Data Collection using Web Scraping & Visualization for Job Market Analysis

A Case Study on Data Scientist Jobs in India from Naukri.com

Author: Muhammad Kamran Syed
Course: Foundation of Information
Program: University of Arizona, Master of Science, Machine Learning
Batch: Summer 2024
Project: Project 1, Job Market Data Extraction & Analysis
Date: Tuesday, August 20, 2024

Table of Contents

Introduction

The job market is a dynamic environment where trends change rapidly, making it crucial for professionals and organizations to stay updated. One old yet workable solution to gather up-to-date job market data is through web scraping. This report outlines the process of using web scraping techniques using Selenium and Beautiful Soup libraries with Python, to collect and analyze data related to Data Scientist job postings in India from Naukri.com. Then utilization of this data to provide insights.

Overview of Web Scraping Techniques

Web scraping is the process of automatically extracting publicly listed data from websites. Nowadays, due to their wide popularity, use cases, ease of use, speed, security, and effectiveness in separating the data from the presentation logic, web APIs are the preferred way to request data between 2 systems. However, there are still use cases when we are required to fall back to Web Scraping techniques, thus making it an essential tool for businesses and individuals seeking to gather large amounts of data quickly and efficiently. A few of such use cases are:

1. Job Market Analysis
2. E-Commerce Price Monitoring
3. Market Research
4. Sentiment Analysis
5. News Aggregation

However, web scraping also involves understanding and obliging the legal and ethical considerations that comes with extracting data from public websites.

Tools, Technologies, and Data-Source

Naurki.com: I choose to collect data from Naukri.com, because it doesn't require you to log in, plus most of my required info is available on the Job Listing page, hence I don't have to follow a drill-down approach. Plus, it lets you navigate its pages using a simple https://www.{base-url}/{job-title}-jobs-in-{location}-{page-no} scheme.

Python: A general-purpose high-level programming language that is widely used for data collection and analysis, especially within the Data Science community.

Selenium: A powerful tool/library for automating web browsers. Selenium implements the W3C WebDriver protocol and allows users to navigate web pages, interact with elements, and scrape data from dynamic websites.

Beautiful Soup: A Python library for parsing HTML and XML documents.

lxml: A Python library that works with Beautiful Soup to parse HTML/XML documents.

Matplotlib.Pyplot: Another Python library for data visualization.

Seaborn: A Python library built on top of the Matplotlib library for better-looking visualization.

Pandas: A Python library for data access, storage, transformation, and manipulation.

re: A Python library for Regular Expressions.

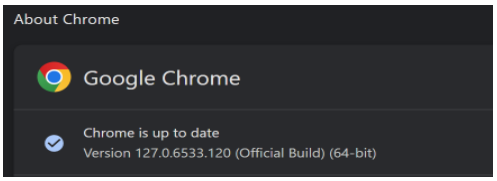
Methodology

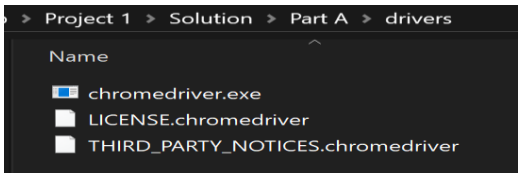
Most websites reject simple web scraping attempts such as using Python's built-in 'requests' library. Initially, I attempted to scrape it using the request and beautiful soup combo by setting the user-agent header. However, it was not allowing me to scrape more than the 1st page with 20 records only. This data was not enough to come up with any significant analysis, hence, I decided to use Selenium. This however involves several steps, beginning with Downloading WebDriver, Setup and Configuring, Navigating Naukri.Com, Data Extraction, and Storage:

Data Collection Process

The data collection process involved the following detailed steps:

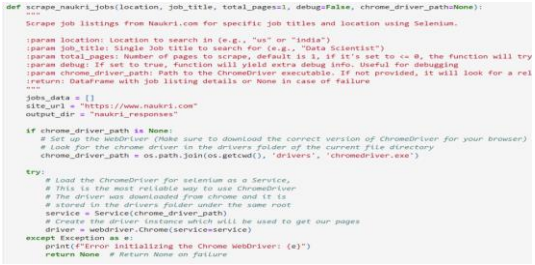
1. ***Downloading a suitable Selenium WebDriver:*** All the popular Web Browsers, supports the W3C WebDriver protocol. For this project, I decided to use the Selenium Chrome WebDriver from Google. Firstly, we need to check the installed Chrome Browser Version and your OS platform, then download the same suitable version of Selenium WebDriver for your browser and platform. In my case the Web Driver version was 127.0.6533.120, running on the Windows X64 platform. Hence, I downloaded the WebDriver for Chrome for the same version and saved it on my local drive in the Drivers folder of my project home directory so that I could load it using a relative path from within Python code.





