)
- Win_Rate: "0.667" (Calculated from 'Game' sheet)
- Same: True (Both records and win percentages match)

Insights:

➢ Consistency: The 'Same' column being True across all rows indicates that the data is consistent between the game logs and the season summaries. This means the game-by-game data correctly aggregates to match the overall season statistics.

➢ Performance Trends: You can observe fluctuations in performance over the years. For example:

- 2002: High performance with a record of "32-21" and a win rate of 0.604.
- 2010: Lower performance with a record of "17-39" and a win rate of 0.304.

- 2022: Exceptional performance with a record of "48-14" and a win rate of 0.774.

Win Rate Calculation:

- ➢ Adjusted Wins: For seasons with ties (e.g., 1999 with "20-27-1"), adjusted wins are calculated as Wins+0.5×Ties. For 1999, adjusted wins are 20+0.5×1=20.5.
- ➢ Win Rate: For 1999, the win rate is 20.5/48 = 0.427.

Conclusion:

The table provides a thorough comparison between individual game data and summarized season records, ensuring data accuracy and consistency. It allows the university or analysts to verify that their records are correctly maintained and reflects the performance trends of the UMTerps baseball team over the years.

The output presents a summary of the University of Maryland Terrapins (UMTerps) baseball team's performance from 1999 to 2023, categorized by game locations (home, away, and neutral sites). Here's a breakdown of the columns and what they represent:

	Year	Home	Home_WLT	Away	Away_WLT	Neutral	Neutral_WLT
0	1999	9-15-0	9-15	11-12-1	11-12-1	0-0-0	NaN
1	2000	11-13	11-13	9-14	9-14	0-0	NaN
2	2001	3-15	3-15	12-20	12-20	0-0	NaN
3	2002	23-5	23-5	9-16	9-16	0-0	NaN
4	2003	13-20	13-20	7-13	7-13	0-0	NaN
5	2004	15-14	15-14	7-20	7-20	0-0	NaN
6	2005	14-16	14-16	12-15	12-15	0-0	NaN
7	2006	16-9	16-9	10-21	10-21	0-0	NaN
8	2007	19-14	19-14	7-16	7-16	0-0	NaN
9	2008	21-12	21-12	9-14	9-14	0-0	NaN
10	2009	21-9	21-9	6-18	6-18	0-0	NaN
11	2010	11-21	11-21	6-18	6-18	0-0	NaN
12	2011	15-16	15-16	6-19	6-19	0-0	NaN
13	2012	21-10	21-10	11-14	11-14	0-0	NaN
14	2013	20-12	20-12	10-13	10-13	0-0	NaN
15	2014	21-7	21-7	13-14	13-14	6-2	6-2
16	2015	16-9	16-9	14-11	14-11	12-4	12-4
17	2016	15-11	15-11	11-14	11-14	4-2	4-2
18	2017	20-3	20-3	9-14	9-14	9-6	9-6
19	2018	13-13	13-13	9-17	9-17	2-0	2-0
20	2019	11-15	11-15	16-11	16-11	2-3	2-3
21	2020	7-2	7-2	2-3	2-3	1-0	1-0
22	2021	16-4	16-4	9-7	9-7	5-7	5-7
23	2022	27-4	27-4	19-7	19-7	2-3	2-3
24	2023	19-9	19-9	19-7	19-7	4-5	4-5

Columns Explained:

1. Year:
    - Represents the year of the baseball season.
2. Home:
    - Indicates the record of games played at the home stadium.
    - Format: Wins-Losses-Ties (e.g., "9-15-0").
3. Home_WLT:
    - Represents the combined wins-losses-ties record for home games.
    - Format: Wins-Losses (e.g., "9-15").
4. Away:
    - Indicates the record of games played away from the home stadium.
    - Format: Wins-Losses-Ties (e.g., "11-12-1").
5. Away_WLT:

- Represents the combined wins-losses-ties record for away games.
- Format: Wins-Losses (e.g., "11-12").

