Data Analyst Test

Readme

Selenium

Approach

- The NewsScraper class contains all the methods used to scrape.
- The init method sets all the class variables required from the user
- init_record method initializes the dictionary which represents 1 row
- wait_for_element is a helper method to wait for the particular elements to load before scraping
- get_news_urls gets all the urls in a page
- save_data converts the list of dictionaries to a pandas df and saves as excel
- scrape is the main method (runner function) which calls all other methods. For each page, I get all the article urls and then loop through them and get the required data. Then go to the next page till num_page number of times or if the next button is not clickable then the loop breaks.

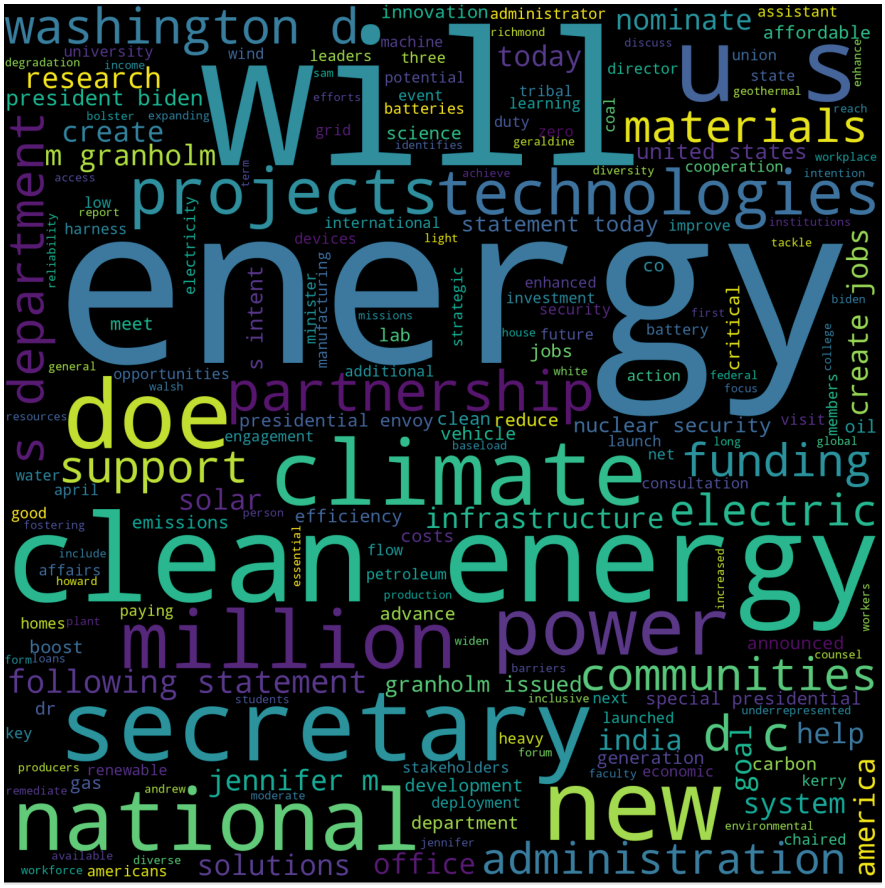
- Regarding cleaning and structuring the data, I made converted the date to the required format as given in the sample dataset. Also made the links clickable us df.style

- I did some error handling like making sure the type of the fields and parameters are correct using try except and also wait for page to load the elements.

- I also generated a word cloud from the scraped data. Most of the articles scraped talk about clean energy, power, India, America, new technology etc

Sample Output

```
sw ❯ python main.py
Page: 1    Num articles: 4
Page: 2    Num articles: 4
Page: 3    Num articles: 4
Page: 4    Num articles: 4
Page: 5    Num articles: 4
Page: 6    Num articles: 4
Page: 7    Num articles: 4
Page: 8    Num articles: 4
Page: 9    Num articles: 4
Page: 10    Num articles: 4
Total articles scraped: 40
Data saved successfully.
Time elapsed:  48.21 seconds
```



Scrapy

Approach

- I created 2 spiders, 1 for crawling all article urls and saved them in article_urls.json
- Another spider called ArticleContent to scrape the content from each article url stored in the json and finally save it as pdf
- The cleaning and structuring methods are similar to the selenium way.
- Most of the error handling etc are being handled by Scrapy itself unlike in selenium.
- The final output has been stored in **‘articles.csv’** inside NewsScraperScrapy