MLB Baseball Data Analyses

Introduction - Presentation , Code

In our analysis of baseball data, we delve into various facets, including player performance, salaries, college affiliations, and historical trends. The dataset encompasses a rich array of information, incorporating player details, batting and pitching statistics, and college backgrounds. With a focus on Python programming, leveraging the pandas and seaborn libraries, we employ data manipulation techniques such as merging tables, filtering data, and computing relevant metrics to extract meaningful insights. Our exploration spans diverse analyses, from identifying the most improved players and top-salary colleges and uncovering trends in pitcher performance and comparing batting averages over the years.

Dataset

The dataset comprises comprehensive baseball-related information, making it suitable for a multifaceted analysis. Attributes like playerID, yearID, and various performance metrics allow us to explore player evolution, college contributions, and salary distributions.

- People Table: Contains player information, including name details, birth details, batting, and throwing hands. playerID is unique code given to every player.
- Batting Table: Provides statistics such as at-bats, hits, home runs, year, runs bated in, etc.
- Pitching Table: Includes statistics like player id, wins, losses, earned run average, etc.
- CollegePlaying Table: Contains information about players' college affiliations as player id, school id, year.
- Salaries Table: Includes players' salaries, team, year.
- Schools Table: Provides information about colleges school id, team name.

Analysis Technique

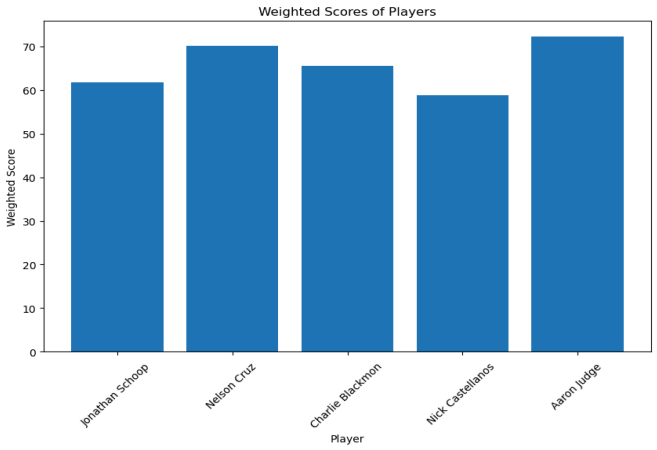
We utilized Python programming along with pandas and seaborn libraries for data manipulation and visualization. Our approach involves merging relevant tables, filtering data, and calculating metrics to derive meaningful insights.

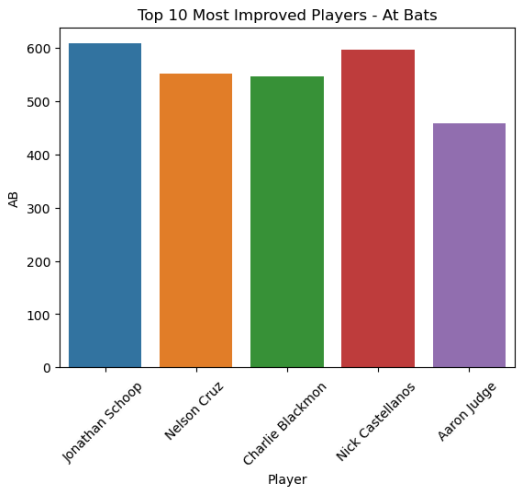
Analysis 1: Most Improved Player Throughout Their Career

Technique- We first read in our data, set up our columns, dropped NANs, and got the players first and last year. We merged the two, and then subtracted their last year from their first year, and set it as a new variable. Then we sorted, and merged with the People.csv to get their names.

Analyzed the batting statistics of baseball players and calculated a weighted score for each player based on home runs (HR), runs batted in (RBI), stolen bases (SB), walks (BB), and hits (H). The most improved player is determined by the highest weighted score. We then thought of how we define 'improvement'. What makes a player good. We decided that HR and RBI were pretty important, so we gave that a higher stake in the total score. But we also included Stolen Bases, Walks, and Hits in their total score.

