TRADING WITH AI

Stock Market Screening and Analysis: Using Web Scraping, Neural Networks, and Regression Analysis in Investing

Increasing portfolio returns using python

Ibinabo Bestmann · Follow

Published in The Startup

16 min read · Aug 24, 2020

Listen Share

Stock markets tend to react very quickly to a variety of factors such as news, earnings reports, etc. While it may be prudent to develop trading strategies based on fundamental data, the rapid changes in the stock market are incredibly hard to predict and may not conform to the goals of more short term traders. This study aims to use data science as a means to both

identify high potential stocks, as well as attempt to forecast future prices/price movement in an attempt to maximize an investor's chances of success.

In the first half of this analysis, I will introduce a strategy to search for stocks that involves identifying the highest-ranked stocks based on trading volume during the trading day. I will also include information based on twitter and sentiment analysis in order to provide an idea of which stocks have the maximum probability of going up in the near future. The next half of the project will attempt to apply forecasting techniques to our chosen stock(s). I will apply deep learning via a **Long short term memory (LSTM) neural network**, which is a form of a recurrent neural network (RNN) to predict close prices. Finally, I will also demonstrate how **simple linear regression** could aid in forecasting.

Part 1: Stock screening

Let's begin by web-scraping data on the most active stocks in a given time period, in this case, one day. Higher trading volume is more likely to result in bigger price volatility which could potentially result in larger gains. The main python packages to help us

with this task are the *yahoo_fin*, *alpha_vantage,* and *pandas* libraries.

```
# Import relevant packages
import yahoo_fin.stock_info as ya
from alpha_vantage.timeseries import
TimeSeries
from alpha_vantage.techindicators import
TechIndicators
from alpha_vantage.sectorperformance
import SectorPerformances
import pandas as pd
import pandas_datareader as web
import matplotlib.pyplot as plt
from bs4 import BeautifulSoup
import requests
import numpy as np
```

```
# Get the 100 most traded stocks for the
trading day
movers = ya.get_day_most_active()
movers.head()
```

	Symbol	Name	Price (Intraday)	Change	% Change	Volume	Avg Vol (3 month)	Market Cap	PE Ratio (TTM)
0	CSCO	Cisco Systems, Inc.	42.72	-5.38	-11.19	89447000.0	22215000.0	1.80377e+11	16.94
1	SRNE	Sorrento Therapeutics, Inc.	13.01	0.60	4.83	80883000.0	52273000.0	3.149e+09	NaN
2	AMD	Advanced Micro Devices, Inc.	81.84	-0.77	-0.93	57093000.0	64091000.0	9.6085e+10	159.22
3	BAC	Bank of America Corporation	26.35	-0.38	-1.42	54573000.0	73060000.0	2.28299e+11	12.68
4	GE	General Electric Company	6.60	-0.12	-1.79	51798000.0	99811000.0	5.7772e+10	NaN

Stocks with the largest trading volume for the trading day

The *yahoo_fin* package is able to provide the top 100 stocks with the largest trading volume. We are interested in stocks with a positive change in price so let's filter based on that.

```
movers = movers[movers['% Change'] >= 0]
movers.head()
```

	Symbol	Name	Price (Intraday)	Change	% Change	Volume	Avg Vol (3 month)	Market Cap	PE Ratio (TTM)
1	SRNE	Sorrento Therapeutics, Inc.	13.01	0.60	4.83	80883000.0	52273000.0	3.149e+09	NaN
5	AAPL	Apple Inc.	460.04	8.00	1.77	50105000.0	36787000.0	1.967T1.967T1.967T1.967T1.967T1.967T1.967T1.96...	34.89
6	NIO	NIO Limited	13.36	0.00	0.00	49377000.0	118525000.0	1.5824e+10	NaN
11	PLUG	Plug Power Inc.	11.36	0.73	6.87	35324000.0	24319000.0	4.163e+09	NaN
12	PCG	PG&E Corporation	9.62	0.34	3.66	31023000.0	22652000.0	1.9105e+10	NaN

Stocks with the largest trading volume for the trading day, filtered by the price change

Excellent! We have successfully scraped the data using the yahoo_fin python module. it is often a good idea to see if those stocks are also generating attention, and what kind of attention it is to avoid getting into false rallies. We will scrap some sentiment data courtesy of sentdex. Sometimes sentiments may lag due to source e.g News article published an hour after the event, so we will also utilize tradefollowers for their twitter sentiment data. We will process both lists independently and combine them. For both the

sentdex and tradefollowers data we use a 30 day time period. Using a single day might be great for day trading but increases the probability of jumping on false rallies. NOTE: Sentdex only has stocks that belong to the S&P 500. Using the *BeautifulSoup* library, this process is made fairly simple.

```
res =
requests.get('http://www.sentdex.com/finan
cial-analysis/?tf=30d')
soup = BeautifulSoup(res.text)
table = soup.find_all('tr')

# Initialize empty lists to store stock
symbol, sentiment and mentions

stock = []
sentiment = []
mentions = []
sentiment_trend = []

# Use try and except blocks to mitigate
missing data
for ticker in table:
    ticker_info = ticker.find_all('td')

    try:

stock.append(ticker_info[0].get_text())
    except:
        stock.append(None)
    try:

sentiment.append(ticker_info[3].get_text()
```

```
)
    except:
        sentiment.append(None)
    try:

mentions.append(ticker_info[2].get_text())
    except:
        mentions.append(None)
    try:
        if (ticker_info[4].find('span',
{"class":"glyphicon  glyphicon-chevron-
up"})):
            sentiment_trend.append('up')
        else:
            sentiment_trend.append('down')
    except:
        sentiment_trend.append(None)
```

