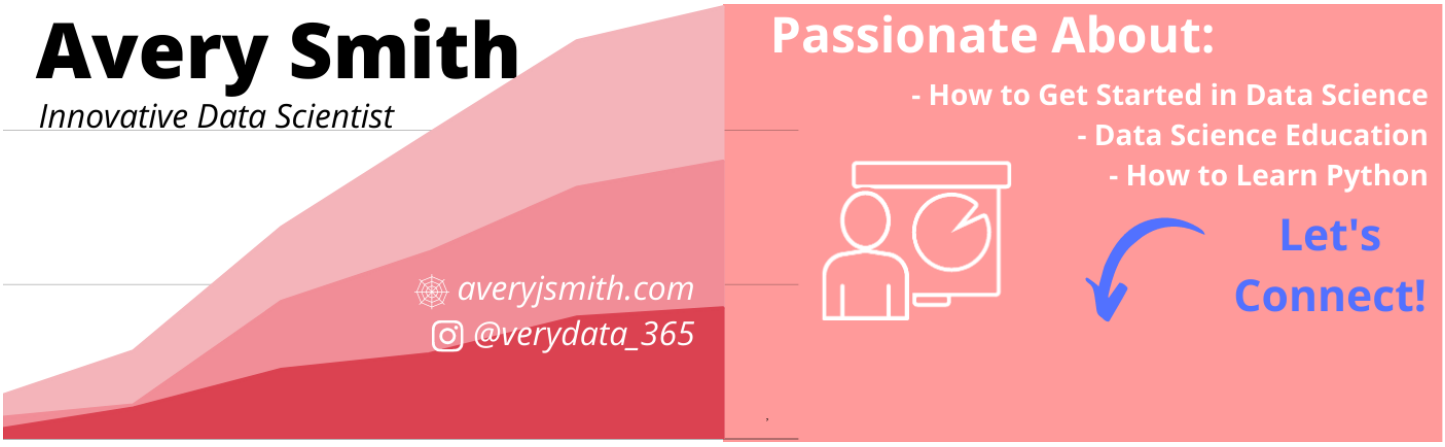


Predicting the Stock Market Using Python

In late July 2020, I decided to see how easily I could predict the stock market. I gave myself 10 days to come up with something viable.

The results were interesting. Some stocks proved hard to predict. Others proved much easier. This isn't perfect, but it is a start!

Here's a report that captures my findings!

TL;DR:
I tried using linear regression, random forest, and artificial neural nets to predict the price of Amazon (AMZN), ExxonMobil (XOM), Tesla (TSLA), DOMO (DOMO), SNAP (SNAP), Uber (UBER), and Zoom (ZM).

I used 'High', 'Low', 'Close', 'Open', 'Volume' data with additional technical indicators and time factors. I tried to predict the prices 1 day out, 5 days out, and 10 days out.

I had mixed success with the algorithms. Some worked great, others not so much.

About Me:

My name is Avery Smith. I'm obsessed with data science! I currently work as a data scientist for ExxonMobil, and have a lot of fun projects on the side.

I really like to teach and help others so reach out and connect! Find me on LinkedIn, Instagram, Medium, or Twitter.

You can find out more about me, and more of my projects on my website, averyjsmith.com.

Problem Statement:

Millions of people attempt to play the stock market every day. Stock traders constantly buy and sell stocks in aim of capitalizing on price fluctuations to make profits. The practice is relatively effortless and can lead to large returns, or losing it all. In the past, increase in IoT (Internet of things) and increased computer and internet speeds have driven an upsurge in algorithmic trading. Algorithmic trading uses a computer following a set of instructions to place trades to hopefully generate

profits at a speed and frequency that is impossible for a human. Algorithmic trades now account for 80% of all stock movement [1].

Being able to have accurate stock price predictions is even more important in uncertain economic times like 2020. It isn't possible to anticipate everything, but being able to make unemotional, data-driven decisions in the face of uncertainty provides peace, and hopefully profits.

This study explores the possibility of predicting what a stock's price might be a few days later. This prediction will be largely based on the current day's performance and technical and date parameters. Several prediction techniques will be used and compared for accuracy, ease to implement, and easy of understanding.

Data Source:

To evaluate the feasibility of predicting stocks, many data sources could be useful. Geopolitical stability, Twitter sentiment, proprietary financial statements, and just stock data have been used previously to predict stock data. In this study, we simply used vanilla stock data to start. This data was retrieved using the Python package *yfinance* [2] that collects the original measurements of 'Date', 'Open', 'High', 'Low', 'Close', 'Volume', and 'Dividends' parameters for any stock.

Methodology:

The goal of this study was to evaluate how well different machine learning algorithms could be used to predict the price of a stock, some days into the future. The predicting variable will be considered to be the

closing price some *n* days into the future. The input variables will be today's parameters and some additional financial parameters and date features. The first step will be to calculate these technical and date features. The data will then be scaled as required and preprocessed to create dummy variables for categorical variables where needed. Because we need to train the models, the data will be spilt into training and testing sections. Then the algorithms will be implemented and an error term and model accuracy will evaluate how our models performed. Linear regression, random forest, and Multi-layer Perceptron Neural Nets will be used to predict the stock prices. The models will be trained and evaluated several different stocks including Amazon (AMZN), ExxonMobil (XOM), Tesla (TSLA), DOMO (DOMO), SNAP (SNAP), Uber (UBER), and Zoom (ZM). Each stock will have its own model as we will consider each as an independent actor with unique characteristics and patterns.

Feature Generation:

Having the 'High' and 'Low' is useful, but doesn't capture all the variance in tomorrow's stock close; there are other factors to consider. The date and technical indicators also play a role in evaluating to buy or sell stocks. For example, certain quarters typically have higher earnings that could raise the stock price. Or, there could be momentum signs that trigger indicating a large sell off is likely to occur.

To generate the date meta-data, a Python package called *fastai* [3] was used. It was able to evaluate the data's date and add flags for day of the week, day or the year, month, quarter, start of week, start of year, ect. The entire date indicators are shown in Table X.

Table 1: Date Meta Data

Year	Dayofweek	Is_quarter_end
Month	Dayofyear	Is_quarter_start
Week	Is_month_end	Is_year_end
Day	Is_month_start	Is_year_start
Elapsed		

The technical indicators were added via the Python package *ta* [4]. Financial parameters featuring momentum, volume, trends, and volatility were generated based on the basic stock data. An entire list of the technical indicators is given below. To understand what each of these mean, the package's glossary proves detailed and information rich [5].

