Capstone Project Report on

Stanford Researchers Database Analysis and Ranking

submitted in partial fulfillment (8 Course Credits) of the

requirements for the award of the degree of

Masters of Technology

in

Computer Science and Engineering

By:

Dwaipayan Mondal (MT23035)

Parnita Bokade (MT23055)

Pragati Agrawal (MT23059)

Prolay Shankar Mazumder (MT23065)

under the guidance of

Dr.N. Arul Murugan

(Associate Professor, CB)

Department of Computer Science & Engineering

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY **DELHI**

Table of Contents

1. **Abstract** -- 2

2. **Relevant Tools** --- 2

3. **Web Scraping** -- 3

4. **Data Analysis/Methodologies** ------------------------------- 7

5. **Results** -- 14

6. **Implication**—---19

7. **Future Work**—--19

8. **Conclusion**—---19

9. **References**—---20

1. ABSTRACT

In the modern digital age, data in various forms such as numbers, words, images, and symbols are produced at an unprecedented rate—estimated at 2.5 quintillion bytes every day. This explosion of data has led to the rise of data science, a discipline focused on gathering, analyzing, and interpreting data to derive insights and inform decision-making. Data science has become integral to numerous sectors, including marketing, sports, healthcare, education, and finance, playing a crucial role for corporations, governments, and individuals.

Despite the abundance of detailed information on university ranking websites, these platforms often lack comprehensive statistics or visual representations to assist users in making informed decisions. Leveraging data science techniques and the Python programming language, this report aims to address these issues by utilizing Python libraries to collect and analyze data from the Stanford Researchers database. The goal is to rank subjects, university, departments, publication and professors based on various factors like H-index, i10 index, Total citations, Journal impact factor etc.

2. RELEVANT TOOLS

2.1 Python libraries for Web Scraping and Data Analysis

1) **Cloudscraper**: This library is an extension of the popular **requests** library, designed to bypass anti-bot measures used by some websites. It was used to handle web scraping tasks efficiently.
2) **BeautifulSoup**: A library for parsing HTML and XML documents, BeautifulSoup is used to extract data from web pages. In this project, it was used to parse HTML content and extract relevant information from the AD Scientific Index and Google Scholar pages.
3) **Requests**: A simple yet powerful HTTP library for making web requests in Python. It was used for sending HTTP requests to web servers to fetch HTML content.

2.2 Concurrency Tools

1) **Threading and ThreadPoolExecutor**: Python's threading module and **ThreadPoolExecutor** from the concurrent.futures module were used to manage concurrent web scraping tasks. These tools helped speed up the scraping process by allowing multiple pages to be fetched and processed simultaneously.

2.3 Flask for Web Development

A lightweight web framework for Python, Flask was used to build a web application for presenting the scraped and processed data. It allowed for the development of a dynamic website where users could interact with the data, such as viewing top-ranked professors, exploring subject-specific rankings, and seeing college rankings.

2.4 Project-Specific Functionalities

1) **Data Scraping and Parsing**: The code demonstrates robust methods for scraping data from web pages, handling pagination, and parsing HTML content to extract meaningful data.
2) **Google Scholar Integration**: Special attention was given to fetching Google Scholar profiles and research papers of professors, showcasing the integration of external data sources to enrich the dataset.
3) **Data Presentation**: The use of Flask to create a web interface for presenting the data highlights the importance of making data accessible and interpretable to users.

3. WEB SCRAPING

3.1 Web Scraping

Web scraping is an automated method used to collect vast amounts of data from websites for purposes such as data analysis, training machine learning algorithms, market research, and more. Typically, the data extracted is in an unstructured HTML format, which is then converted into structured data, such as a spreadsheet or a database, making it usable for various applications.

Web scraping involves extracting data from websites using automated software or tools. This process requires writing code to automate the navigation of web pages, identify relevant data, and extract it for further analysis. By transforming unstructured data into a structured format, web scraping enables efficient data processing and utilization across different fields.

3.2 Operation of Web Scrapers

Web scrapers can extract either all the data or specific data from a target website. The process begins by providing the URLs of the websites from which data needs to be extracted. This step identifies the target sites. Next, the structure of the website must be analyzed to identify the specific data to be extracted, along with the relevant tags and attributes in the HTML or XML code.

Once the data is identified, the scraper program accesses the website, extracts the required data, and saves it in the specified format, such as a CSV file or JSON file. This systematic approach ensures that the desired data is efficiently and accurately collected and stored for further use.

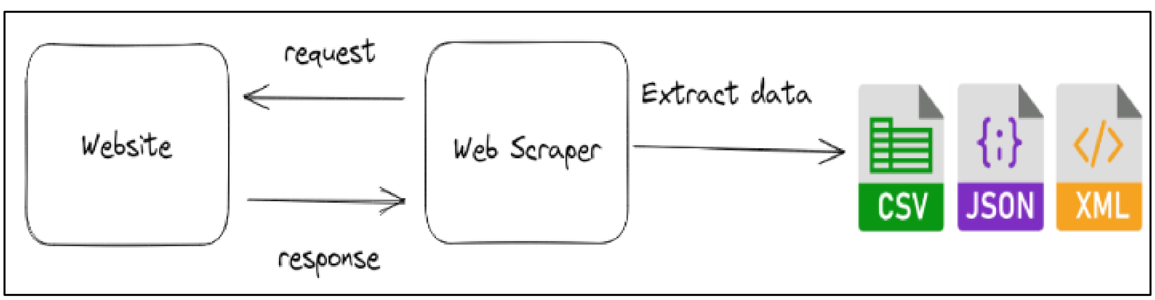


Fig 1: Web scraper workflow diagram

3.3 Web Scraper Program

3.3.1 Installing the Beautiful Soup

Before creating web scrapers, we ensure that the latest version of Python is downloaded from the official Python website. Once Python is installed, we install BeautifulSoup4 using pip, the Python package installer, with the following command:

pip install beautifulsoup4

This command will install the latest version of BeautifulSoup and its dependencies. After the installation is complete, we can verify it by entering the following command in the Python shell:

from bs4 import BeautifulSoup

This confirms that BeautifulSoup 4 has been successfully installed and is ready to use for web scraping projects.