Initial Setup and Configuration: After that, I wrote the code for Setting up Python, Selenium, Beautiful Soup plus other required packages, and configured the Selenium WebDriver. Selenium WebDriver was configured to automate the browsing process. This included loading & setting up the ChromeDriver for Google Chrome and defining the target URL (Naukri.com).

```
In [9]: import os

        from selenium import webdriver
        from selenium.webdriver.chrome.service import Service
        from selenium.webdriver.common.by import By
        from selenium.webdriver.support.ui import WebDriverWait
        from selenium.webdriver.support import expected_conditions as EC

        import time
        from bs4 import BeautifulSoup
        import pandas as pd
        from pprint import pprint
        import re

        import seaborn as sns
        import matplotlib.pyplot as plt
```



```
def scrape_naukri_jobs(location, job_title, total_pages=1, debug=False, chrome_driver_path=None):
    """
    Scrape job listings from Naukri.com for specific job titles and location using Selenium.

    :param location: Location to search in (e.g., "us" or "india")
    :param job_title: Single Job title to search for (e.g., "Data Scientist")
    :param total_pages: Number of pages to scrape, default is 1, if it's set to <= 0, the function will try
    :param debug: If set to true, function will yield extra debug info. Useful for debugging
    :param chrome_driver_path: Path to the ChromeDriver executable. If not provided, it will look for a rel
    :return: DataFrame with job listing details or None in case of failure
    """
    jobs_data = []
    site_url = "https://www.naukri.com"
    output_dir = "naukri_responses"

    if chrome_driver_path is None:
        # Set up the WebDriver (Make sure to download the correct version of ChromeDriver for your browser)
        # Look for the chrome driver in the drivers folder of the current file directory
        chrome_driver_path = os.path.join(os.getcwd(), 'drivers', 'chromedriver.exe')

    try:
        # Load the ChromeDriver for selenium as a Service,
        # This is the most reliable way to use ChromeDriver
        # The driver was downloaded from chrome and it is
        # stored in the drivers folder under the same root
        service = Service(chrome_driver_path)
        # Create the driver instance which will be used to get our pages
        driver = webdriver.Chrome(service=service)
    except Exception as e:
        print(f"Error initializing the Chrome WebDriver: {e}")
        return None  # Return None on failure
```

2. ***Navigating the Website:*** Using Selenium to navigate Naukri.com, enter the search team for "Data Scientist" jobs in "India", and interact with dynamic elements such as pagination. Selenium was passed a dynamically constructed URL by a Python function to navigate to the next page (if available). The function can either parse a fixed number of pages or could use the information available on Naukri.Com's currently scraped page to deduce if the next page is available.

3. ***Data Extraction:*** Using Beautiful Soup and lxml parser to parse the HTML content and extract relevant job details, such as titles, company, company ratings, locations, experience, ratings, salary range, job description, posted date, and required skill set. I, decided to extract 50 pages (950+) of Job listings from Naukri.com and saved them in a CSV file. Since collecting all the info would take a significant amount of time.

```
# Get/Load the Page in Chrome Driver
driver.get(url)
# Allow the initiated page to load before start scraping.
# Doing it less than X seconds will result in you getting
# bloked by the service provider or the site itself. Plus
# If page is not loaded fully we will no be able to parse
# the html content correctly.
time.sleep(10)

# Load the HTML content with BeautifulSoup using lxml parser
soup = BeautifulSoup(driver.page_source, 'lxml')

# Verify the soup object exists before parsing
if soup is None: return None

# Find all job listings div elements on the page
jobs_div = soup.find_all('div', class_='srp-jobtuple-wrapper')
```

```
# Extract job-related details for each listed job on the page
for job in jobs_div:
    try:
        a_title = job.find('a', class_='title')
        title = a_title.text.strip() if a_title else 'No job title'
        a_company = job.find('a', class_='comp-name')
        company = a_company.text.strip() if a_company else 'No company info'
        span_loc = job.find('span', class_='locWdth')
        st_job_loc = span_loc.text.strip() if span_loc else 'No location info'
        # Job location is a comma and in some cases hyphen separated multivalue string
        job_locs = [loc.strip() for loc in re.split(r'[,-]', st_job_loc)]
        span_exp = job.find('span', class_='expwdth')
        experience = span_exp.text.strip() if span_exp else 'No experience info'
        span_sal = job.find('span', class_='sal-wrap').find('span', class_='sal').find('span')
        salary = span_sal.text.strip() if span_sal else 'No salary info'
        span_desc = job.find('span', class_='job-desc')
        desc = span_desc.text.strip() if span_desc else 'No description'
        span_date = job.find('span', class_='job-post-day')
        postDay = span_date.text.strip() if span_date else 'No posting period'
        a_rating = job.find('a', class_='rating')
        rating = 'No rating'
        if a_rating:
            span_rating = a_rating.find('span', class_='main-2')
            rating = span_rating.text.strip() if span_rating else 'No rating'
```

