Job Analysis

Overview:

Job Analysis Using LinkedIn Data with Selenium repository! This project demonstrates how to automate the extraction of job data from LinkedIn using Selenium, perform data analysis, and present the findings in a visually appealing manner.

Table of Contents:

Introduction:

The process of performing job analysis using data extracted from LinkedIn. Selenium, a powerful web scraping tool, will be used to automate the extraction of job data. The data will then be analyzed to identify trends, skills demand, and other relevant insights. This guide is designed to help you understand each step in the process and ensure that your presentation of the findings is both informative and visually appealing.

Prerequisites:

- Before you begin, ensure you have the following installed on your machine:
- Python 3.x
- Selenium library
- WebDriver for your browser (e.g., ChromeDriver for Google Chrome)
- A LinkedIn account (ensure you comply with LinkedIn's terms of service)

Installation:

1. Clone the Repository:

git clone https://github.com/Ankitjaiswal1411/LinkedIn-Job-Scraper

cd job-analysis-linkedin-selenium

2. Install Required Libraries:

pip install -r requirements.txt

```
from selenium import webdriver
import time
import pandas as pd
import os
```

Usage:

1. Setting Up Selenium

Download the appropriate WebDriver for me browser. For Chrome, I can download ChromeDriver .

2. Logging into LinkedIn

Create a 'config.py' file with my LinkedIn credentials:

```
# config.py
USERNAME = 'your_email'
PASSWORD = 'your_password'
```

3. Running the Script

Run the main script to start the data extraction process:

```
python main.py
```

Data Extraction:

Main.py

```
# Import Packages
from selenium import webdriver
import time
import pandas as pd
```

```
import os

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

# Use this URL and change city or role accordingly
url1 =
'https://www.linkedin.com/jobs/search?keywords=Marketing%20Dat
a%20Analyst&location=Berlin%2C%20Berlin%2C%20Germany&geoId=106
967730&trk=public_jobs_jobs-search-bar_search-
submit&position=1&pageNum=0'

# Specify the path to chromedriver
chrome_driver_path =
r'C:\\Users\\ankit\\Downloads\\chromedriver-
win64\\chromedriver-win64\\chromedriver.exe'

# Set up the web driver and get the URL
service = Service(chrome_driver_path)
driver = webdriver.Chrome(service=service)
driver.implicitly_wait(10)
driver.get(url1)

# Find number of job listings
y = driver.find_elements(By.CLASS_NAME, 'results-context-
header__job-count')[0].text
print(y)

n = pd.to_numeric(y.replace(',', ''))
print(n)

# Loop to scroll through all jobs and click on see more jobs
button for infinite scrolling
i = 2
while i <= int((n+200)/25)+1:
    driver.execute_script("window.scrollTo(0,
document.body.scrollHeight);")
    i += 1

    try:
        send = driver.find_element(By.XPATH, "//button[@aria-
label='Load more results']")
        driver.execute_script("arguments[0].click();", send)
        time.sleep(3)
    except:
        pass
        time.sleep(5)
```

```
# Create empty lists for company name, job title, and location
companyname = []
titlename = []
locationname = []

# Find and append company names, titles, and locations to the
lists
try:
    for i in range(n):
        company = driver.find_elements(By.CLASS_NAME, 'base-
search-card__subtitle')[i].text
        companyname.append(company)
        title = driver.find_elements(By.CLASS_NAME, 'base-
search-card__title')[i].text
        titlename.append(title)
        location = driver.find_elements(By.CLASS_NAME, 'job-
search-card__location')[i].text
        locationname.append(location)
except IndexError:
    print("no")
```

Data Analysis:

Use pandas to analyze the extracted data and generate insights.

Analysis.py

```
# Import Packages
from selenium import webdriver
import time
import pandas as pd
import os
# Create dataframes for company name, title, and location
companyfinal = pd.DataFrame(companyname, columns=["company"])
titlefinal = pd.DataFrame(titlename, columns=["title"])
locationfinal = pd.DataFrame(locationname,
columns=["location"])

# Join the lists
x = companyfinal.join(titlefinal).join(locationfinal)
print(x)

# Save file in your directory
x.to_csv('linkedin.csv')
```

```
# Find job links and append them to a list
jobList = driver.find_elements(By.CLASS_NAME, 'base-
card__full-link')
hrefList = [e.get_attribute('href') for e in jobList]

print(hrefList)

linklist = pd.DataFrame(hrefList, columns=["joblinks"])
linklist.to_csv('linkedinlinks.csv')

# Close the driver
driver.close()
```

Practices:

Compliance: Always ensure your scraping practices comply with LinkedIn's terms of service.

Error Handling: Implement robust error handling to manage any unexpected issues during scraping.

