**Web Scraping Project**

Most Runs in ODI Cricket History

Name : Kamesh.K

Domain : Data analytics

Language : Python

Tool : Jupyter Notebook

Mentor : Mr.Madhavan Siva

**Table of Contents**

1. Introduction
2. Project Requirements
3. Installation of Dependencies
4. Web Scraping Process
5. Data Processing
6. Data Visualization
7. Conclusion
8. References

**1. Introduction**

**1.1 Project Overview:**

This project focuses on extracting and analyzing One Day International (ODI) cricket player statistics from a sports website. The key objective is to automate the collection of cricket data, organize it for analysis, and generate visual insights into player performances.

**1.2 Key Objectives:**

* Data Collection: Scraping cricket statistics (e.g., most runs, matches played, averages) using web scraping techniques.
* Data Analysis: Organizing the scraped data and performing statistical analysis to identify trends.
* Visualization: Creating graphical representations of the data to illustrate insight**s.**

1. **Project Requirements**

**2.1. Software & Environment**

* **Python (3.x)**: The project is built using Python, so you’ll need Python 3.x installed.
* **Jupyter Notebook (Optional)**: Recommended for organizing and running the project in an interactive environment.

**3 . Installation of Dependencies**

* **Requests**

Used to send HTTP requests to fetch the content of the webpage.

* **BeautifulSoup (bs4)**

Part of the bs4 package, it is used for parsing HTML and extracting useful information from web pages.

* **Pandas**

A powerful library used for data manipulation and analysis. In this project, it’s used for storing and organizing scraped data in a DataFrame.

* **Seaborn**

A library for statistical data visualization, built on top of matplotlib. It provides high-level functions to create beautiful visualizations easily.

* **Matplotlib**

A comprehensive library for creating static, animated, and interactive plots. It is essential for plotting data visualizations.

**4. Web Scraping Process**

The web scraping process in this project involves fetching data from a website, parsing the HTML content,

* **Sending HTTP Request to the Website**

The requests library is used to send an HTTP GET request to retrieve the webpage's HTML content.

* **Parsing HTML Content with BeautifulSoup**

The BeautifulSoup library from bs4 is used to parse the HTML content retrieved from the website.

* **Locating and Extracting Data**

You need to locate the specific HTML tags where the data (such as cricket statistics) is stored.

* **Extracting Specific Data (Player Statistics)**

In this project, the focus is on extracting cricket player statistics (such as player names, runs scored, and matches played). Each table cell contains specific data.

* **Storing Extracted Data**

The extracted data (player statistics) is stored in a pandas DataFrame for easy manipulation and further analysis.

**5. Data Processing**

Once the data has been scraped from the website, it is important to process and clean it before performing analysis and visualization.

**5.1 Organizing Data into a Pandas DataFrame**

After extracting the raw data from the webpageyou need to organize it into a structured format.

* 1. **Cleaning the Data**

The data extracted from the webpage might contain unwanted characters, missing values, or incorrectly formatted data.

**Steps in Data Cleaning**

* Removing Extra Whitespace: Sometimes data might have trailing or leading spaces, which need to be removed.
* Handling Missing Values: Some data might be incomplete so you'll need to handle missing values either by filling them or removing the corresponding rows.
* Converting Data Types: Often, numbers are extracted as strings. Converting them to numerical data types (e.g., int or float) makes analysis easier.
  1. **Handling Duplicates**

During data extraction, there is a possibility that duplicate entries might be included (e.g., the same player appearing multiple times)

* 1. **Adding New Columns or Features**

Depending on the analysis, you might need to add new calculated columns.

**5.5 Saving the Processed Data**

Once the data is cleaned and processed, it’s important to save it for future use. You can export the DataFrame to a CSV file.

**6. Data Visualization**

Once the data is scraped, cleaned, and processed, the next step is to visualize it to gain meaningful insights.

* **Bar Plot: Comparison of Players' Runs**

A bar plot is useful for comparing the total runs scored by different players. It provides a clear visual representation of the difference in runs between top players.

* **Line Plot: Runs vs Matches**

A line plot helps visualize the relationship between the number of matches played by a player and their total runs.

.

* **Scatter Plot: Runs per Match**

A scatter plot shows the relationship between two continuous variables.

* **Histogram: Distribution of Runs**

A histogram is used to show the distribution of a single variable, like the number of runs. .

* **Joint Plot: Matches vs Runs with Averages**

A joint plot is useful for visualizing the bivariate relationship between two variables along with their individual distributions. .

* **Violin Plot: Distribution of Player Averages**

A violin plot is a combination of a box plot and a KDE plot, showing the distribution of data across different categories.

**7.Conclusion**

The project aimed to analyze cricket player statistics using web scraping, data processing, and visualization techniques. Through this process, valuable insights into player performance, consistency, and career progress were derived.

**8.Reference**

* **Github Repository:**[**https://github.com/Kamesh2430/web-scraping/blob/main/Mini%20Project.ipynb**](https://github.com/Kamesh2430/web-scraping/blob/main/Mini%20Project.ipynb)
* **Csv Data**

[**https://github.com/Kamesh2430/web-scraping/blob/main/cricketdatas.csv**](https://github.com/Kamesh2430/web-scraping/blob/main/cricketdatas.csv)