Table 2: Financial Indicators

Stock Splits	volatility_bbl	trend_macd	trend_mass_index	trend_aroon_ind	others_dr
volume_adi	volatility_bbw	trend_macd_signal	trend_cci	trend_psar_up	others_dlr
volume_obv	volatility_bbp	trend_macd_diff	trend_dpo	trend_psar_down	others_cr
volume_cmf	volatility_bbhi	trend_sma_fast	trend_kst	trend_psar_up_indicator	
volume_fi	volatility_bbli	trend_sma_slow	trend_kst_sig	trend_psar_down_indicator	
momentum_mfi	volatility_kcc	trend_ema_fast	trend_kst_diff	momentum_rsi	
volume_em	volatility_kch	trend_ema_slow	trend_ichimoku_conv	momentum_tsi	
volume_sma_em	volatility_kcl	trend_adx	trend_ichimoku_base	momentum_uo	
volume_vpt	volatility_kcw	trend_adx_pos	trend_ichimoku_a	momentum_stoch	
volume_nvi	volatility_kcp	trend_adx_neg	trend_ichimoku_b	momentum_stoch_signal	
volume_vwap	volatility_kchi	trend_vortex_ind_pos	trend_visual_ichimoku_a	momentum_wr	
volatility_atr	volatility_kcli	trend_vortex_ind_neg	trend_visual_ichimoku_b	momentum_ao	

volatility_bbm	volatility_dcl	trend_vortex_ind_diff	trend_aroon_up	momentum_kama	
volatility_bbh	volatility_dch	trend_trix	trend_aroon_down	momentum_roc	

With some extra data instead of just basic stock data, the data set was prepared to be spilt into a training set and a testing set.

Training / Testing:
Because this data is largely time series based, the training/testing split was done chronologically, not randomly. The first 75% of the data was used as training while the last 25% were used as testing. This was particularly interesting as the stock market has been quick schizophrenic the last six months.

Evaluation and Final Results:

Perhaps astoundingly, many models performed really well. In general, the MLP was the most accurate, with linear regression not too far behind, with the random forest being the least accurate.

Some stocks proved more difficult to predict than others. Uber was particularly difficult to predict, while Amazon seemed surprisingly simple. Predicting more days into the future proved much more difficult than predicting just a day into the future.

The figures below illustrate the prior comments, emphasizing the strength of the MLP on tech stocks like Snap, Amazon, and Tesla, especially with a short date range.

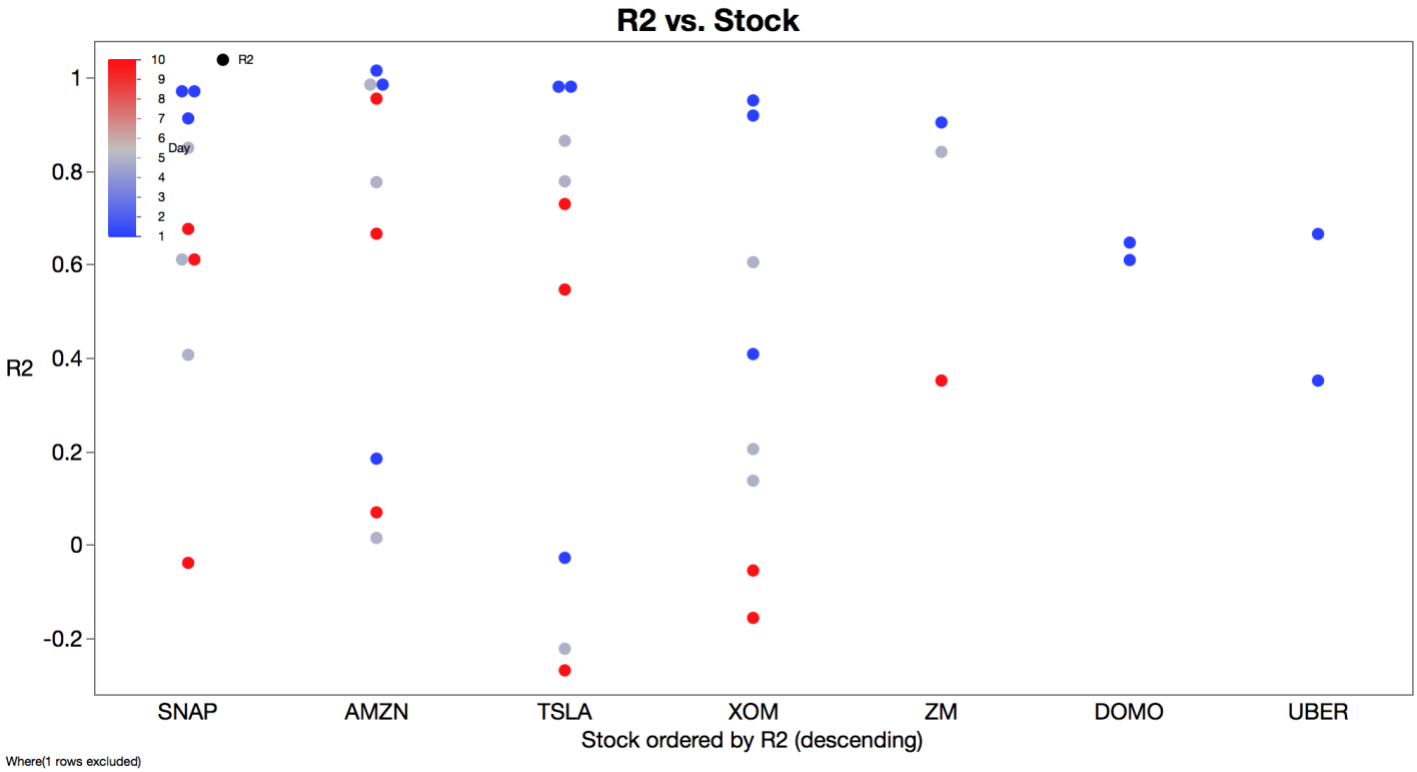


Figure 1: R2 of Prediction by Stock (colored by n days out)