4. ***Data Storage:*** After extraction, I Stored the extracted data in a CSV file format to avoid scraping it every time program is executed and performing further analysis. For this project, I scraped 900+ records of posted jobs from Naukri.com and saved them in a CSV file on the local disk.

```
# Example usage
job_titles = ["Data Scientist"]
location = "India"
num_pages = 50  # Adjust the number of pages you want to scrape

for title in job_titles:
    df_file_name = f"naukri_{title}_{location}_jobs.csv"

    # Checks if the csv data doesn't exists than scrape it from Naukri and save in local csv file
    if not os.path.exists(df_file_name):
        df = scrape_naukri_jobs(location, title, num_pages)
        # Save the data to a CSV file, so that next time we don't have to scrape it again
        if df is not None and len(df) > 0:
            df.to_csv(df_file_name, index=False)
            print(f"Scraping completed. Data saved to {df_file_name}.")
        else:
            print(f"DataFrame has no record for title {title}")
    else: # If csv data is already available just load it.
        print(f'Loading from already saved file {df_file_name}')
        df = pd.read_csv(df_file_name)

df.info()
```

```
Loading from already saved file naukri_Data Scientist_India_jobs.csv
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 990 entries, 0 to 989
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Title            990 non-null    object
 1   Company          990 non-null    object
 2   Company_Rating   990 non-null    object
 3   Company_Reviews  990 non-null    object
 4   Location         990 non-null    object
 5   Salary_Range     990 non-null    object
 6   Experience       990 non-null    object
 7   Post_Day         990 non-null    object
 8   Tags             990 non-null    object
 9   Description      990 non-null    object
dtypes: object(10)
memory usage: 77.5+ KB
```

```
Title,Company,Company_Rating,Company_Reviews,Location,Salary_Range,Experience,Post_Day,Tags,Description
Lead Data Scientist /  Lead Machine learning Engineer,Atyeti,4.3,129,"['Hybrid', 'Pune(Kharadi)']",Not disclosed,8-12 Yrs,Just Now,"['Tensorflow'
Data Scientist / Data Analyst,.,No rating,No reviews,['Remote'],12-16 Lacs PA,3-5 Yrs,1 Day Ago,"['Data Science', 'NLP', 'Machine Learning', 'AI
Data Scientist,Rlabs Enterprise Services,3.7,137,"['Hybrid', 'Hyderabad', 'Bengaluru']",Not disclosed,4-9 Yrs,Just Now,"['Machine Learning', 'Pyt
Data Scientist - BLR/ HYD/ GGN,Genpact,3.9,27557,"['Hybrid', 'Hyderabad', 'Gurugram', 'Bengaluru']",Not disclosed,6-11 Yrs,6 Days Ago,"['Data Sci
Data Scientist,Johnson and Johnson Kenvue,4.3,31,['Bengaluru'],Not disclosed,4-6 Yrs,3 Days Ago,"['Supply chain', 'Computer science', 'Data analy
Data Scientist III,Walmart,3.9,2164,['Bengaluru'],Not disclosed,4-9 Yrs,5 Days Ago,"['Supply chain', 'Prototype', 'data science', 'Machine learni
Data Scientist I,Honeywell,4.0,348,['Bengaluru'],Not disclosed,3-6 Yrs,5 Days Ago,"['Computer science', 'SQL queries', 'tableau', 'KPI', 'Analyti
Data Scientist,Johnson and Johnson Kenvue,4.3,31,['Bengaluru'],Not disclosed,3-5 Yrs,6 Days Ago,"['Supply chain', 'Computer science', 'Data analy
Data Scientist,Calsoft,3.2,307,"['Pune(Pune Mumbai Highway)', 'Bengaluru']",Not disclosed,5-7 Yrs,2 Days Ago,"['Natural Language Processing', 'Ma
Lead Data Scientist - Machine Learning/Artificial Intelligence,Tetrahed Inc,No rating,No reviews,['Bengaluru'],Not disclosed,7-10 Yrs,3 Days Ago,
Data Scientist,Soul Ai,3.6,6,['Remote'],Not disclosed,0 Yrs,6 Days Ago,"['Data Science', 'Business Intelligence', 'Data Engineering', 'Natural La
Data Scientist,Blend360 India,3.3,23,['Hyderabad'],Not disclosed,4-6 Yrs,1 Day Ago,"['Marketing Analytics', 'Machine Learning', 'Python', 'SQL',
Data Scientist - Phd only,NDS Infoserv,2.3,8,['Mumbai'],12-15 Lacs PA,0-2 Yrs,4 Days Ago,"['Natural Language Processing', 'Phd', 'Deep Learning',
Data Scientist,IBM,4.1,20301,['Pune'],Not disclosed,10-15 Yrs,5 Days Ago,"['time series analysis', 'python', 'natural language processing', 'fore
Data Scientist,AXA Global Business Services,3.8,1388,"['Hybrid', 'Pune', 'Bengaluru']",Not disclosed,4-9 Yrs,3 Days Ago,"['Machine Learning', 'Da
Data Scientist: Artificial Intelligence,IBM,4.1,20301,['Gurugram'],Not disclosed,5-8 Yrs,3 Days Ago,"['python', 'data analytics', 'tableau', 'dat
Data Scientist: Artificial Intelligence,IBM,4.1,20301,['Gurugram'],Not disclosed,5-9 Yrs,5 Days Ago,"['python', 'data analytics', 'tableau', 'dat
```

