Web Scraping and Analysis on Trending Stocks

Sagar Limbu

Why Web Scraping?

- Using automation tools like ‘Selenium’ we can collect data from websites
- I have used Selenium to scrap information from the stock website which provides forecasting on price target, weight, etc. and other factors that drive the stock up/ down
- Because not all websites offer API (Application Program Interface)
- Some websites require Subscription Fees to check the forecasting information from Analyst’s research

Objectives:

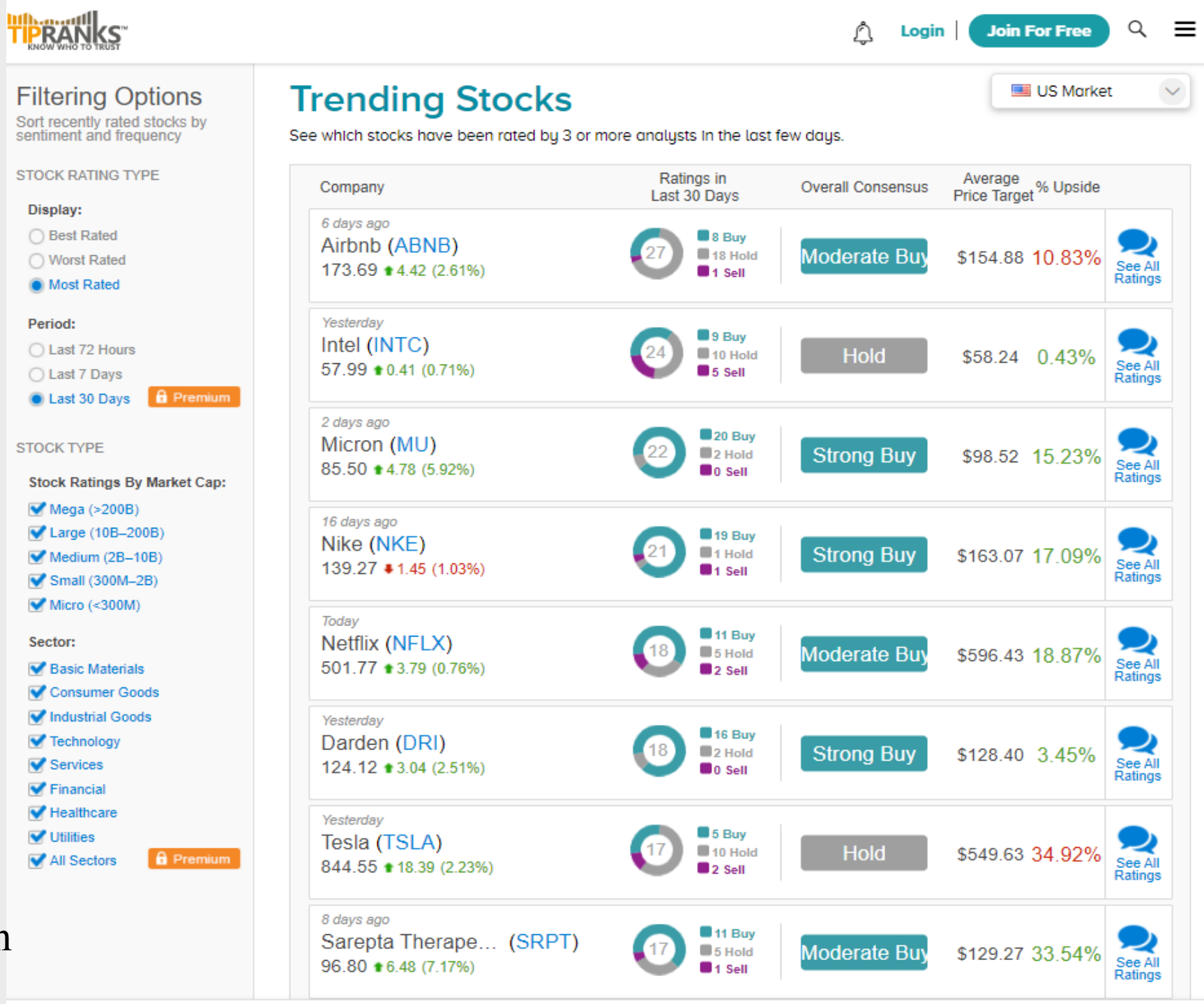
- Extracting the info on Trending stocks from the popular website to perform analysis
- Using Selenium Web Driver for web scraping
- Data Cleaning, Feature Engineering and Analysis
- **NOTE**: The Sole purpose of this project is to showcase the skills on Web Scraping with no intention of any illegal act

