Machine Learning Engineer Nanodegree Capstone Project

Predicting Stock Prices with LSTM

Tarek A. Atwan

August 2021

1. Definition

Project Overview

During the COVID-19 pandemic period in 2020, there were a lot of interesting and unexpected behaviors in the stock market that have extended into 2021. The pandemic impact was more like an accelerators, propelling some companies forward at record speed while others were collapsed at plummeted as if they were hit by a tornado[1]

The pandemic also shifted the consumer behavior forcing them to adapt at an accelerated speed, thus impacting many firms who were not ready to embrace such a dramatic and quick change. This drift have benefits several market sectors at the expense of others. Examples include the reliance on online shopping (e-commerce) for almost anything and everything, subscribing to video streaming and gaming services for entertainment, and phenomenon of accepting the idea that people can work remotely at the comfort of their home.

During the pandemic, there was another interesting phenomneon that made predicting stock market even less stable and more volatile. This was headlined in many news outlets describing the mass interest from the general public to jump into the stock market for day trading[2]. Many of those investors were not your typical day trader or investor, they do not apply statistics analysis, or focus on fundamental analysis, nor are they interested in technical analysis or technical indicators. They were pretty much riding a wave based on gut instinct, speculations, hype, and other factors.

As the impact of COVID-19 on the stock market hit hard in 2020, it's repel effect still plays into 2021. In this paper, the goal is to investigate the potential of using machine learning from past data to predict future stock prices. More specifically, the intersect is to examine Long-Short Term Memory (LSTM) neural network on time series stock data to see if these proven algorithms and approaches are still valid for making predictions.

The paper will investigate the use of classical time series analysis techniques on hand picked stocks (top and worst performers of 2020). These classical techniques include Monte Carlo simulation[3], Autoregressive Integrated Moving Average (ARIMA)[4], and Vector Autoregressive (VAR)[5]. These will help build an intuition on these statistical approaches and act as benchmark to the more complex deep learning techniques using LSTM[6].

We will obtain the stock data using a AlphaVantage service to pull historical stock data. In particular, we will use a Python wrapper SDK that leverages the AlphaVantage API to make our calls to extract data for each of the selected stocks. Throughout the analysis, we will be using the adjusted closing price. AlphaVantage also allows to pull additional data such a list of Technical Indicators, Fundamental data, and Sector data. During the process, we will be comparing univariate time series techniques and multivariate time series techniques and combine additional features (signals) to test our models. The first half to he journey will focus on building a benchmark with classical time series techniques before diving and exploring LSTM models and their robustness to pick up signals from the noise.

Problem Statement

The paper's goal is to investigate the potential of predicting stock prices using historical data. Historical data in this context includes the COVID-19 2020 period which was an anomaly in itself. We explored two approaches: Historical from 2010-2021 and from 2019-2021 to see how the models behave. We hold out data from 2021 to see if the model can learn from past data to predict prices in 2021 post the COVID-19 peak impact period.

Our base or benchmark for the models will be Linear Regression. But we will also attempt to compare the models against each other. These models include statistical methods such as ARIMA, VAR, and Monte Carlo simulation. We will use Facebook Prophet given it’s popularity for Time Series forecasting. Input data will include Adjusted Closing price, RSI (Relative Strength Index), and in the case of LSTM we will feed multiple stocks that seemed to be moving together to predict one stock of interest to determine if they are good predictors.

More specifically, this papers attempt to compare multiple models, and compare the final results from each with a more complex LSTM model. In total we will evaluate 11 models in total.

The table below summarizes the experiments using APPL stock prices.

Model	**Description**
ARIMA	Train on data from 2019-2020 (pre-Covid) and see if we can predict 2021
ARIMA	Train on data from 2019 to March 2021 for early days in the Post-Covid
ARIMA	Train on data from 2020 to June 2021 to cover the Pandemic and Post Pandemic period
ARIMA	Train on all data up to 2021-05-31 and hold-out June 2021 for testing. This is what we will use with other models.
VAR	Train on all data up to 2021-05-31 and hold-out June 2021 for testing. For Multivariate we will include the RSI value.
Prophet	Univariate. Train on all data up to 2021-05-31 and hold-out June 2021 for testing.
Prophet	Multivariate. Train on all data up to 2021-05-31 and hold-out June 2021 for testing. For Multivariate we will include the RSI value
Prophet	Using an additional regressor for RSI. Train on all data up to 2021-05-31 and hold-out June 2021 for testing
LSTM	3 LSTM layers with dropouts using train, val, and test data sets training on 2-day window
LSTM	3 LSTM layers with dropouts using train, val, and test data sets training on 5-day window
Linear Regression	Train on all data up to 2021-05-31 and hold-out June 2021 for testing. We will use a 2-day window.

Once we evaluate the models and realize which model has the best potential, we plan to use the model to train on S&P500 data and use to forecast other stocks.

Metrics

Since we are predicting continue values it makes it a regression problem. We will use regression related metrics to measure and evaluate the performance of our models.

- **Root Mean Square Error**

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2} \tag{1}$$

Where

n = number of data points

Y_i = observed values

$\hat{Y}_i$ = predicted values

- **Mean Absolute Percentage Error**

$$\text{MAPE} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - E_t}{A_t}\right| \tag{2}$$

where

n = number of times the summation iteration happens

A_t = observed value

F_t = predicted value

- R^2 **(R Squared)**

$$R^2 = 1 - \frac{RSS}{TSS} \tag{3}$$

where

RSS = sum of squared residuals

TSS = total sum of squares

When comparing the different models we will compare against those three scores. BIC and AIC will be observed, but for initial judgement it will be MAPE, RMSE, and R^2. We will lean more toward MAPE score to determine best model [7]. MAPE is a good measure of how accurate the forecast model is since it measure the accuracy as a percentage, and a good measure to forecast error.

---

2. Analysis

Data Exploration

We picked a handful of stocks based on their performance in 2020. These stocks were either strong performers or weak performers.

Strong performers : ['TSLA', 'AMZN', 'NFLX', 'NVDA', 'AAPL', 'GME']

Weak performers : ['CCL', 'MRO', 'UAL', 'SLB', 'OKE', 'FARM', 'GLBS']

Additional information regarding the data pulled from AlphaVantage:

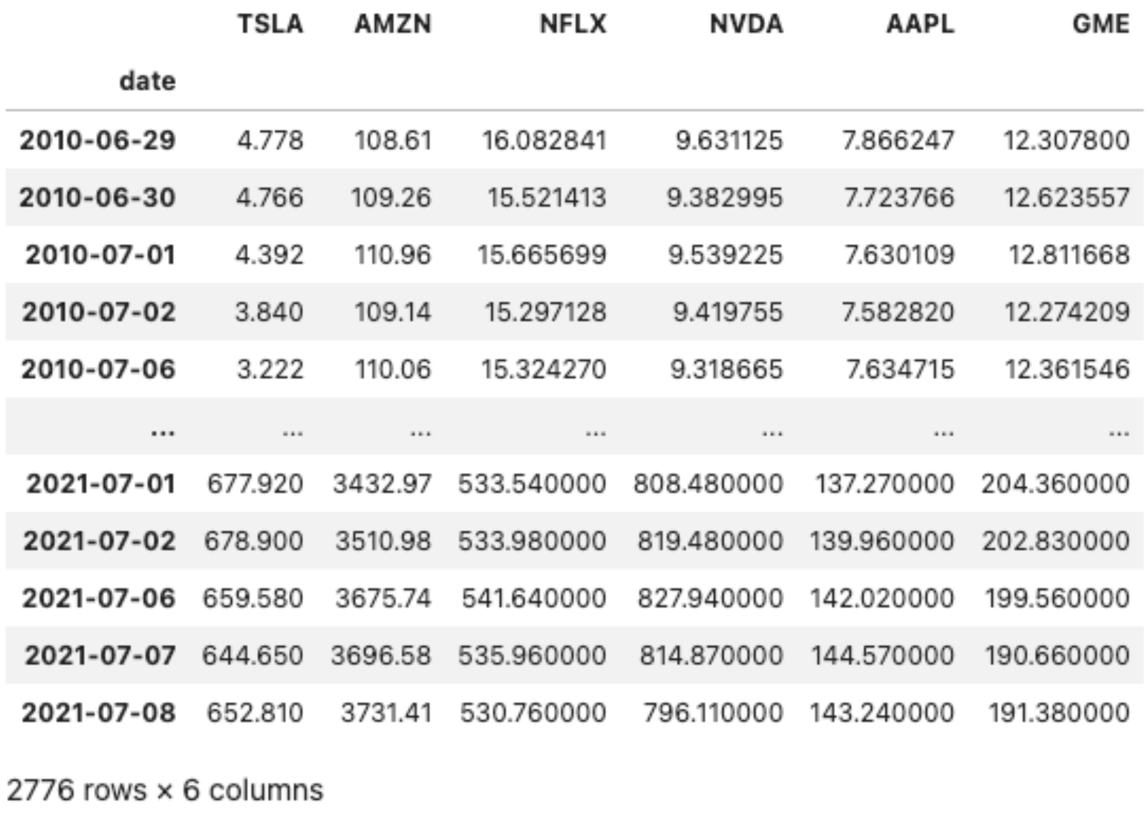
```
Number of records: 2276 trading days
Start date: 2010-06-29
End date: 2021-07-08
Columns:  1. open 2. high 3. low  4. close  5. adjusted close 6. volume 7.
dividend amount  8. split coefficient
Data Frequency: Daily
Additional Features: RSI
```

We will be using Adjusted Close price moving forward.

The API allows us to call each stock individually. We used a function to make a call for each stock, extract the adjusted closing price, and then finally concatenate the results into one DataFrame where columns are the symbol name as shown in the following image.