Challenges and Solutions

While scraping data from Naukri.com, several challenges were encountered:

1. Dynamic Content: Some page content was loaded dynamically. Selenium's ability to interact with JavaScript allowed for proper handling of such content.

2. Use of Cookies and Headers: Websites like Naukri.Com uses Cookies and Headers to deduce the request origin from a browser, a simple Request and Beautiful Soap-based scraping would not work and we have to use a browser emulator environment like Selenium Web Driver.

3. Timeouts and Delays: Managing timeouts and implementing delays between actions helped avoid being blocked by the website.

4. Broken Pages: During the execution of the program if there is latency or the Page is not loaded correctly, the program's attempts to scrape rendered HTML gets failed. We need to add appropriate delays after the call to load the URL using WebDriver to make sure the page is loaded fully before we attempt to parse the HTML.

5. Hidden Job Listings: The total number of Jobs count does not match the actual scraped Jobs. This is due to hidden job listings which is not listed on the Job Page. Hence the total number of jobs details parsed are not exactly a match with the Job listing Page * Per Page Job Count. We ignored such discrepancies.

7. Missing / Inconsistent Info: Not all the listed jobs contain all the required info. For example, most of the jobs are without any Salary-Range, single job posting contains multiple locations data, Required Experience format is not consistent, almost all the data is categorical in nature instead of numerical such as Experience, Salary, etc. I used data transformation to partly overcome such issues in the analysis phase.

```python
# Function to normalize and combine titles
def normalize_titles(title):
    if re.match(r'^(lead|lead/|/lead|manager)', title, re.IGNORECASE):
        return "Lead Data Scientist"
    elif re.match(r'^(sr\.?|senior)', title, re.IGNORECASE):
        return "Senior Data Scientist"
    elif re.match(r'^(data scientist)', title, re.IGNORECASE):
        return "Data Scientist"
    elif re.match(r'^(staff)', title, re.IGNORECASE):
        return "Staff Data Scientist"
    elif re.search(r'\b(opening|junior|nlp|ai|expert|assistant)\b', title, re.IGNORECASE):
        return "Data Scientist"
    return title.title()  # Return title in Pascal Case
```