6. Neutral:
   - Indicates the record of games played at neutral sites (neither home nor away).
   - Format: Wins-Losses-Ties (e.g., "0-0-0").
7. Neutral_WLT:
   - Represents the combined wins-losses-ties record for games played at neutral sites.
   - Format: Wins-Losses (e.g., "0-0").

Outcome Analysis:

Example Entries:

1. 1999:
   - Home: "9-15-0" (9 wins, 15 losses, 0 ties)
   - Home_WLT: "9-15" (Combined wins-losses record for home games)
   - Away: "11-12-1" (11 wins, 12 losses, 1 tie)
   - Away_WLT: "11-12" (Combined wins-losses record for away games)
   - Neutral: "0-0-0" (No games played at neutral sites)
   - Neutral_WLT: NaN (No data available for neutral site games)
2. 2022:
   - Home: "27-4" (27 wins, 4 losses)
   - Home_WLT: "27-4" (Combined wins-losses record for home games)
   - Away: "19-7" (19 wins, 7 losses)
   - Away_WLT: "19-7" (Combined wins-losses record for away games)
   - Neutral: "2-3" (2 wins, 3 losses)

- Neutral_WLT: "2-3" (Combined wins-losses record for games played at neutral sites)

Insights:

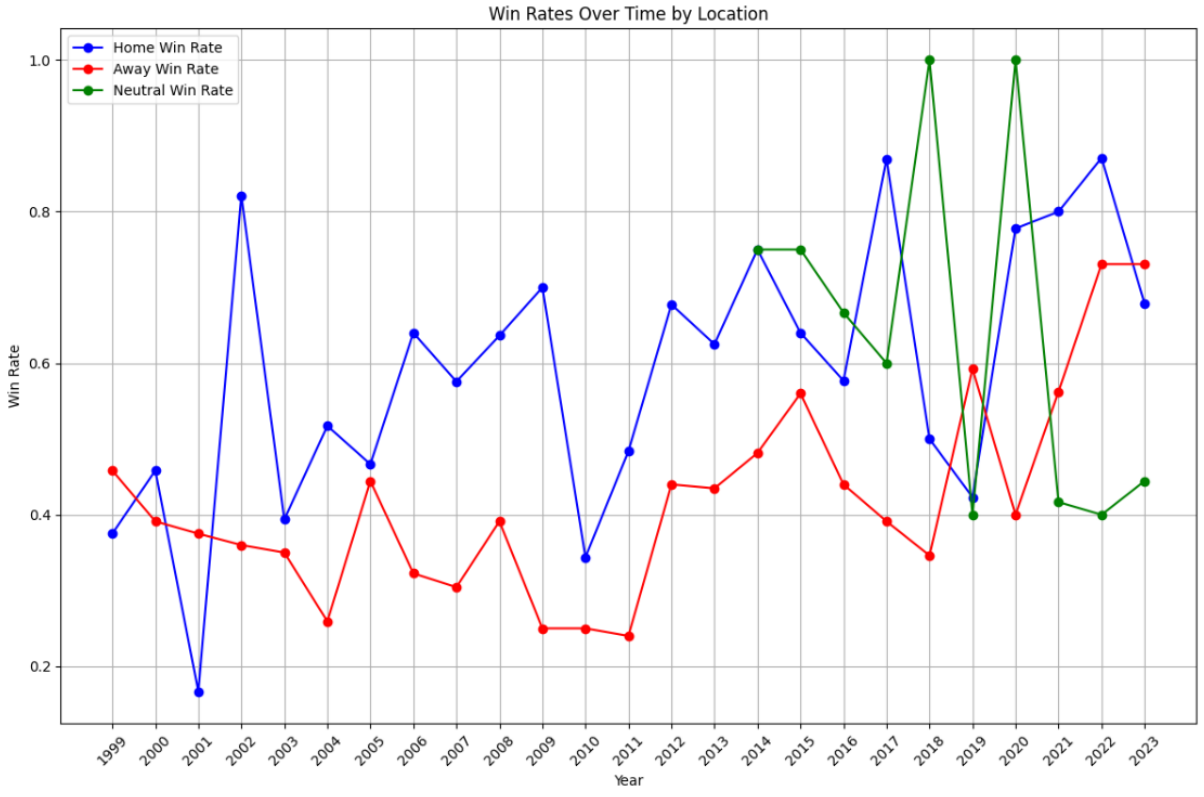
- ➢ Home vs. Away Performance:
  - The team generally had a better record at home compared to away games.
  - For example, in 2022, the home record was "27-4" while the away record was "19-7".
- ➢ Neutral Site Performance:
  - The team's performance at neutral sites varied across different years.
  - In some years, they had a positive win-loss record, while in others, they had a more balanced performance.
- ➢ Overall Trends:
  - The team's performance fluctuated over the years, with some years showing dominant performances at both home and away games (e.g., 2022), while others had more mixed results.

