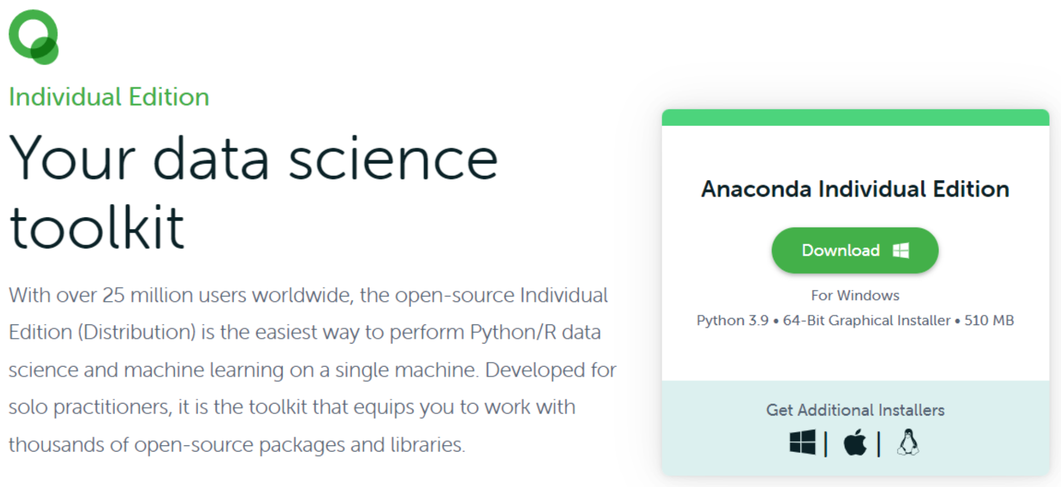
Hello, welcome to our STOCK SMART!

Our tool is based on python script. Before you start your trading, please look at our dashboard first. We gathered all the relevant and important information to help you trade SMART!

Prep. Software installation

- Please download and install the latest Anaconda Individual Edition.

  https://www.anaconda.com/products/individual



Individual Edition

Your data science toolkit

With over 25 million users worldwide, the open-source Individual Edition (Distribution) is the easiest way to perform Python/R data science and machine learning on a single machine. Developed for solo practitioners, it is the toolkit that equips you to work with thousands of open-source packages and libraries.

Anaconda Individual Edition

Download
For Windows
Python 3.9 • 64-Bit Graphical Installer • 510 MB
Get Additional Installers

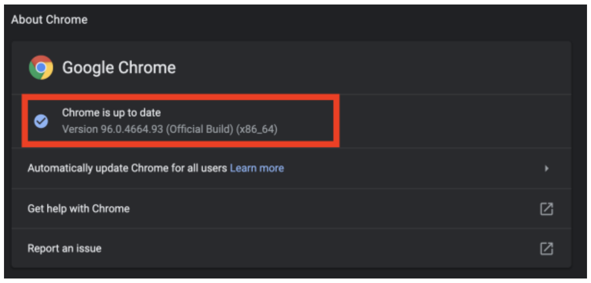
- We choose Dash to let our customers pick the stock they are interested.

  Dash is **a python framework created by plotly for creating interactive web applications**. With Dash, you only need python to create interactive dashboards. Dash is open source and applications built using this framework are viewed on the web browser.

Part I. Where to Start? Let us check out the news!

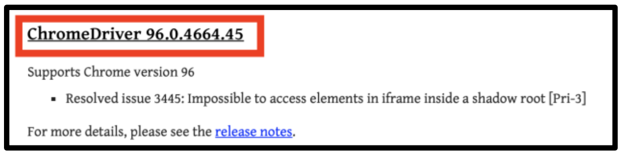
To retrieve the latest headlining news, we need to interact with the target news site to ensure that we get correct and enough information. To do so, we need to first download the package and set up the environment.

1. Download the WebDriver that works with your Browser's specific version from this link
   a. To find the version of your browser, please refer to Version in the Setting page



Chrome is up to date
Version 96.0.4664.93 (Official Build) (x86_64)

b. From the website from the above link, please download the WebDriver that works with your browser version.

   i. 