Out[5]:

	Symbol	Sentiment	direction	Mentions
0	None	None	None	None
1	SP500	very good	up	124814
2	NOK	very good	down	8996
3	SNE	good	down	1616
4	MAT	very good	up	1309
...	...	...	...	...
419	GCI	very good	up	215
420	ZMH	very good	down	215
421	BWA	very good	down	208
422	JDSU	very good	up	27
423	NDAQ	good	down	24

424 rows × 4 columns

Sentiment data obtained from sentdex

We then combine these results with our previous results about the most traded stocks with positive price changes on a given day. This done using a left join of this data frame with the original *movers* data frame

```
top_stocks = movers.merge(company_info,
on='Symbol', how='left')
top_stocks.drop(['Market Cap','PE Ratio
```

```
(TTM)'], axis=1, inplace=True)
top_stocks
```

	Symbol	Name	Price (Intraday)	Change	% Change	Volume	Avg Vol (3 month)	Sentiment	direction	Mentions
0	SRNE	Sorrento Therapeutics, Inc.	13.010	0.600	4.83	80883000.0	52273000.0	NaN	NaN	NaN
1	AAPL	Apple Inc.	460.040	8.000	1.77	50105000.0	36787000.0	good	down	315
2	NIO	NIO Limited	13.360	0.000	0.00	49377000.0	118525000.0	NaN	NaN	NaN
3	PLUG	Plug Power Inc.	11.360	0.730	6.87	35324000.0	24319000.0	NaN	NaN	NaN
4	PCG	PG&E Corporation	9.620	0.340	3.66	31023000.0	22652000.0	good	down	325
5	NKLA	Nikola Corporation	45.970	3.160	7.38	23682000.0	22461000.0	NaN	NaN	NaN
6	NCLH	Norwegian Cruise Line Holdings Ltd.	15.340	0.110	0.72	23591000.0	59927000.0	NaN	NaN	NaN
7	NOK	Nokia Corporation	5.060	0.010	0.20	21909000.0	33151000.0	very good	down	8996
8	AUY	Yamana Gold Inc.	5.980	0.200	3.46	21349000.0	19437000.0	NaN	NaN	NaN
9	TSLA	Tesla, Inc.	1621.000	66.240	4.26	19974000.0	13162000.0	good	up	322
10	CCL	Carnival Corporation & Plc	15.220	0.030	0.20	19766000.0	47043000.0	good	down	315

A merged data frame containing the biggest movers and their sentiment information (if available)

The *movers* data frame had a total of 45 stocks but for brevity only 10 are shown here. A couple of stocks pop up with both very good sentiments and an upwards trend in favourability. ZNGA, TWTR, and AES (not shown) for instance stood out as potentially good picks. Note, the mentions here refer to the number of times the stock was referenced according to the internal metrics used by sentdex. Let's attempt supplementing this information with some data based on twitter. We get stocks that showed the strongest twitter sentiments within a time period of 1 month and were considered bullish.

	Symbol	Sector	Twit_Bull_score
0	AAPL	Technology	6,004
1	AMZN	Consumer Services	4,063
2	MSFT	Technology	3,464
3	AMD	Technology	3,318
4	FB	Technology	2,682
...	...	...	...
95	WLL	Energy	122
96	SNE	Consumer Durables	122
97	HL	Basic Industries	121
98	CMCSA	Consumer Services	119
99	EA	Technology	119

100 rows × 3 columns

Twitter sentiment data for stocks classified as bullish. Obtained from tradefollowers

Twit_Bull_score refers to the internal scoring used at tradefollowers to rank stocks that are considered bullish, based on twitter sentiments, and can range from 1 to as high as 10,000 or greater. With the twitter sentiments obtained, I'll combine it with our sentiment data to get an overall idea of the stocks and their sentiments

	Symbol	Name	Price (Intraday)	Change	% Change	Volume	Avg Vol (3 month)	Sentiment	direction	Mentions	Sector	Twit_Bull_score
0	SRNE	Sorrento Therapeutics, Inc.	13.010	0.600	4.83	80883000.0	52273000.0	NaN	NaN	NaN	NaN	NaN
1	AAPL	Apple Inc.	460.040	8.000	1.77	50105000.0	36787000.0	good	down	315	Technology	6.004
2	NIO	NIO Limited	13.360	0.000	0.00	49377000.0	118525000.0	NaN	NaN	NaN	NaN	NaN
3	PLUG	Plug Power Inc.	11.360	0.730	6.87	35324000.0	24319000.0	NaN	NaN	NaN	Capital Goods	554
4	PCG	PG&E Corporation	9.620	0.340	3.66	31023000.0	22652000.0	good	down	325	NaN	NaN
5	NKLA	Nikola Corporation	45.970	3.160	7.38	23682000.0	22461000.0	NaN	NaN	NaN	NaN	NaN
6	NCLH	Norwegian Cruise Line Holdings Ltd.	15.340	0.110	0.72	23591000.0	59927000.0	NaN	NaN	NaN	NaN	NaN
7	NOK	Nokia Corporation	5.060	0.010	0.20	21909000.0	33151000.0	very good	down	8996	Technology	379
8	AUY	Yamana Gold Inc.	5.980	0.200	3.46	21349000.0	19437000.0	NaN	NaN	NaN	Basic Industries	248
9	TSLA	Tesla, Inc.	1621.000	66.240	4.26	19974000.0	13162000.0	good	up	322	NaN	NaN
10	CCL	Carnival Corporation & Plc	15.220	0.030	0.20	19766000.0	47043000.0	good	down	315	Transportation	775