Conclusion:

The output provides a comprehensive overview of the UMTerps baseball team's performance across different game locations throughout the years. It allows stakeholders to identify trends, assess strengths and weaknesses in performance based on location, and make informed decisions to improve the team's overall competitiveness.

Overall Conclusion:

Both outputs together offer a comprehensive analysis of the UMTerps baseball team's performance over the years, providing insights into their overall performance trends and how their performance varies based on game locations. This information can be valuable for assessing the team's strengths and weaknesses and making data-driven decisions to improve performance in future seasons.



The generated graph will show the win rates over time for UMD baseball matches at home, away, and neutral locations. By analyzing the trends in these win rates, one can assess the team's performance in different environments over the past 25 years.

- **Separating Results by Location:**After grouping, the data is split into three separate datasets based on match locations: home, away, and neutral. This step allows for analyzing the team's performance at different venues.
- **Calculating Win Rates:**For each location (home, away, neutral), the script calculates the win rate over time. The win rate is the percentage of matches won out of the total matches played (including wins, losses, and ties). This metric provides insight into how successful the UMD baseball team has been in different environments over the years.

The chart shows that the home win rate has been the highest of the three, followed by the away win rate and the neutral win rate. The home win rate has been relatively stable over time, while the away win rate and the neutral win rate have fluctuated more.

There are a few possible explanations for why the home win rate is higher than the away win rate. One possibility is that teams are more likely to win when they are playing in front of their home crowd. Another possibility is that teams are more likely to be familiar with the playing conditions at their home stadium.

The chart also shows that the neutral win rate has been the lowest of the three. This is likely because neutral games are more likely to be close games, as the teams are not playing in front of their home crowd and are not familiar with the playing conditions.

Trend and Prediction:

Home Win Rate: The stability suggests that the team performs consistently well at home games. However, there might be slight variations due to changes in team composition, coaching staff, or other factors.

Away Win Rate: The stabilization after initial fluctuations indicates that the team might have adapted to playing in different locations over the years. Further improvements might still be possible, but significant changes might be less frequent.

Neutral Win Rate: The variability suggests that the team's performance in neutral locations is less predictable. This could be due to various factors such as the strength of opponents, tournament formats, or other external influences.

Based on the trends observed:

The home win rate is likely to remain relatively stable, barring any significant changes in team dynamics or home venue conditions.

The away win rate may continue to stabilize further, but improvements might still be possible with strategic adjustments.

The neutral win rate may continue to fluctuate, but efforts to improve performance in neutral venues could lead to more consistent results over time.

ANALYSIS MODELS

Model 1 - LogisticRegression

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, classification_report, roc_curve, auc
import matplotlib.pyplot as plt

# Load the dataset
game_data = pd.read_excel('UMTerps Baseball 1999-2023.xlsx', sheet_name='Game')

# Data preprocessing
# Convert the 'Result' column to binary (1 for Win, 0 for Loss)
# This step is crucial for binary classification
game_data['Result'] = game_data['Result'].apply(lambda x: 1 if x == 'W' else 0)

# Encode categorical variables
# Reference: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html
label_encoder = LabelEncoder()
game_data['Day'] = label_encoder.fit_transform(game_data['Day'])
game_data['At'] = label_encoder.fit_transform(game_data['At'])