   ChromeDriver 96.0.4664.45

   Supports Chrome version 96

   - Resolved issue 3445: Impossible to access elements in iframe inside a shadow root [Pri-3]

   For more details, please see the release notes.

   ii. Save the file to a folder, of which the folder directory will be used later

c. Lastly, install Selenium package using below terminal command:

   i. pip install selenium

2. With the required package installed correctly, we can now import the "wsj.py" script to your main script and call the function using "wsj.get_news()"
   a. First, the function will ask the user to provide the local file directory to which they stored their downloaded WebDriver. Please enter the full absolute path, including the webdriver name and its extension. e.g. "/Users/taimingl/Downloads/chromedriver.exe"
   b. The user then will be prompted to enter their interested stock ticker. "AAPL"
   c. Lastly, the function will print the five latest news for the given ticker

Part II. Looking at some Historical Data!

Run the following commands on your terminal.

pip install pandas_datareader

pip install pandas_ta

Input the name of the stock you're interested in to check and sit back!

Part III. Using Linear Regression to predict stock Closing price

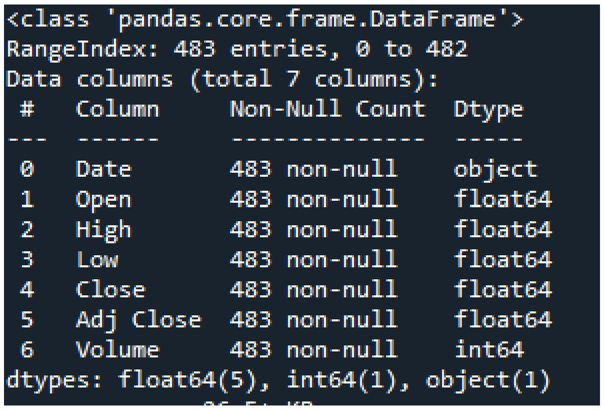
Once you open the StockSmart.py file and please run the code from under section:

```
186    # Paid Service: Stock analysis
```

This program generates a data table which contains your chosen ticker's stock price information, see below.

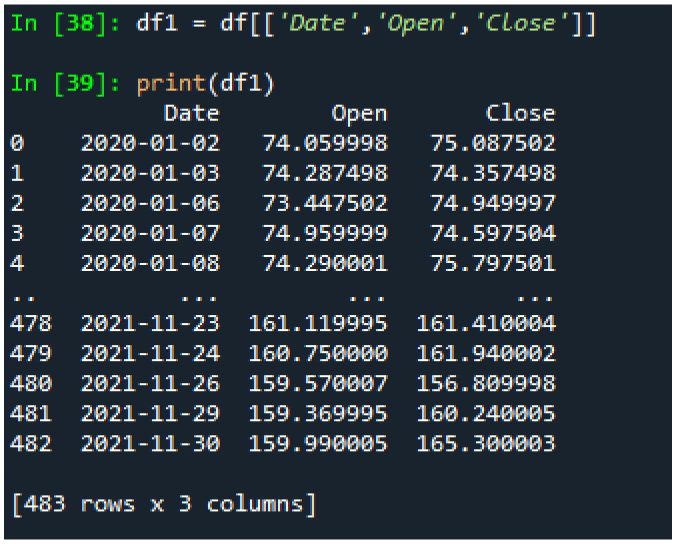
```
tickers = ['AAPL']   #'TSLA','TWTR', 'GOOG','MSFT','AMZN'     #Choose your interested stock's ticker then input here
```

```
interval = '1d'      #interval can be adjusted as needed. Here we choose '1d' since this program is to help daily trade.

for ticker in tickers:

query_string = f'https://query1.finance.yahoo.com/v7/finance/download/{ticker}?period1={period1}&period2={period2}&interval={interval}&events=history&includeAdjustedClose=true'

    df = pd.read_csv(query_string)



print(df.info())
```