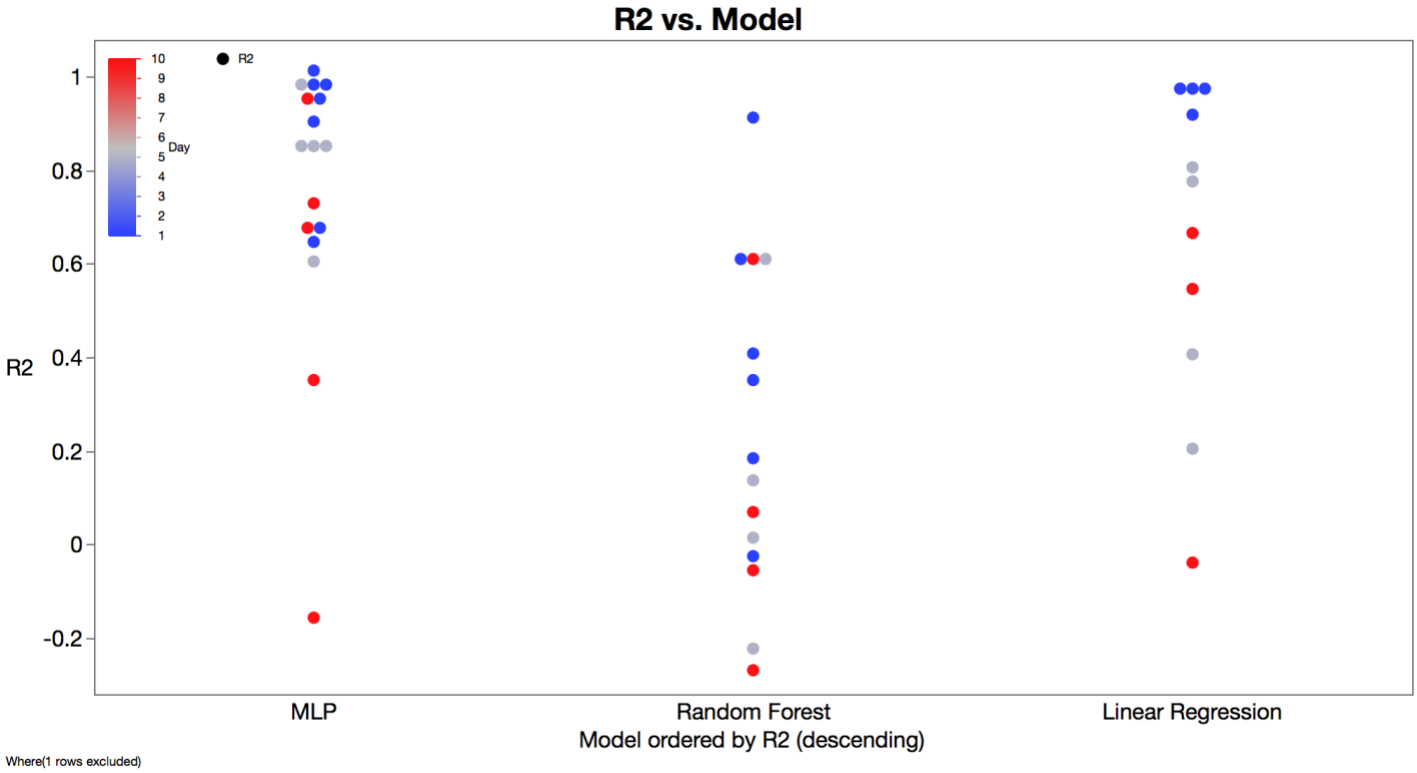


Figure 2: R2 of Prediction by Model Type (colored by n days out)

A total of 13 models landed an R^2 above 0.9. It is notable that many models had poor results, even yielding negative R^2. Having a negative R^2 is never a good thing and may seem mathematically possible, but all it truly means is the fit is very poor [6]. To further understand the details of the results, a table summarizing all models is given below:

Table 3: Summary of Model Results

Stock	Day	Model	R2
AMZN	1	MLP	0.99375216
TSLA	1	MLP	0.98360587
AMZN	5	MLP	0.98061411
TSLA	1	Linear Regression	0.97736714

SNAP	1	Linear Regression	0.97361107
AMZN	1	Linear Regression	0.97245307
SNAP	1	MLP	0.96780837
AMZN	10	MLP	0.95499217
XOM	1	MLP	0.95113252
XOM	1	Linear Regression	0.91876331
SNAP	1	Random Forest	0.91266134
ZM	1	MLP	0.90398633
TSLA	5	MLP	0.86477299
SNAP	5	MLP	0.84971713
ZM	5	MLP	0.84104003
TSLA	5	Linear Regression	0.77803297
AMZN	5	Linear Regression	0.77619589
TSLA	10	MLP	0.72946027
SNAP	10	MLP	0.67616029
AMZN	10	Linear Regression	0.66592002
UBER	1	MLP	0.66533811
DOMO	1	MLP	0.64693224
SNAP	5	Random Forest	0.61817999
DOMO	1	Random Forest	0.60956244

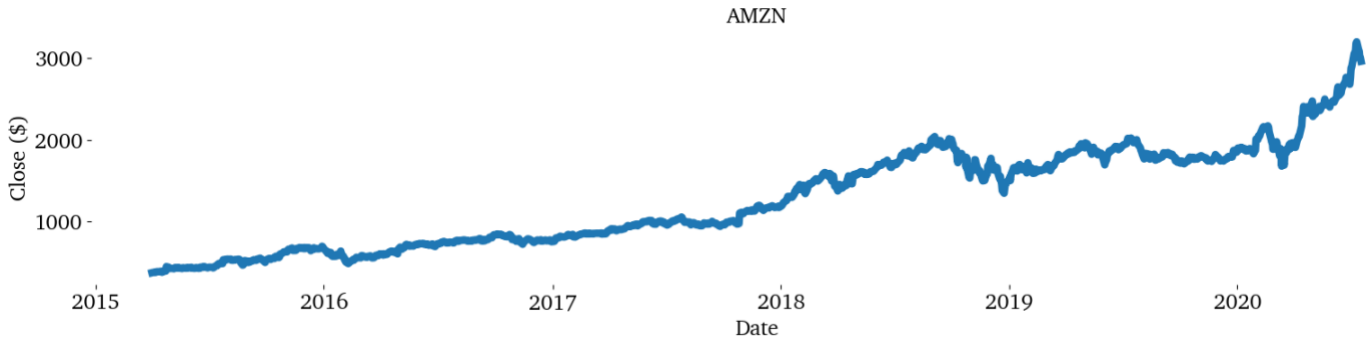
XOM	5	MLP	0.60502313
SNAP	10	Random Forest	0.60331407
TSLA	10	Linear Regression	0.54642354
XOM	1	Random Forest	0.40834109
SNAP	5	Linear Regression	0.40666655
ZM	10	MLP	0.35164462
UBER	1	Random Forest	0.35153674
XOM	5	Linear Regression	0.20499665
AMZN	1	Random Forest	0.18445782
XOM	5	Random Forest	0.13732508
AMZN	10	Random Forest	0.06957171
AMZN	5	Random Forest	0.01475803
TSLA	1	Random Forest	-0.0277741
SNAP	10	Linear Regression	-0.0385873
XOM	10	Random Forest	-0.0548579
XOM	10	MLP	-0.1562955

TSLA	5	Random Forest	-0.2223643
TSLA	10	Random Forest	-0.2686145
ZM	1	Random Forest	-0.4036952
DOMO	5	Random Forest	-0.4113695
UBER	10	Random Forest	-0.5062918
ZM	10	Linear Regression	-0.5206224
ZM	10	Random Forest	-0.5512454
ZM	5	Random Forest	-0.5889274
XOM	10	Linear Regression	-0.716773
DOMO	5	MLP	-0.8240214
DOMO	1	Linear Regression	-0.9247827
UBER	5	Random Forest	-0.9626587
DOMO	10	Random Forest	-1.3450069
UBER	10	MLP	-2.9815505
DOMO	10	MLP	-3.3744434
UBER	5	MLP	-4.5167892

DOMO	5	Linear Regression	-4.566908
ZM	5	Linear Regression	-53.978339
DOMO	10	Linear Regression	-55.772948
UBER	5	Linear Regression	-445.62981
UBER	1	Linear Regression	-1030.8394
UBER	10	Linear Regression	-3804.9193
ZM	1	Linear Regression	-10585169

Because the models were most accurate on Amazon, a complete depiction of the models will be presented.

Amazon:



Most of the world is familiar with Amazon. The founder and CEO, Jeff Bezos, is now the richest person in the world. They have a plethora of

business lines ranging from their most famous logistics e-commerce side to video streaming to online web hosting.