# Handle 'TBA' values in the 'Time' column
# Replace 'TBA' with the median hour of the day
# This step is necessary to handle missing or placeholder values in the 'Time' column
time_data = pd.to_datetime(game_data['Time'], errors='coerce', format='%H:%M:%S').dt.hour
median_hour = time_data.median()
game_data['Time'] = game_data['Time'].replace('TBA', median_hour).astype(str).str.extract(r'(\d+)').astype(int)

# Define features and target variable, excluding 'Opponent'
# X contains the features: 'Day', 'Time', 'At'
# y contains the target variable: 'Result'
X = game_data[['Day', 'Time', 'At']]
y = game_data['Result']

# Split the data into training and testing sets
# Reference: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the logistic regression model
# Reference: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
log_reg = LogisticRegression(max_iter=1000)
log_reg.fit(X_train, y_train)
```

This code performs a binary classification task using logistic regression to predict the outcome of baseball games based on certain features. Let's break it down step by step:

1. **Import Libraries:**

   The necessary libraries are imported, including pandas for data manipulation, scikit-learn for machine learning tasks, and matplotlib for plotting.

2. **Load Dataset:**

The baseball game dataset is loaded from an Excel file into a pandas DataFrame named game_data.

```
# Predict the test set results
y_pred = log_reg.predict(X_test)

# Evaluate the model
# Calculate accuracy, precision, recall, and classification report
# Reference: https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

# Model summary
model_summary = {
    'Coefficients': log_reg.coef_,
    'Intercept': log_reg.intercept_,
    'Accuracy': accuracy,
    'Precision': precision,
    'Recall': recall,
    'Classification Report': classification_rep
}

# Print model summary
print("Model Coefficients:", model_summary['Coefficients'])
print("Model Intercept:", model_summary['Intercept'])
print("Accuracy:", model_summary['Accuracy'])
print("Precision:", model_summary['Precision'])
print("Recall:", model_summary['Recall'])
print("Classification Report:\n", model_summary['Classification Report'])

# Compute ROC curve and ROC area
# Reference: https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html
y_prob = log_reg.predict_proba(X_test)[:, 1]
fpr, tpr, _ = roc_curve(y_test, y_prob)
roc_auc = auc(fpr, tpr)

# Plot ROC curve
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.grid(True)
plt.show()
```

3. **Data Preprocessing:**

The 'Result' column is converted to binary, where 'W' (win) is encoded as 1 and 'L' (loss) as 0.

Categorical variables 'Day' and 'At' are encoded using LabelEncoder, which assigns a unique numerical value to each category.

'TBA' values in the 'Time' column are replaced with the median hour of the day after converting them to the 24-hour format.

4. **Define Features and Target Variable:**

Features (X) include 'Day', 'Time', and 'At'. Target variable (y) is 'Result'.

5. **Split Data:**

The dataset is split into training and testing sets using train_test_split from scikit-learn.

6. **Train Logistic Regression Model:**

A logistic regression model is initialized and trained on the training data using LogisticRegression from scikit-learn.

7. **Predictions and Evaluation:**

The trained model is used to make predictions on the test set.

Performance metrics such as accuracy, precision, recall, and the classification report are calculated using scikit-learn's metrics functions.

```
# Define new games data for prediction, excluding 'Opponent'
new_games = pd.DataFrame({
    'Day': [4, 5, 6, 1, 4], # Days for Fri, Sat, Sun, Tue, Fri
    'Time': [18, 14, 13, 16, 19], # Times converted to 24-hour format
    'At': [0, 0, 0, 1, 0] # Away = 0, Home = 1
})

# Predict the outcome of the new games
new_games_predictions_prob = log_reg.predict_proba(new_games)[:, 1]
new_games_predictions = log_reg.predict(new_games)