3.3.2 Scraping Data from URL

Step 1: The process begins by creating a **cloudscraper** instance, which is an enhanced version of the **requests** library designed to bypass anti-bot mechanisms commonly found on websites. This scraper is used to send HTTP requests and retrieve HTML content from the target URLs.

Step 2: Fetching and Parsing Data from AD Scientific Index- The base URL for Stanford University's page on the AD Scientific Index is defined. A range of pages is specified, and the ThreadPoolExecutor is employed to concurrently fetch data from these pages. Each page contains a table listing scientists and their profiles.

Step 3: Extracting Google Scholar URLs - Once the initial data is collected, the code focuses on extracting Google Scholar URLs for each scientist. For each scientist's profile URL, the scraper sends a request to retrieve the HTML content of their profile page. The profile page is parsed to find the div containing the Google Scholar URL. If found, this URL is added to the scientist's data.

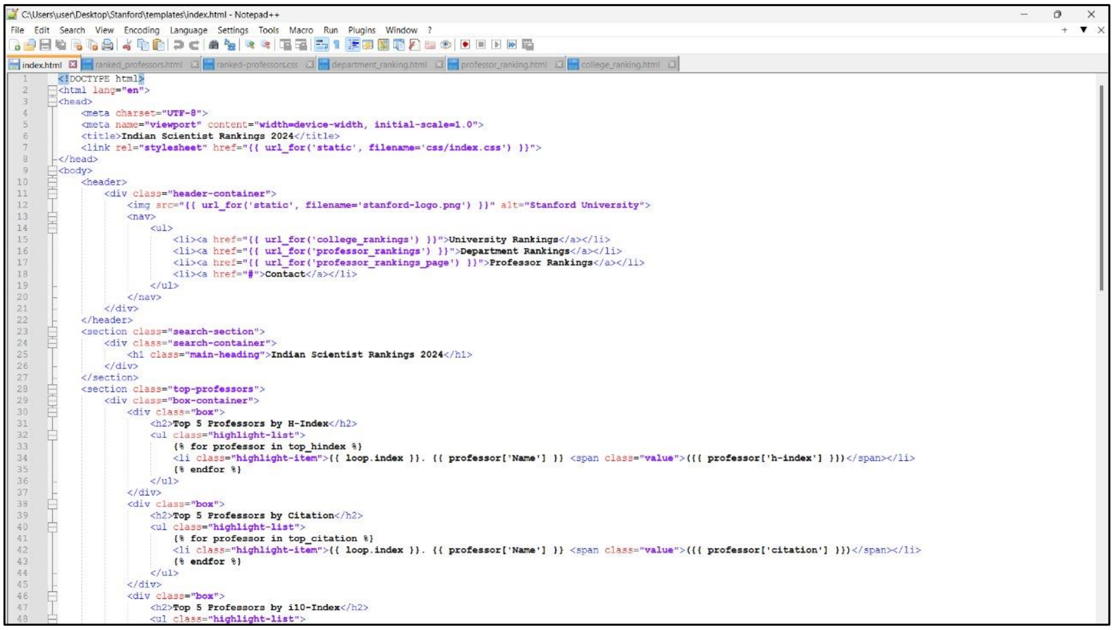
Step 4: The final dataset is neatly organized and saved as a CSV file for further analysis or presentation.

```python
import cloudscraper
from bs4 import BeautifulSoup
import pandas as pd
import threading
from concurrent.futures import ThreadPoolExecutor, as_completed

# Base URL of the Stanford University page on AD Scientific Index
base_url = "https://www.adscientificindex.com"

# List to store all scientists' data
data = []
data_lock = threading.Lock()

# Function to parse the scientist data from a single page
def parse_page(soup):
    table = soup.find('table', {'class': 'table table-striped table-bordered table-sm'})
    if table:
        rows = table.find_all('tr')
        page_data = []
        for row in rows[1:]:  # Skip the header row
            cols = row.find_all('td')
            parsed_cols = []
            profile_url = ""
            for col in cols:
                if col.find('a'):
                    a_tag = col.find('a')
                    if 'subject=' in a_tag['href']:
                        subject = col.find('a').text.strip()
                        parsed_cols.append(subject)
```

Fig 2: Code snippet for demo web scrapper

```python
# Create a cloudscraper instance
scraper = cloudscraper.create_scraper()

# Use ThreadPoolExecutor to fetch and parse pages concurrently
num_threads = 10
pages = range(0, 10000, 50)

with ThreadPoolExecutor(max_workers=num_threads) as executor:
    future_to_page = {executor.submit(fetch_and_parse_page, page): page for page in pages}
    for future in as_completed(future_to_page):
        page_data = future.result()
        with data_lock:
            data.extend(page_data)

# Use ThreadPoolExecutor to fetch Google Scholar URLs concurrently
with ThreadPoolExecutor(max_workers=num_threads) as executor:
    future_to_url = {executor.submit(fetch_google_scholar_url, row[-1]): row for row in data}
    for future in as_completed(future_to_url):
        google_scholar_url = future.result()
        row = future_to_url[future]
        row.append(google_scholar_url)

# Analyze the structure of the data to ensure consistent columns
max_columns = max(len(row) for row in data)

# Ensure all rows have the same number of columns
for row in data:
    while len(row) < max_columns:
        row.append('')
```

Fig 3: Code snippet for demo web scrapper (used multi-threading)