Analysis:

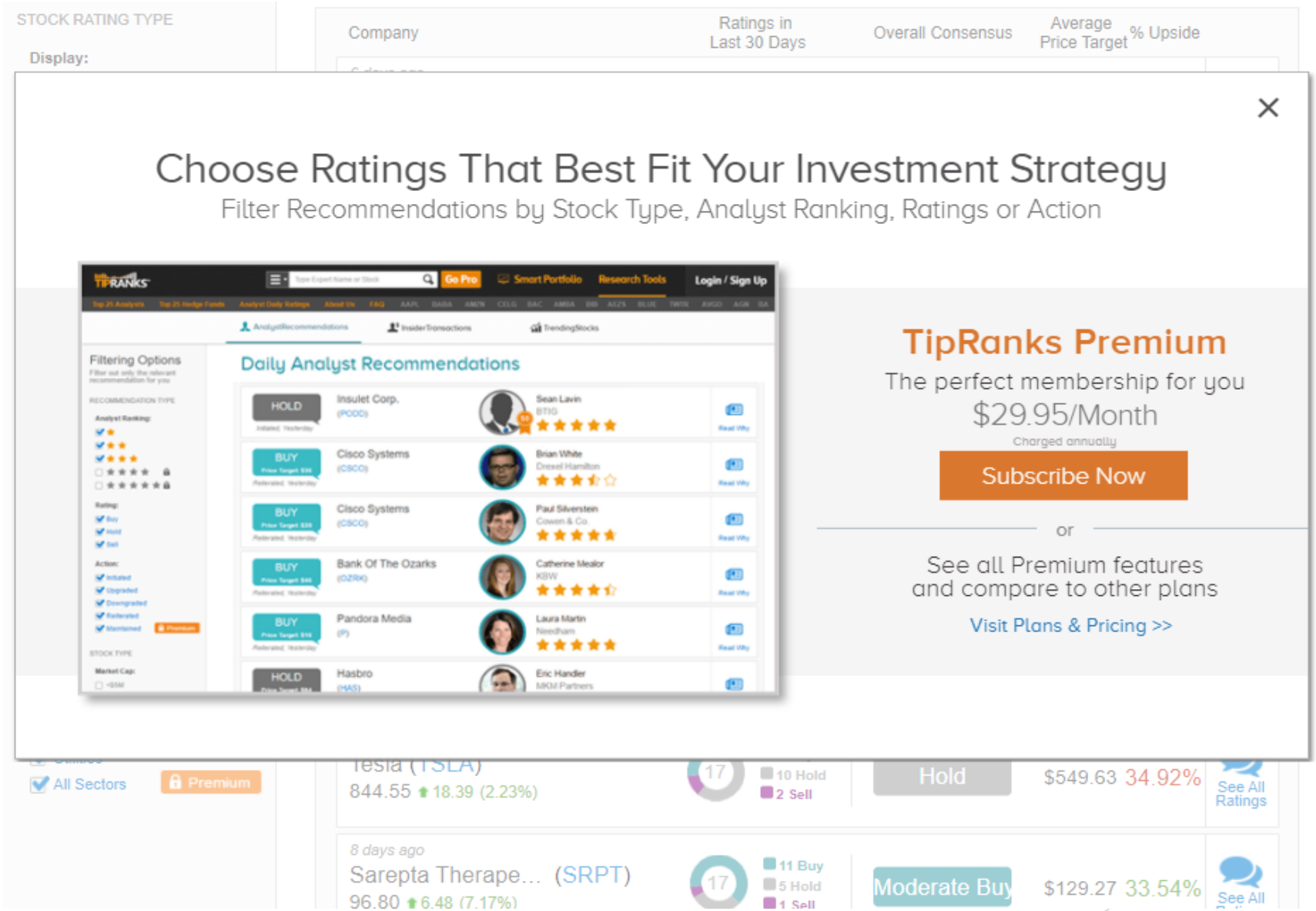
I’m sure lot of people have got involved on Stock Trading more than before in these past few months. Especially when the market was down by more than S&P500(-33%), Nasdaq (-26%) in mid March.

