COINMARKETCAP 24H TOP10 TREEMAP

Project Link: https://github.com/AyFait/Cmc_Scrape_Top10_HeatMap_TreeMap

Introduction

The purpose of this project was to scrape the top 10 daily gainers from CoinMarketCap over a 24-hour interval. This task cannot be done initially without a paid API. The project involved creating a Python script to automate the data scraping and analysis process.

Required Libraries

•Selenium •Numpy •Pandas •Time •Datetime •Squarify •Matplotlib •Plotly

Methodology

1. Importing The Required Libraries

```
import pandas as pd
import time
from datetime import datetime
import numpy as np
import squarify
import matplotlib.pyplot as plt
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
```

Required Libraries for the static TreeMap

```
import pandas as pd
import time
from datetime import datetime
import numpy as np
import plotly.express as px
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
#from webdriver_manager.chrome import ChromeDriverManager
```

Required Libraries for the hovering TreeMap

2. Initializing Chrome Driver

I began by initializing a Chrome driver to browse the CoinMarketCap website using Selenium.

```
# Declare browser
chrome_driver_path = '/home/a0x0bc1/Downloads/chromedriver-linux64/chromedriver'
service = Service(chrome_driver_path)# Create a Service object
driver = webdriver.Chrome(service=service) # Pass the Service object to the webdriver.Chromefrom selenium.webdriver.common.by import By
```

3. Scraping the Data

Using Selenium, I navigated to the CoinMarketCap website and scraped the data for the top 10 gainers under the 24-hour time frame. This data was extracted as an HTML table.

```
#define url of page to exract data fromfrom selenium.webdriver.common.by import By
url='https://coinmarketcap.com/gainers-losers/'
driver.get(url)
time.sleep(5) #Sleep for few seconds so, by that time, the webpage gets loaded.
ranking =  driver.find_elements(By.XPATH, '//*[@id="__next"]/div[2]/div/div[2]/div/div[2]/div/div[1]/div/table')# get element by XPATH
```

4. Data Cleaning and Structuring

I encountered issues with the scraped data, such that the data was in a single row list and also some merged columns. To rectify this, I used the NumPy module to rearrange and reshape the data into the desired number of rows and columns.

```
#Resahaping the data list to # cols, should be indented or not?
lst1 = rowData[5:] #popping headers
expectedRows = len(lst1) // 4 #divide the list by no of expected cols to get no of expected rows
lst = np.array(lst1)#turning the simngle rowdata list into an array
reshpd = lst.reshape(int(expectedRows), 4) #to get row x col
'''print(reshpd)'''
reshpdtrimmed = reshpd[:, :-1]#'Price' '24h%' 'Vol(24h)' were merged together so I popped them
forth = np.array([row[3].split() for row in reshpd])#now splitting 'Price' '24h%' 'Vol(24h)' on their own
'''print(reshpdtrimmed)'''
joined = np.concatenate((reshpdtrimmed, forth), axis = 1)#coming together making the perfect 2D array
'''print(joined)'''
first10 = np.array(joined[:10])#only need the first 10
'''print(first10)'''
```

5. Creating a Pandas DataFrame

The resulting 2D array from NumPy was converted into a Pandas DataFrame. This tabular form allowed for better manipulation and analysis of the data.

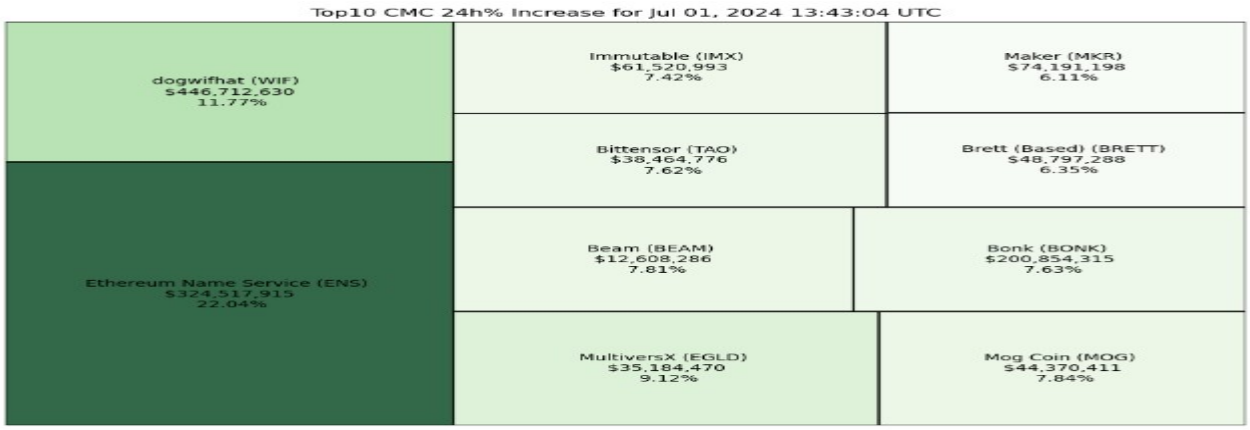
```
#passing to nympy Dataframe to get tabular form
df = pd.DataFrame(first10, columns=['CmcRank', 'Name', 'Symbol', 'Price', '24h%', 'Vol(24h)'])
#Initially getting errors of ValueError: 5 columns passed, passed data had 125 columns
#so needed to break the  cols down into 5
print(df)

df['24h%'] = df['24h%'].str.rstrip('%').astype(float)
```