We have two DataFrames. One for top performers and one for weak performers.

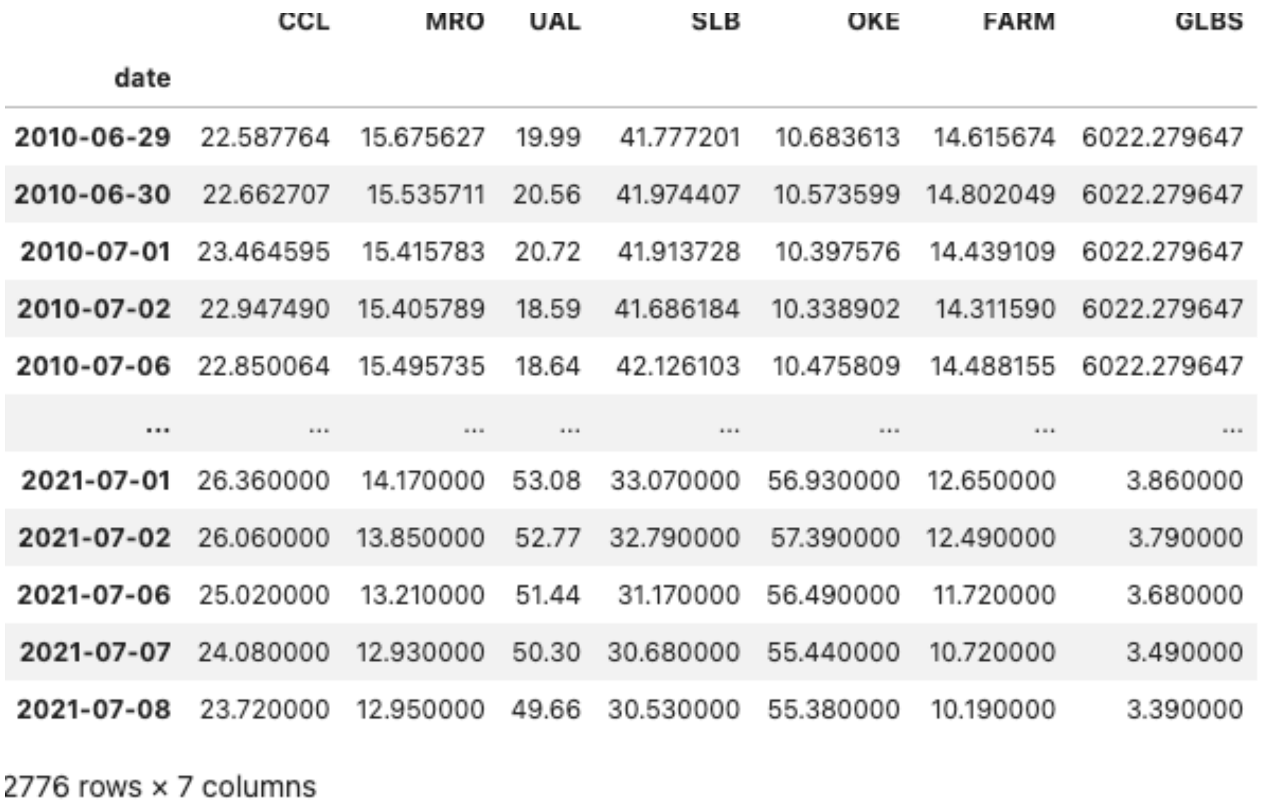
2020 Top Performers



date	TSLA	AMZN	NFLX	NVDA	AAPL	GME
2010-06-29	4.778	108.61	16.082841	9.631125	7.866247	12.307800
2010-06-30	4.766	109.26	15.521413	9.382995	7.723766	12.623557
2010-07-01	4.392	110.96	15.665699	9.539225	7.630109	12.811668
2010-07-02	3.840	109.14	15.297128	9.419755	7.582820	12.274209
2010-07-06	3.222	110.06	15.324270	9.318665	7.634715	12.361546
...	...	...	...	...	...	...
2021-07-01	677.920	3432.97	533.540000	808.480000	137.270000	204.360000
2021-07-02	678.900	3510.98	533.980000	819.480000	139.960000	202.830000
2021-07-06	659.580	3675.74	541.640000	827.940000	142.020000	199.560000
2021-07-07	644.650	3696.58	535.960000	814.870000	144.570000	190.660000
2021-07-08	652.810	3731.41	530.760000	796.110000	143.240000	191.380000

2776 rows × 6 columns

2020 Weak Performers



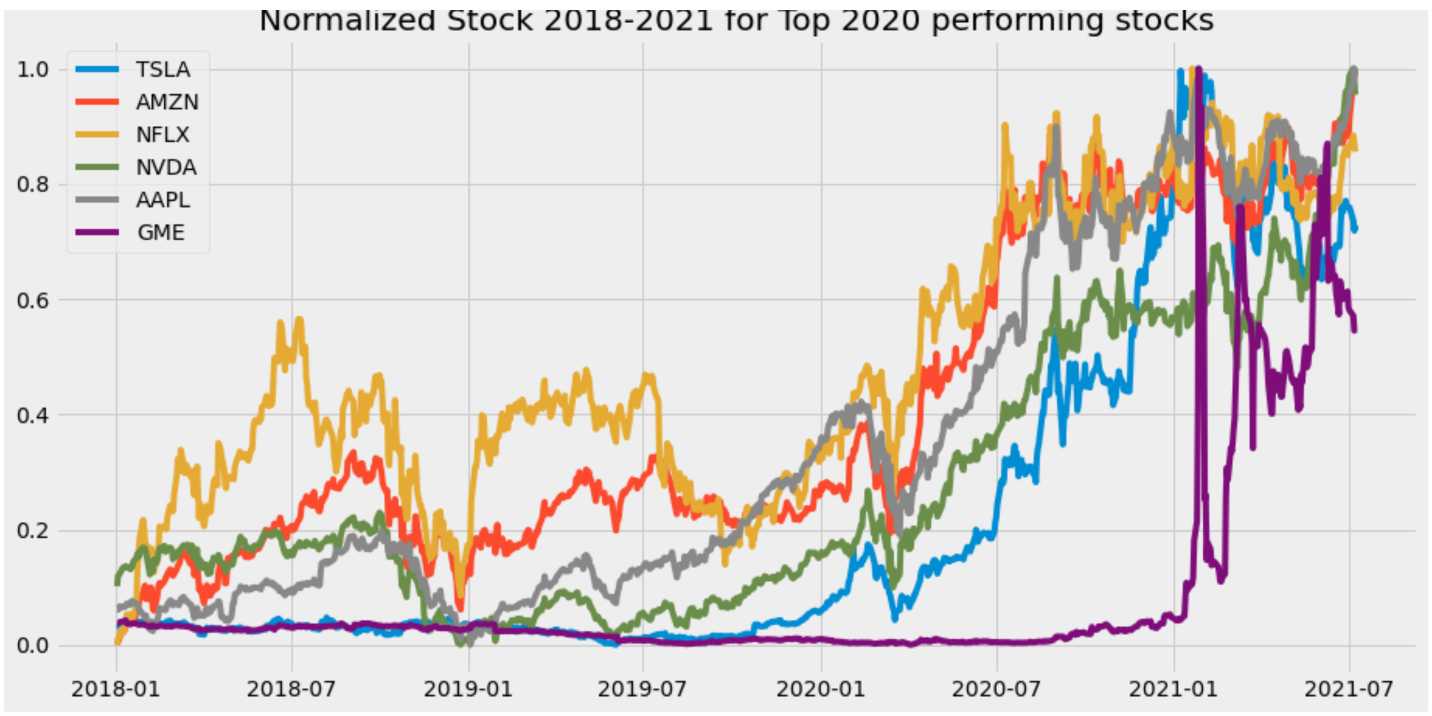
date	CCL	MRO	UAL	SLB	OKE	FARM	GLBS
2010-06-29	22.587764	15.675627	19.99	41.777201	10.683613	14.615674	6022.279647
2010-06-30	22.662707	15.535711	20.56	41.974407	10.573599	14.802049	6022.279647
2010-07-01	23.464595	15.415783	20.72	41.913728	10.397576	14.439109	6022.279647
2010-07-02	22.947490	15.405789	18.59	41.686184	10.338902	14.311590	6022.279647
2010-07-06	22.850064	15.495735	18.64	42.126103	10.475809	14.488155	6022.279647
...	...	...	...	...	...	...	...
2021-07-01	26.360000	14.170000	53.08	33.070000	56.930000	12.650000	3.860000
2021-07-02	26.060000	13.850000	52.77	32.790000	57.390000	12.490000	3.790000
2021-07-06	25.020000	13.210000	51.44	31.170000	56.490000	11.720000	3.680000
2021-07-07	24.080000	12.930000	50.30	30.680000	55.440000	10.720000	3.490000
2021-07-08	23.720000	12.950000	49.66	30.530000	55.380000	10.190000	3.390000

2776 rows × 7 columns

We will be using the two DataFrames for exploratory data analysis to gain an intuition. When building our models we will focus mostly on the top performers.