Results- After calculation, we graphed our results, and we noted that the most improved player base on our metrics was Aaron Judge. Of our top 5 players, most lied within the range of 60-70. Though, if we calculate best as the most At Bats, the graph changes, and Jonathan Schoop becomes thewinner, at a total of above 600 At bats



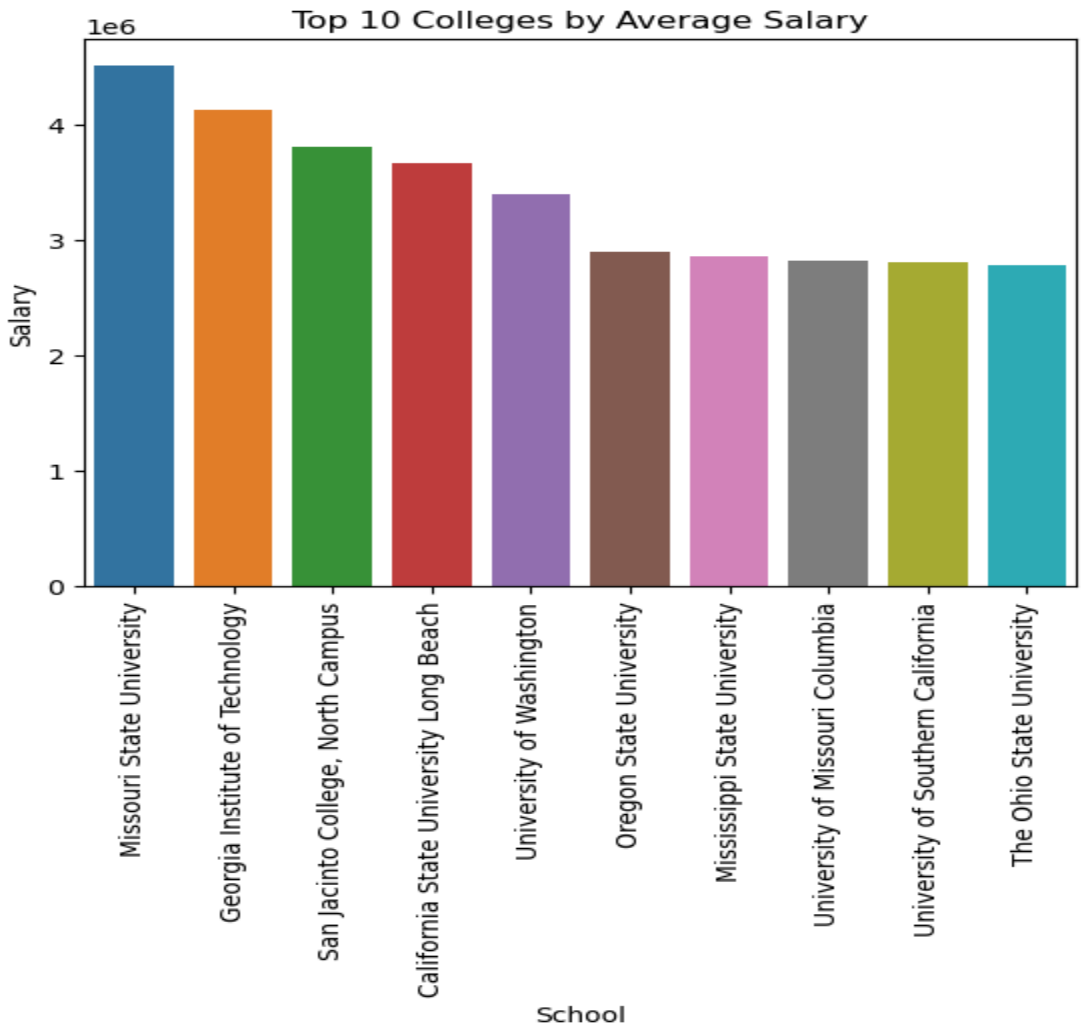


Analysis 2: Top Colleges by Average Salary

Technique- We first loaded our datasets, CollegePlaying.csv and Salaries.csv. Then we merged, grouped by college, and got rid of schools with less than 5 players. We did this as one college had one player from it, and that player had a massive salary. We then calculated the mean for each college, and displayed them in a graph.