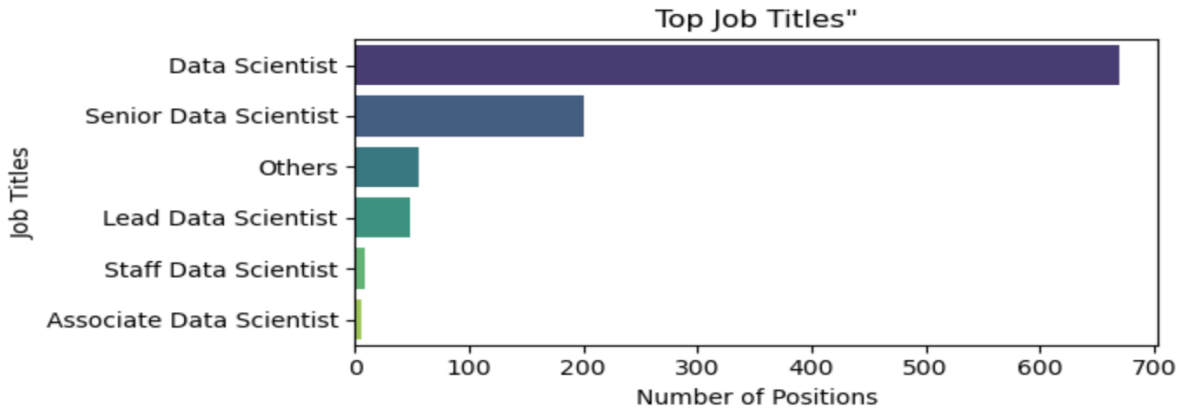
```python
ideal_job_location = (ideal_job_titles
    .Location
    .apply(lambda x: eval(x.strip().title()))
    .explode()
)

# Count the occurrences of each location
location_counts = ideal_job_location.value_counts()

# Replace locations with fewer than 10 occurrences with "Others"
normalized_locations = ideal_job_location.apply(lambda x: x if location_counts[x] >= 10 else 'Others')
```

```python
ideal_job_skills = (ideal_job_titles
    .Tags
    .apply(lambda x: eval(x.strip().title()))
    .explode()
)

# Count the occurrences of each skill
skill_counts = ideal_job_skills.value_counts().to_dict()

# Replace skills with only one occurrence with "Others"
normalized_skills = ideal_job_skills.apply(lambda x: x if skill_counts.get(x,0) > 10 else 'Others')
```

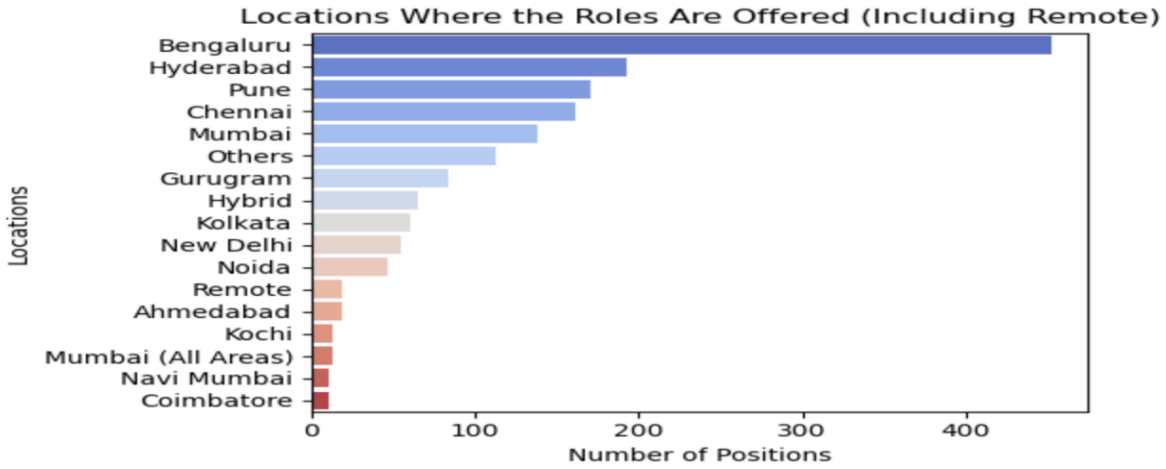
```python
# Function to convert salary range to midpoint
def convert_to_midpoint(salary_range):
    if 'Lacs PA' in salary_range:
        # Extract the numeric part for the range
        min_salary, max_salary = salary_range.replace('Lacs PA', '').strip().split('-')
        min_salary = float(min_salary.strip())
        max_salary = float(max_salary.strip())
        return ((min_salary + max_salary) / 2) * 100  # Calculate midpoint in thousands
    return None

# Step 1: Filter out rows where Salary_Range is 'Not disclosed' or 'Unpaid'
ideal_job_salary = ideal_job_titles[
    ~ideal_job_titles.Salary_Range.isin(['Not disclosed', 'Unpaid'])
].head(10)
# Step 2: Convert Salary_Range to midpoint for each row
```