Data Privacy: Handle sensitive data responsibly and ensure you do not violate any privacy policies.

Conclusion:

This guide provides a comprehensive approach to performing job analysis using LinkedIn data with Selenium. By following these steps, you can extract valuable job market insights and present them in a visually appealing manner. Happy analyzing!

Summary:

This project aims to automate the extraction of job data from LinkedIn using Selenium, perform data analysis to identify trends and insights, and present the findings in a visually appealing format. This process helps users understand the job market and the demand for various skills.

Key Features:

3. **Automated Data Extraction: Uses Selenium to log into LinkedIn, search for job postings, and extract relevant job data.**
4. **Data Storage: Stores extracted job data in a CSV file for easy access and analysis.**
5. **Data Analysis: Utilizes pandas to analyze the data and generate insights, such as job distribution by location and top hiring companies.**
6. **Data Visualization: Employs Matplotlib and Seaborn to create visual representations of the analyzed data, enhancing the presentation and interpretation of findings.**

Components:

1. Prerequisites

7. **Python 3.x**
8. **Selenium library**
9. **WebDriver for the browser (e.g., ChromeDriver)**
10. **LinkedIn account**

2. Installation

11. **Clone the repository and install required libraries from requirements.txt.**

3. Usage

12. **Configure LinkedIn credentials in a config.py file.**
13. **Run the main script (main.py) to log in, search for jobs, and extract job data into a CSV file.**

4. Data Analysis

14. **Analyze the extracted data using analysis.py.**
15. **Generate insights such as job location distribution and top hiring companies.**

5. Visualization

16. **Create visual representations of the data using Matplotlib and Seaborn.**
17. **Examples include bar charts for job locations and top hiring companies.**

6. Best Practices

18. **Ensure compliance with LinkedIn's terms of service.**
19. **Implement robust error handling.**
20. **Handle data privacy responsibly.**

Project Structure:

- **main.py: Script for logging into LinkedIn, searching for jobs, and extracting job data.**

- **analysis.py: Script for analyzing the extracted job data and generating visualizations.**
- **config.py: File for storing LinkedIn login credentials.**
- **requirements.txt: List of required Python libraries.**
- **images/: Directory for storing generated visualizations.**

Contribution:

- **Contributions are welcome, and guidelines are provided in the CONTRIBUTING.md sfile.**

Excel dashboard to visualize job analysis data extracted from LinkedIn can provide a comprehensive and interactive way to present the insights. Here is a step-by-step guide to convert the CSV data to Excel and create a dashboard:

1. Extract and Save Data as CSV

Ensure you have the job data extracted and saved as a CSV file (linkedin_jobs.csv). This can be done using the main.py script provided in the previous steps.

2. Convert CSV to Excel

Use Python to convert the CSV file to an Excel file:

```
import pandas as pd

# Load the CSV data
df = pd.read_csv('linkedin_jobs.csv')

# Save to Excel
df.to_excel('linkedin_jobs.xlsx', index=False)
```

3. Create an Excel Dashboard

Open the Excel file (linkedin_jobs.xlsx) in Microsoft Excel and follow these steps:

a. Load Data:

21. **Open Excel and load the linkedin_jobs.xlsx file.**
22. **Click on the "Data" tab and then "Get Data" > "From File" > "From Workbook".**
23. **Select linkedin_jobs.xlsx and import the data into Excel.**

b. Create Pivot Tables:

1. **Go to the "Insert" tab and select "PivotTable".**
2. **Choose the data range from your imported data and place the PivotTable in a new worksheet.**
3. **Create different PivotTables to analyze various aspects of the data:**

- **Job Locations Distribution:**
- **Rows: Location**
- **Values: Count of Location**

Top Companies Hiring:

- ***Rows: Company***
- **Values: Count of Company**

c. Create Charts:

1. Job Locations Distribution:

- **Select the PivotTable created for job locations.**
- **Go to the "Insert" tab and choose a bar chart.**
- **Customize the chart with titles, labels, and colors.**

2. Top Companies Hiring:

- **Select the PivotTable created for top companies.**
- **Go to the "Insert" tab and choose a bar chart.**
- **Customize the chart with titles, labels, and colors.**

d. Create the Dashboard Layout:

- **Create a new worksheet named "Dashboard".**
- **Copy and paste the charts from the previous steps into the "Dashboard" worksheet.**
- **Arrange the charts and add titles, labels, and any additional notes or insights.**

e. Add Interactive Elements:

1. **Slicers:**

- **Select a PivotTable and go to the "PivotTable Analyze" tab.**
- **Click on "Insert Slicer" and choose fields like Location, Company, etc.**
- **Use slicers to filter data interactively.**
-

PivotCharts:

