IPL Statistics Web Scraping Project Report

Tapaswi Satyapanthi
College of Engineering
Northeastern University
satyapanthi.t@northeastern.edu

1 Introduction

The Indian Premier League (IPL) is a professional Twenty20 cricket league in India, founded in 2008 by the Board of Control for Cricket in India (BCCI). IPL features franchise teams representing different Indian cities, with top cricket players from around the world participating. The league is one of the most popular and lucrative cricket competitions globally, attracting millions of viewers and large sponsorship deals. At the end of each IPL season, top-performing players are recognized with prestigious awards such as the Orange Cap (for the batsman with the highest runs) and the Purple Cap (for the bowler with the most wickets).

This project focuses on scraping detailed statistics of both batsmen and bowlers from all IPL seasons, spanning from 2009 to 2024. The data collected for batsmen includes runs scored, matches and innings played, batting position, highest score, average runs, strike rate, centuries, half-centuries, and boundaries (4s and 6s). For bowlers, the data includes wickets taken, matches and innings played, overs bowled, runs conceded, bowling average, and economy rate.

2 Project Motivation

The goal of this project is to create a comprehensive dataset of IPL player statistics, which can then be used for future data analysis projects. While this current project focuses solely on scraping the data, it lays the groundwork for insightful analytics to explore trends, compare players' performances over time, and even predict outcomes based on past statistics.

3 Prerequisites

To run this project, a Firefox browser is required as the scraping process is not compatible with Chrome.

4 Methodology

The IPL statistics website has dedicated pages for each season. The methodology involves visiting each of these pages and scraping the data for both batsmen and bowlers. This was accomplished using two key Python libraries:

- **Selenium**: Used to navigate web pages and simulate user interactions, such as clicking buttons and drop-down menus.

- **BeautifulSoup**: Employed to parse the HTML content of the web pages and extract the relevant data from the tables.

Key Steps:

1. **Page Navigation**: Selenium is used to visit each season's web page, interact with the buttons and drop-down menus to view both batsmen and bowler data, and retrieve the table of statistics.
2. **Data Extraction**: Once the page loads the required data, BeautifulSoup parses the HTML table and extracts it into a structured format.
3. **Handling Web Elements**: Four critical web elements were handled: the "Accept Cookies" button, the "View All" button, the stats table, and a dropdown to switch between batsman and bowler data. Selenium interacted with these elements using CSS selectors, XPATH strategies, and JavaScript execution.
4. **Rate Limiting and Proxy Management**: Due to website rate limiting, proxy servers were employed to ensure continued access without blocking.
5. **Retry Mechanism**: A retry mechanism was implemented to handle JavaScript errors on some season pages, ensuring all data was eventually scraped.

5 Solution Design

Wait Times: Adding wait times to pause execution between steps allowed the content to fully load and mimicked human behavior to prevent the bot from being detected.

Scraping Batsmen and Bowlers: Each season page defaults to showing batsmen data. To retrieve bowler data, the project programmatically switches the dropdown to "Purple Cap" using Selenium.

Data Storage: The raw HTML data for each season is stored in text files. Each season has two files: one for batsmen data and one for bowlers. These are saved in the `./scraped_data` directory.

6 Addressing Rate Limit Issues

To overcome rate limit blocks, proxy servers were leveraged. By rotating between different IP addresses, the project avoided detection. The list of proxy servers was scraped from the website "free-proxy-list.net" using BeautifulSoup, and the data was processed using Pandas to select fresh IP addresses every 10 minutes. This method allowed smooth scraping without interruption due to IP bans.

7 Handling Website Errors

The website being scraped is slow and prone to errors, especially JavaScript issues. To address these:

- **Selenium Interrupts**: Used to introduce wait conditions, ensuring the page components fully loaded before interacting with them.
- **Retry Mechanism**: For failed pages, a retry mechanism was implemented to track errors and automatically attempt scraping again until all years were successfully processed.

8 Data Storage and Transformation

Raw Data: The scraped HTML data is stored in the `./scraped_data` directory as text files, with separate files for batsmen and bowlers for each season.

Transformation: The raw HTML data is processed using BeautifulSoup and Pandas to convert it into structured CSV files, which are then saved in the `./transformed_data` directory. This ensures the data is ready for future analysis.

9 Project Structure

- `scrapper.py`: Main file for scraping data from the IPL stats website.
- `proxies.py`: Script for scraping and managing proxy servers.
- `transform.py`: Script for transforming raw HTML data into CSV format.
- `./log`: Contains log files for both the scraping and transformation processes.
- `./scraped_data`: Directory containing raw HTML data.
- `./transformed_data`: Directory containing CSV files of the processed data.

10 Challenges Faced

- **Rate Limiting**: Websites block requests if they detect too many coming from the same IP. To overcome this, proxy servers were utilized, providing fresh IP addresses for each request. The list of proxy servers was scraped and refreshed periodically, ensuring uninterrupted scraping.
- **Website Slowness and Errors**: The target website was slow and frequently encountered JavaScript errors. Selenium's interrupts helped manage slowness by waiting for elements to load, while a retry mechanism was implemented to handle random JavaScript errors, ensuring all data was eventually scraped.

11 Conclusion

This IPL web scraping project successfully gathers a wealth of statistical data on both batsmen and bowlers from all IPL seasons, setting the stage for further data analysis. Despite the challenges posed by rate limiting, website errors, and slow load times, the project was able to handle these issues through innovative solutions like proxy servers, Selenium interrupts, and a retry mechanism.

This robust dataset can now be used to perform various types of data analysis, from tracking player performance over time to identifying patterns and trends in IPL seasons, providing valuable insights into one of the world's most exciting cricket leagues.