Analysis and Interpretation of Data

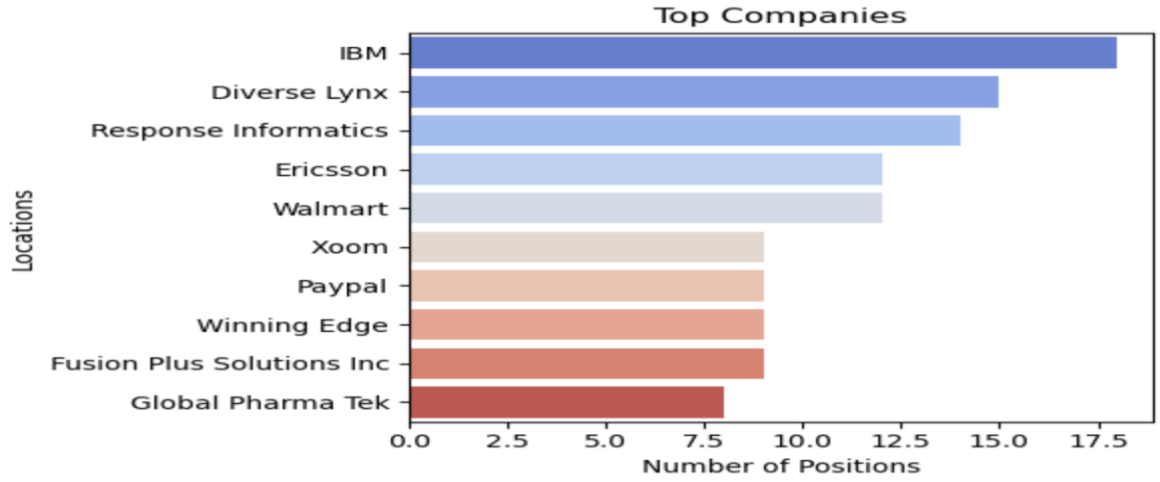
The collected data was analyzed to identify trends in the Data Scientist job market in India. Preliminary analysis revealed key insights such as the top job titles, top 10 companies, top job locations, top 15 required skills, salary ranges, and required experience for Data Scientists jobs. Below are some of the details graphs:



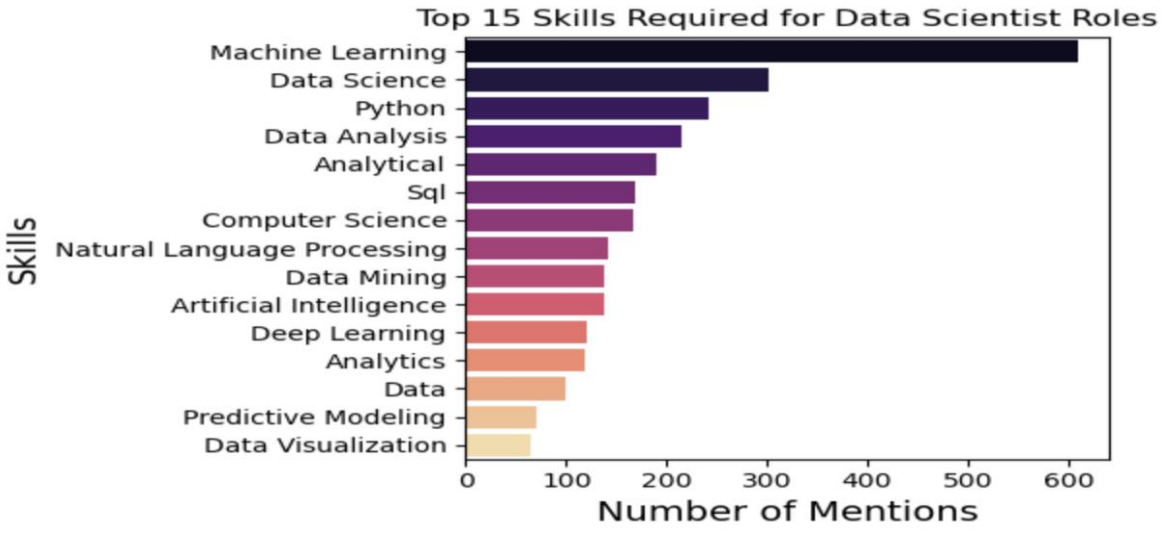
Most of the job listings are for Data Scientist and Senior Data Scientist roles