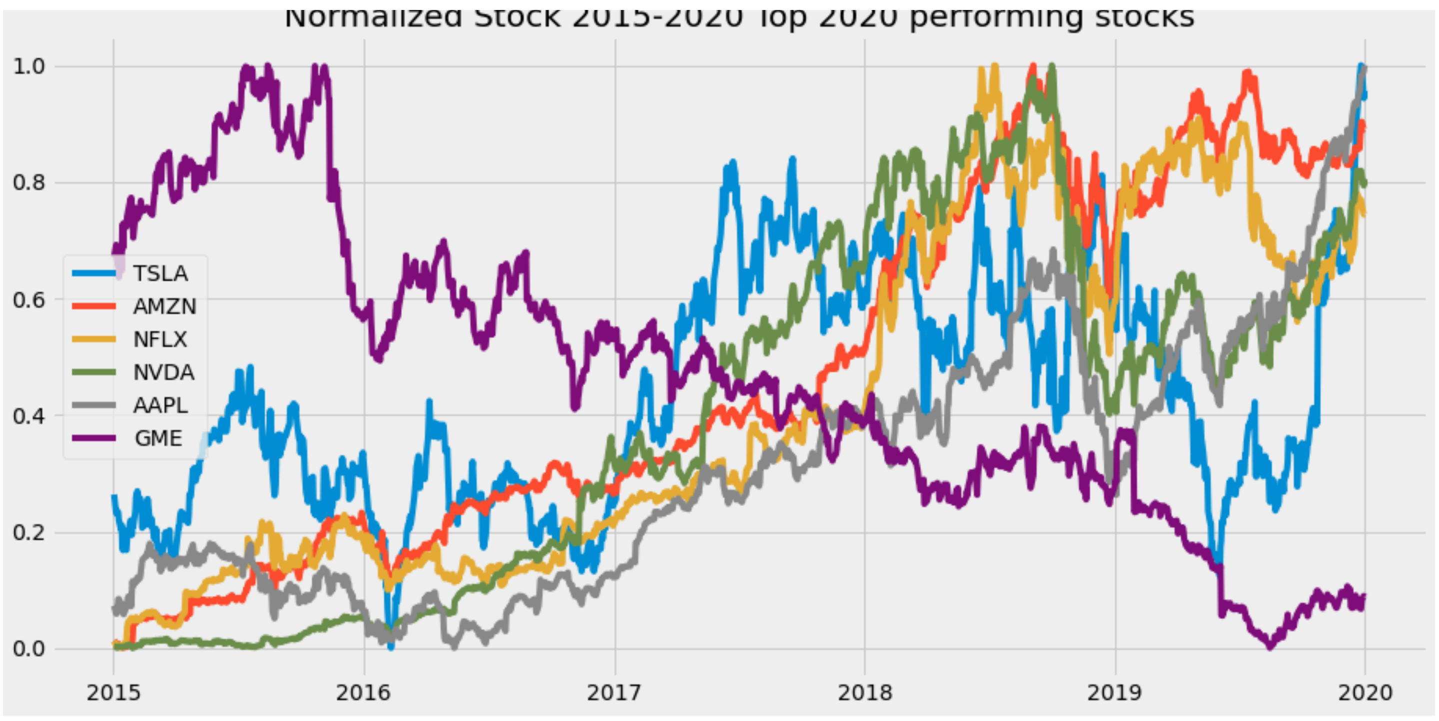
Exploratory Data Visualization

Visualizing the overall price (adjusted closing) movement among the strong performance shows an interesting behavior and synergy on how these stock moved

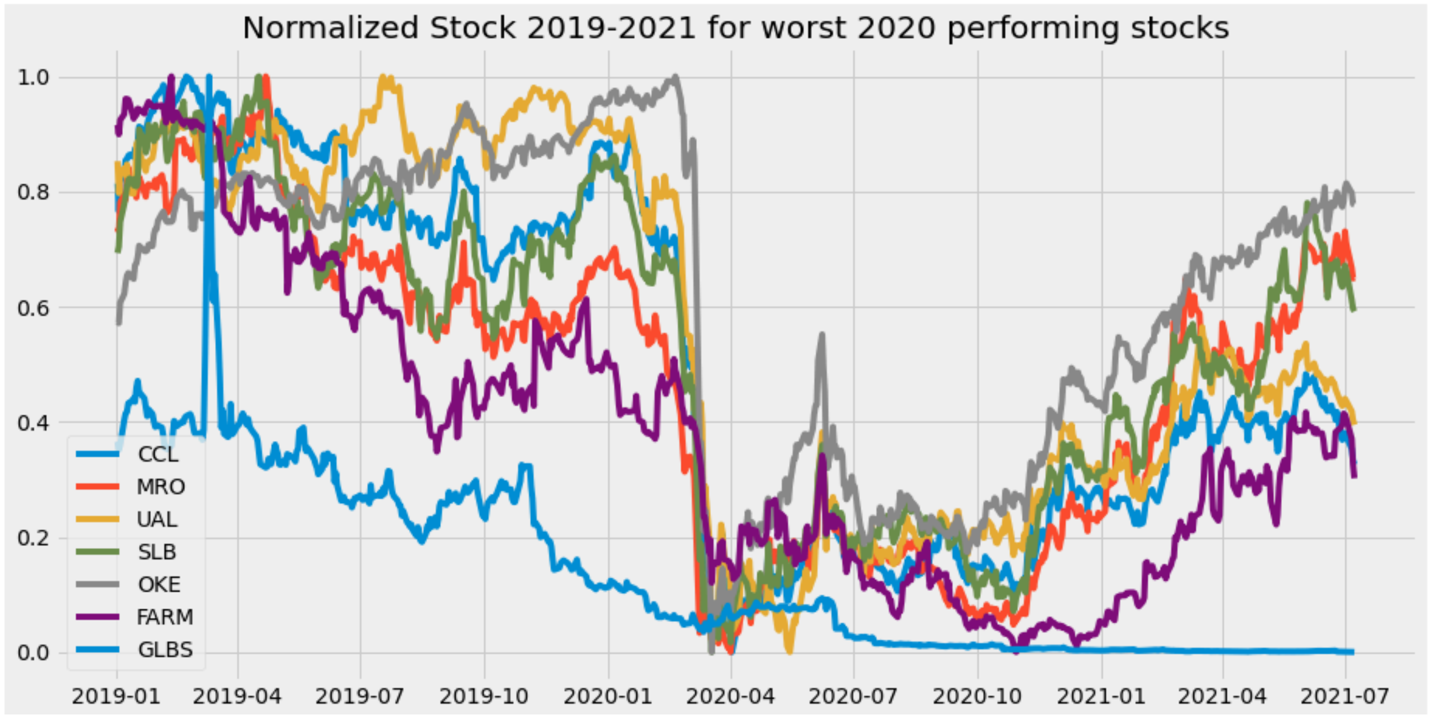


With the exception of GME, these stocks were performing pretty well but took a bug jump in terms of growth in 2020. TSLA, AMZN, NFLX, NVDA, and APPL were moving in the same direction and showing a similar trend. This can indicate the technology sector was getting a major boost from investors during the pandemic.

If we zoom out a little, and go back toward 2015 and observe the performance, we can see that GME was an anomaly. Every indication shows a stock that is plummeting. The other technology stocks were showing growth and a trend upwards but at a normal (slow) pace before the pandemic.



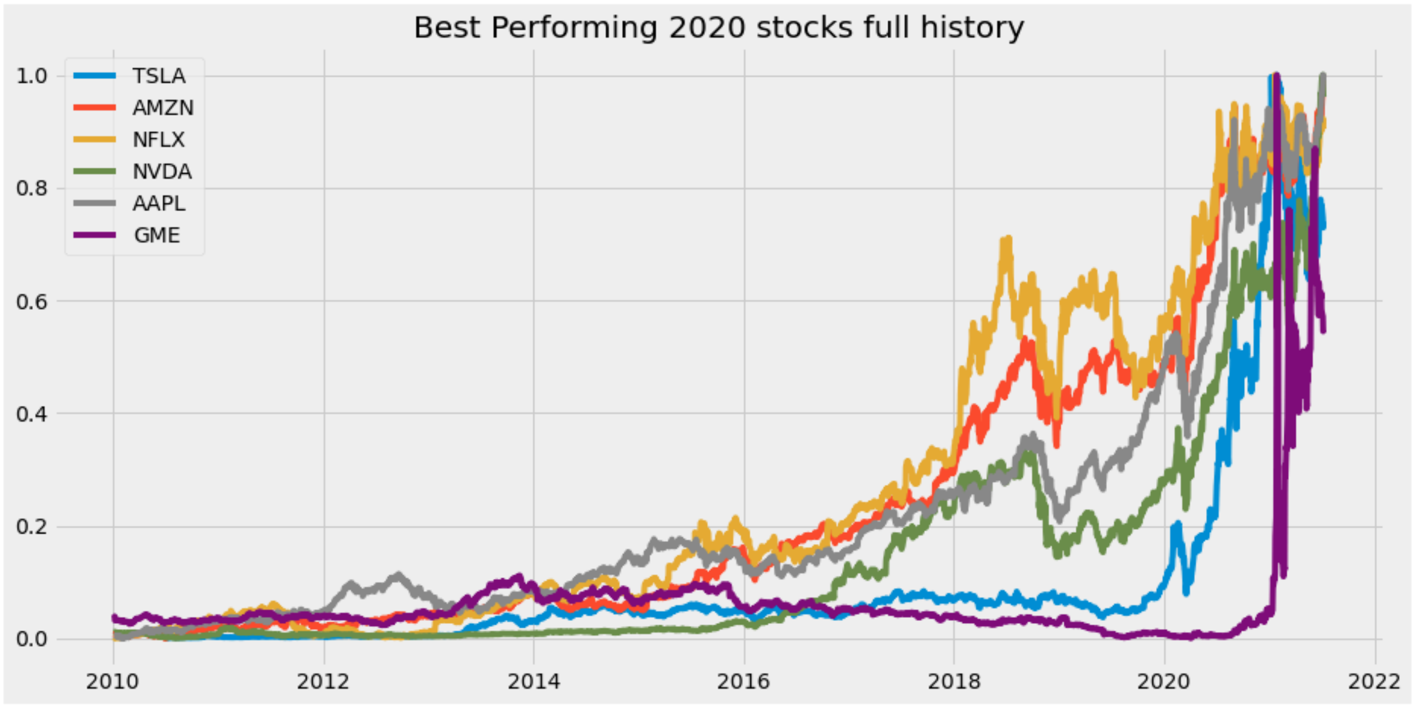
If we examine the weak performers right before the pandemic, we can see that they were headed downward but at a normal pace, right until the pandemic (March 2020) when that accelerated.