A merged data frame containing the biggest movers and their sentiment information (if available). Only the first 10 stocks shown

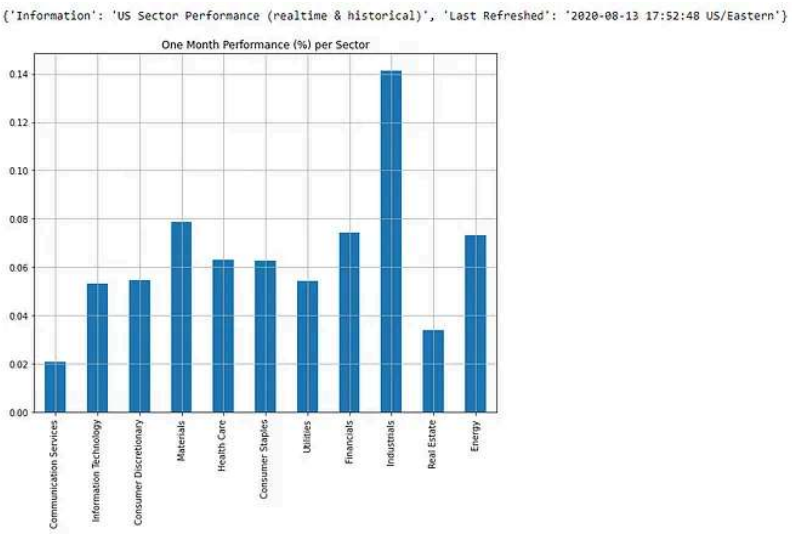
For completeness, stocks classified as having large momentum and their sentiment data will also be included in this analysis. These were scraped and merged with the *Final_list* data frame.

	Symbol	Name	Price (Intraday)	Change	% Change	Sentiment	direction	Mentions	Sector_x	Twit_Bull_score	Sector_y	Twit_mom
0	SRNE	Sorrento Therapeutics, Inc.	13.010	0.600	4.83	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	AAPL	Apple Inc.	460.040	8.000	1.77	good	down	315	Technology	6,004	Technology	14,100
2	NIO	NIO Limited	13.360	0.000	0.00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	PLUG	Plug Power Inc.	11.360	0.730	6.87	NaN	NaN	NaN	Capital Goods	554	Capital Goods	1,187
4	PCG	PG&E Corporation	9.620	0.340	3.66	good	down	325	NaN	NaN	NaN	NaN
5	NKLA	Nikola Corporation	45.970	3.160	7.38	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	NCLH	Norwegian Cruise Line Holdings Ltd.	15.340	0.110	0.72	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7	NOK	Nokia Corporation	5.060	0.010	0.20	very good	down	8996	Technology	379	Technology	624
8	AUY	Yamana Gold Inc.	5.980	0.200	3.46	NaN	NaN	NaN	Basic Industries	248	Basic Industries	419
9	TSLA	Tesla, Inc.	1621.000	66.240	4.26	good	up	322	NaN	NaN	Capital Goods	27,183
10	CCL	Carnival Corporation & Plc	15.220	0.030	0.20	good	down	315	Transportation	775	Transportation	1,717
11	PENN	Penn National Gaming, Inc.	53.350	3.590	7.21	NaN	NaN	NaN	NaN	NaN	NaN	NaN
12	MCRB	Seres Therapeutics, Inc.	28.580	5.020	21.31	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13	FB	Facebook, Inc.	261.300	1.410	0.54	poor	down	319	Technology	2,682	Technology	6,977
14	KGC	Kinross Gold Corporation	8.810	0.290	3.40	NaN	NaN	NaN	NaN	NaN	NaN	NaN
15	HL	Hecla Mining Company	6.110	0.470	8.33	NaN	NaN	NaN	Basic Industries	121	NaN	NaN
16	SPCE	Virgin Galactic Holdings, Inc.	18.770	0.610	3.36	NaN	NaN	NaN	NaN	NaN	NaN	NaN
17	BBD	Banco Bradesco S.A.	3.950	0.010	0.25	NaN	NaN	NaN	NaN	NaN	NaN	NaN
18	GOLD	Barrick Gold Corporation	27.140	0.960	3.67	NaN	NaN	NaN	NaN	NaN	Basic Industries	2,444
19	ZNGA	Zynga Inc.	9.250	0.140	1.54	very good	up	353	Technology	494	Technology	716
20	INO	Inovio Pharmaceuticals, Inc.	14.390	0.640	4.65	NaN	NaN	NaN	NaN	NaN	NaN	NaN
21	ET	Energy Transfer LP	6.550	0.000	0.00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
22	MGM	MGM Resorts International	21.340	0.210	0.99	very good	down	381	Consumer Services	536	Consumer Services	1,158
23	DKNG	DraftKings Inc.	36.050	2.310	6.85	NaN	NaN	NaN	NaN	NaN	NaN	NaN
24	FSLY	Fastly, Inc.	79.740	3.790	4.99	NaN	NaN	NaN	NaN	NaN	NaN	NaN
25	SNAP	Snap Inc.	21.900	0.040	0.18	NaN	NaN	NaN	Technology	566	Technology	1,580
26	VRT	Vertiv Holdings Co.	16.660	0.900	5.71	NaN	NaN	NaN	NaN	NaN	NaN	NaN
27	CMCSA	Comcast Corporation	43.390	0.050	0.12	good	down	316	Consumer Services	119	NaN	NaN
28	NVAX	Novavax, Inc.	133.280	8.860	7.12	NaN	NaN	NaN	NaN	NaN	Health Care	2,524
29	AG	First Majestic Silver Corp.	11.870	0.470	4.12	NaN	NaN	NaN	NaN	NaN	NaN	NaN
30	TWTR	Twitter, Inc.	37.820	0.380	1.01	very good	up	322	NaN	NaN	Technology	3,042
31	DDOG	Datadog, Inc.	82.420	5.810	7.58	NaN	NaN	NaN	NaN	NaN	NaN	NaN
32	PTON	Peloton Interactive, Inc.	65.710	1.350	2.10	NaN	NaN	NaN	NaN	NaN	NaN	NaN
33	PYPL	PayPal Holdings, Inc.	193.070	1.750	0.91	NaN	NaN	NaN	Finance	607	Finance	1,106
34	EPD	Enterprise Products Partners L.P.	18.810	0.010	0.05	NaN	NaN	NaN	NaN	NaN	NaN	NaN
35	NVDA	NVIDIA Corporation	457.720	0.110	0.02	good	down	315	Technology	1,315	Technology	2,753
36	SQ	Square, Inc.	143.190	4.170	3.00	NaN	NaN	NaN	Technology	1,627	Technology	3,294
37	CTL	CenturyLink, Inc.	10.850	0.060	0.56	good	up	315	NaN	NaN	NaN	NaN
38	NLOK	NortonLifeLock Inc.	22.970	0.450	2.02	NaN	NaN	NaN	NaN	NaN	NaN	NaN
39	RCL	Royal Caribbean Group	58.500	1.310	2.29	NaN	NaN	NaN	NaN	NaN	NaN	NaN
40	INFY	Infosys Limited	12.780	0.020	0.16	NaN	NaN	NaN	NaN	NaN	NaN	NaN
41	OPK	OPKO Health, Inc.	4.855	0.135	2.86	NaN	NaN	NaN	Health Care	504	Health Care	1,007
42	VIAC	ViacomCBS Inc.	26.690	0.450	1.71	NaN	NaN	NaN	NaN	NaN	NaN	NaN
43	CLF	Cleveland-Cliffs Inc.	5.990	0.000	0.00	good	down	321	NaN	NaN	NaN	NaN
44	AES	The AES Corporation	17.790	0.530	3.07	very good	up	318	NaN	NaN	NaN	NaN
45	HMY	Harmony Gold Mining Company Limited	5.930	0.210	3.67	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Final merged data frame of the largest moving stocks and their associated sentiments (if present)