Results-

We showed the top 10 Colleges, and we noticed that the lower 5 of the 10 were all sitting around 3 million, but there is a pretty constant increase until we get to Missouri State University, whose average was $4.5 million. This is significant, because as we look at the average salary for players from the bottom schools, we note that they only have average salaries of 700,000.

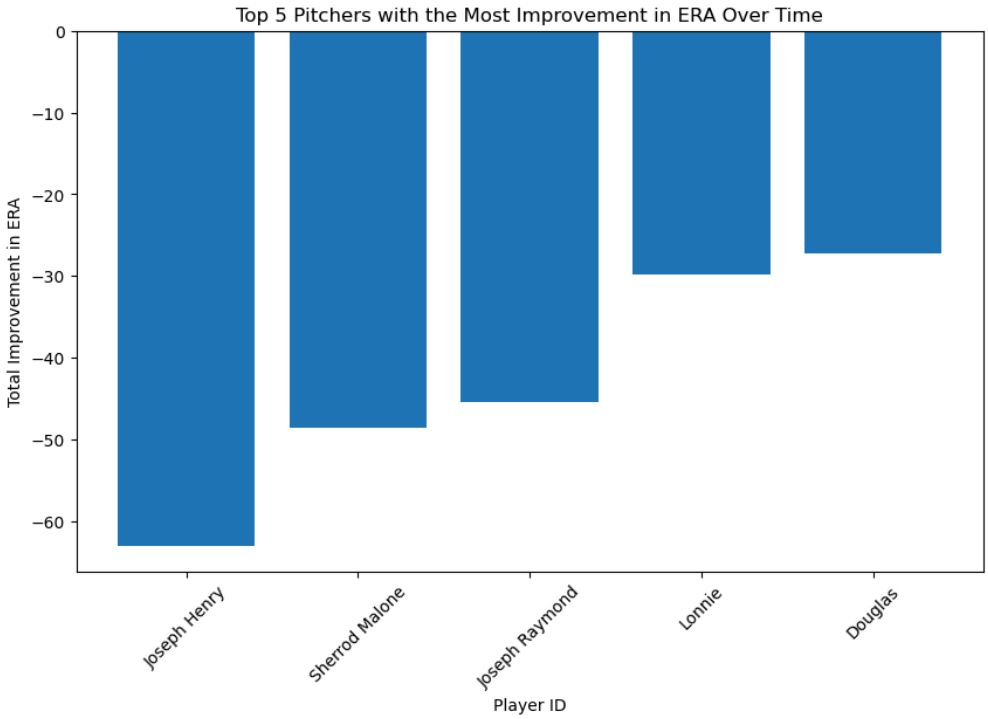


Analysis 3: Improved top 5 Pitchers Over the Years

Technique- We In the dataset we extract Earned Run Average (ERA) for every player. Then, we calculated the average ERA for each player over different years. Next, we assessed the improvement in ERA for each player by comparing their performance in consecutive years. To focus on experienced players, we filtered out those with fewer than ten seasons. Afterward, we determined the total improvement in ERA for each player by summing up these year-to-year improvements. Finally, we identified and selected the top 5 pitchers who showed the most significant overall improvement in their ERA across multiple seasons

Results- The top 5 pitchers with the most significant improvement in Earned Run Average (ERA) over the years are as follows:

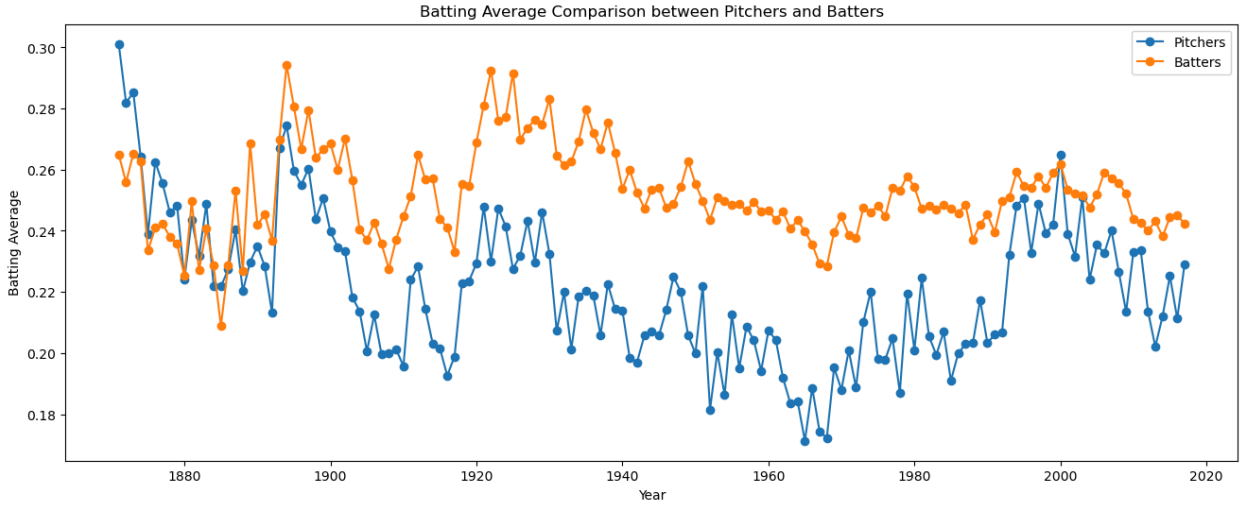
If coaches know which pitchers have gotten much better at preventing runs (ERA improvement), they can figure out what training methods worked. This may also helps teams decide to keep or get players who improved, thinking they'll make the team better.



Analysis 4: Batting Averages of Pitchers and Batters Over the Years

Technique- For finding batting average for pitcher and batters over the years we need to file namely 'Batting.csv' and 'Pitching.csv'. Subsequently, we created two distinct DataFrames, by filtering players in the 'Batting' table based on whether pitcher unique id were not present(only batters) and were present in the 'Pitching' table (batters and pitcher), respectively. Further, we computed the batting average (H/AB) for both groups, namely pitchers and batters. To ensure data quality, we eliminated rows where 'BattingAveragepitchers' and 'BattingAveragebatters' were either 0 or contained missing values. The next step involved merging the 'pitchers' and 'batters' DataFrames based on 'playerID' and 'yearID' by joining table. Finally, we aggregated the mean batting averages for pitchers and batters for each year.

Results-



Role Specialization:There's been a trend in the major leagues to exclude pitchers when calculating team batting averages, as pitchers generally have lower averages than position players. This is because pitchers specialize in pitching and spend less time honing their batting skills compared to batters.

Live Ball Era (1920's): This era began with a series of rule changes in 1920 that made the ball more "lively", leading to a dramatic rise in offensive statistics.