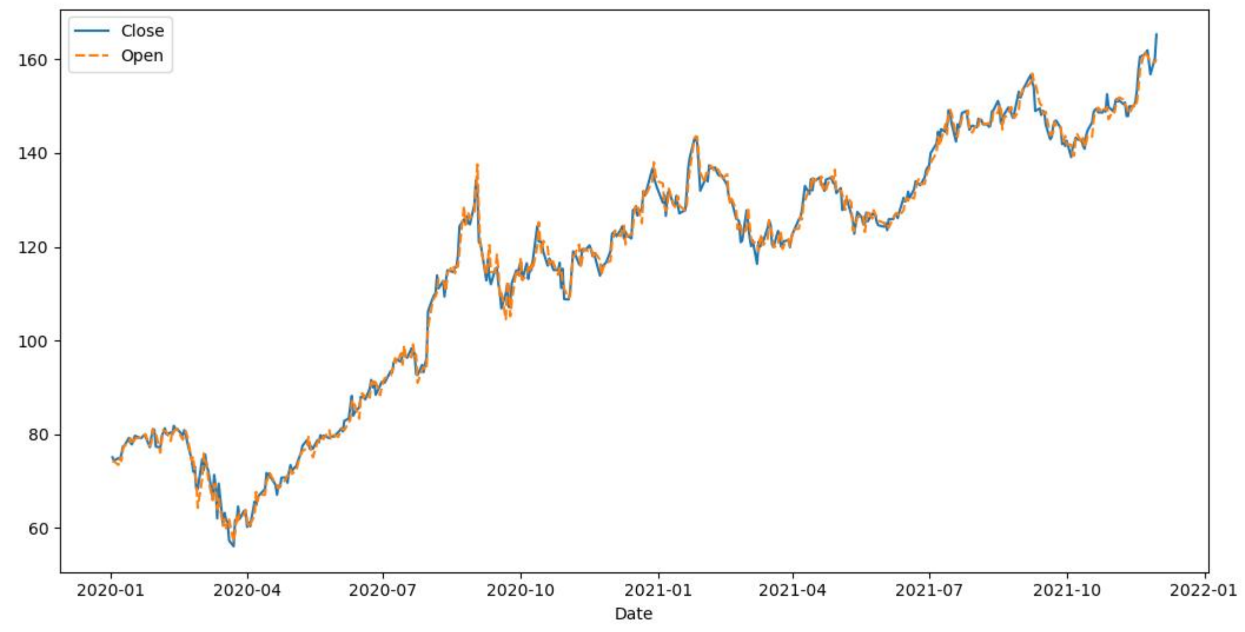
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 483 entries, 0 to 482
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       483 non-null    object
 1   Open       483 non-null    float64
 2   High       483 non-null    float64
 3   Low        483 non-null    float64
 4   Close      483 non-null    float64
 5   Adj Close  483 non-null    float64
 6   Volume     483 non-null    int64
dtypes: float64(5), int64(1), object(1)
```

Next, let us trim the data set by removing unnecessary data fields.



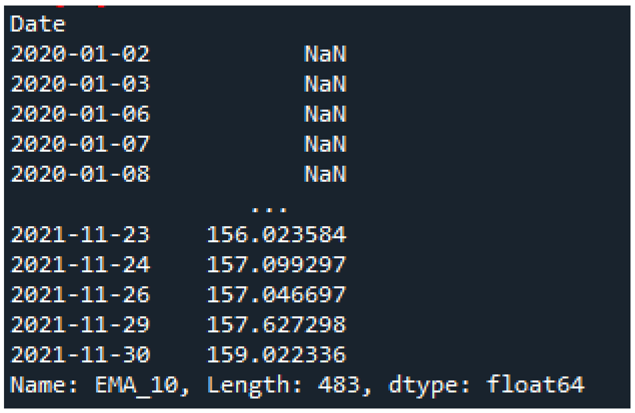
```
In [38]: df1 = df[['Date','Open','Close']]

In [39]: print(df1)
           Date        Open       Close
0    2020-01-02   74.059998   75.087502
1    2020-01-03   74.287498   74.357498
2    2020-01-06   73.447502   74.949997
3    2020-01-07   74.959999   74.597504
4    2020-01-08   74.290001   75.797501
..          ...         ...         ...
478  2021-11-23  161.119995  161.410004
479  2021-11-24  160.750000  161.940002
480  2021-11-26  159.570007  156.809998
481  2021-11-29  159.369995  160.240005
482  2021-11-30  159.990005  165.300003

[483 rows x 3 columns]
```

Let us plot the data to check on the actual data trend:



To increase the model reliability, let us use technical indicator from pandas_ta library.