Our list now contains even more information to help us with our trades. Stocks that it suggests might generate positive returns include TSLA, ZNGA, and TWTR as their sentiments are positive, and they have relatively decent twitter sentiment scores. As an added measure, we can also obtain information on the sectors to see how they've performed. Again, we will use a one month time period for comparison. The aforementioned stocks belong to the Technology and consumer staples sectors.

```
# Checking sector performances

sp =
SectorPerformances(key='0E66O7ZP6W7A1LC9',
output_format='pandas')
plt.figure(figsize=(8,8))
data, meta_data = sp.get_sector()
print(meta_data)
data['Rank D: Month
Performance'].plot(kind='bar')
plt.title('One Month Performance (%) per
Sector')
plt.tight_layout()
plt.grid()
plt.show()
```

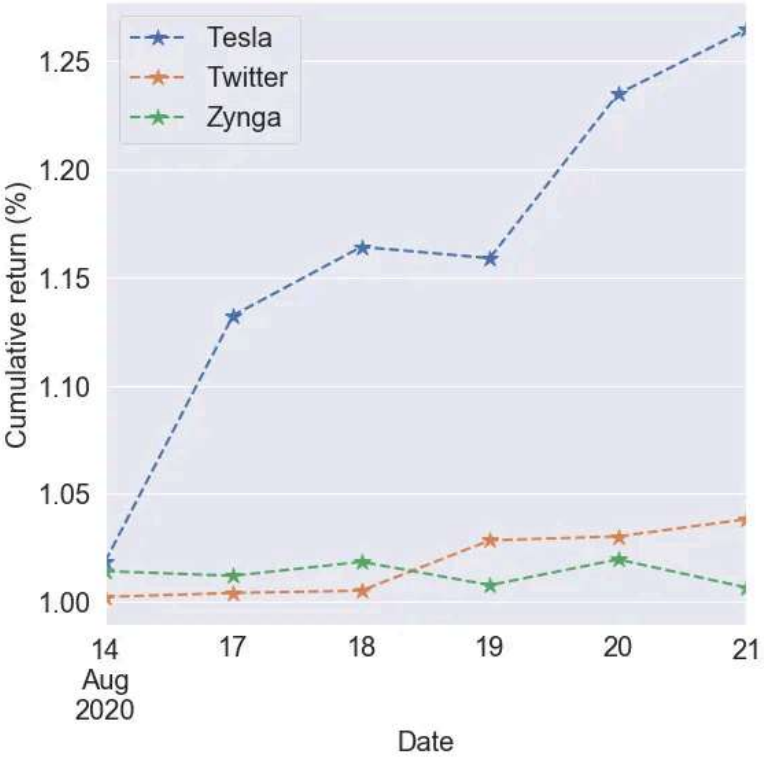


One-month sector performance (July till August) obtained using the alpha_vantage python package

The industrials sector appears to be the best performing in this time period. Consumer staples sector appears to be doing better than the information technology sector, but overall they are up which bodes well for potential investors. Of course with stocks, a combination of news, technical and fundamental analysis, as well as knowledge of the given product/sector, is needed to effectively invest but this system is designed to find stocks that are more likely to perform.

To validate this notion, let's construct a mock portfolio using TSLA, ZNGA, and TWTR as our chosen stocks

and observe their cumulative returns. This particular analysis began on the 13th of August, 2020 so that will be the starting date of the investments.