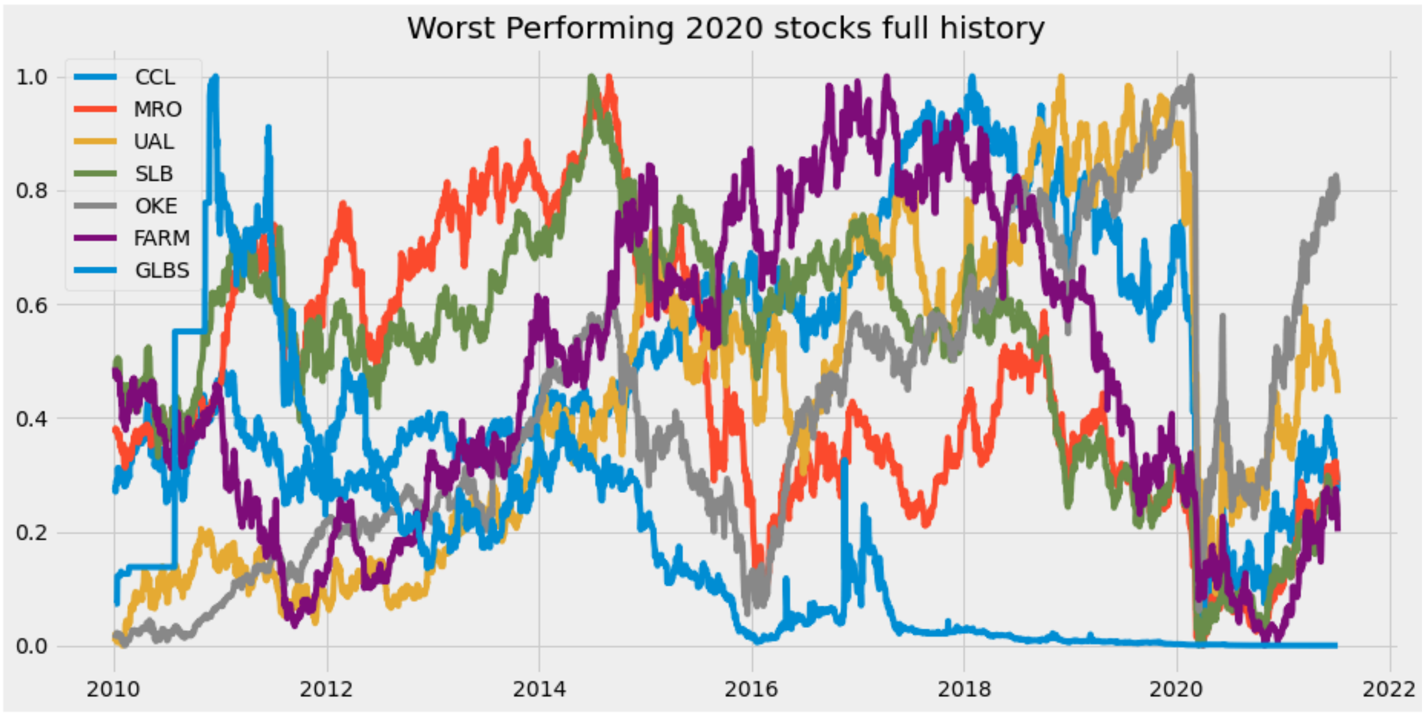


There are signs that some of these stock were trying to recover but never back to the pre-covid period as show in 2019.

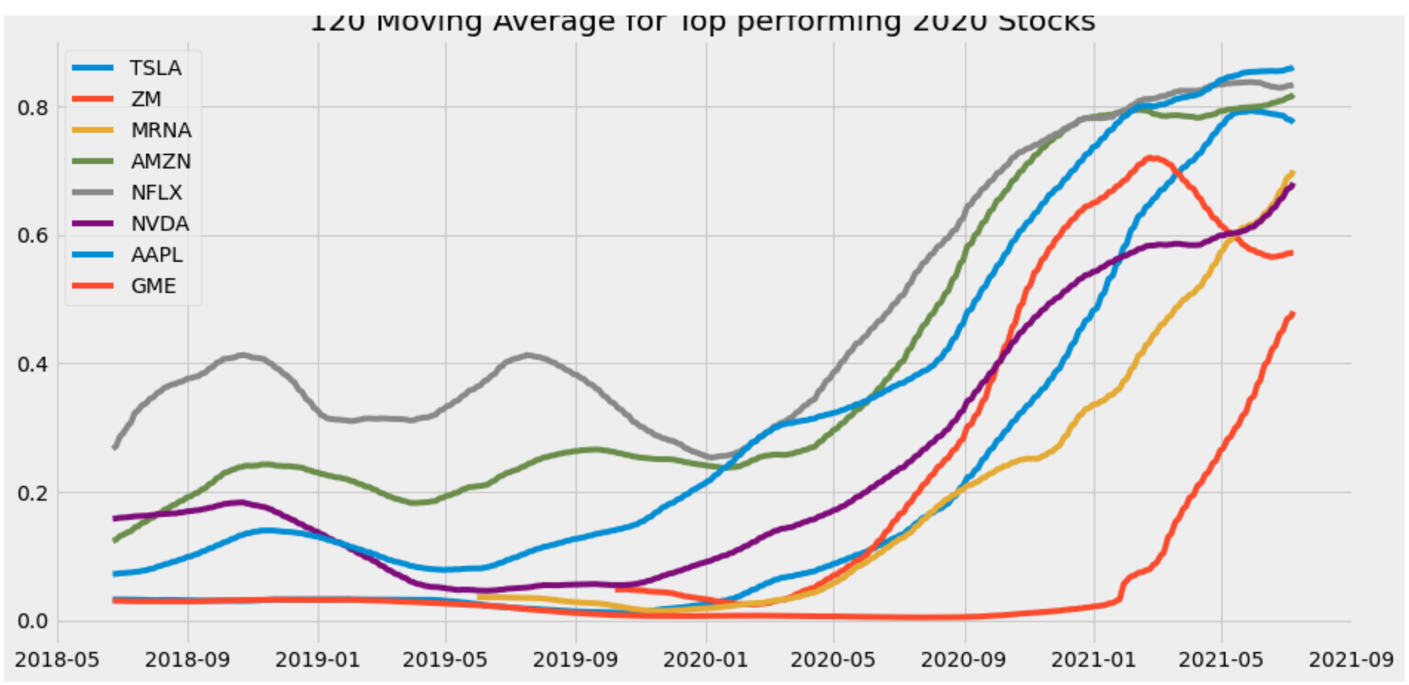
In the following we look at the entire history of the stock that we are allowed to

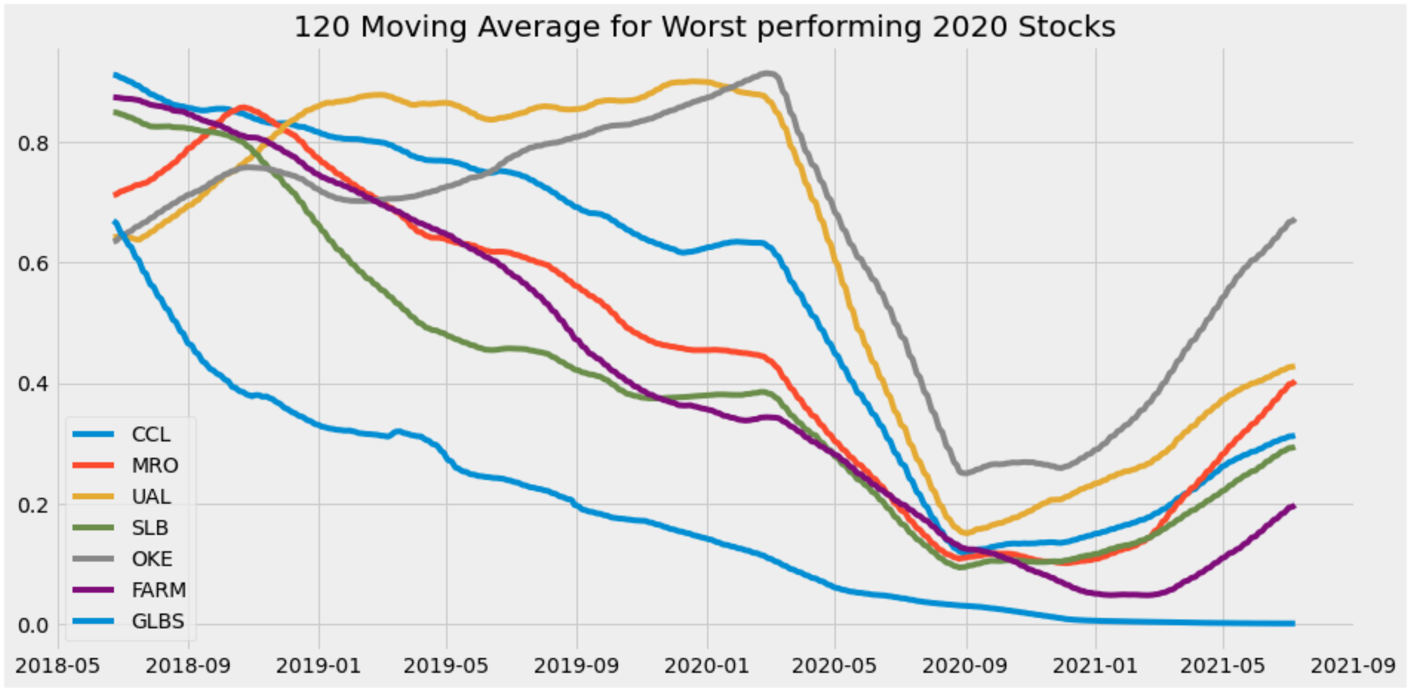
In the following we look at the entire history of the stock that we are allowed to capture from AlphaVantage.