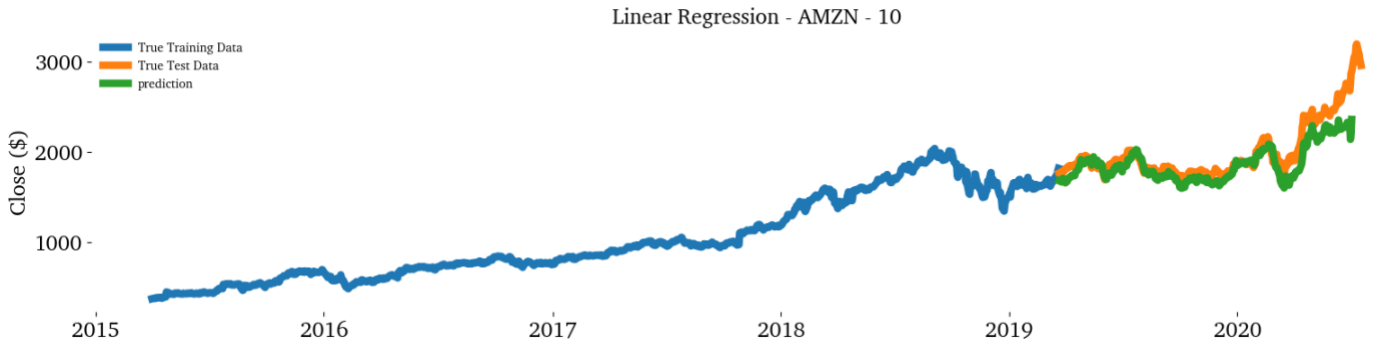
The stock has recently nearly doubled in a short few months due to factors mostly related to how life has changed in 2020 with COVID-19. Despite the rapid, recent changes to the stock's behavior, the models seemed to understand the patterns for the most part. Before showing the performance on individual date ranges and model methods, including a prediction on a time series graph, a table summarizing the predictions is given below:

Table 4: Summary of Results for Amazon

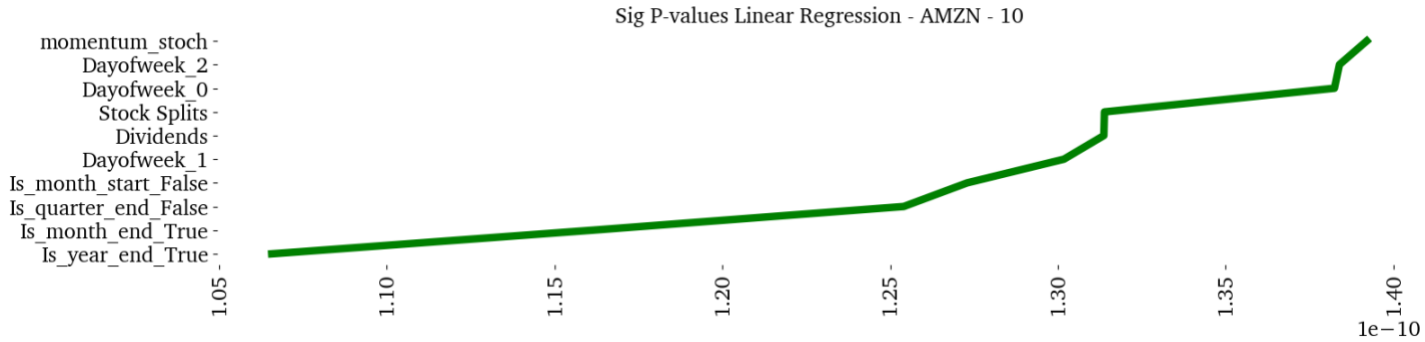
Stock	Day	Model	R2	MSE
AMZN	1	Linear Regression	0.972453	2783.158
AMZN	1	Random Forest	0.184458	82396.95
AMZN	1	MLP	0.993752	631.2404
AMZN	5	Linear Regression	0.776196	19843.71
AMZN	5	Random Forest	0.014758	87356.99
AMZN	5	MLP	0.980614	1718.86
AMZN	10	Linear Regression	0.66592	23506.31
AMZN	10	Random Forest	0.069572	65466.17
AMZN	10	MLP	0.954992	3166.811

The first prediction was used to see if today's factors were indicative of what the closing cost will be ten trading days into the future. The linear regression, random forest, and MLP Neural Net methods will be shown and discussed.

10-Day Forecast:

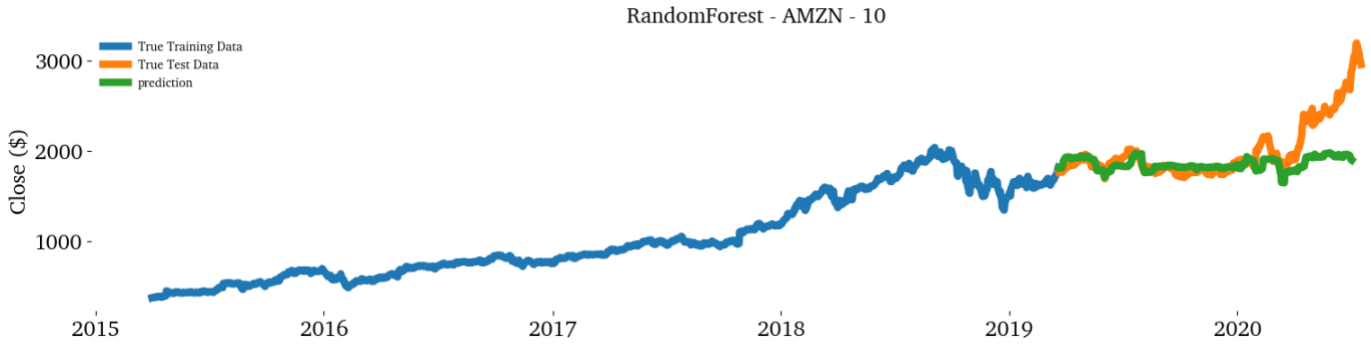


It appears the linear regression predictions at 10 days out were relatively accurate, especially with direction. However, it looks like it wasn't able to continue to predict at the higher slopes that the stock has recently been experiencing. This somewhat makes sense. Amazon really hasn't experience as steep of growth as it currently is and hence, no previous slope can compensate for the current rapid development.