7-day cumulative returns for Tesla, Twitter, and Zynga using close prices from yahoo finance.

Overall all three investments would have netted a positive net return over the 7-day period. The tesla stock especially experienced a massive jump around

the 14th and 19th respectively, while twitter shows a continued upward trend.

> *Please note that this analysis is only a guide to find potentially positive return generating stocks and does not guarantee positive returns. It is still up to the investor to do the research.*

Part 2: Forecasting close price using an LSTM

In this section, I will attempt to apply deep learning to a stock of our choosing to forecast future prices. At the time this project was conceived (early mid-late July 2020), the AMD stock was selected as it experienced really high gains around this time. First I obtain stock data for our chosen stock. Data from 2014 data up till August of 2020 was obtained for our analysis. Our data will be obtained from yahoo using the *pandas_webreader* package

```
# Downloading stock data using the
pandas_datareader library

from datetime import datetime

stock_dt =
web.DataReader('AMD','yahoo',start,end)
stock_dt.reset_index(inplace=True)
stock_dt.head()
```

	Date	High	Low	Open	Close	Volume	Adj Close
0	2014-12-31	2.70	2.64	2.64	2.67	11177900	2.67
1	2015-01-02	2.67	2.67	2.67	2.67	0	2.67
2	2015-01-05	2.70	2.64	2.67	2.66	8878200	2.66
3	2015-01-06	2.66	2.55	2.65	2.63	13912500	2.63
4	2015-01-07	2.65	2.54	2.63	2.58	12377600	2.58

AMD stock price information

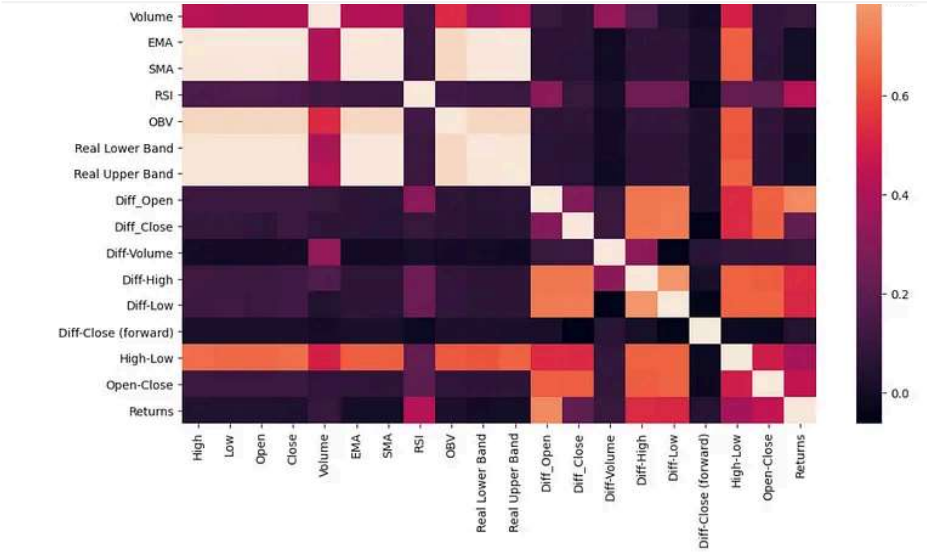
Next, some technical indicators were obtained for the data using the *alpha_vantage* python package.

```
# How to obtain technical indicators for a
stock of choice
# RSI
t_rsi =
TechIndicators(key='0E66O7ZP6W7A1LC9',outp
ut_format='pandas')
data_rsi, meta_data_rsi =
t_rsi.get_rsi(symbol='AMD',
interval='daily',time_period = 9,

series_type='open')

# Bollinger bands
t_bbands =
TechIndicators(key='0E66O7ZP6W7A1LC9',outp
ut_format='pandas')
data_bbands, meta_data_bb =
t_bbands.get_bbands(symbol='AMD',
interval='daily', series_type='open',
time_period=9)
```

This can easily be functionalized depending on the task. To learn more about technical indicators and how they are useful in stock analysis, I welcome you to explore investopedia's articles on different technical indicators. Differential data was also included in the list of features for predicting stock prices. In this study, differential data refers to the difference between price information on 2 consecutive days (price at time **t** and price at time **t-1**).

Feature engineering

Let's visualize how the generated features relate to each other using a heatmap of the correlation matrix.



Correlation matrix of stock features

The closing price has very strong correlations with some of the other price information such as opening price, highs, and lows. On the other hand, the differential prices aren't as correlated. We want to limit the amount of colinearity in our system before running any machine learning routine. So feature selection is a must.

I utilize two means of feature selection in this section. Random forests and mutual information gain. Random forests are very popular due to their relatively good accuracy, robustness as well as simplicity in terms of utilization. They can directly measure the impact of each feature on the accuracy of the model and in essence, give them a rank. Information gain, on the other hand, calculates the reduction in entropy from transforming a dataset in some way. Mutual information gain essentially evaluates the gain of each variable in the context of the target variable.

Random forest regressor

Let’s make a temporary training and testing data sets and run the regressor on it.

```
# Feature selection using a Random forest
regressor
X_train, X_test,y_train, y_test =
train_test_split(X,y,test_size=0.2,random_
state=0)

feat =
SelectFromModel(RandomForestRegressor(n_es
timators=100,random_state=0,n_jobs=-1))
feat.fit(X_train,y_train)

X_train.columns[feat.get_support()]
```

The regressor essentially selected the features that displayed a good correlation with the close price which is the target variable. However, although it selected the most important we would like information on the information gain from each variable. An issue with using random forests is it tends to diminish the importance of other correlated variables and may lead to incorrect interpretation. However, it does help reduce overfitting. Using the regressor, only three features were seen as being useful for prediction; High, Low, and Open prices.