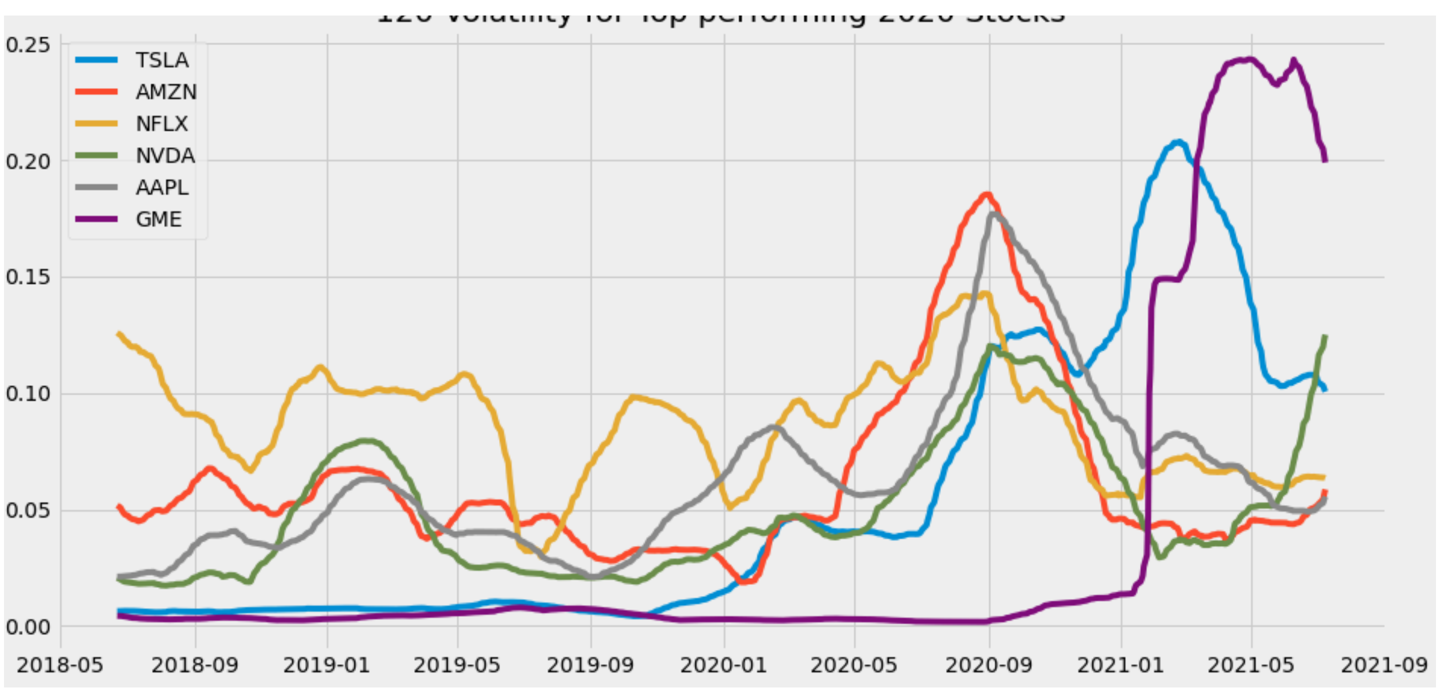


To better capture the trend we can use simple moving average (a smoothing technique). Here we take the 120 moving average for the top and worst performers to gain a better intuition in terms of overall direction and trend.

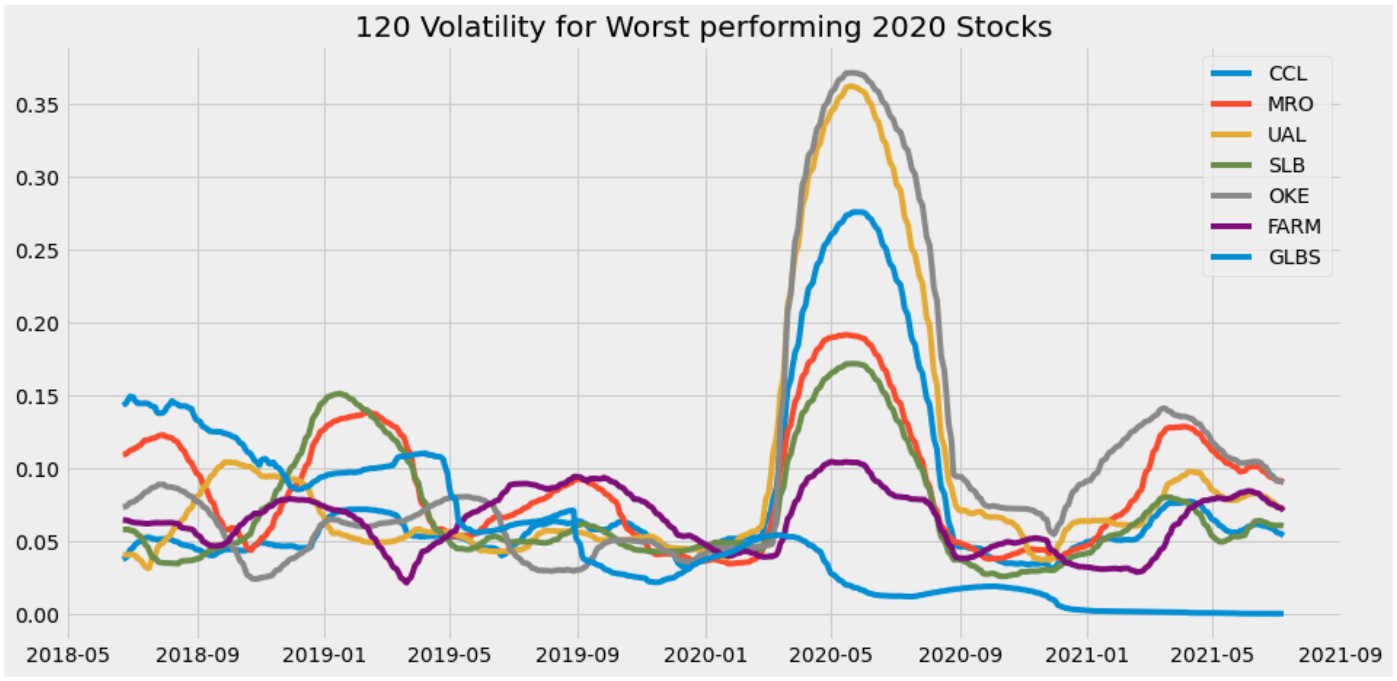




We calculate the volatility by calculating the standard deviation. Similar to the simple moving average, we will use a window of 120 days to get a sense of overall volatility for both groups.



Interesting for the top performers volatility overall seems steady, though some increases (spikes) around August-September of 2020 given the overall market sentiment and perception it makes sense.



On the other hand, the preceding figure shows how the weak performers volatility went extremely high early months of the pandemic which when compare to the 120 day moving average price movement we can see the impact on the price.

Algorithms and Techniques

Algorithms and Techniques

ARIMA

Autoregressive Integrated Moving Average (ARIMA) is a popular time series forecasting technique. It combines an AutoRegressive (AR) model and a Moving Average (MA) model. Unlike ARMA (AR+MA) which expects the time series data to be stationary, ARIMA has an Integrated (I) which is for the differencing. Differencing is a technique to make a non-stationary time series data into stationary. ARIMA requires three parameters (p, d, and q). To select values for AR(p) and MA(q) we can use PACF and ACF plots. Order q for the MA can be obtained from ACF plot, while order p for AR can be obtained from PACF plot.

Vector AutogRession (VAR)

This is a statistical model technique similar to ARIMA which can handle multivariate time series data. ARIMA model works with univariate time series data. VAR can help us capture the relationship between multiple quantities as they change over time.

Linear Regression (AutoRegressive Model)

Linear Regression is a very basic model, yet it should not be underestimated because it can be a very useful tool used by many analysts. Every model has assumptions. For example an ARMA model expects data to be stationary for example, a Linear Regression model has some assumptions that needs to be met in order for the results to be meaningful. One of those assumptions is little or no autocorrelation in the data. In Time Series data, autocorrelation is a big factor as we are observing correlation between a variable (e.g. price_t) and it's lagged version (e.g. price_{t-1}). This implies that Linear Regression may not capture trends properly.

For this reason we will be using a special type of Linear Regression called AutoRegressive Model (Regressing on itself or past values of itself to be more specific).