Investing money on the right stock can be daunting. Thus, many investors look for websites where analysts make predictions and give insights on the stocks.

Here, you can see one of the top websites (Tipranks) with the information on Analyst’s research on the Trending Stocks (ETFs)



PROBLEM: It requires **Subscription Fee** to **FILTER** the Stocks based on Rating, Stock-Cap Size, Sectors, etc.



Approach:

Scraping data from the website to retrieve information on stocks to **manipulate,** study, and filter the data (**which requires subscription from website)**

Libraries used for Web Scraping & Analysis:

In [1]:
```
from selenium import webdriver
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.common.by import By
from selenium.common.exceptions import TimeoutException
```

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [3]:
```
url= 'https://www.tipranks.com/trending-stocks'
```

Locating Web Elements:

- It is an important step to browse the webpage first and locate the web elements before performing Web Scraping
- Following example shows data extraction from the web element '**Class Name**' using '**By**' method. Using the same method for different Class Names

```
# LOCATE each INDIVIDUAL element from the WEB element

# stock INFO:
locator_stock_info= (By.CLASS_NAME,'client-components-ratings-trending-stock-row__stockInfo')

# stock Consensus
locator_stock_consensus= (By.CLASS_NAME,'client-components-ratings-trending-stock-row__consensus')

# stock Target Price
locator_stoc_tgt_price= (By.CLASS_NAME, 'client-components-ratings-trending-stock-row__priceTarget')

# Stock Rating
locator_stock_weigth= (By.CLASS_NAME,'client-components-ratings-trending-stock-row__trendGraph')
```

```
# create while loop
while True:

    # use browser script for iteration over the browser DIV element
    browser.execute_script(
    'arguments[0].scrollTop= arguments[0].scrollHeight', div_element
    )

    # use TRY and catch error method to ITERATE using the CLASS

    try:
        # RECORD each iteration and wait for 3 seconds until ITERATIng other element
        stock_info= WebDriverWait(browser, 3).until(
            # USING THe located elmenets from the SECtion
            get_element(locator_stock_info, stock_info_cnt +1)
        )

        # stock consesnus
        stock_cons= WebDriverWait(browser, 3).until(
            get_element(locator_stock_consensus, stock_cons_cnt +1)
        )

        # stock PRICE target
        stock_tgt_price= WebDriverWait(browser, 3).until(
            get_element(locator_stoc_tgt_price, stock_price_cnt +1)
        )

        # stock Weight cnt
        stock_wgt_total_cnt= WebDriverWait(browser, 3).until(
            get_element(locator_stock_weigth, stock_weight_cnt+ 1)
        )

    except TimeoutException as EC:
        break

    # update the COLLECTED record info UNLESS returns FALSE
    # number of RECORDS collected
    stock_info_cnt= len(stock_info)
    stock_cons_cnt= len(stock_cons)
    stock_price_cnt= len(stock_tgt_price)
    stock_weight_cnt = len(stock_wgt_total_cnt)
```

Iteration over the Web Element using ‘browser_execute Script’ to extract the Text field (Unstructured Raw Data)

- Full Code on **GitHub**:
- https://github.com/sagarlimbu0/My_project/blob/main/WEB-SCRAPING_tutorial_exercise/Trending_stocks_webscraping/Trending_stocks_webscraping.ipynb

Creating a Class Objects

- Class '**get_element**' is used in the previous page for iterating the web elements
- It uses the **call** function after object is instantiated over the browser element

```
# Creating a CLASs that iterates over the Browser element
class get_element(object):

    # initializing
    # locator: to locate elements
    # n: to get the NUMBER of elements
    def __init__(self, locator, n):
        self.locator= locator
        self.n= n

    # call function for Iteration using the BROWSER to
    # find element by LOCATING each WEB element
    def __call__(self, browser):

        element= browser.find_elements(*self.locator)

        # compare the number of elements
        if len(element) >= self.n:
            return element
        else:
            return False
```