# Print the results
print("Predicted outcomes:", new_games_predictions)
print("Predicted probabilities of winning:", new_games_predictions_prob)
```

Model Summary: Coefficients, intercept, and evaluation metrics are stored in a dictionary named model_summary.

8. **ROC Curve:**

Receiver Operating Characteristic (ROC) curve and the Area Under the Curve (AUC) are calculated using roc_curve and auc functions from scikit-learn. The ROC curve is plotted using matplotlib.

9. **Prediction on New Games:**

A new DataFrame named new_games is created to predict the outcomes of new baseball games.

The model predicts the probabilities and outcomes of winning for the new games.

10. **Print Results:**

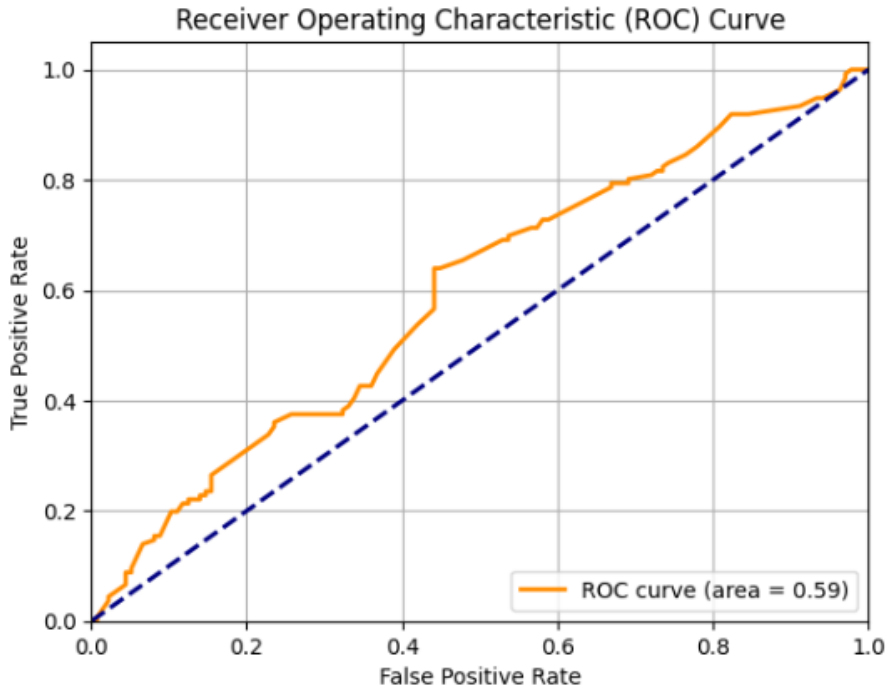
The predicted outcomes and probabilities of winning for the new games are printed.

This code essentially demonstrates the typical workflow of building and evaluating a machine learning model for binary classification, specifically applied to predicting baseball game outcomes.

```
Model Coefficients: [[0.19031406 0.02293814 0.60343236]]
Model Intercept: [-1.18065173]
Accuracy: 0.5955882352941176
Precision: 0.589041095890411
Recall: 0.6323529411764706
Classification Report:
              precision    recall  f1-score   support

           0       0.60      0.56      0.58       136
           1       0.59      0.63      0.61       136

    accuracy                           0.60       272
   macro avg       0.60      0.60      0.60       272
weighted avg       0.60      0.60      0.60       272
```