```
df1.ta.ema(close='Close', length = 10, append=True)
```



```
Date
2020-01-02           NaN
2020-01-03           NaN
2020-01-06           NaN
2020-01-07           NaN
2020-01-08           NaN
                     ...
2021-11-23    156.023584
2021-11-24    157.099297
2021-11-26    157.046697
2021-11-29    157.627298
2021-11-30    159.022336
Name: EMA_10, Length: 483, dtype: float64
```

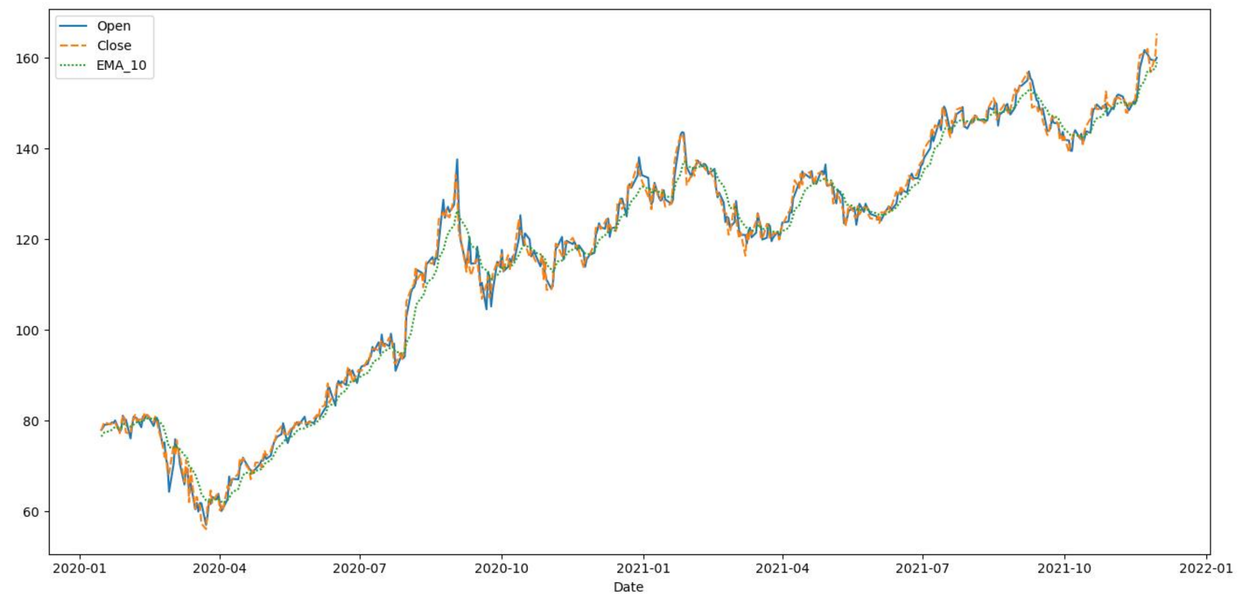
We now have a new column in our data titled "EMA_10." This is our newly-calculated value representing the exponential moving average calculated over a 10-day period.

As you can see, the first 9 records do not have EMA_10, instead of "NaN", since there weren't preceding values from which the EMA could be calculated.

Given our goal of predicting real-world pricing, we're going to just drop all the rows where we have NaN values and use a slightly smaller dataset by taking the following approach:

```
df = df.iloc[9:]
```

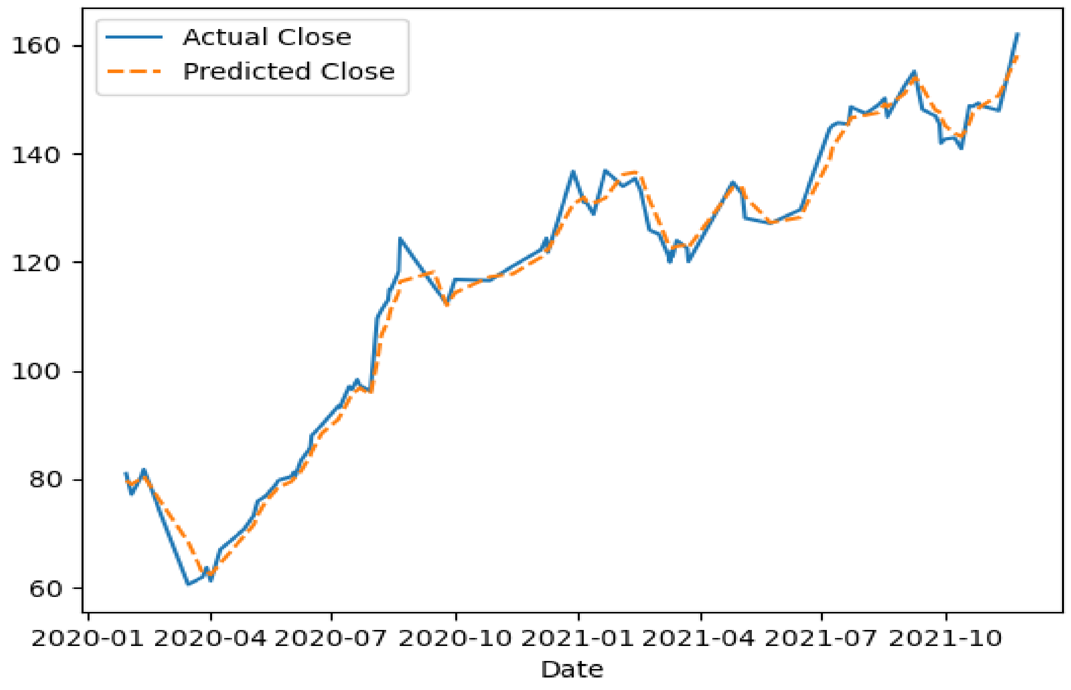
After we dropped the first 9 data points, we plot the chart below and as you can clearly see that the EMA tracks nicely.



Test-Train Split

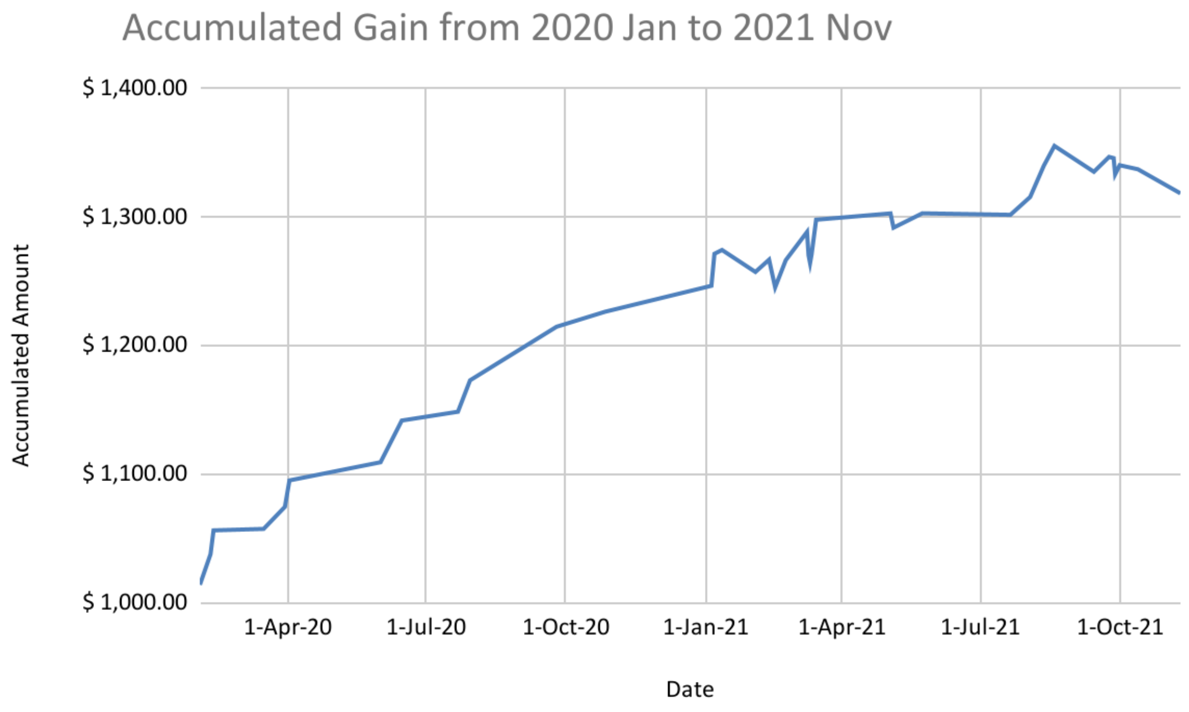
Using eighty percent of data for training and the remaining twenty percent for testing is common in machine learning models.

```
X_train, X_test, y_train, y_test = train_test_split(df1[['EMA_10']], df1['Close'], test_size =.2)
```



Lastly, let us export the data to Excel for you to take a closer look 😊

```
xlwriter = pd.ExcelWriter('Stock Prediction based on Daily Stock Closing Price and EMA.xlsx', engine = 'openpyxl')
```



*Notes: the above Accumulated Gain is based on our initial investment capital $1,000

** Please check your root folder to check all these graphs and sheets generated by the code.

Contributed by:

Ananya Raghupathy	akraghup@andrew.cmu.edu
Haoyu Wang	haoyuw2@andrew.cmu.edu
Taiming Liu	taimingl@andrew.cmu.edu
Ruben Giro	rgiro@andrew.cmu.edu