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Indian Scientist Rankings 2024</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/index.css') }}">
</head>
<body>
    <header>
        <div class="header-container">
            <img src="{{ url_for('static', filename='stanford-logo.png') }}" alt="Stanford University">
            <nav>
                <ul>
                    <li><a href="{{ url_for('college_rankings') }}">University Rankings</a></li>
                    <li><a href="{{ url_for('professor_rankings') }}">Department Rankings</a></li>
                    <li><a href="{{ url_for('professor_rankings_page') }}">Professor Rankings</a></li>
                    <li><a href="#">Contact</a></li>
                </ul>
            </nav>
        </div>
    </header>
    <section class="search-section">
        <div class="search-container">
            <h1 class="main-heading">Indian Scientist Rankings 2024</h1>
        </div>
    </section>
    <section class="top-professors">
        <div class="box-container">
            <div class="box">
                <h2>Top 5 Professors by H-Index</h2>
                <ul class="highlight-list">
                    {% for professor in top_hindex %}
                    <li class="highlight-item">{{ loop.index }}. {{ professor['Name'] }} <span class="value">({{ professor['h-index'] }})</span></li>
                    {% endfor %}
                </ul>
            </div>
            <div class="box">
                <h2>Top 5 Professors by Citation</h2>
                <ul class="highlight-list">
                    {% for professor in top_citation %}
                    <li class="highlight-item">{{ loop.index }}. {{ professor['Name'] }} <span class="value">({{ professor['citation'] }})</span></li>
                    {% endfor %}
                </ul>
            </div>
            <div class="box">
                <h2>Top 5 Professors by i10-Index</h2>
                <ul class="highlight-list">
```

Fig 4: HTML content of Homepage of Stanford Researchers website

4. DATA ANALYSIS/METHODOLOGIES

4.1 The Process of Data Analysis

Step 1: Data Cleaning and Preparation - The data scraped from the AD Scientific Index and Google Scholar is loaded into a Pandas DataFrame. Certain columns, such as citation and i10-index, are cleaned by removing commas and converting the data to integer types to facilitate numerical operations. The code ensures that all rows have the same number of columns by dynamically adjusting the structure, filling in missing values where necessary.

Step 2: Sorting and Ranking - The data is sorted based on various academic metrics such as h-index, citation, and i10-index. For each metric, the top 5 professors are identified and stored in dictionaries for easy retrieval and display. The professors are grouped by their subject areas, and the average score for each department is calculated. Departments are then ranked based on these average scores to identify the top-performing departments.

Step 3: Flask Web Application for Data Presentation - The Flask web application serves an index page that displays the top 5 professors by h-index, citation, and i10-index, as well as the top 5 departments and colleges. This provides users with a quick overview of the key academic figures and departments at Stanford. Users can select different criteria (e.g., h-index, citation, i10-index) to dynamically sort and view the data. The sorted data is paginated and displayed, allowing users to navigate through large datasets efficiently. The application provides a dedicated page for each subject, showing the rankings of professors within that subject. This allows users to explore academic performance within specific fields of study. Another page in the application presents the rankings of colleges, sorted by their overall rank.

Step 4: The final step of the data analysis process is to visualize and share the insights. Depending on what you share, the result of data analysis provides useful information for organizations and highlights all the details which were gathered and analyzed.

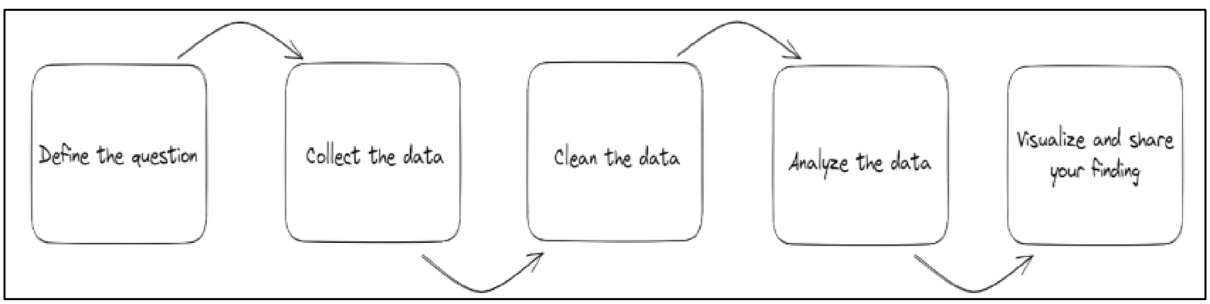


Fig 5: The process of a data analysis diagram

4.2 Professor Ranking Data Analysis

```python
import pandas as pd
from scipy.stats import rankdata

# Load the data
data_path = 'professors_data.csv'
professors_data = pd.read_csv(data_path)

# Convert the columns to strings first to ensure we can use .str methods
professors_data['citation'] = professors_data['citation'].astype(str)
professors_data['h-index'] = professors_data['h-index'].astype(str)
professors_data['i10-index'] = professors_data['i10-index'].astype(str)

# Remove commas and convert columns to integers
professors_data['citation'] = professors_data['citation'].str.replace(',', '').astype(int)
professors_data['h-index'] = professors_data['h-index'].str.replace(',', '').astype(int)
professors_data['i10-index'] = professors_data['i10-index'].str.replace(',', '').astype(int)

# Group by Subject and calculate the number of professors in each subject
professors_data['num_professors'] = professors_data.groupby('Subject')['Subject'].transform('count')

# Normalize and calculate the score
professors_data['normalized_citation'] = professors_data['citation'] / professors_data['num_professors']
professors_data['normalized_h-index'] = professors_data['h-index'] / professors_data['num_professors']
professors_data['normalized_i10-index'] = professors_data['i10-index'] / professors_data['num_professors']

# Calculate a combined score (equal weights for all metrics)
professors_data['score'] = (professors_data['normalized_citation'] + professors_data['normalized_h-index'] +
                            professors_data['normalized_i10-index']) / 3

# Rank within each subject using scipy.stats.rankdata to ensure ties have the same rank
def compute_rank(group):
    group['rank'] = rankdata(-group['score'], method='dense').astype(int)
    return group
professors_data = professors_data.groupby('Subject').apply(compute_rank)