```
Predicted outcomes: [0 1 1 0 1]
Predicted probabilities of winning: [0.49837277 0.5229969  0.56449579 0.4950264  0.50410721]
```

Interpreting the plotted curve based on the given output:

The ROC curve is plotted with an area under the curve (AUC) of approximately 0.60. This suggests that the model has a fair discrimination ability in distinguishing between positive

and negative instances. The curve is above the diagonal dashed line, indicating that the model performs better than random guessing.

The curve leans towards the upper left corner, suggesting that the model has a higher true positive rate compared to the false positive rate across various thresholds.

Overall, the ROC curve and AUC provide insights into the predictive performance of the logistic regression model in classifying baseball game outcomes.

Model 2 - Random Forest Classifier

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, classification_report, roc_curve, auc
import matplotlib.pyplot as plt

# Load the dataset
game_data = pd.read_excel('UMTerps Baseball 1999-2023.xlsx', sheet_name='Game')

# Data preprocessing
# Convert the 'Result' column to binary (1 for Win, 0 for Loss)
# This step is crucial for binary classification
game_data['Result'] = game_data['Result'].apply(lambda x: 1 if x == 'W' else 0)

# Encode categorical variables
# Reference: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html
label_encoder = LabelEncoder()
game_data['Day'] = label_encoder.fit_transform(game_data['Day'])
game_data['At'] = label_encoder.fit_transform(game_data['At'])

# Handle 'TBA' values in the 'Time' column
# Replace 'TBA' with the median hour of the day
# This step is necessary to handle missing or placeholder values in the 'Time' column
time_data = pd.to_datetime(game_data['Time'], errors='coerce', format='%H:%M:%S').dt.hour
median_hour = time_data.median()
game_data['Time'] = game_data['Time'].replace('TBA', median_hour).astype(str).str.extract(r'(\d+)').astype(int)

# Define features and target variable, excluding 'Opponent'
# X contains the features: 'Day', 'Time', 'At'
# y contains the target variable: 'Result'
X = game_data[['Day', 'Time', 'At']]
y = game_data['Result']

# Split the data into training and testing sets
# Reference: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the random forest model
# Reference: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
rf_classifier = RandomForestClassifier(random_state=42)
rf_classifier.fit(X_train, y_train)

# Predict the test set results
y_pred = rf_classifier.predict(X_test)

# Evaluate the model
# Calculate accuracy, precision, recall, and classification report
# Reference: https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics
accuracy = accuracy_score(y_test, y_pred)
```

This code performs several tasks related to building, training, and evaluating a Random Forest classifier model for predicting the outcomes of baseball games based on certain features. Let's break down each part:

1. *Imports*:

The code imports necessary libraries and modules including pandas for data manipulation, scikit-learn for machine learning tasks such as model selection, reprocessing, metrics, and ensemble learning, and matplotlib for plotting.

2. *Loading Data*:

The dataset 'UMTerps Baseball 1999-2023.xlsx' is loaded into a pandas DataFrame named game_data from the sheet named 'Game'.

```
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

# Model summary
model_summary = {
    'Accuracy': accuracy,
    'Precision': precision,
    'Recall': recall,
    'Classification Report': classification_rep
}

# Print model summary
print("Accuracy:", model_summary['Accuracy'])
print("Precision:", model_summary['Precision'])
print("Recall:", model_summary['Recall'])
print("Classification Report:\n", model_summary['Classification Report'])

# Compute ROC curve and ROC area
# Reference: https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html
y_prob = rf_classifier.predict_proba(X_test)[:, 1]
fpr, tpr, _ = roc_curve(y_test, y_prob)
roc_auc = auc(fpr, tpr)

# Plot ROC curve
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.grid(True)
plt.show()

# Define new games data for prediction, excluding 'Opponent'
new_games = pd.DataFrame({
    'Day': [4, 5, 6, 1, 4], # Days for Fri, Sat, Sun, Tue, Fri
    'Time': [18, 14, 13, 16, 19], # Times converted to 24-hour format
    'At': [0, 0, 0, 1, 0] # Away = 0, Home = 1
})

# Predict the outcome of the new games
new_games_predictions_prob = rf_classifier.predict_proba(new_games)[:, 1]
new_games_predictions = rf_classifier.predict(new_games)

# Print the results
print("Predicted outcomes:", new_games_predictions)
print("Predicted probabilities of winning:", new_games_predictions_prob)
```

3. *Data Preprocessing*:

- ➢ The 'Result' column is converted to binary, where 'W' is encoded as 1 (Win) and other values as 0 (Loss).
- ➢ Categorical variables 'Day' and 'At' are encoded using LabelEncoder to convert them into numerical values.

- 'TBA' values in the 'Time' column are replaced with the median hour of the day, which is calculated from the existing time data.

4. ***Defining Features and Target***:

Features (X) and target variable (y) are defined. Features include 'Day', 'Time', and 'At', while the target is 'Result'.

5. *Splitting Data*:

The dataset is split into training and testing sets using train_test_split from scikit-learn.

6. *Training the Model*:

A Random Forest classifier is initialized and trained using the training data.