Prophet

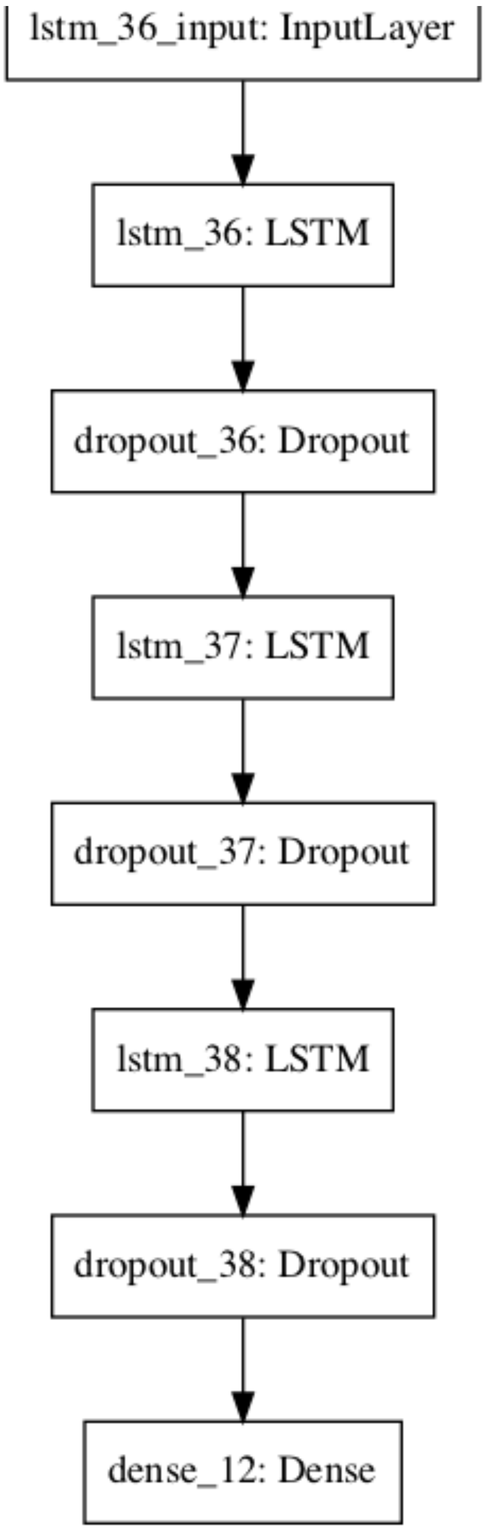
At it's core Prophet is an additive regression model with four main components [8] which can also be called Generalized Additive Model (GAM).

> *In statistics, a generalized additive model (GAM) is a generalized linear model in which the linear response variable depends linearly on unknown smooth functions of some predictor variables, and interest focuses on inference about these smooth functions. GAMs were originally developed by Trevor Hastie and Robert Tibshirani[1] to blend properties of generalized linear models with additive models.*
>
> *[Wikipedia]*

Long Short-Term Memory (LSTM)

LSTM is a special kind of neural network (deep learning architecture) that is can handle sequential data (such as text or time series) with memory. We use Time Series data to forecast the future because we believe the past influence the future, and hence the ability for the neural network to remember what it learned from past events is important. LSTM solved an issue with Recurrent Neural Networks (RNNs) related to lack of ability to remember called "**Vanishing Gradient**".

The architecture for the LSTM is highlighted in the following image:


lstm_36_input: InputLayer
lstm_36: LSTM
dropout_36: Dropout
lstm_37: LSTM
dropout_37: Dropout
lstm_38: LSTM
dropout_38: Dropout
dense_12: Dense

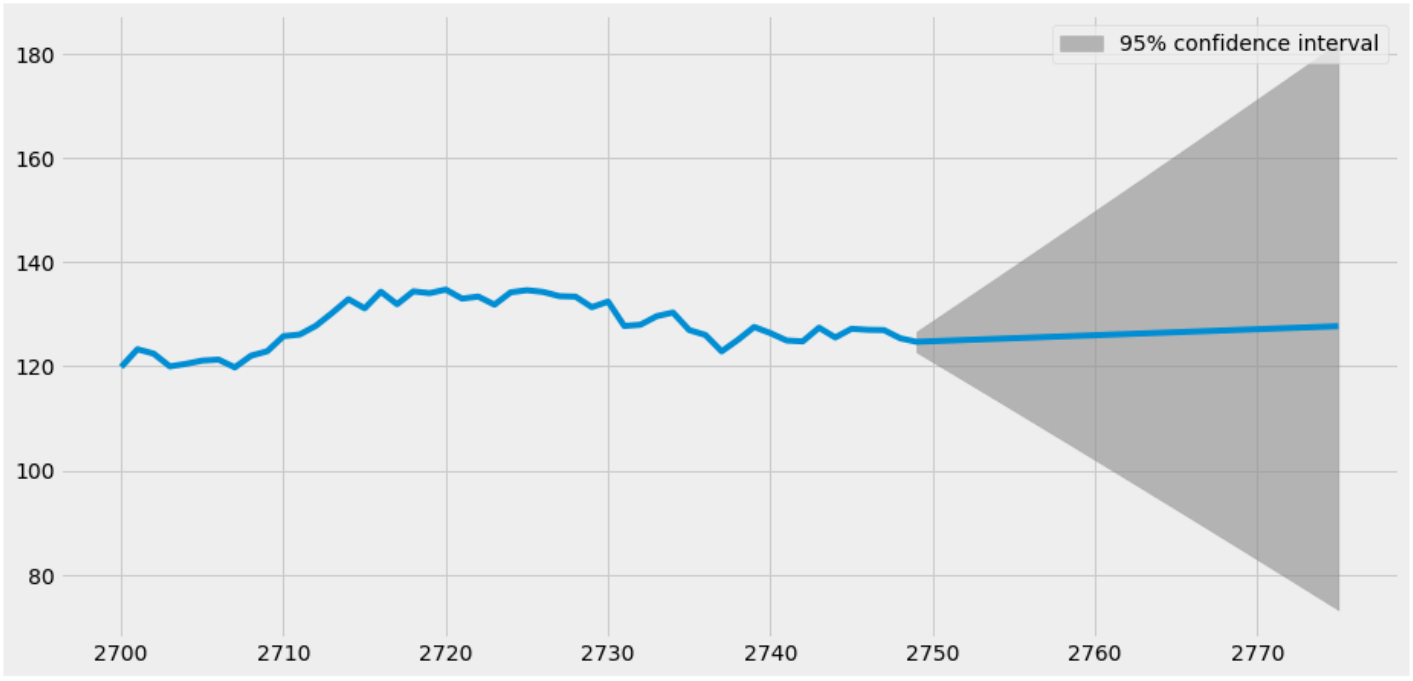
Benchmark

Benchmark

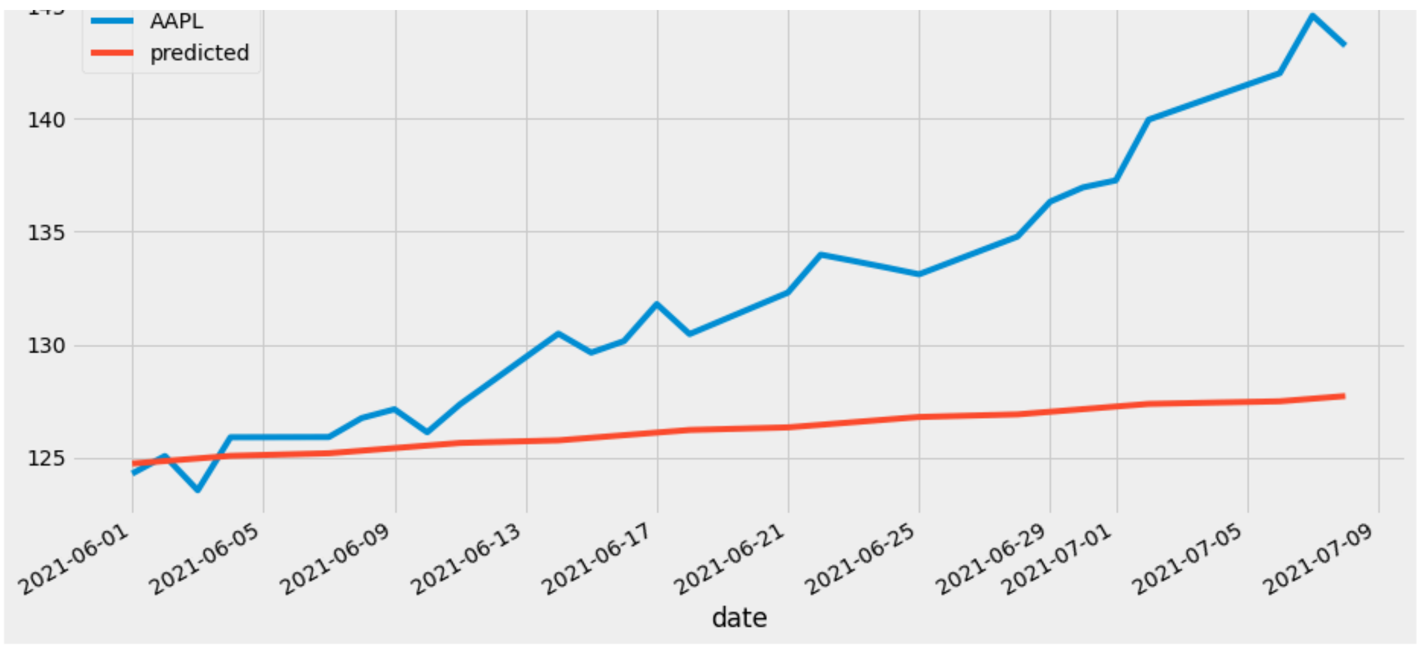
The benchmark model for this project would be an AutoRegressive model. Though, the intent of this paper is to evaluate multiple models to highlight which model has the best potential, we will incorporate linear regression (AutReg) into the mix to compare all the models. The original quest here is to see if a more complex model like LSTM (Deep Neural Network) are superior to classical statistical methods such as ARIMA, VAR, and even Facebook Prophet library.

We will use an AR model of lag (1) for the benchmark. We will train the AR model on the data up and until `2021-05-31` and then use the model to predict the remaining dates `2021-06-01 to 2021-07-8`. We will capture RMSE, R^2 and MAPE.