# Save cleaned and processed data
processed_data_path = 'normalised_data.csv'
professors_data.to_csv(processed_data_path, index=False)

print("Data cleaning and processing complete. Check", processed_data_path, "for the results.")
```

Fig 6: Code snippet for Professor ranking based on H-index, i10 index and Total Citation

The above figure consists of code that efficiently processes and ranks professors based on three key academic metrics: h-index, i10-index, and total citations. The steps involve cleaning and normalizing the data, followed by calculating a combined score for comprehensive ranking.

Step 1: Data Cleaning and Preparation - The process begins by loading the raw data from a CSV file and converting the metrics (citation, h-index, and i10-index) from strings to integers, after removing any commas. This ensures that the data is in a consistent format suitable for numerical operations.

Step 2: Normalization of Metrics - To account for variations in the number of professors across different subjects, the metrics are normalized by dividing each professor's citation, h-index, and i10-index by the total number of professors in their respective subject.

Step 3: Calculation of Combined Score - A combined score is then calculated by taking the average of the normalized metrics. This score provides an overall assessment of each professor's academic impact, considering citations, h-index, and i10-index equally.

Step 4: Ranking Within Subjects - Using the combined score, professors are ranked within their respective subjects. The ranking is performed using the rankdata function from the scipy.stats module, which assigns ranks in such a way that ties receive the same rank.

Step 5: Saving the Processed Data - The final step involves saving the cleaned, normalized, and ranked data to a new CSV file. This processed data includes the original metrics, normalized values, combined score, and the ranks within each subject.

4.3 Department Ranking Data Analysis

```
import pandas as pd
from scipy.stats import rankdata

# Load the processed data
data_path = 'normalised_data.csv'
professors_data = pd.read_csv(data_path)

# Group by College and Department, then sum the normalized metrics
department_scores = professors_data.groupby(['University / Institution', 'Subject']).agg(
    total_normalized_citation=('normalized_citation', 'sum'),
    total_normalized_h_index=('normalized_h-index', 'sum'),
    total_normalized_i10_index=('normalized_i10-index', 'sum')
).reset_index()

# Calculate the combined score for each department
department_scores['score'] = (
    department_scores['total_normalized_citation'] +
    department_scores['total_normalized_h_index'] +
    department_scores['total_normalized_i10_index']
) / 3

# Rank the colleges department-wise based on the combined score
department_scores['rank'] = department_scores.groupby('Subject')['score'].rank(
    method='dense', ascending=False).astype(int)

# Save the ranked data
ranked_data_path = 'processed_professors_data.csv'
department_scores.to_csv(ranked_data_path, index=False)

print("Department ranking complete. Check", ranked_data_path, "for the results.")
```

Fig 7: Code snippet for Department Ranking

The above figure consists of code which performs a comprehensive ranking of departments within various universities based on aggregated academic metrics. This process involves grouping data by university and department, summing normalized metrics, calculating combined scores, and ranking the departments within each subject.

Step 1: Data Aggregation by Department – Initially, the code loads the processed data containing normalized academic metrics for individual professors. It then groups this data by university and department, summing the normalized citation counts, h-index, and i10-index for each department.

Step 2: Calculation of Combined Scores - For each department, a combined score is calculated by averaging the total normalized citation, h-index, and i10-index.

Step 3: Ranking Within Subjects - Departments are ranked within their respective subjects based on the combined score. The ranking is performed using the rank function from the Pandas library, with the dense method ensuring that ties receive the same rank.

Step 4: Saving the Ranked Data - The final ranked data, including the university name, department, summed normalized metrics, combined score, and rank, is saved to a new CSV file.

4.4 College Ranking Data Analysis

```python
import pandas as pd
from scipy.stats import rankdata

# Load the processed data
data_path = 'processed_professors_data.csv'
professors_data = pd.read_csv(data_path)

# Sum up the scores for all the departments for each college
college_scores = professors_data.groupby('University / Institution').agg(
    total_score=('score', 'sum')
).reset_index()

# Rank the colleges based on the summed scores
college_scores['rank'] = rankdata(-college_scores['total_score'], method='dense').astype(int)

# Sort by rank to ensure the ranking is in order
college_scores = college_scores.sort_values('rank')

# Select the desired columns for the output
output_columns = ['rank', 'University / Institution', 'total_score']
college_scores = college_scores[output_columns]

# Save the ranked data
ranked_colleges_path = 'ranked_colleges.csv'
college_scores.to_csv(ranked_colleges_path, index=False)

print("College ranking complete. Check", ranked_colleges_path, "for the results.")
```

Fig 8: Code snippet for College Ranking

The above figure consists of code that calculates and ranks colleges based on the aggregated scores of their respective departments, offering a holistic view of institutional academic performance. This process involves grouping data by college, summing departmental scores, and ranking colleges based on the total scores.

Step 1: Aggregation of Departmental Scores - The code begins by loading processed data, which includes the scores of various departments within different colleges. It then groups this data by college, summing the scores of all departments within each institution.

Step 2: Calculation and Ranking of Total Scores - Using the aggregated departmental scores, the code calculates a total score for each college. The colleges are then ranked based on these total scores. The rankdata function from the scipy.stats module is used to assign ranks, with the dense method ensuring that ties receive the same rank.

Step 3: Sorting and Output - To ensure the rankings are presented in an easily interpretable format, the colleges are sorted by rank. The final output includes the rank, college name, and total score for each institution.

4.5 Impact Score Calculation Analysis

```python
import pandas as pd

# Load the merged CSV file
merged_data = pd.read_csv('Impact Factor Calculation\merged_professors_profiles_with_JCI_upto10000.csv')

# Calculate the total JCI and total count of research papers for each profile name
summary_data = merged_data.groupby('Profile Name').agg(
    Total_Impact_Factor=('JCI', 'sum'),
    Total_Publications=('Research Paper', 'count')
).reset_index()

# Calculate the impact score
summary_data['Impact_Score'] = summary_data['Total_Impact_Factor'] / summary_data['Total_Publications']

# Save the summary dataframe to a new CSV file
summary_data.to_csv('Impact Score Calculation\profile_impact_summary_upto10000.csv', index=False)