Mutual information gain

Now quantitatively see how each feature contributes to the prediction

```
from sklearn.feature_selection import
mutual_info_regression, SelectKBest

mi =
mutual_info_regression(X_train,y_train)
mi = pd.Series(mi)
mi.index = X_train.columns
mi.sort_values(ascending=False,inplace=Tru
e)

mi
```

```
Low                    3.708476
High                   3.706451
Open                   3.165398
EMA                    2.671979
Real Upper Band        2.600057
SMA                    2.553970
OBV                    2.549028
Real Lower Band        2.498755
Volume                 0.586351
High-Low               0.546105
Diff-Low               0.395561
Open-Close             0.381547
Diff_Open              0.362054
Diff-High              0.352471
Diff_Close             0.324437
RSI                    0.208567
Diff-Volume            0.111994
Returns                0.104227
Diff-Close (forward)   0.015581
dtype: float64
```

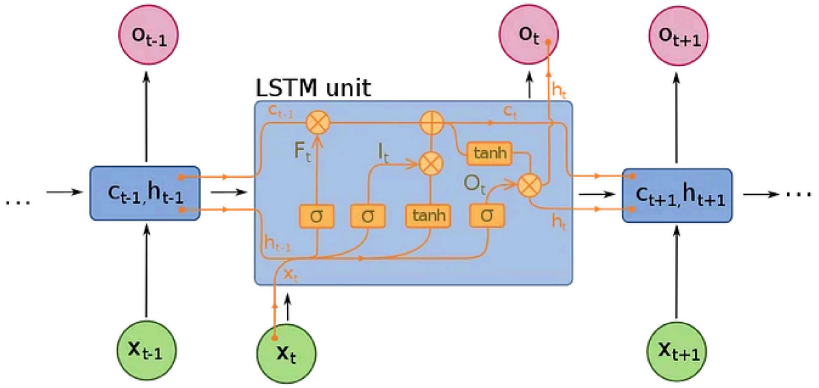
Mutual information gain from each feature towards predicting close price

The results validate the results using the random forest regressor, but it appears some of the other variables also contribute a decent amount of

information. In terms of selection, only features with contributions greater than 2 are selected, resulting in 8 features

Data preparation for the LSTM

In order to construct a Long short term memory neural network (LSTM), we need to understand its structure. Below is the design of a typical LSTM unit.



The architecture of a typical LSTM unit. Data source: Wikimedia

As mentioned earlier, **LSTM's** are a special type of recurrent neural network (**RNN**). Recurrent neural networks (**RNN**) are a special type of neural network in which the output of a layer is fed back to the input layer multiple times in order to learn from the past data. Basically, the neural network is trying to learn data that follows a sequence. However, since the RNNs utilize past data, they can become computationally

expensive due to storing large amounts of data in memory. The **LSTM** mitigates this issue, using gates. It has a cell state, and 3 gates; **forget**, **input** and **output** gates.

The **cell state** is essentially the memory of the network. It carries information throughout the data sequence processing. Information is added or removed from this cell state using gates. Information from the previous hidden state and current input are combined and passed through a sigmoid function at the **forget** gate. The sigmoid function determines which data to keep or forget. The transformed values are then multiplied by the current **cell state**.

Next, the information from the previous hidden state (**H_(t-1)**)combined with the input (**X_t**) is passed through a sigmoid function at the **input** gate to again determine important information, and also a *tanh* function to transform data between -1 and 1. This transformation helps with the stability of the network and helps deal with the vanishing/exploding gradient problem. These 2 outputs are multiplied together, and the output is added to the current cell state with the sigmoid function applied to it to give us our new cell state for the next time step.

Finally, the information from the hidden state combined with the current input is combined and a sigmoid function applied to it. In addition, the new **cell state** is passed through a *tanh* function to transform the values and both outputs are multiplied to determine the new hidden state for the next time step at the **output** gate.

Now we have an idea of how the **LSTM** works, let's construct one. First, we split our data into training and test set. An 80 – 20 split was used for the training and test sets respectively in this case. The data were then scaled using a *MinMaxScaler*.

Structure of data input for LSTM

The figure below shows how the sequential data for an LSTM is constructed to be fed into the network. Data source: Althelaya et al, 2018

Modified from Althelaya et al, 2018. Data input structure for LSTM

The idea is since stock data is sequential, information on a given day is related to information from previous days. Basically for data at time **t**, with a window size of **N**, the target feature will be the data point at time **t**, and the feature will be the data points [**t-1**, **t-N**]. We then sequentially move forward in time using this approach. This applies for the short-term prediction. For long term prediction, the target feature will be **Fk** time steps away from the end of the window where **Fk** is the number of days ahead we want to predict. We, therefore, need to format our data that way.

```
# Formatting input data for LSTM
def format_dataset(X, y, time_steps=1):
    Xs, ys = [], []
    for i in range(len(X) - time_steps):
        v = X.iloc[i:(i +
time_steps)].values
        Xs.append(v)
        ys.append(y.iloc[i + time_steps])
    return np.array(Xs), np.array(ys)
```

Following the work of Althelaya et al, 2018, a time window of 10 days was used for **N**.