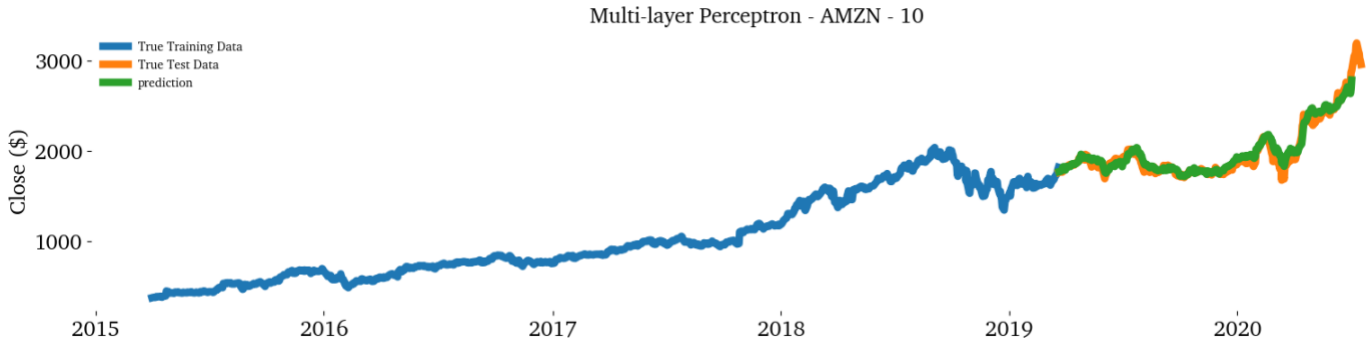


One of the benefits to linear models, is it is interpretable. Interestingly, we see several of the time parameters as highly significant. It appears the

year, month, quarter ends, as well as day of the weeks were important to the model.



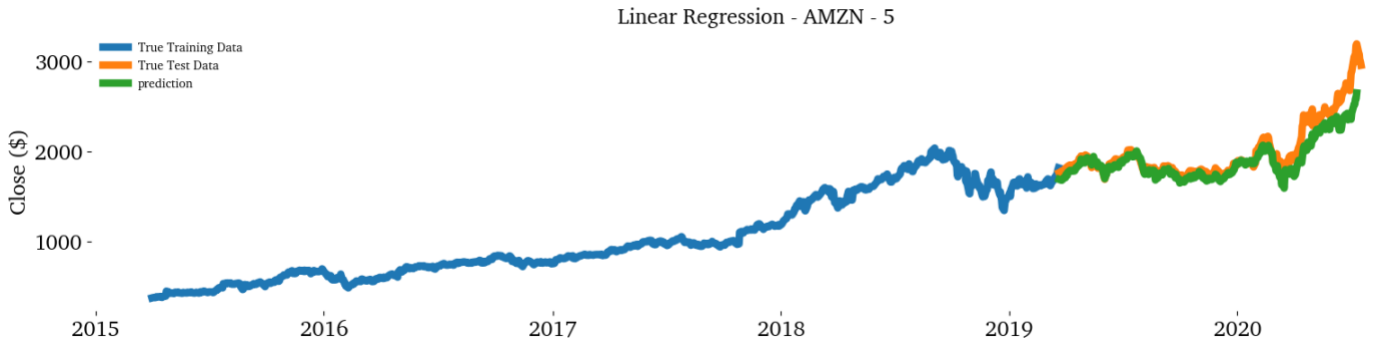
Random forest didn't do as well as linear regression. Despite experimenting with different depths and number of trees, the model struggled to not overfit the training data. It also appeared to really capture trends as many of the random forest predictions remained relatively flat.



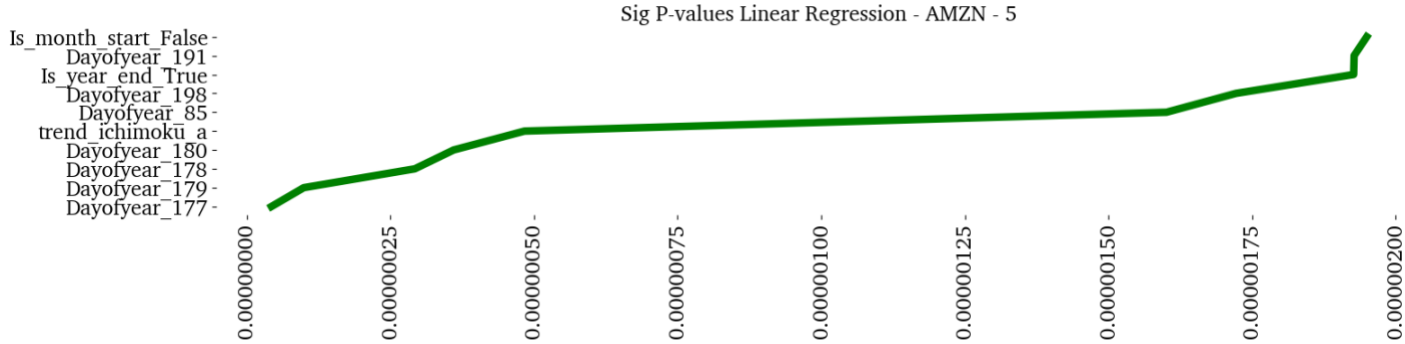
The MLP performed incredibly well, even at 10 days out. It ended up with an $R^2 = 0.95$. It appears to truly trace the actuals, with the exception of the magnitude of some of the more dramatic days.

5-Day Forecast:

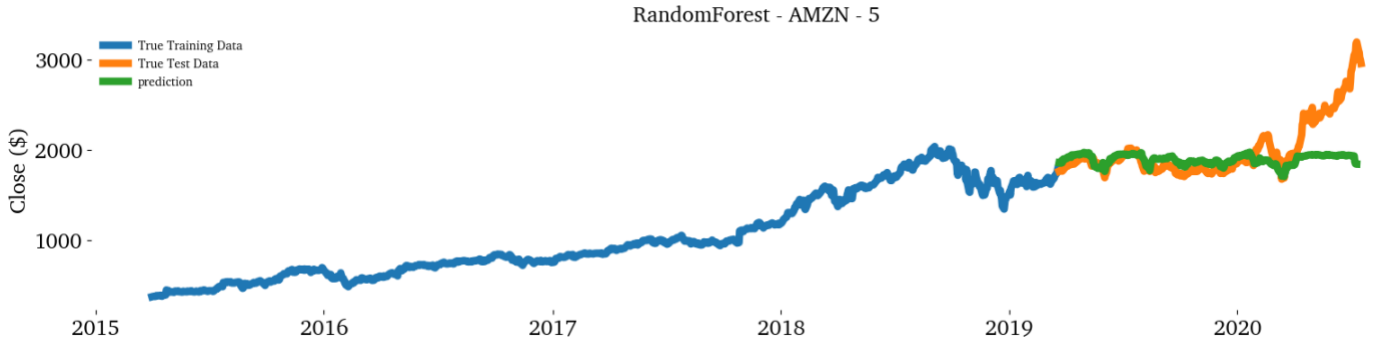
Next, a simple five-day forecast shall be assessed.