Feature Engineering

5 days ago\nAirbnb(ABNB)\n173.694.42(2.61%)	27\n8 Buy\n18 Hold\n1 Sell
Today\nIntel(INTC)\n57.990.41(0.71%)	24\n9 Buy\n10 Hold\n5 Sell
Yesterday\nMicron(MU)\n85.504.78(5.92%)	22\n20 Buy\n2 Hold\n0 Sell
15 days ago\nNike(NKE)\n139.271.45(1.03%)	21\n19 Buy\n1 Hold\n1 Sell
in a day\nNetflix(NFLX)\n501.773.79(0.76%)	18\n11 Buy\n5 Hold\n2 Sell
Today\nDarden(DRI)\n124.123.04(2.51%)	18\n16 Buy\n2 Hold\n0 Sell
Today\nTesla(TSLA)\n844.5518.39(2.23%)	17\n5 Buy\n10 Hold\n2 Sell
7 days ago\nSarepta Therapeutics(SRPT)\n96.806...	17\n11 Buy\n5 Hold\n1 Sell
Today\nJPMorgan Chase & Co.(JPM)\n138.040.60(0...	16\n12 Buy\n4 Hold\n0 Sell
8 days ago\nChange Healthcare(CHNG)\n24.000.00	15\n1 Buy\n14 Hold\n0 Sell
11 days ago\nConstellation Brands(STZ)\n222.91...	15\n11 Buy\n4 Hold\n0 Sell
Today\nWells Fargo(WFC)\n32.630.59(1.84%)	14\n9 Buy\n5 Hold\n0 Sell
Today\nPNC Financial(PNC)\n156.001.22(0.79%)	14\n3 Buy\n10 Hold\n1 Sell
6 days ago\nTwitter(TWTR)\n45.930.75(1.66%)	14\n4 Buy\n10 Hold\n0 Sell
Today\nApple(AAPL)\n127.830.69(0.54%)	13\n6 Buy\n5 Hold\n2 Sell
Yesterday\nPeloton Interactive(PTON)\n150.147....	13\n11 Buy\n1 Hold\n1 Sell
4 days ago\nL Brands(LB)\n45.730.53(1.17%)	13\n6 Buy\n6 Hold\n1 Sell
5 days ago\nZscaler(ZS)\n212.000.94(0.45%)	13\n7 Buy\n6 Hold\n0 Sell

- After extracting the data from webpage, we can see **Unstructured** Raw Data.
- Thus, it requires Feature Engineering and Data Preprocessing technique to make a clean data