# Print a message to indicate the task is completed successfully
print("The profile impact summary has been saved to 'profile_impact_summary_upto10000.csv' successfully.")
```

Fig 9: Code Snippet for Impact Score Calculation

The above figure consists of code that focuses on calculating an impact score for academic profiles by analyzing their research publications and Journal Citation Indicator (JCI). The process involves loading merged data, calculating total JCI and publication count, and then computing the impact score for each profile.

Step 1: Data Loading and Aggregation - The code begins by loading a merged CSV file that contains detailed information about academic profiles, including their research papers and JCI values. It then groups the data by profile name, calculating the total JCI and the total number of research papers for each profile.

Step 2: Calculation of Impact Score - For each profile, the impact score is calculated by dividing the total JCI by the total number of publications. This score reflects the average impact of a profile's research output, offering a normalized metric to compare the research influence across different profiles.

Step 3: Saving and Output - The resulting summary data, which includes the profile name, total JCI, total publications, and impact score, is saved to a new CSV file.

5. RESULTS

Total Professor/Scientists	91946
Total Department	203
Total College/ University	3963
Total Publications	3068163 (~ 3.06 million)
Total Journals	21507

Fig 10: Table containing total Data scraped

The analysis performed across various stages involved the extraction, cleaning, processing, and ranking of data from academic profiles, departments, and colleges. The results reflect the comprehensive evaluation of academic performance based on key metrics such as citations, h-index, i10-index, and Journal Citation Indicator (JCI).

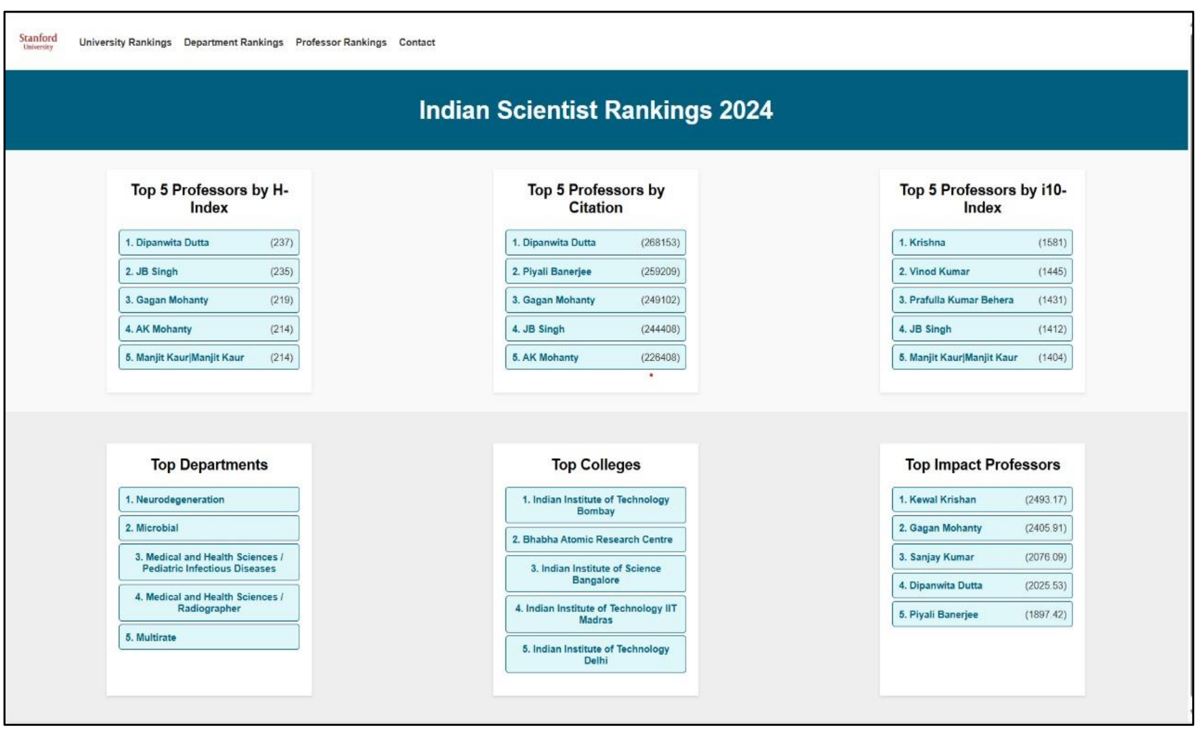


Fig 11: Homepage for Stanford Ranking System

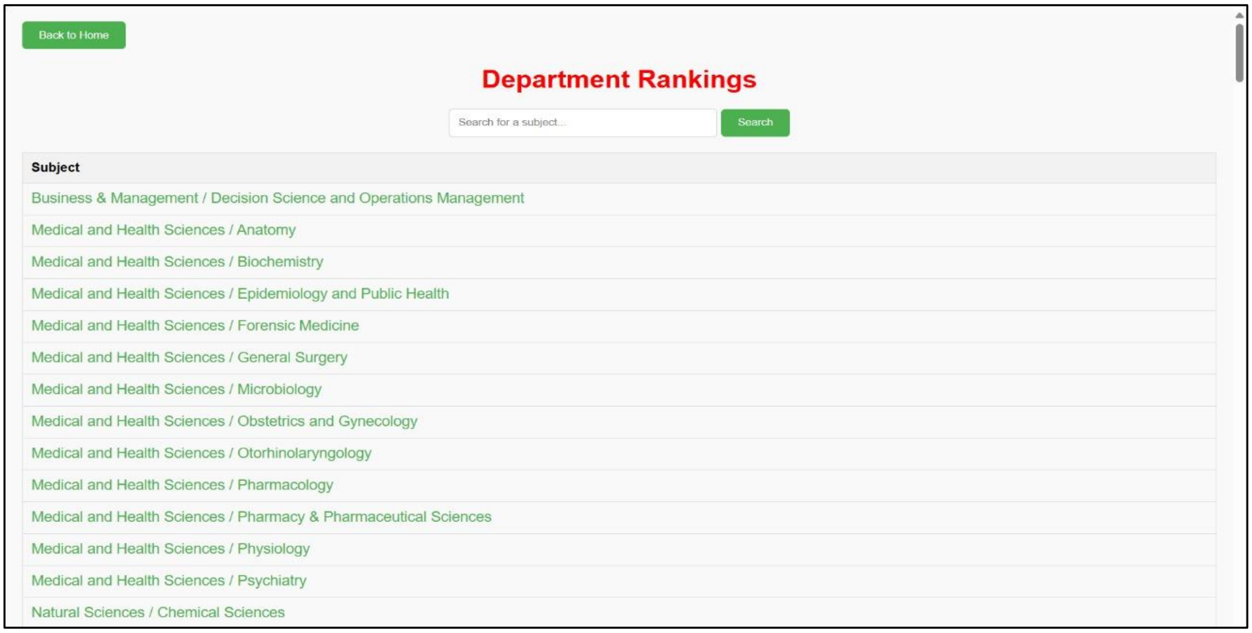
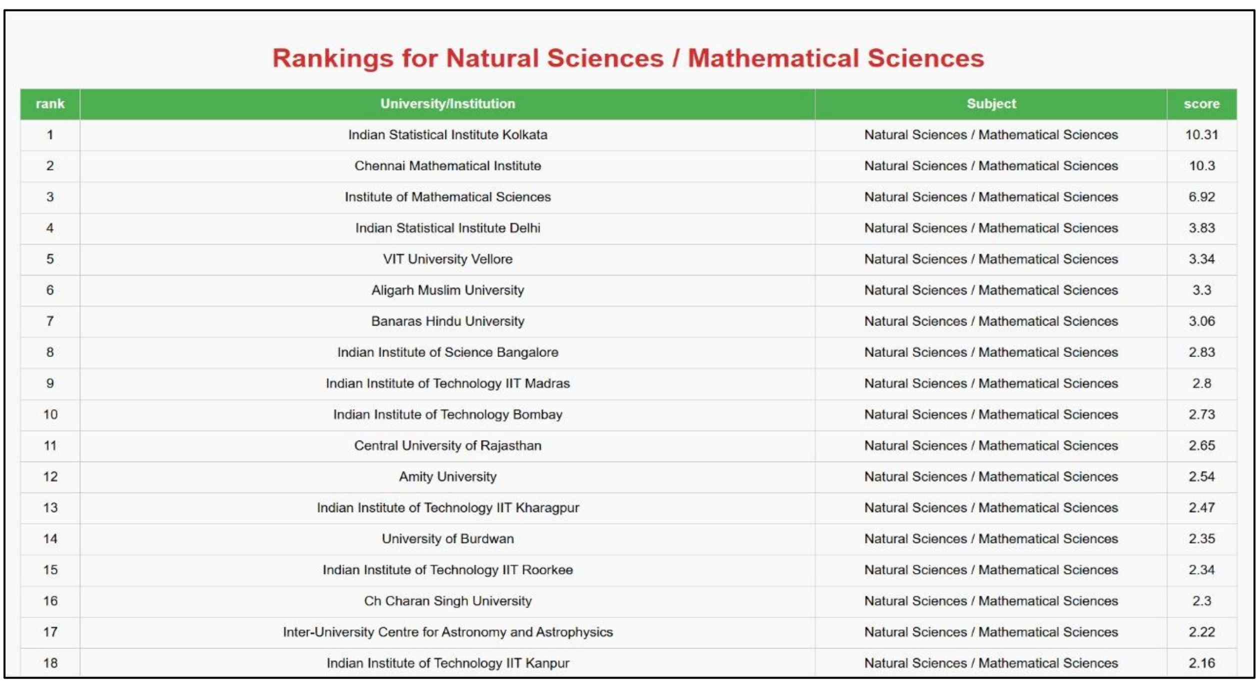


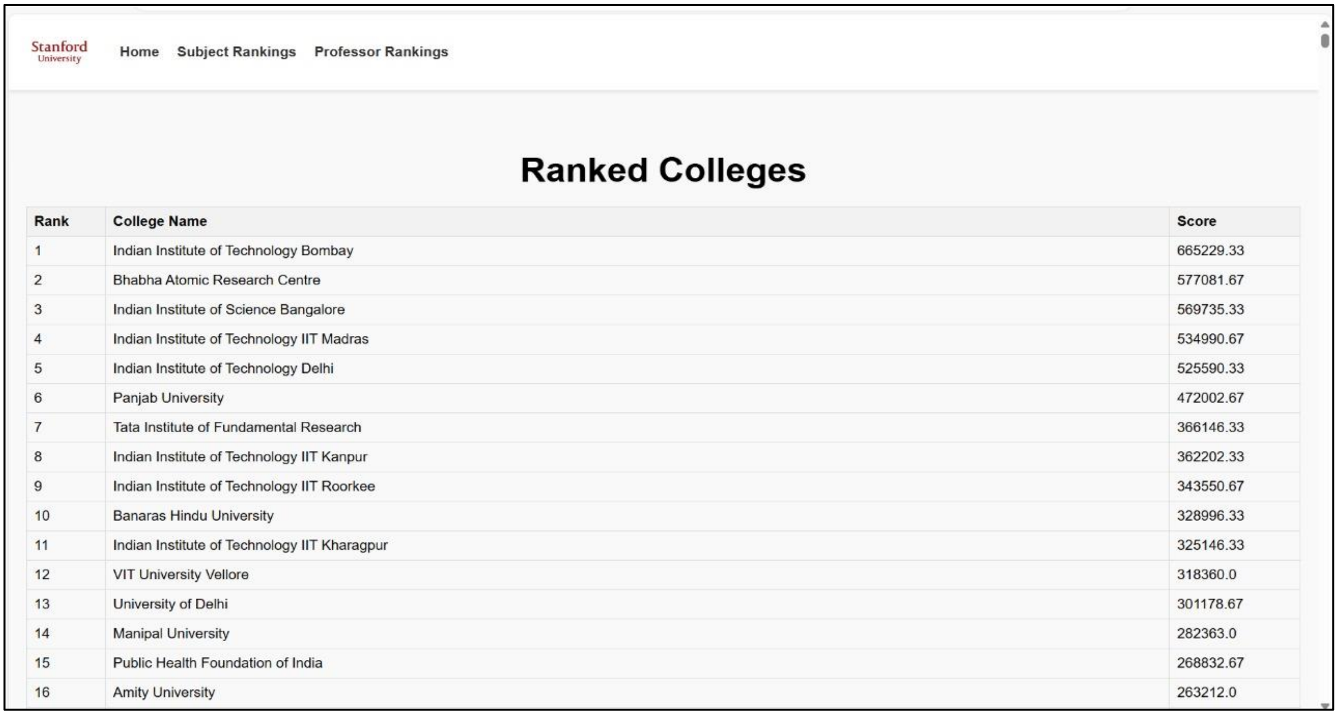
Fig 12: Department Ranking Page



Rankings for Natural Sciences / Mathematical Sciences

rank	University/Institution	Subject	score
1	Indian Statistical Institute Kolkata	Natural Sciences / Mathematical Sciences	10.31
2	Chennai Mathematical Institute	Natural Sciences / Mathematical Sciences	10.3
3	Institute of Mathematical Sciences	Natural Sciences / Mathematical Sciences	6.92
4	Indian Statistical Institute Delhi	Natural Sciences / Mathematical Sciences	3.83
5	VIT University Vellore	Natural Sciences / Mathematical Sciences	3.34
6	Aligarh Muslim University	Natural Sciences / Mathematical Sciences	3.3
7	Banaras Hindu University	Natural Sciences / Mathematical Sciences	3.06
