data

August 26, 2025

[10]:
```
import pandas as pd
import requests
from bs4 import BeautifulSoup
import pybaseball as pyb
import json
from datetime import datetime
import time
import os

import re


pyb.cache.enable()
```

[3]:
```
team_batting = pyb.team_batting(2024)
team_batting = team_batting[['Team', 'G', 'AB', 'R', 'H', 'HR', 'RBI', 'SB',␣
  ↪'OBP', 'SLG']]
print("Team Batting Data Sample:")
print(team_batting.head())
```

```
Team Batting Data Sample:
  Team     G    AB    R     H   HR  RBI   SB    OBP    SLG
0  LAD  2403  5522  842  1423  233  815  136  0.335  0.446
1  ARI  2436  5522  886  1452  211  845  119  0.337  0.440
2  NYY  2304  5450  815  1352  237  782   88  0.333  0.429
3  PHI  2317  5534  784  1423  198  750  148  0.325  0.425
4  BAL  2412  5567  786  1391  235  759   98  0.315  0.435
```

[4]:
```
team_pitching = pyb.team_pitching(2024)
team_pitching = team_pitching[['Team', 'W', 'L', 'ERA', 'IP', 'SO', 'WHIP',␣
  ↪'FIP']]
print("\nTeam Pitching Data Sample:")
print(team_pitching.head())
```

```

Team Pitching Data Sample:
  Team   W   L   ERA      IP    SO  WHIP   FIP
0  ATL  89  73  3.49  1443.1  1553  1.20  3.44
1  SEA  85  77  3.49  1433.0  1416  1.08  3.73
```

```
2  CLE  92  69  3.61  1428.0  1410  1.20  3.98
3  DET  86  76  3.63  1447.0  1354  1.16  3.70
4  MIL  93  69  3.65  1446.0  1373  1.23  4.19
```

[5]:
```
dodgers_logs = pyb.schedule_and_record(2024, 'LAD')
dodgers_logs = dodgers_logs[['Date', 'Opp', 'W/L', 'R', 'RA', 'Inn', 'GB',␣
  ↪'Home_Away']]
print("\nDodgers Game Log Sample:")
print(dodgers_logs.head())
```

```
http://www.baseball-reference.com/teams/LAD/2024-schedule-scores.shtml

Dodgers Game Log Sample:
                Date Opp W/L     R    RA   Inn      GB Home_Away
1  Wednesday, Mar 20 SDP   W   5.0   2.0   9.0  up 0.5         @
2   Thursday, Mar 21 SDP   L  11.0  15.0   9.0    Tied      Home
3   Thursday, Mar 28 STL   W   7.0   1.0   9.0    Tied      Home
4     Friday, Mar 29 STL   W   6.0   3.0   9.0    Tied      Home
5   Saturday, Mar 30 STL   L   5.0   6.0  10.0    Tied      Home
```

```
/home/cjanua/Passport_Repos/SportsBetting/.venv/lib/python3.13/site-
packages/pybaseball/team_results.py:75: FutureWarning: A value is trying to be
set on a copy of a DataFrame or Series through chained assignment using an
inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['Attendance'].replace(r'^Unknown$', np.nan, regex=True, inplace = True) #
make this a NaN so the column can benumeric
```

[11]:
```
def scrape_oddsportal(url):
    headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) Chrome/
  ↪91.0.4472.124'}
    try:
        response = requests.get(url, headers=headers)
        response.raise_for_status()
        soup = BeautifulSoup(response.text, 'html.parser')

        # Find all tables (no specific 'table-main' class in provided HTML)
        tables = soup.find_all('table')
        odds_data = []
        for table in tables:
            # Look for rows with game data (skip headers and non-game rows)
            rows = table.find_all('tr')
            for row in rows:
                cols = row.find_all('td')
                if len(cols) >= 4:  # Ensure row has game, odds, and bookmakers
                    # Extract game details (teams and score)
                    game_cell = cols[0].text.strip()
                    # Use regex to split teams and scores (e.g., "New York␣
↪Yankees 6-7 Los Angeles Dodgers")
                    match = re.match(r'(.+?)\s+(\d+)\s*-\s*(\d+)\s+(.+)',␣
↪game_cell)
                    if match:
                        team1, score1, score2, team2 = match.groups()
                    else:
                        continue  # Skip rows without proper game format

                    # Extract odds and bookmakers
                    moneyline_home = cols[1].text.strip() if len(cols) > 1 else␣
↪'N/A'
                    moneyline_away = cols[2].text.strip() if len(cols) > 2 else␣
↪'N/A'
                    over_under = cols[3].text.strip() if len(cols) > 3 else 'N/
↪A'
                    bookmakers = cols[4].text.strip() if len(cols) > 4 else 'N/
↪A'

                    # Extract date and time (from parent div or previous row)
                    date_row = row.find_previous('tr', class_='table-dummyrow')
                    game_date = date_row.text.strip() if date_row else 'Unknown'

                    odds_data.append({
                        'Date': game_date,
                        'Team1': team1,
                        'Score1': score1,
                        'Team2': team2,
                        'Score2': score2,
                        'Moneyline_Home': moneyline_home,
                        'Moneyline_Away': moneyline_away,
                        'Over_Under': over_under,
                        'Bookmakers': bookmakers
                    })

        return pd.DataFrame(odds_data)
    except Exception as e:
        print(f"Error scraping odds: {e}")
        return pd.DataFrame()
```