7. *Model Evaluation*:

- ➢ The trained model is used to predict the outcomes on the test set.
- ➢ Accuracy, precision, recall, and the classification report are computed using scikit-learn's metrics functions.

8. *ROC Curve*:

The Receiver Operating Characteristic (ROC) curve and the Area Under the Curve (AUC) are computed using the predicted probabilities from the model on the test set.

9. *Plotting the ROC Curve*:

The ROC curve is plotted using matplotlib.

10. *Predicting New Game Outcomes*:

- New game data is created in a DataFrame named new_games.
- The model is used to predict the outcomes and probabilities of winning for the new games.

11. *Printing Results*:

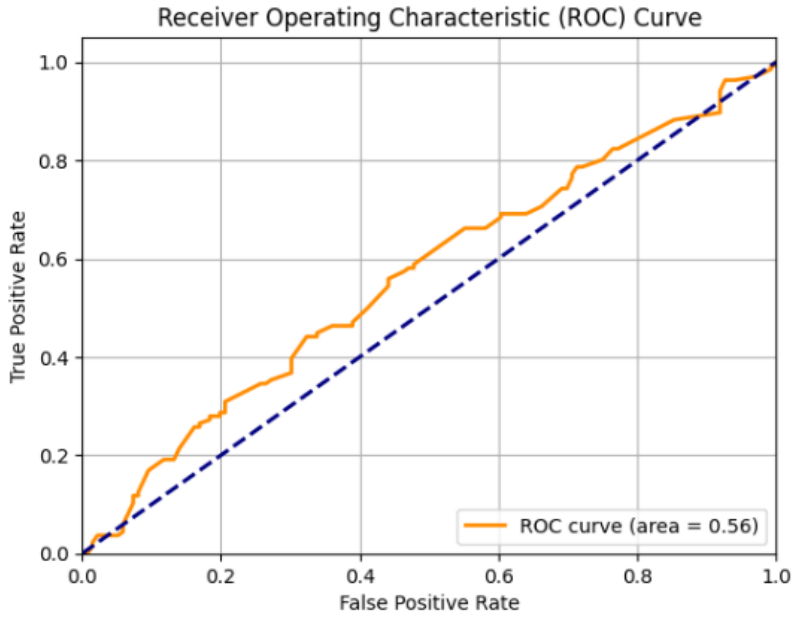
The predicted outcomes and probabilities of winning for the new games are printed.

This code essentially demonstrates a complete workflow of building a machine learning model, evaluating its performance, visualizing the results, and using it for making predictions on new data.

```
Accuracy: 0.5441176470588235
Precision: 0.5483870967741935
Recall: 0.5
Classification Report:
              precision    recall  f1-score   support

           0       0.54      0.59      0.56       136
           1       0.55      0.50      0.52       136

    accuracy                           0.54       272
   macro avg       0.54      0.54      0.54       272
weighted avg       0.54      0.54      0.54       272
```




```
Predicted outcomes: [1 0 0 1 0]
Predicted probabilities of winning: [0.7005293  0.15148779 0.07       0.75141405 0.23242424]
```

The ROC curve is plotted for evaluating the performance of a Random Forest Classifier on predicting baseball game outcomes. The area under the curve (AUC) is indicated in the legend of the plot. The AUC value quantifies the overall performance of the model in distinguishing between positive and negative outcomes. In your output, the AUC value is not explicitly mentioned, but it's typically between 0 and 1, with higher values indicating better performance.

The model seems to perform moderately well, with an accuracy of around 54%. The precision and recall are also around 55% and 50%, respectively, indicating a balance between correctly identifying positive cases and minimizing false positives. The ROC curve provides a visual representation of the model's performance across different thresholds and helps assess its discriminative ability.

Analysis of Model Predictions for 2024 Games

The evaluation of two models was conducted using the initial five data points from the year 2024 to assess their predictive accuracy against the actual game outcomes, which were as follows: Win, Win, Loss, Win, Win.