- **Create PivotCharts from your PivotTables for more interactive filtering.**

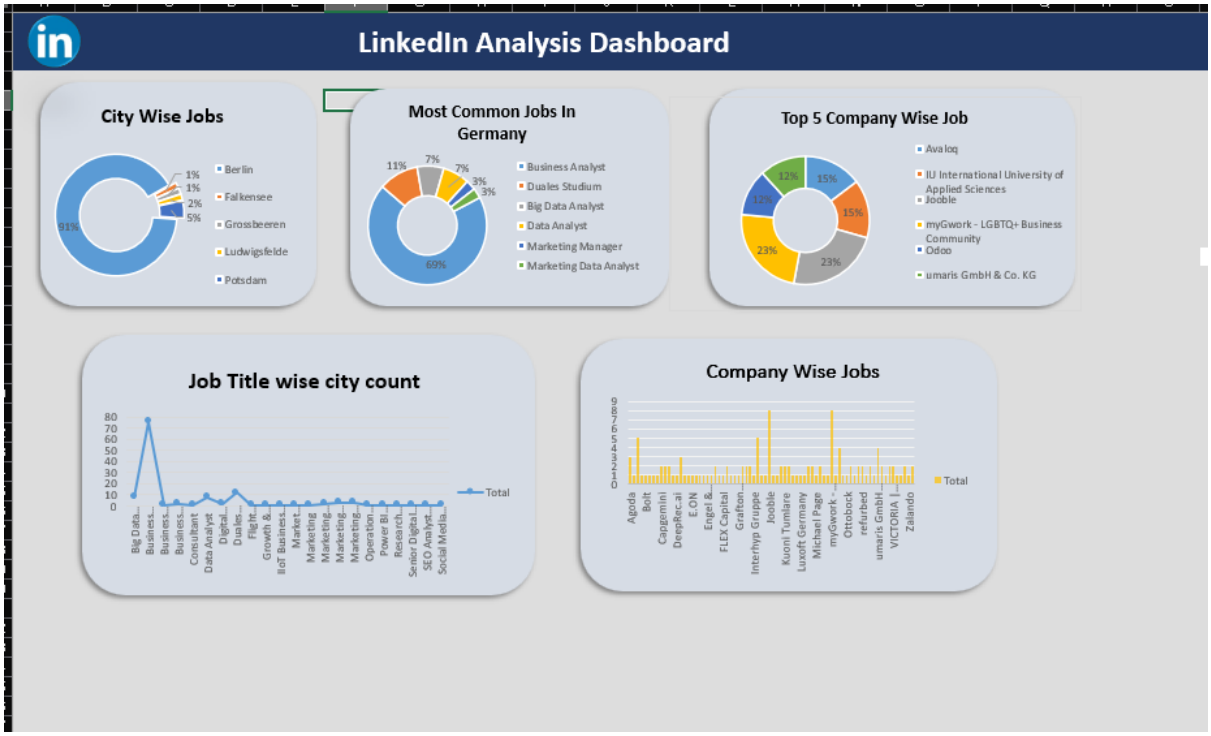
Summary of Steps in Excel:

- **Load the data into Excel.**
- **Create PivotTables for different analyses.**
- **Create charts from the PivotTables.**
- **Arrange the charts in a new worksheet to form a dashboard.**
- **Add slicers and PivotCharts for interactivity.**

Best Practices

- **Clean Data: Ensure your data is clean and structured properly before creating PivotTables.**
- **Consistency: Use consistent chart styles and colors for better readability.**
- **Interactivity: Use slicers and PivotCharts to allow users to interact with the data.**
- **Annotations: Add annotations or notes to highlight key insights in your dashboard.**

DashBoard


LinkedIn Analysis Dashboard
City Wise Jobs
1%
1%
2%
5%
91%
Berlin
Falkensee
Grossbeeren
Ludwigsfelde
Potsdam
Most Common Jobs In Germany
11%
7%
7%
3%
3%
69%
Business Analyst
Duales Studium
Big Data Analyst
Data Analyst
Marketing Manager
Marketing Data Analyst
Top 5 Company Wise Job
12%
15%
15%
12%
23%
23%
Avaloq
IU International University of Applied Sciences
Jooble
myGwork - LGBTQ+ Business Community
Odoo
umaris GmbH & Co. KG
Job Title wise city count
80
70
60
50
40
30
20
10
0
Total
Company Wise Jobs
Agoda
Bolt
Capgemini
DeepRec.ai
E.ON
Engel &
FLEX Capital
Grafton
Interhyp Gruppe
Jooble
Kuoni Tumlare
Luxoft Germany
Michael Page
myGwork -
Ottobock
refurbed
umaris GmbH
VICTORIA |
Zalando
Total