[12]:
```
odds_url = "https://www.oddsportal.com/baseball/usa/mlb-2024/results/"
odds_df = scrape_oddsportal(odds_url)
if not odds_df.empty:
    print("\nBetting Odds Sample:")
    print(odds_df.head())
    odds_df.to_csv('mlb_odds_2024.csv', index=False)
else:
    print("No odds data retrieved. Check URL or site restrictions.")
```

```
No odds data retrieved. Check URL or site restrictions.
```

[13]:
```
def get_weather_data(city, date, api_key):
    try:
        # Convert date to Unix timestamp for API
        date_obj = datetime.strptime(date, '%d %b %Y')
        timestamp = int(date_obj.timestamp())
        url = f"http://api.openweathermap.org/data/2.5/weather?
 ↪q={city}&dt={timestamp}&appid={api_key}&units=metric"
        response = requests.get(url)
        response.raise_for_status()
        data = response.json()
        return {
            'City': city,
            'Date': date,
            'Temperature_C': data['main']['temp'],
            'Wind_Speed_ms': data['wind']['speed'],
            'Humidity': data['main']['humidity']
        }
    except Exception as e:
        print(f"Error fetching weather for {city} on {date}: {e}")
        return None
```

[14]:
```
api_key = "YOUR_OPENWEATHER_API_KEY"  # Replace with your key
weather_data = get_weather_data("Los Angeles", "30 Oct 2024", api_key)
if weather_data:
    weather_df = pd.DataFrame([weather_data])
    print("\nWeather Data Sample:")
    print(weather_df)
    weather_df.to_csv('mlb_weather_sample_2024.csv', index=False)
```

```
Error fetching weather for Los Angeles on 30 Oct 2024: 401 Client Error:
Unauthorized for url: http://api.openweathermap.org/data/2.5/weather?q=Los%20Ang
eles&dt=1730260800&appid=YOUR_OPENWEATHER_API_KEY&units=metric
```

[15]:
```
print("\nData Availability Summary:")
print(f"- Team Batting: {len(team_batting)} teams, columns: {list(team_batting.
  ↪columns)}")
print(f"- Team Pitching: {len(team_pitching)} teams, columns:␣
  ↪{list(team_pitching.columns)}")
print(f"- Game Logs (Dodgers): {len(dodgers_logs)} games, columns:␣
  ↪{list(dodgers_logs.columns)}")
print(f"- Betting Odds: {len(odds_df)} games, columns: {list(odds_df.columns)}")
print(f"- Weather Data: {'Available' if weather_data else 'Not Available'}")
```

```
Data Availability Summary:
- Team Batting: 30 teams, columns: ['Team', 'G', 'AB', 'R', 'H', 'HR', 'RBI',
'SB', 'OBP', 'SLG']
- Team Pitching: 30 teams, columns: ['Team', 'W', 'L', 'ERA', 'IP', 'SO',
'WHIP', 'FIP']
- Game Logs (Dodgers): 162 games, columns: ['Date', 'Opp', 'W/L', 'R', 'RA',
'Inn', 'GB', 'Home_Away']
- Betting Odds: 0 games, columns: []
- Weather Data: Not Available
```

[16]:
```
if not odds_df.empty:
    # Mock merge (simplified; adjust based on actual data alignment)
    merged_df = dodgers_logs.copy()
    merged_df['Moneyline_Home'] = 'N/A'  # Placeholder; replace with actual␣
  ↪merge logic
    print("\nMerged Game Logs and Odds Sample:")
    print(merged_df.head())
    merged_df.to_csv('merged_mlb_data_2024.csv', index=False)
else:
    print("Cannot merge; no odds data available.")
```

```
Cannot merge; no odds data available.
```

[ ]:
```
```