Here is the prediction plot of the AutReg model using statsmodel library



And here is the comparison between Actual vs Predicted



And the output scores

```
RMSE : 7.673165105240683
R2 : -0.7427030977199409
MAPE : 0.04398642842957195
```

3. Methodology

Data Preprocessing

We used AlphaVantage Python wrapper to pull data for the each individual stock. The API only allows 5 calls per minute so we build a function that sleeps (pauses) for 60 seconds before continuing with the calls for each symbol provided. The data returned from the function is in a dictionary. We also captured some metadata as well with each returned price data.

Several of steps include extracting Adjusted Closing price, merging the data into one DataFrame where the header is the different symbols.

For each model different data preprocessing needs took place.

- VAR model expects a stationary dataset so we used log differencing

```
data_stationary = np.log(data).diff().dropna()
```

- LSTM model expects the data to be normalized so we applied MinMaxScaler from the Scikit-Learn library

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
scaler.fit(x)
x_train = scaler.transform(x_train)
x_test = scaler.transform(x_test)
x_val = scaler.transform(x_val)
scaler.fit(y)
y_train = scaler.transform(y_train)
y_test = scaler.transform(y_test)
y_val = scaler.transform(y_val)
```

- Splitting the data by date slices. Since this is Time Series we cannot shuffle the data or randomly split. In different scenarios as we tested different hypothesis we made different data slices. Overall, in the majority of the cases the training data includes historical data up and until `2021-05-31`, and the testing data `2021-06-01 to 2021-06-08` which is 27 days. We are trying to predict 22-27 days generally. Though for Stock data this is too ambitious and most professionals would be satisfied with 2-3 days look ahead, we want to explore the potential of the model in this type of scenario.

Implementation

For all the models we created, we captured three main scores and stored them into a Python dictionary object. We then converted into a Pandas DataFrame for comparison.

For scoring and evaluation we used the Scikit-Learn library.

```
from sklearn.metrics import mean_squared_error,
mean_absolute_percentage_error, r2_score
```

Additionally, when constructing our LSTM network we specified similar metrics/scores so we can observe the changes over each epoch for plotting and evaluation purposes. Here is the LSTM metrics defined

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.MeanSquaredError(),
              metrics=[tf.keras.losses.MeanAbsolutePercentageError(),
                                       tf.keras.losses.MeanSquaredError()])
```

We also used an EarlyStopping technique for the LSTM

```
callback = tf.keras.callbacks.EarlyStopping(
    monitor="loss",
    min_delta=0,
    patience=8,
    verbose=0,
    mode="auto",
    baseline=None,
    restore_best_weights=False,
)

history = model.fit(
            x_train,
            y_train,
            epochs = 50,
            shuffle=False,
            validation_data=(x_val, y_val),
            batch_size=30,
            verbose=1,
            callbacks=[callback]
        )
```

And specifically for the LSTM we used a train, validation, and test data sets as defined here

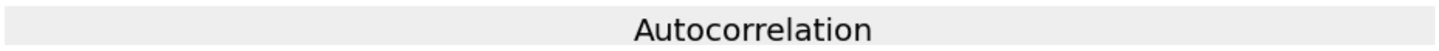
```
    split_1 = int(0.85 * len(x))
    train_x = x[:split_1]
    x_test = x[split_1:]
    train_y = y[:split_1]
    y_test = y[split_1:]

    split_2 = int(0.90 * len(train_x))
    x_train = train_x[:split_2]
    x_val = train_x[split_2:]
    y_train = train_y[:split_2]
    y_val = train_y[split_2:]
```

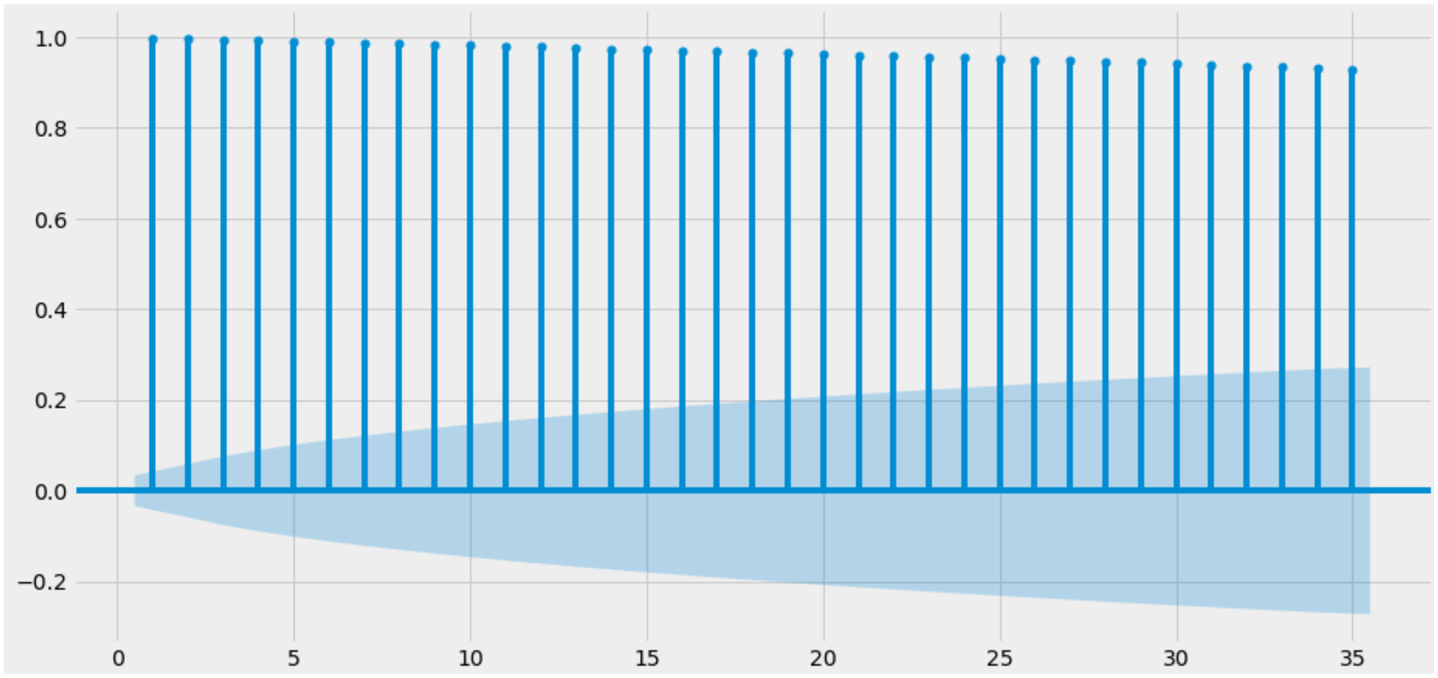
From the preceding code, we split the data 85% for training and 15% for test. Then we further split the training dataset into a train and validation at (90%, 10%) split.

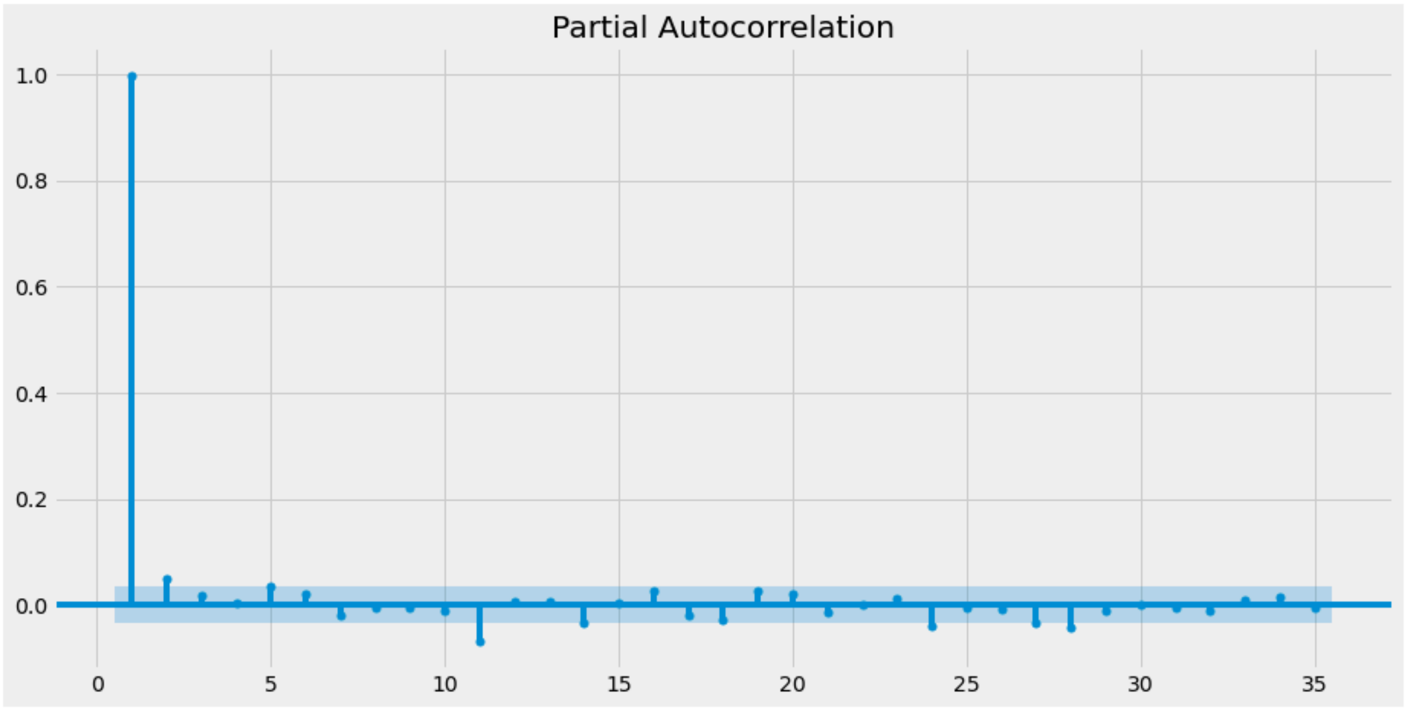
Here is a summary of the libraries and models used