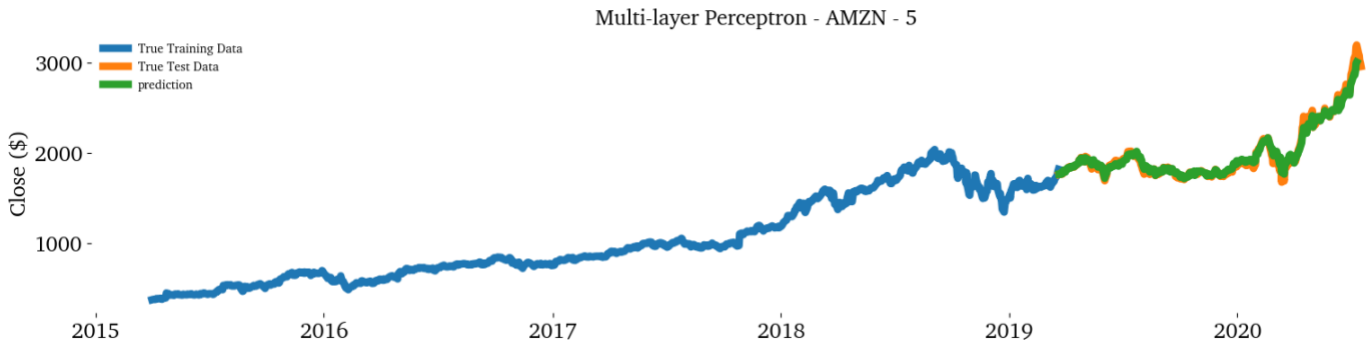
The linear regression predictions at five days appeared pretty similar to what they did at 10 days, however, it looks like some of the more dramatic peaks and troughs were more properly capture. The R^2 improved from 0.66 to 0.77.



Predicting Amazon five days out seems to have a couple of days in the year, just over the half way point, that seem to have highly predictive power. The Ichimoku trend also comes into play.



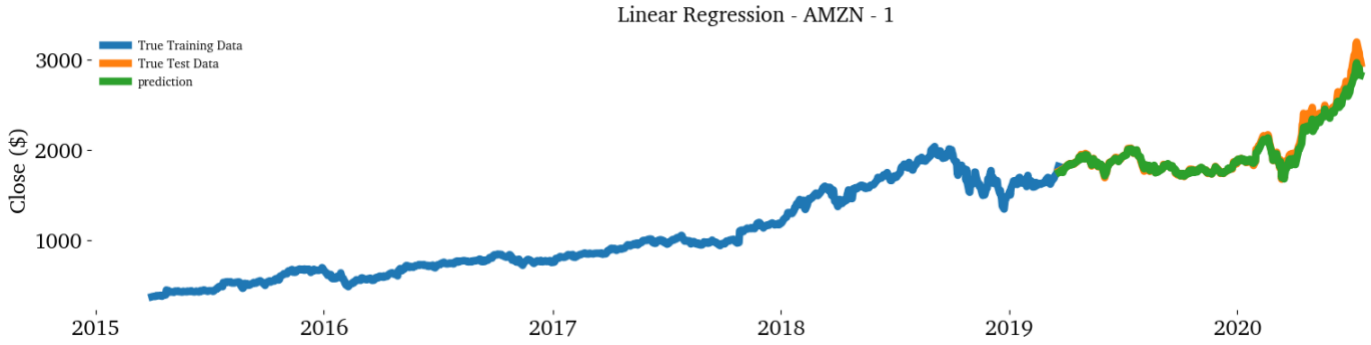
The random forest actually performed worse at five days then what it did on ten days. It still struggled with predicting any of the larger trends or fluctuations, and remained relatively flat.



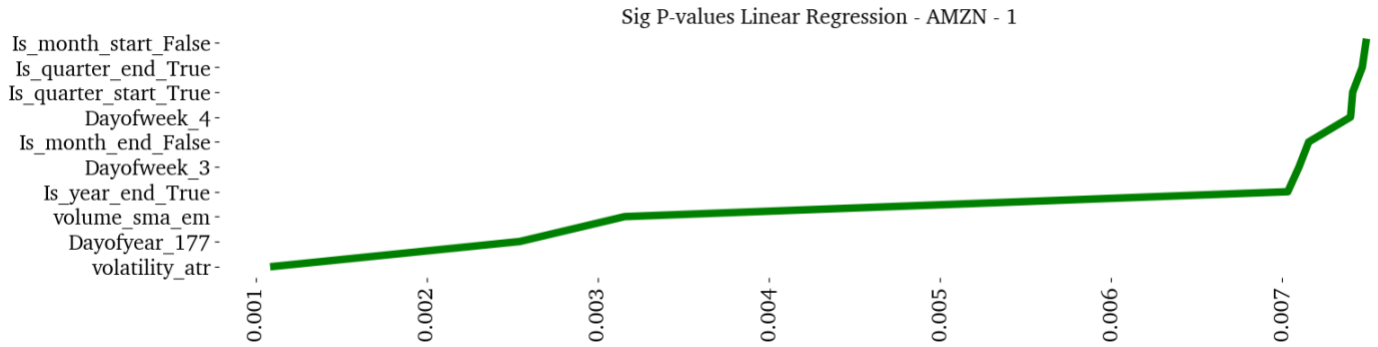
Once again, the MLP performed great at five days, capturing 98% of the variance. It appears this model truly captures most of the price fluctuation and could be used to trade for prophit.

1-Day Forecast:

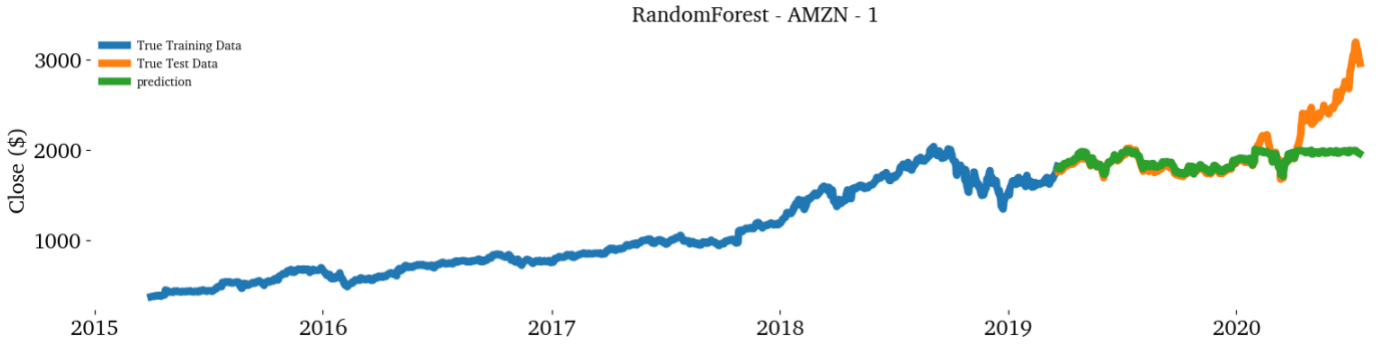
Finally, the one-day forecast will be analyzed. This forecast will be the shortest, but also the most accurate.