The year 1968 was known as the "Year of the Pitcher", with pitchers such as Bob Gibson and Denny McLain dominating the game. This could have led to a decrease in batting averages for pitchers as they focused more on their pitching skills.
This events can tell us how changes in game rules and world events can make a major impact on the overall game for both batter and pitcher

Technical

Data Preparation

In the data preparation phase, Python's Pandas library is utilized to load baseball-related datasets. The primary datasets include 'Batting.csv', 'Pitching.csv', 'CollegePlaying.csv', 'Salaries.csv', 'People.csv','Batting.csv', and 'Schools.csv.' The *pd.read_csv* function is employed to read CSV

files into Pandas DataFrames. *isna(), dropna()* function is used to remove nan or missing valuer in dataset. If require to merge tables *pd.merge()* is used which mearge two dataset on condition.

Analysis Process

1. Most Improved Player Analysis:
   - Initially, player data is loaded and cleaned.
   - Player progress is calculated based on various baseball statistics.
   - Weighted scores are computed, and the most improved player is identified.

     # Calculate weighted score for each player
     battingFirstLastName['Score'] = (battingFirstLastName['HR'] * 0.3) + (battingFirstLastName['RBI'] * 0.3) + (battingFirstLastName['SB'] * 0.1) + (battingFirstLastName['BB'] * 0.1) + (battingFirstLastName['H'] * 0.1)
   - Results are visualized using a bar plot.
2. College Salary Analysis:
   - College playing and salary datasets are merged.
   - Schools with less than 5 players are filtered out.

     ```
     # Filter out schools with less than 5 players
     valid_schools = college_players_count[college_players_count >= 10].index
     ```
   - The count and mean salary for players from each college are calculated.

     ```
     # Filter the merged_data dataframe to include only valid schools
     merged_data_filtered = merged_data[merged_data['schoolID'].isin(valid_schools)]

     college_salaries = merged_data_filtered.groupby('schoolID')['salary'].sum()

     college_salaries = merged_data_filtered.groupby('schoolID')['salary'].mean()
     ```
   - The top and bottom colleges by average salary are visualized.
3. Improved Pitchers Analysis:
   - Pitching data is loaded and filtered.
   - Average ERA is calculated for each player, and improvement is computed.

```
# Calculate average ERA for each player
average_era = pitching.groupby(['playerID', 'yearID'])['ERA'].mean().reset_index()
# Calculate improvement by finding the difference in ERA between consecutive years
average_era['ERA_difference'] = average_era.groupby('playerID')['ERA'].diff()
```

   - Players with at least ten seasons are selected.
   - Top 5 improved pitchers are identified and visualized.

```
# Calculate the total improvement for each player
total_improvement = improvement_over_time.groupby('playerID')['ERA_difference'].sum().reset_index()
# Sort by total improvement and select the top 10 players
top_5_improved_pitchers = total_improvement.sort_values(by='ERA_difference').head(5)
```

4. Batting Averages Analysis:
    - Batting and pitching datasets are loaded and filtered.
    - Identify pitchers who are batters and only batters.

```
# Extract unique playerIDs from the Pitching table
pitchers_player_ids = pitching['playerID'].unique()
# Filter pitcher in Batting table(players who are pitcher and batters)
pitchers = batting[batting['playerID'].isin(pitchers_player_ids)].copy()
# Filter Batting table for remaining players (not in the Pitching table)
batters = batting[~batting['playerID'].isin(pitchers_player_ids)].copy()
```

    - Batting averages for pitchers and batters are calculated.

```
# Calculate Batting Average (H/AB) for both groups
pitchers['BattingAveragepitchers'] = pitchers['H'] / pitchers['AB']
batters['BattingAveragebatters'] = batters['H'] / batters['AB']
```

    - Rows with zero or NaN averages are removed.
    - A comparison of batting averages between pitchers and batters is visualized over the years.

```
# Batting average is calculate for BattingAveragepitchers and BattingAveragebatters and groupd by yearID
avg_batting=players.groupby('yearID').agg({'BattingAveragepitchers':'mean','BattingAveragebatters':'mean'}).reset_index()
```

Conclusion

Our analysis provides a holistic view of baseball-related data, ranging from individual player performance to historical trends. The results are presented in a visually appealing manner, facilitating easy interpretation.