8	Indian Institute of Science Bangalore	Natural Sciences / Mathematical Sciences	2.83
9	Indian Institute of Technology IIT Madras	Natural Sciences / Mathematical Sciences	2.8
10	Indian Institute of Technology Bombay	Natural Sciences / Mathematical Sciences	2.73
11	Central University of Rajasthan	Natural Sciences / Mathematical Sciences	2.65
12	Amity University	Natural Sciences / Mathematical Sciences	2.54
13	Indian Institute of Technology IIT Kharagpur	Natural Sciences / Mathematical Sciences	2.47
14	University of Burdwan	Natural Sciences / Mathematical Sciences	2.35
15	Indian Institute of Technology IIT Roorkee	Natural Sciences / Mathematical Sciences	2.34
16	Ch Charan Singh University	Natural Sciences / Mathematical Sciences	2.3
17	Inter-University Centre for Astronomy and Astrophysics	Natural Sciences / Mathematical Sciences	2.22
18	Indian Institute of Technology IIT Kanpur	Natural Sciences / Mathematical Sciences	2.16

Fig 13: Ranking for "Rankings for Natural Sciences / Mathematical Sciences" Department



Rank	College Name	Score
1	Indian Institute of Technology Bombay	665229.33
2	Bhabha Atomic Research Centre	577081.67
3	Indian Institute of Science Bangalore	569735.33
4	Indian Institute of Technology IIT Madras	534990.67
5	Indian Institute of Technology Delhi	525590.33
6	Panjab University	472002.67
7	Tata Institute of Fundamental Research	366146.33
8	Indian Institute of Technology IIT Kanpur	362202.33
9	Indian Institute of Technology IIT Roorkee	343550.67
10	Banaras Hindu University	328996.33
11	Indian Institute of Technology IIT Kharagpur	325146.33
12	VIT University Vellore	318360.0
13	University of Delhi	301178.67
14	Manipal University	282363.0
15	Public Health Foundation of India	268832.67
16	Amity University	263212.0

Fig 14: College Ranking Page

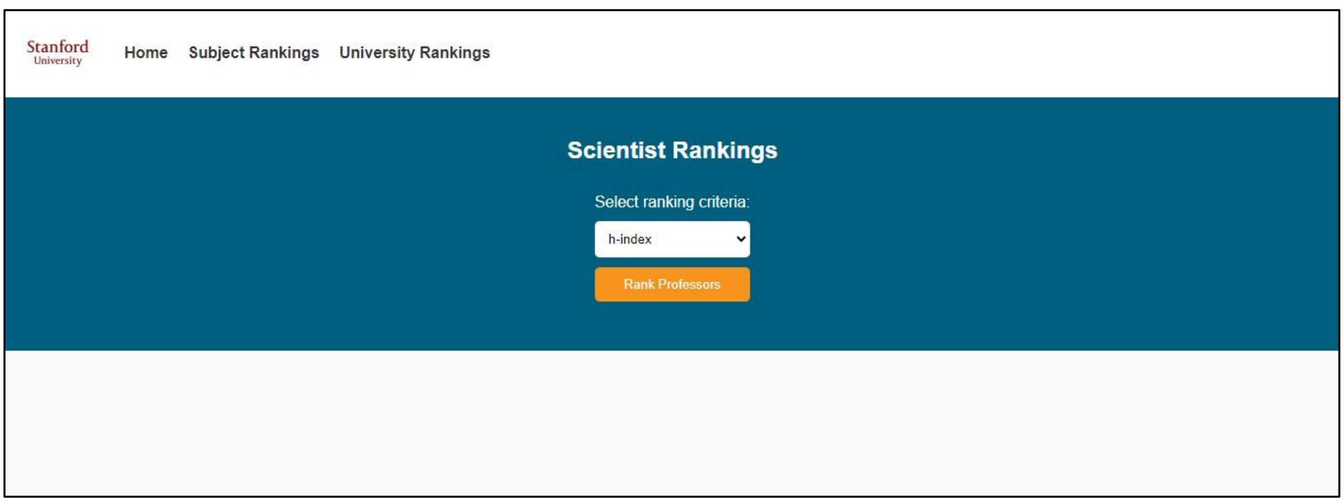


Fig 15: Professor Ranking Homepage

Back to Professor Rankings

Ranked Professors

Rank	Name	Country	University / Institution	Subject	H-Index	i10-Index	Citation
1	Dipanwita Dutta	India	Bhabha Atomic Research Centre	Natural Sciences / Physics	237	1320	268153
2	JB Singh	India	Panjab University	Natural Sciences / Physics	235	1412	244408
3	Gagan Mohanty	India	Tata Institute of Fundamental Research	Natural Sciences / Physics	219	1337	249102
4	AK Mohanty	India	Bhabha Atomic Research Centre	Natural Sciences / Physics	214	1011	226408
5	Manjit Kaur|Manjit Kaur	India	Panjab University	Natural Sciences / Physics	214	1404	211817
6	Anirban Saha	India	Shoolini University of Biotechnology and Management Sciences	Natural Sciences / Physics	212	873	185465
7	Prafulla Kumar Behera	India	Indian Institute of Technology IIT Madras	Natural Sciences / Physics	206	1431	220810