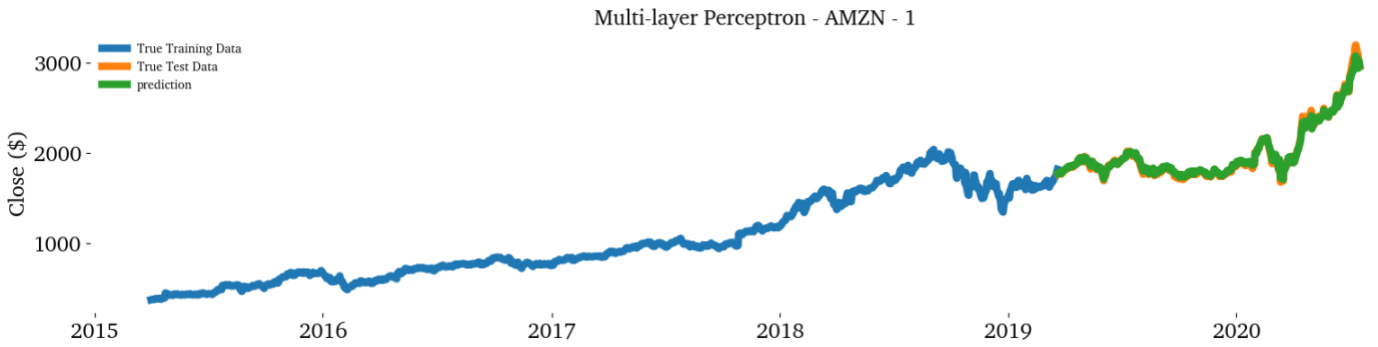
The linear regression model proved very accurate, jumping from an R^2 of 0.77 to 0.97! Most of the variation in tomorrow's price can be explained by the indicators from the day before.



Volatility, Volume SMA, and some of the dates play key roles in the one-day out prediction model.



Once again, the random forest failed to understand Amazon's most recent burst in price. It proved incapable of predicting that steep of slope changes.



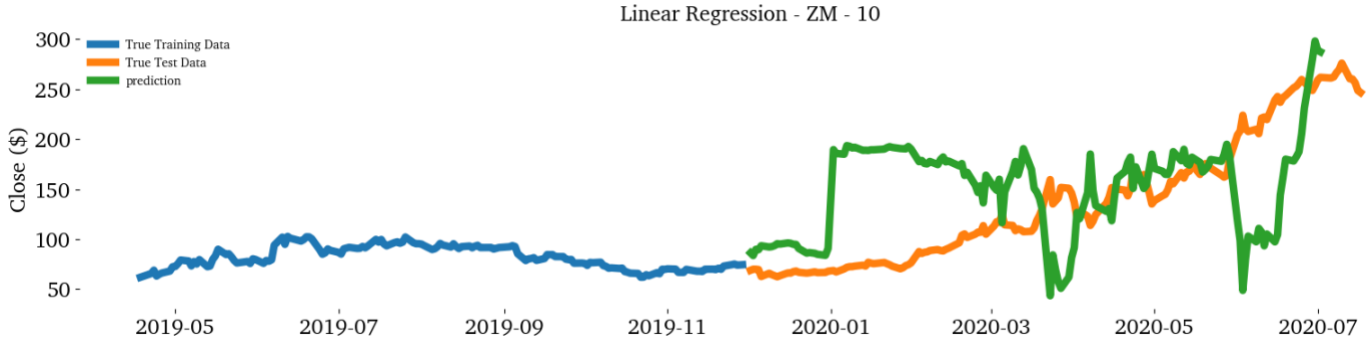
The MLP was accurate at five days, but even more accurate at one day with an R^2 of 0.99! This means only 1% of variance remained unaccounted for.

It seems that MLP or linear regression could be used to trade the Amazon stock for profit. It appears accurate enough in which buying with a predicted price of *n* days later would lead to gains. It is unclear with the random forest struggled, but errors to overfitting the training data is a possible culprit.

Conclusion:

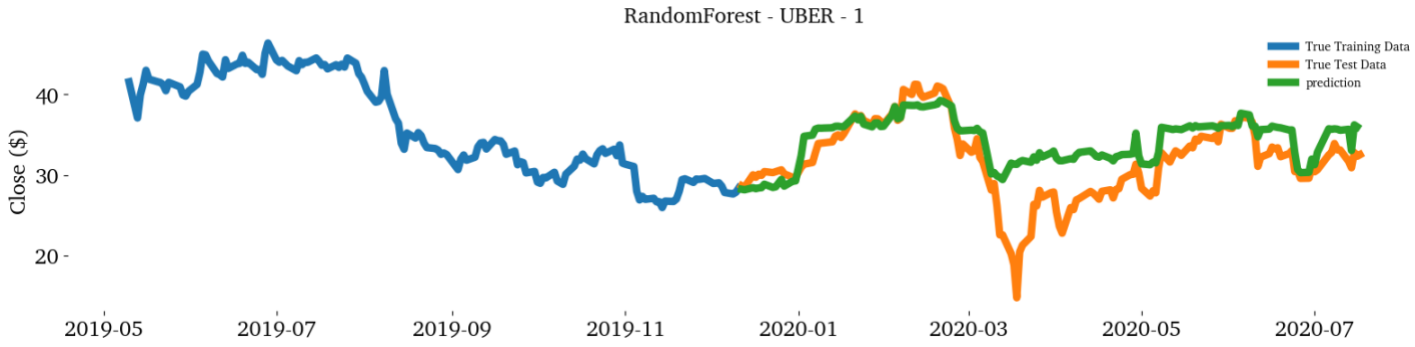
More than just Amazon proved accurate. Stocks like Tesla and Snap also had high average predictions across models and hold periods. Uber, Zoom, and DOMO did not prove as easy to predict. The other stocks' results can be seen in the sorted table above by accuracy, or in individual tables in the appendix.

Some particular cases may be of interest and are shown below.

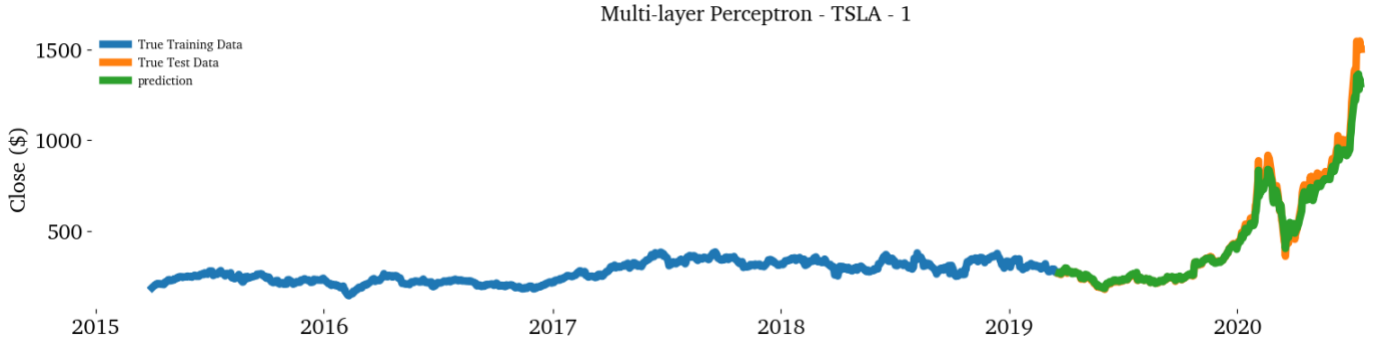


Zoom is another interesting stock that has soared due to COVID-19 situations. It is interesting to see how the linear regress model tried to account for the changes since the training data was all pre 2020. The