Building the LSTM model

The new installment of TensorFlow (Tensorflow 2.0) via Keras has made the implementation of deep learning models much easier than in previous installments. We will apply a bidirectional LSTM as they have been shown to more effective in certain applications (see Althelaya et al, 2018). This due to the fact that the network learns using both past and future data in 2 layers. Each layer performs the operations using reversed time steps to each other. The loss function, in this case, will be the mean squared error, and the adam optimizer with the default learning rate is applied. 20% of the training set will also be used as a validation set while running the model.

```
# Design and input arguments into the LSTM
model for stock price prediction

from tensorflow import keras

# Build model. Include 20% drop out to
minimize overfitting
model = keras.Sequential()
model.add(keras.layers.Bidirectional(
        keras.layers.LSTM(
          units=32,
          input_shape=
(X_train_lstm.shape[1],
X_train_lstm.shape[2]))
  )
)
```

```
model.add(keras.layers.Dropout(rate=0.2))
model.add(keras.layers.Dense(units=1))

# Define optimizer and metric for loss
function
model.compile(optimizer='adam',loss='mean_
squared_error')

# Run model
history = model.fit(
    X_train_lstm, y_train_lstm,
    epochs=90,
    batch_size=40,
    validation_split=0.2,
    shuffle=False,
    verbose=1
)
```

Note we do not shuffle the data in this case. That's because stock data is sequential i.e the price on a given day is dependent on the price from previous days. Now let's have a look at the loss function to see how the validation and training data fair

The loss function for training and validation sets as a function of epochs

The validation set appears to have a lot more fluctuations relative to the training set. This is mainly due to the fact that most neural networks use stochastic gradient descent in minimizing the loss function which tends to be stochastic in nature. In addition, the data is batched which can influence the behavior of the solution. Of course, ideally, these hyper-parameters need to be optimized, but this is merely a simple demonstration. Let's have a look at the solution

Predicted vs actual closing prices for AMD stock

Overall, it seems the predictions follow the trend of the actual prices. Of course with more epochs and hyperparameter tuning, we could achieve better results. Let's take a closer look

Predicted vs actual closing prices for AMD stock. A sub-section of the results

At first glance, we notice the LSTM has some implicit autocorrelation in its results since its predictions for a given day are very similar to those of the previous day. The predictions essentially lag the true results. Its basically showing that the best guess of the model is very similar to previous results. This should not be a surprising result; The stock market is influenced by a number of factors such as news, earnings reports,

mergers, etc. Therefore, it is a bit too chaotic and stochastic to be accurately modeled because it depends on so many factors, some of which can be sporadic i.e positive or negative news. Therefore, in my opinion, this may not be the best way to predict stock prices. Of course with major advances in AI, there might actually be a way of doing this using LSTMs, but I don't think the hedge funds will be sharing their methods with outsiders anytime soon.

Part 3: Estimating price movement with regression analysis

Of course, we could still make an attempt to have an idea of what the possible price movements might be. In this case, I will utilize the differential prices as there is less variance compared to using absolute prices. This is a somewhat naive way but let's try it nevertheless. Let's explore some relationships

Differential close price as a function of some stock-price information. The X-axis is denoted by the legends in each subplot

Again to reiterate, the differential prices relate to the difference between the price at time **t** and the previous day value at time **t-1**. The differential high, differential low, differential high-low, and differential open-close price appear to have a linear relationship with the differential close. But only the differential open-close would be useful in analysis. This because on a given day (time **t**), we can not know what the highs or lows are beforehand till the end of the trading

day. Theoretically, those values could end up coinciding with the closing price. However, we do know the open value at the start of the trading period. I will apply ridge regression to make our results more robust. Our target variable will be the differential close price, and our feature will be the differential open-close price.

```
from sklearn.linear_model import
LinearRegression, Ridge, Lasso
from sklearn.model_selection import
GridSearchCV, cross_val_score
import sklearn
from sklearn.metrics import
median_absolute_error, mean_squared_error

# Split data into training and test
X_train_reg, X_test_reg, y_train_reg,
y_test_reg = train_test_split(X_reg,
y_reg, test_size=0.2, random_state=0)

# Use a 10-fold cross validation to
determine optimal parameters for a ridge
regression
ridge = Ridge()
alphas = [1e-15,1e-8,1e-6,1e-5,1e-4,1e-
3,1e-2,1e-
1,0,1,5,10,20,30,40,45,50,55,100]
params = {'alpha': alphas}

ridge_regressor =
GridSearchCV(ridge,params,
```

```
scoring='neg_mean_squared_error',cv=10)
ridge_regressor.fit(X_reg,y_reg)

print(ridge_regressor.best_score_)
print(ridge_regressor.best_params_)
```

The neg_mean_squared_error returns the negative version of the mean squared error. The closer it is to 0, the better the model. The cross-validation returned an *alpha* value of 1e-15 and a score of approximately **-0.4979.** Now let's run the actual model and see our results

```
regr = Ridge(alpha=1e-15)
regr.fit(X_train_reg, y_train_reg)

y_pred = regr.predict(X_test_reg)
y_pred_train = regr.predict(X_train_reg)

print(f'R^2 value for test set is
{regr.score(X_test_reg,y_test_reg)}')
print(f'Mean squared error is
{mean_squared_error(y_test_reg,y_pred)}')

plt.scatter(df_updated['Open-Close']
[1:],df_updated['Diff_Close'][1:],c='k')
plt.plot(df_updated['Open-Close'][1:],
(regr.coef_[0] * df_updated['Open-Close']
[1:] + regr.intercept_), c='r' );
plt.xlabel('Open-Close')
plt.ylabel('Diff-Close')
```

Differential open-close vs differential close price for AMD stock

The model corresponds to a slope of **$0.9599** and an intercept of **$0.01854.** This basically says that for every $1 increase in price between the opening price on a given day, and its previous closing price we can expect the closing price to increase by roughly a dollar from its closing price the day before. I obtained a mean square error of **0.58** which is fairly moderate all things considered. This R^2 value of 0.54 basically says 54% of the variance in the differential close price is explained by the differential open-close price. Not so bad so far. Note that the adjusted R^2 value was roughly equal to the original value.