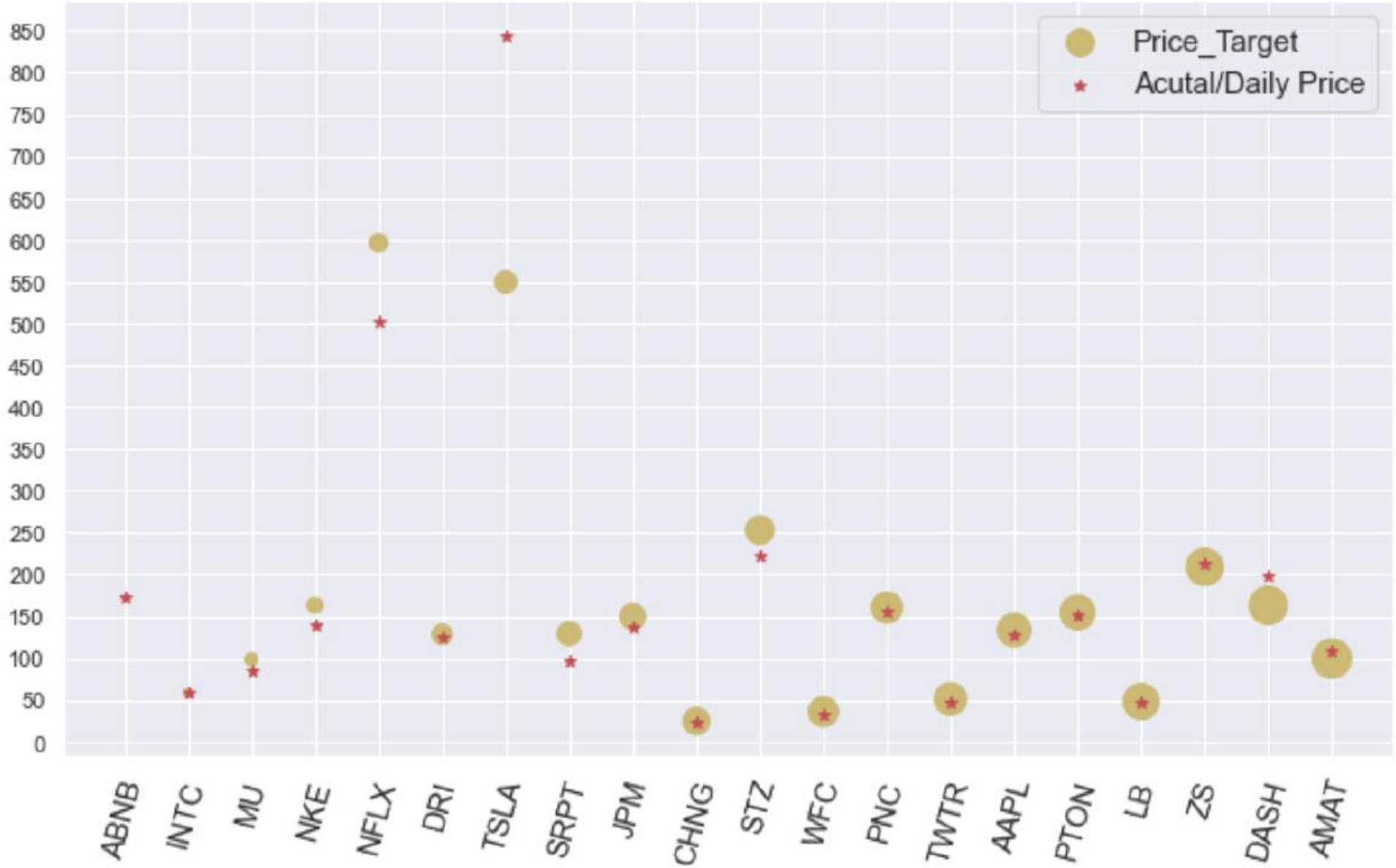
Feature Engineering (contd.)

- Compiling the Strings and Data Extraction using **Python Regular Expression** (**re.compile**) module

```
def extract_sym(stk):

    # checks over the strings that are UPPERCASE repeated Consecutively
    split_wd= re.compile('\(+[A-Z]+\)')
    symbl= split_wd.findall(stk)

    # Stored on LIST
    symbl_extracted= symbl[0]

    # Replace any Signs
    new_sym= symbl_extracted.replace('(','')
    new_sym= new_sym.replace(')','')
    return new_sym
```

Cleaned Dataset after compiling

Price_Target	Consensus	stock_weigh	stock	Daily_price	Analyst_weigh
154.88	M	27\n8 Buy\n18 Hold\n1 Sell	ABNB	173.6940	1
58.24	H	24\n9 Buy\n10 Hold\n5 Sell	INTC	57.9900	0
98.52	S	22\n20 Buy\n2 Hold\n0 Sell	MU	85.5040	2
163.07	S	21\n19 Buy\n1 Hold\n1 Sell	NKE	139.2710	2
596.43	M	18\n11 Buy\n5 Hold\n2 Sell	NFLX	501.7730	1
128.40	S	18\n16 Buy\n2 Hold\n0 Sell	DRI	124.1230	2
549.63	H	17\n5 Buy\n10 Hold\n2 Sell	TSLA	844.5518	0
129.27	M	17\n11 Buy\n5 Hold\n1 Sell	SRPT	96.8060	1
149.67	M	16\n12 Buy\n4 Hold\n0 Sell	JPM	138.0400	1
24.66	H	15\n1 Buy\n14 Hold\n0 Sell	CHNG	24.0000	0
253.00	M	15\n11 Buy\n4 Hold\n0 Sell	STZ	222.9120	1


Price_Target
Acutal/Daily Price
850
800
750
700
650
600
550
500
450
400
350
300
250
200
150
100
50
0
ABNB
INTC
MU
NKE
NFLX
DRI
TSLA
SRPT
JPM
CHNG
STZ
WFC
PNC
TWTR
AAPL
PTON
LB
ZS
DASH
AMAT