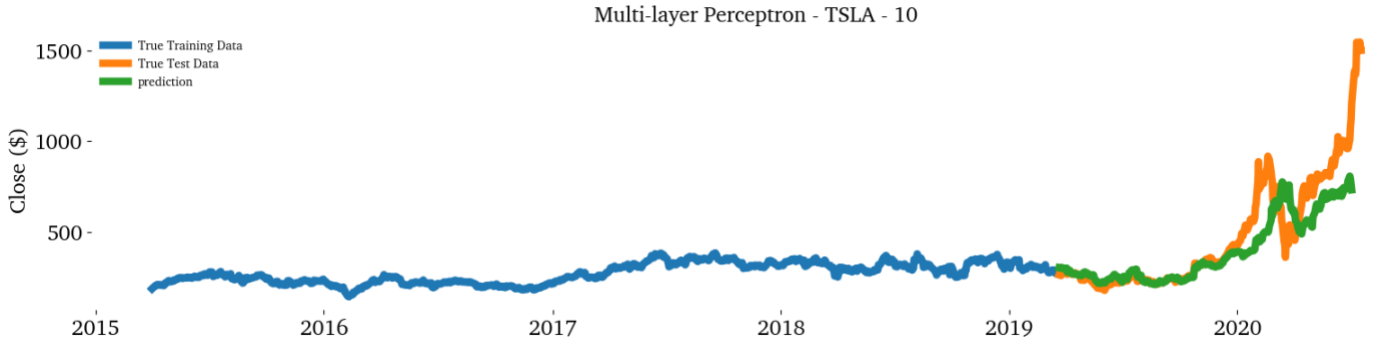
model definitely has some biases where it really mis-predicts as shown by the green and orange lines having large gaps.



The random forest model rarely performed well, but it actually did decently with Uber at a prediction of one day out. It does miss the large downward spike in March of 2020 (COVID-19 related), but other than that, it tracks the price decently well.



Tesla is currently a stock market favorite, and it is clear why. Recently the stock has been going nearly vertically upwards! The MLP one-day out does a pretty good job, at least directionally.



Ten-days out is much harder to do. The Tesla stock is changing so quickly that the algorithm is unable to keep up!

Future Enhancement:

For this study, only one day's set of parameters was used to predict a future price. It would be interesting to use a larger subset, of previous days, perhaps the entire past week's parameters. This could be especially useful if trying to predict the stock in the longer view point, maybe a month out, or even six months, or perhaps even a year. It would have also been interesting to try even more stocks, perhaps in industries that weren't included. Pharma, financial, traditional auto, and semiconductor stocks would be interesting to analyze. With the models that were created, it would be interesting to dive deeper to further understand what input parameters are driving the predictions.

Next steps in this process would include seeing what an average daily profit would look like. If today's price was below the predicted price in the future, the stock should be purchased and sold for profit hopefully. If today's price is above the predicted price in the future, the stock should shorted for the later date. These average profits could be calculated for the testing data and compared. At the end of the day, knowing a stock's future price isn't useful unless profit is reachable.

Finally, more models could be tested and created, or perhaps different parameters and tuning could be used. A LSTM (long short term memory) neural net would be of special interest.

Appendix:

Stock	*Day*	*Model*	*R2*	*MSE*
TSLA	1	Linear Regression	0.977367	2046.945
TSLA	1	Random Forest	-0.02777	92953.2
TSLA	1	MLP	0.983606	1482.706
TSLA	5	Linear Regression	0.778033	17348.29
TSLA	5	Random Forest	-0.22236	95536.42
TSLA	5	MLP	0.864773	10568.95
TSLA	10	Linear Regression	0.546424	29504.25
TSLA	10	Random Forest	-0.26861	82520.87
TSLA	10	MLP	0.72946	17598.07

Stock	*Day*	*Model*	*R2*	*MSE*
SNAP	1	Linear Regression	0.973611	0.317019
SNAP	1	Random Forest	0.912661	1.049229
SNAP	1	MLP	0.967808	0.386729
SNAP	5	Linear Regression	0.406667	6.577244

SNAP	5	Random Forest	0.61818	4.232566
SNAP	5	MLP	0.849717	1.665922
SNAP	10	Linear Regression	-0.03859	9.673346
SNAP	10	Random Forest	0.603314	3.694711
SNAP	10	MLP	0.67616	3.016225

Stock	***Day***	***Model***	***R2***	***MSE***
DOMO	1	Linear Regression	-0.92478	79.82177
DOMO	1	Random Forest	0.609562	16.19166
DOMO	1	MLP	0.646932	14.64191
DOMO	5	Linear Regression	-4.56691	221.7655
DOMO	5	Random Forest	-0.41137	56.22387
DOMO	5	MLP	-0.82402	72.66244
DOMO	10	Linear Regression	-55.7729	2030.432
DOMO	10	Random Forest	-1.34501	83.86701
DOMO	10	MLP	-3.37444	156.4479

Stock	*Day*	*Model*	*R2*	*MSE*
UBER	1	Linear Regression	-1030.84	23556.39
UBER	1	Random Forest	0.351537	14.8041
UBER	1	MLP	0.665338	7.640168
UBER	5	Linear Regression	-445.63	10471.37
UBER	5	Random Forest	-0.96266	46.01512
UBER	5	MLP	-4.51679	129.3428
UBER	10	Linear Regression	-3804.92	92165.3
UBER	10	Random Forest	-0.50629	36.47682
UBER	10	MLP	-2.98155	96.41844

Stock	*Day*	*Model*	*R2*	*MSE*
XOM	1	Linear Regression	0.918763	12.20698
XOM	1	Random Forest	0.408341	88.90524
XOM	1	MLP	0.951133	7.343039
XOM	5	Linear Regression	0.204997	118.0638

XOM	5	Random Forest	0.137325	128.1135
XOM	5	MLP	0.605023	58.65696
XOM	10	Linear Regression	-0.71677	249.9407
XOM	10	Random Forest	-0.05486	153.5741
XOM	10	MLP	-0.1563	168.3422

Stock	***Day***	***Model***	***R2***	***MSE***
ZM	1	Linear Regression	-1.1E+07	4.26E+10
ZM	1	Random Forest	-0.4037	5655.218
ZM	1	MLP	0.903986	386.8206
ZM	5	Linear Regression	-53.9783	206773.5
ZM	5	Random Forest	-0.58893	5975.955
ZM	5	MLP	0.84104	597.8483
ZM	10	Linear Regression	-0.52062	4990.769
ZM	10	Random Forest	-0.55125	5091.275
ZM	10	MLP	0.351645	2127.939

References:

[1] “Seeking Alpha.” https://seekingalpha.com/article/4230982-algo-trading-dominates-80-of-stock-market

[2] “yfinance.” https://pypi.org/project/yfinance/

[3] "fastai." https://docs.fast.ai/

[4] "ta." https://technical-analysis-library-in-python.readthedocs.io/en/latest/

[5] "ta glossary." https://technical-analysis-library-in-python.readthedocs.io/en/latest/ta.html#momentum-indicators

[6] "negative R^2." https://stats.stackexchange.com/questions/12900/when-is-r-squared-negative