To be truly effective, we need to make further use of statistics. Specifically, let's define a prediction interval around the model. Prediction intervals give you a range for the prediction that accounts for any threshold of modeling error. Prediction intervals are most commonly used when making predictions or forecasts with a regression model, where a quantity is being predicted. We select the 99% confidence interval in this example such that our actual predictions fall into this range 99% of the time. For an in-depth overview and explanation please explore machinelearningmastery

```
def predict_range(X,y,model,conf=2.58):

    from numpy import sum as arraysum

    # Obtain predictions
    yhat = model.predict(X)

    # Compute standard deviation
    sum_errs = arraysum((y - yhat)**2)
    stdev = np.sqrt(1/(len(y)-2) *
sum_errs)

    # Prediction interval (default
confidence: 99%)
    interval = conf * stdev

    lower = []
```

```
    upper = []

    for i in yhat:
        lower.append(i-interval)
        upper.append(i+interval)

return lower, upper, interval
```

Let's see our prediction intervals

Differential open-close vs differential close price for AMD stock with its prediction interval

Our prediction error corresponds to a value of **$1.82** in this example. Of course, the parameters used to obtain our regression model (slope and intercept) also have a

confidence interval which we could calculate, but its the same process as highlighted above. From this plot, we can see that even with a 99% confidence interval, some values still fall outside the range. This highlights just how difficult it is to forecast stock price movements. So ideally we would supplement our analysis with news, technical indicators, and other parameters. In addition, our model is still quite limited. We can only make predictions about the closing price on the same day i.e at time **t**. Even with the large uncertainty, this could potentially prove useful in, for instance, options tradings.

What does this all mean?

The stock screening strategy introduced could be a valuable tool for finding good stocks and minimizing loss. Certainly, in future projects, I could perform sentiment analysis myself in order to get information on every stock from the biggest movers list. There is also some room for improvement in the LSTM algorithm. If there was a way to minimize the bias and eliminate the implicit autocorrelation in the results, I'd love to hear about it. So far, the majority of research on the topic haven't shown how to bypass this issue. Of course, more model tuning and data may help so any experts out there, please give me some feedback. The

regression analysis seems pretty basic but could be a good way of adding extra information in our trading decisions. Of course, we could have gone with a more sophisticated approach; the differential prices almost appear to form an ellipse around the origin. Using a radial SVM classifier could also be a potential way of estimating stock movements.

The codes, as well as the dataset, will be provided here in the not so distant future, and my Linkedin profile can be found here. Thank you for reading!

Ibinabo Bestmann

Data Science | Analytics | Finance | Stock Market | Neural Networks

Follow

Written by Ibinabo Bestmann

75 Followers · Writer for The Startup

Geophysicist and Data engineer with an interest in artificial intelligence and data science

More from Ibinabo Bestmann and The Startup

Ibinabo Bestmann in Data Engineer Things

Setting Up Apache Airflow on Kubernetes for Local Development

In the ever-evolving realm of software development, transformative technologies come in wave after wave,...

14 min read · Jan 13, 2024

56

Maya Sayvanova in The Startup

Want to Freelance for Google?

Google employs 120,000 contractors/freelancers worldwide.

· 4 min read · Apr 21, 2024

2.4K 27

Desiree Peralta in The Startup

This Is the Smartest Move You Can Make With Crypto and Bitcoin Right Now

How to take advantage of the most you can this harvest season without losing everything in the process.

· 9 min read · Apr 12, 2024

Ibinabo Bestmann in Towards Data Science

Facies classification using unsupervised machine learning in geoscience

Facies are uniform sedimentary bodies of rock which are distinguishable enough from each other in terms of physical...

12 min read · Aug 23, 2020

59 1

See all from Ibinabo Bestmann

See all from The Startup

Recommended from Medium

Ramazan Olmez

End-to-End Machine Learning Project: Churn Prediction

The main objective of this article is to develop an end-to-end machine learning project. For a model to be truly useful, it...

18 min read · Feb 23, 2024

Nicky Reinert

How to predict stock or crypto prices in Python. With PyTorch. In 2024. Do it. Now.

Flying and predicting stock prices: two of humanity's greatest dreams. While growing wings seems impossible, let's focus...

13 min read · Jan 1, 2024

Lists

Predictive Modeling w/ Python

20 stories · 1155 saves

Practical Guides to Machine Learning

10 stories · 1396 saves

Coding & Development

11 stories · 595 saves

ChatGPT prompts

47 stories · 1516 saves

Nikos Kafritsas in Towards Data Science

MOMENT: A Foundation Model for Time Series Forecasting, Classification, Anomaly Detection

A unified model that covers multiple time-series tasks

· 11 min read · Apr 27, 2024

Benedict Neo in bitgrit Data Science Publication

Roadmap to Learn AI in 2024

A free curriculum for hackers and programmers to learn AI

11 min read · Mar 11, 2024

Rosaria Silipo in Low Code for Data Science

Is Data Science dead?

In the last six months I have heard this question thousands of time: “Is data science dead?”

6 min read · Mar 11, 2024

2.1K 44

Daniel Wu

Elevate Your Python Data Visualization Skills: A Deep Dive into Advanced Plotly Techniques with...

Data visualization is a crucial aspect of data analysis and exploration. It helps in gaining insights into the underlying...

8 min read · Nov 22, 2023