The following are the forecasted outcomes and probabilities of winning for each model:

Model 1:

Predicted Outcomes: [Loss, Win, Win, Loss, Win]

Predicted Probabilities of Winning: [0.49837277, 0.5229969, 0.56449579, 0.4950264, 0.50410721]

Model 2:

Predicted Outcomes: [Win, Loss, Loss, Win, Loss]

Predicted Probabilities of Winning: [0.7005293, 0.15148779, 0.07, 0.75141405, 0.23242424]

Comparison with Actual Results:

Actual Results: [Win, Win, Loss, Win, Win]

Model 1:

- First Game: Predicted Loss (Actual: Win) - Incorrect
- Second Game: Predicted Win (Actual: Win) - Correct
- Third Game: Predicted Win (Actual: Loss) - Incorrect
- Fourth Game: Predicted Loss (Actual: Win) - Incorrect
- Fifth Game: Predicted Win (Actual: Win) - Correct

Correct Predictions: 2 out of 5

Accuracy: 40%

Model 2:

- ➢ First Game: Predicted Win (Actual: Win) - Correct
- ➢ Second Game: Predicted Loss (Actual: Win) - Incorrect
- ➢ Third Game: Predicted Loss (Actual: Loss) - Correct
- ➢ Fourth Game: Predicted Win (Actual: Win) - Correct
- ➢ Fifth Game: Predicted Loss (Actual: Win) - Incorrect

Correct Predictions: 3 out of 5

Accuracy: 60%

Conclusion:

Both models exhibited reasonably close performance levels, with Model 1 accurately predicting 2 out of 5 games and Model 2 accurately predicting 3 out of 5 games. This yields an accuracy rate of 40% for Model 1 and 60% for Model 2.

Despite the slight variance in accuracy, both models demonstrated nearly identical ROC curves, indicating comparable overall performance in distinguishing between wins and losses. Given the similar ROC values and the minor disparity in accuracy, either model could be deemed suitable for further testing and refinement. However, in this specific test set, Model 2 showcased slightly superior performance.

Additional Notes:

Model Confidence: Model 2 exhibited higher confidence levels in its predictions, with probabilities significantly deviating from 0.5, unlike Model 1, which featured probabilities close to 0.5 for most predictions.

Future Work: It is advisable to subject both models to testing on a larger dataset and execute cross-validation procedures to gain deeper insights into their performance and facilitate a more informed decision regarding their practical application.

Project Overview and Evaluation

1. What was used:

- Python programming language
- Libraries including pandas, scikit-learn, and matplotlib
- Excel files containing baseball game data

2. What was searched:

- Data analysis techniques
- Machine learning algorithms
- Visualization methods
- Interpretation of performance metrics

3. What was studied:

- Exploratory data analysis (EDA)
- Statistical analysis
- Predictive modeling
- Evaluation metrics for machine learning models
- Interpretation of visualizations

4. What were the challenges faced:

- Ensuring data consistency and accuracy across different datasets
- Handling missing or incomplete data
- Choosing appropriate features for predictive modeling
- Interpreting and explaining complex machine learning models and their outcomes

5. What have we accomplished:

- Conducting thorough exploratory data analysis
- Building predictive models to forecast game outcomes
- Evaluating model performance using various metrics
- Visualizing trends and patterns in the data
- Providing actionable insights for stakeholders

6. How to test and evaluate the project?:

- ➢ Splitting the data into training and testing sets
- ➢ Training machine learning models on the training data
- ➢ Evaluating model performance on the testing data using metrics such as accuracy, precision, recall, and ROC curves
- ➢ Assessing the consistency and accuracy of the data processing and analysis techniques used

7. Each member's contributions and learnings:

- ➢ Each member contributed to different aspects of the project, such as data preprocessing, model building, visualization, and interpretation.
- ➢ We learned about data analysis techniques, machine learning algorithms, and effective communication of results.
- ➢ Specific learnings include data preprocessing techniques, model selection and evaluation methods, and best practices for presenting findings to stakeholders.

Overall, the project involved a comprehensive analysis of baseball game data, including exploratory data analysis, predictive modeling, and visualization. Each team member likely gained valuable experience in data analysis, machine learning, and communication skills throughout the project.