8	DP Roy	India	Tata Institute of Fundamental Research	Natural Sciences / Physics	204	1158	165238
9	Krishna	India	Aryabhatta Knowledge University	Engineering & Technology / Computer Science	203	1581	178548

Fig 16: Professor Ranking by H-index

Back to Professor Rankings

Ranked Professors

Rank	Name	Country	University / Institution	Subject	H-Index	i10-Index	Citation
1	Dipanwita Dutta	India	Bhabha Atomic Research Centre	Natural Sciences / Physics	237	1320	268153
2	Piyali Banerjee	India	Indian Institute of Technology Bombay	Natural Sciences / Physics	202	1216	259209
3	Gagan Mohanty	India	Tata Institute of Fundamental Research	Natural Sciences / Physics	219	1337	249102
4	JB Singh	India	Panjab University	Natural Sciences / Physics	235	1412	244408
5	AK Mohanty	India	Bhabha Atomic Research Centre	Natural Sciences / Physics	214	1011	226408
6	Dorairaj Prabhakaran	India	Public Health Foundation of India	Medical and Health Sciences / Cardiology	109	451	224718
7	Prafulla Kumar Behera	India	Indian Institute of Technology IIT Madras	Natural Sciences / Physics	206	1431	220810
8	Prolay Mal	India	National Institute of Science Education and Research	Natural Sciences / Physics	191	977	216487
9	Manjit Kaur|Manjit Kaur	India	Panjab University	Natural Sciences / Physics	214	1404	211817

Fig 17: Professor Ranking by Total Citation

Back to Professor Rankings

Ranked Professors

Rank	Name	Country	University / Institution	Subject	H-Index	i10-Index	Citation
1	Krishna	India	Aryabhatta Knowledge University	Engineering & Technology / Computer Science	203	1581	178548
2	Vinod Kumar	India	Jaypee University of Information Technology Waknaghat	Engineering & Technology / Computer Science	118	1445	84748
3	Prafulla Kumar Behera	India	Indian Institute of Technology IIT Madras	Natural Sciences / Physics	206	1431	220810
4	JB Singh	India	Panjab University	Natural Sciences / Physics	235	1412	244408
5	Manjit Kaur|Manjit Kaur	India	Panjab University	Natural Sciences / Physics	214	1404	211817
6	Gagan Mohanty	India	Tata Institute of Fundamental Research	Natural Sciences / Physics	219	1337	249102
7	Dipanwita Dutta	India	Bhabha Atomic Research Centre	Natural Sciences / Physics	237	1320	268153
8	Sabu Thomas|SABU THOMAS, S.THOMAS, Thomas S	India	Mahatma Gandhi University Kerala	Natural Sciences / Chemical Sciences	142	1303	101041

Fig 18: Professor Ranking by i10 Index

Back to Professor Rankings

Impact Factor Rankings

Rank	Profile Name	University / Institution	Subject	Total Impact Factor
1	Kewal Krishan	Panjab University	Social Sciences / Anthropology	2493.17
2	Gagan Mohanty	Tata Institute of Fundamental Research	Natural Sciences / Physics	2405.91
3	Sanjay Kumar	Central University of South Bihar	Others	2076.09
4	Dipanwita Dutta	Bhabha Atomic Research Centre	Natural Sciences / Physics	2025.53