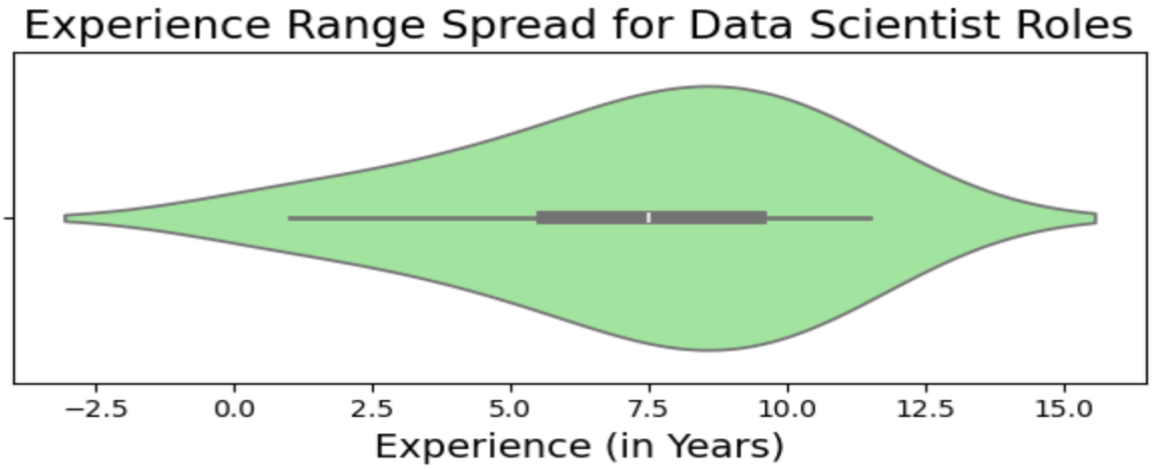
There is a high demand for Data scientist roles in Bengaluru, Hyderabad, Pune, Chennai, and Mumbai areas.



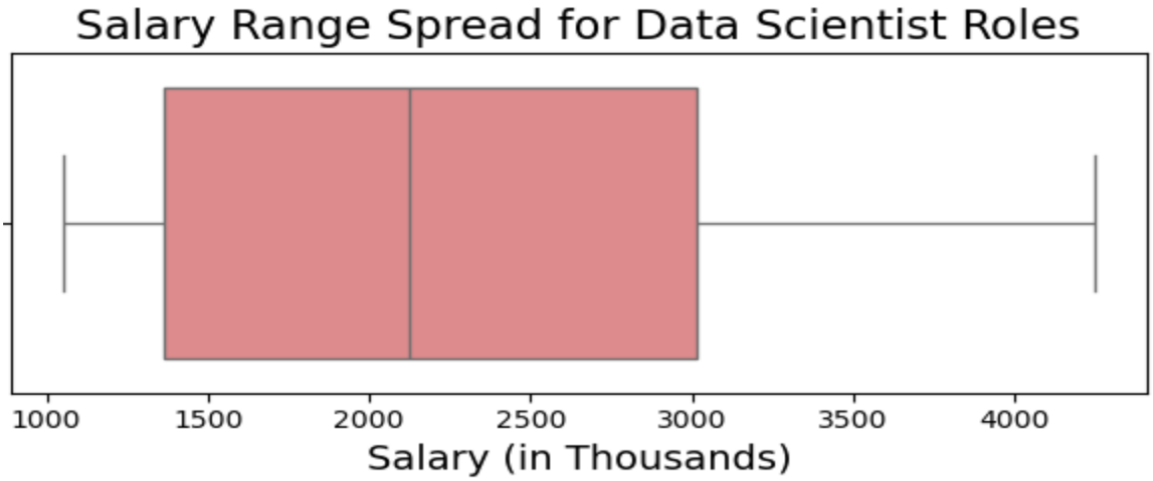
IBM, Diverse Lynx, Response Informatics, Ericsson, Walmart, Xoom, PayPal, Winning Edge, Fusion Plus, and Global Pharma Tek are the top 10 employers for Data Scientists.



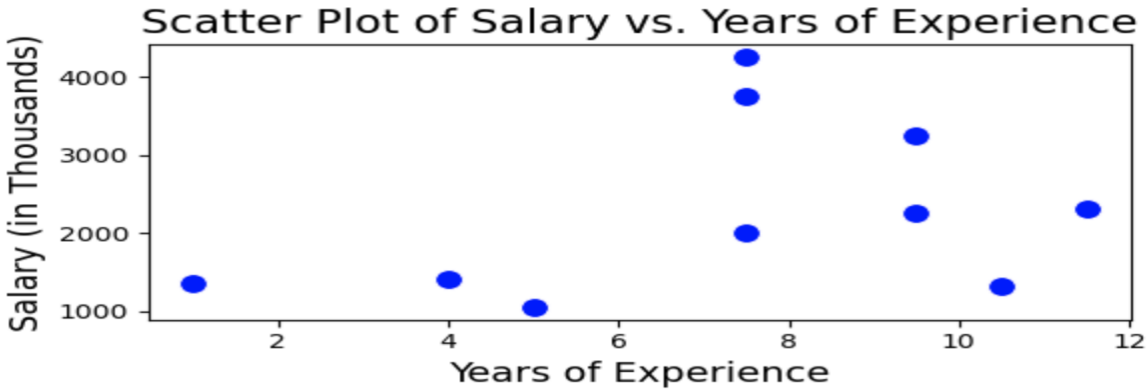
ML, Data Science, Python, Data Analysis and Analytics, SQL, NLP, Computer Science, Data Mining, AI, Predictive Modeling, and Data Visualization are top-in-demand skills for a Career in Data Science.