	CmcRank	Name	Symbol	Price	24h%	Vol(24h)
0	70	Ethereum Name Service	ENS	$31.46	22.04%	$324,517,915
1	41	dogwifhat	WIF	$2.25	11.77%	$446,712,630
2	80	MultiversX	EGLD	$31.21	9.12%	$35,184,470
3	98	Mog Coin	MOG	$0.000001887	7.84%	$44,370,411
4	82	Beam	BEAM	$0.01703	7.81%	$12,608,286
5	54	Bonk	BONK	$0.00002371	7.63%	$200,854,315
6	48	Bittensor	TAO	$277.86	7.62%	$38,464,776
7	40	Immutable	IMX	$1.58	7.42%	$61,520,993
8	56	Brett (Based)	BRETT	$0.1607	6.35%	$48,797,288
9	39	Maker	MKR	$2,606.89	6.11%	$74,191,198

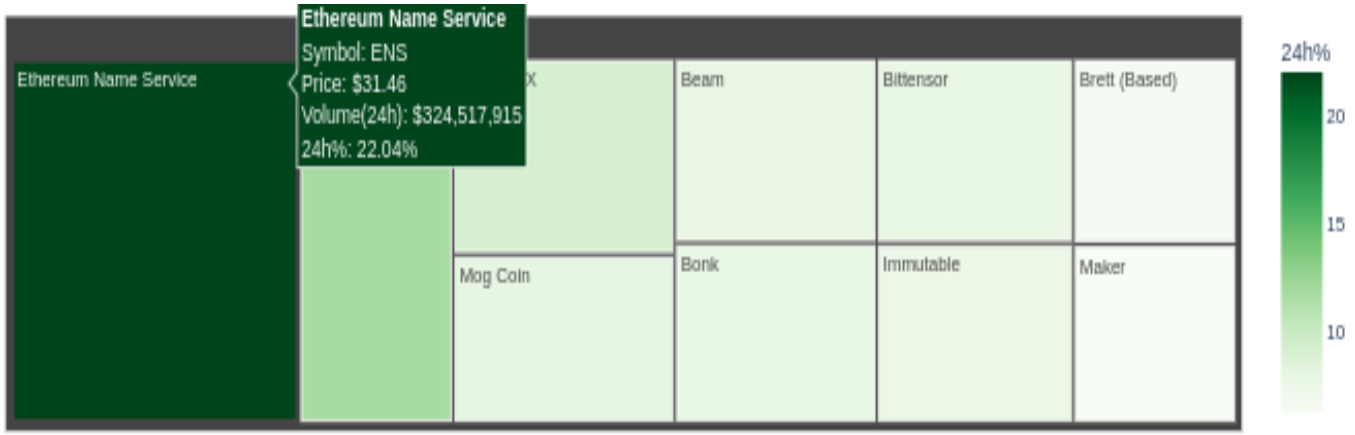
Pandas DataFrame Output

5. Data Visualization

The percentage daily increase column from the Pandas DataFrame was passed into Squarify to generate a treemap, providing a visual representation of each coin's daily percentage increase. To further give it a visual effect, the ***express*** function in Plotly library was used also to generate another treemap.



The static treemap visual representation



The hovering treemap visual representation

Further Improvement on the Code

Future improvements include:

- Making the data frame dynamic, updating in real-time.
- Ensuring that the Matplotlib visualizations update in real-time alongside the data.