5	Piyali Banerjee	Indian Institute of Technology Bombay	Natural Sciences / Physics	1897.42
6	Dorairaj Prabhakaran	Public Health Foundation of India	Medical and Health Sciences / Cardiology	1792.1
7	Vipin Bhatnagar	Panjab University	Natural Sciences / Physics	1638.91
8	Prolay Mal	National Institute of Science Education and Research	Natural Sciences / Physics	1620.76
9	Vanita Noronha	Tata Memorial Centre	Medical and Health Sciences / Medical Oncology	1580.81
10	sunil bansal	Panjab University	Natural Sciences / Physics	1571.57
11	Dr.V Kumar	Indian School of Business Hyderabad	Business & Management / Marketing	1536.12
12	Suneel Dutt	Dr Bhim Rao Ambedkar National Institute of Technology	Natural Sciences / Physics	1510.19
13	Vivekanand Jha	The George Institute for Global Health India	Medical and Health Sciences / Nephrology	1490.7
14	Naba Mondal	Tata Institute of Fundamental Research	Natural Sciences / Physics	1424.98
15	Vijay Patil	Tata Memorial Centre	Medical and Health Sciences / Medical Oncology	1377.11
16	Anirban Saha	Shoolini University of Biotechnology and Management Sciences	Natural Sciences / Physics	1284.68

Fig 19: Professor Ranking by Impact Factor

6. IMPLICATIONS

The results of this study are invaluable for various stakeholders. Academic institutions can leverage these insights to enhance their strategic planning, focusing on strengthening their departments and fostering research excellence. Prospective students and researchers can use the rankings to make informed choices about their educational and career paths, selecting institutions and departments that align with their academic goals.

7. FUTURE WORK

The methodologies and frameworks developed in this study can be extended and refined for broader applications. Future work could involve:

- ➔ Expanding the dataset to include more institutions and a wider range of academic metrics.
- ➔ Incorporating additional factors such as funding, collaboration networks, and societal impact to provide a more holistic evaluation.
- ➔ Developing interactive platforms and dashboards to visualize the data and make it accessible to a wider audience.

8. CONCLUSION

The extensive data analysis carried out in this study provides a comprehensive evaluation of academic performance across multiple dimensions—individual professors, departments, and colleges. Utilizing advanced data scraping, cleaning, and processing techniques, key metrics such as h-index, i10-index, total citations, and Journal Citation Indicator (JCI) were meticulously analyzed to derive meaningful insights.

9. REFERENCES

[1] https://www.adscientificindex.com/?country_code=in

[2] https://flask.palletsprojects.com/en/3.0.x/

[3] https://scholar.google.com/

[4] https://mjl.clarivate.com/home