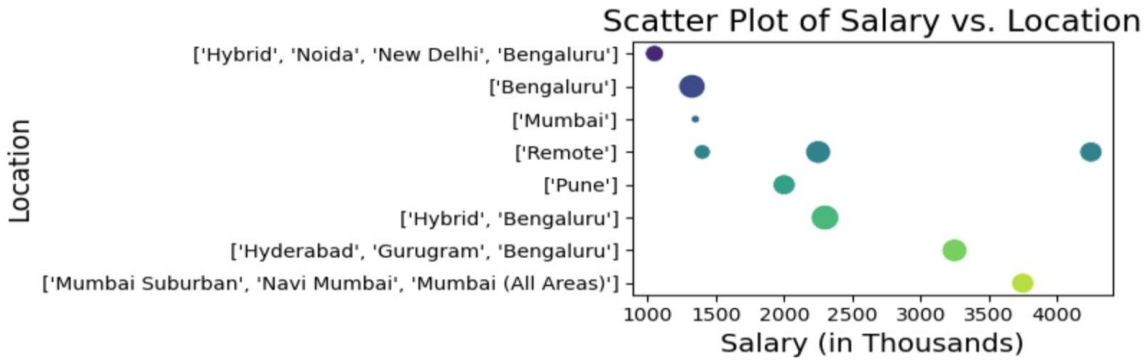


Most Jobs demand an experience of 7 to 10 years.



Average Salary per Annum is around 2.2 million Indian Rupees per Annum. Salary ranges are normally from 1.4 million to 3 million INR per year. However, this data can't be trusted as there are too few Job Postings with a salary range.



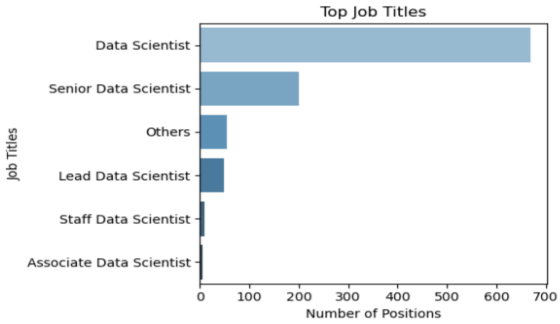


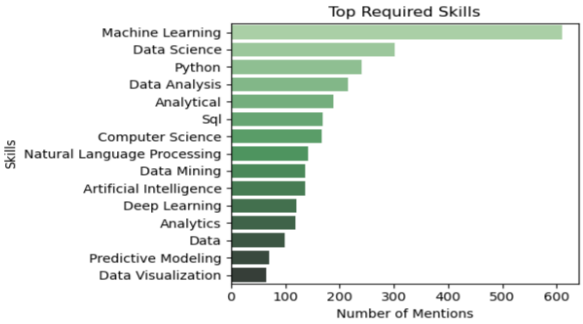
It's not possible to deduce a relationship between, years of experience & salary and location & salary due to insufficient salary data. However, we can assume that a higher-demand location would yield better earning opportunities.

Your Ideal Job:

After looking at different visualizations, we come up with the following parameters for our ideal job:

Top Job Title: Data Scientist.
Required Skills: ML, DS, Python, Data Analysis, SQL, NLP, Data Mining, AI and Deep Learning.
Location to target in India: Bengaluru
Companies to target: IBM, Diverse Lynx, Response Informatics, Ericsson & Walmart
Expected Salary Range: 2.2 million INR per annum (not conclusive due to insufficient data)





Conclusion

Web scraping is a powerful tool for real-time job market analysis. By automating the data collection process, it is possible to gather large datasets that provide valuable insights into market trends. This study demonstrated the effectiveness of using Selenium, Python, and Beautiful Soup to collect job market data, with a focus on Data Scientist positions in India.

References

1. Selenium Documentation: https://www.selenium.dev/documentation/webdriver/
2. Beautiful Soup Documentation: https://www.crummy.com/software/BeautifulSoup/bs4/doc/
3. Real Python: https://realpython.com/beautiful-soup-web-scraper-python/
4. Real Python Seaborn: https://realpython.com/python-seaborn/
4. Chrome Driver: https://developer.chrome.com/docs/chromedriver/downloads
5. W3C Pandas Tutorial: https://www.w3schools.com/python/pandas/default.asp