Library	**Model**
Facebook Prophet	Prophet
Statsmodels	ARIMA
Statsmodels	AR
Statsmodels	VAR
Keras	LSTM

For the ARIMA models we had to determine the value for AR(p) and MA(q) using PACF and ACF plots. We ran the plots for all the stocks and we can see a trend and thus picked the best values we can determine as $p = 1, q = 1$




1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0
5
10
15
20
25
30
35


Partial Autocorrelation
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25
30
35

Refinement

Refinement

Initial analysis and experimentation with date slices was tricky. Assumption was that 2020 (pandemic period) was an anomaly and can skew the results and make it hard to predict. We tested with the simple ARIMA model, and also relying on results from the data exploration, different time slots:

- 2019-2020 then predict early 2021 (March)
- 2019 to mid 2021 to predict remaining months
- 2020 to May 2021
- All historical up to May 2021 to predict later dates

From the initial experiments, a decision made to standardize the approach and focus on 2021 prediction. In other words, we opted to train on historical data up and until end of May 2021 to predict June.

Results

Model Evaluation and Validation

We ran different scenarios for each model and captured the scores for each.

	RMSE	R2	MAPE

	RMSE	R2	MAPE
fb_1	8.624225	-3.745444	0.063526
fb_2	6.092310	-1.368103	0.046379
fb_3	2.533085	0.590611	0.018307
ARIMA_1	3.531296	-2.707079	0.023564
ARIMA_2	6.030342	-3.529083	0.045235
ARIMA_3	6.635800	-1.809463	0.041156
ARIMA_4	5.835365	-1.172565	0.035751
VAR_1	0.009686	-0.062573	2.260753
LSTM_1	40.996503	-1.533111	0.334533
LSTM_2	68.123026	-6.014925	0.517857
LR	7.673165	-0.742703	0.04398

As a reference:

`fb_1`	we ran for univariate time series data
`fb_2`	we ran a multivariate time series data with stock price and RSI
`fb_3`	we used **Additional Regressor** via `add_regressor()` method to add a linear part to the model.
`ARIMA_1`	ARIMA(1,1,1) using 2019-2020 then predict early 2021 (March)
`ARIMA_2`	ARIMA(1,1,1) using 2019 to mid 2021 to predict remaining months
`ARIMA_3`	ARIMA(1,1,1) using 2020 to May 2021
`ARIMA_4`	ARIMA(1,1,1) using all historical up to May 2021 to predict later dates

`VAR` Vector AutoRegressive model for multivariate time series using stock price and RSI. Log differencing applied.

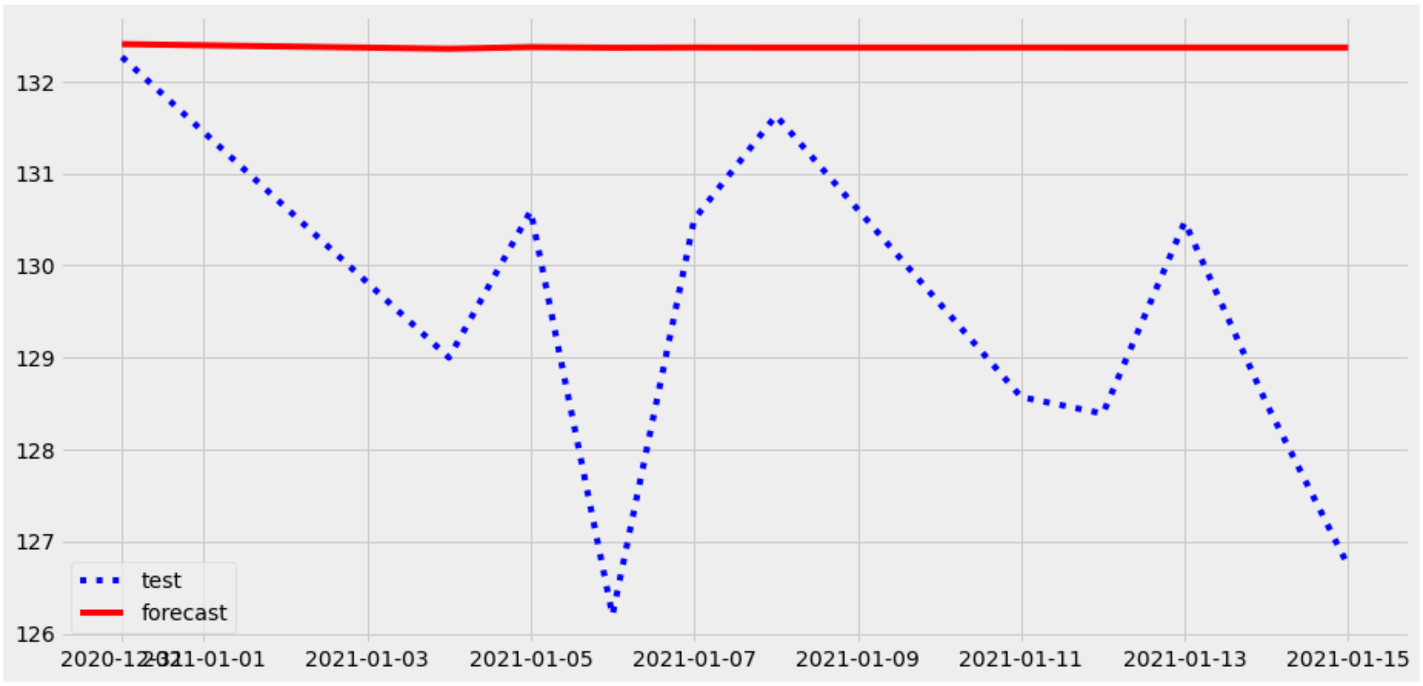
`LSTM_1` LSTM neural network using a window of lag 2 and 2 cells (neurons) per hidden layer

`LSTM_2` LSTM neural network using a window of lag 5 and 5 cells (neurons) per hidden layer

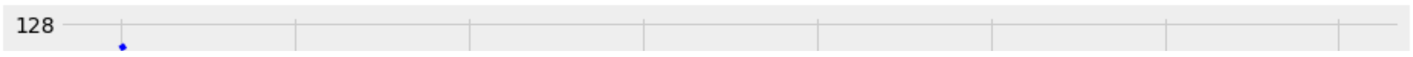
`LR` AutoRegressive (AR) model with Lag = 1

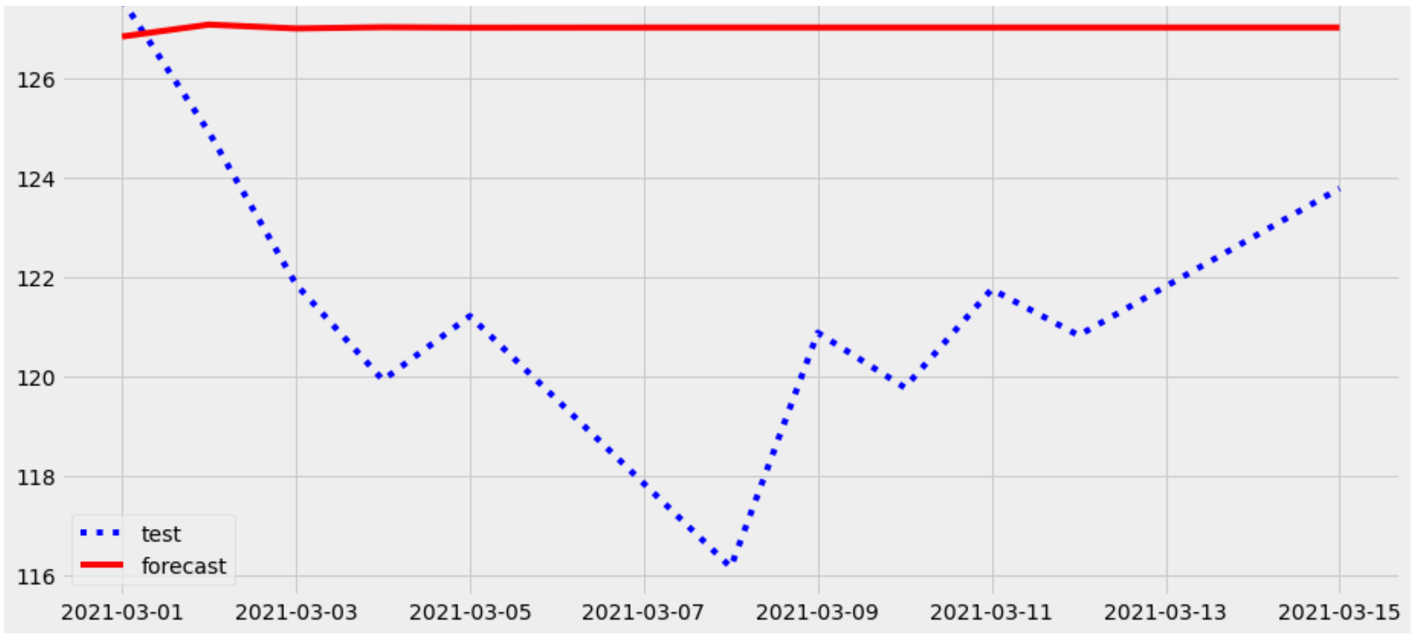
Forecast results examples

ARIMA_1

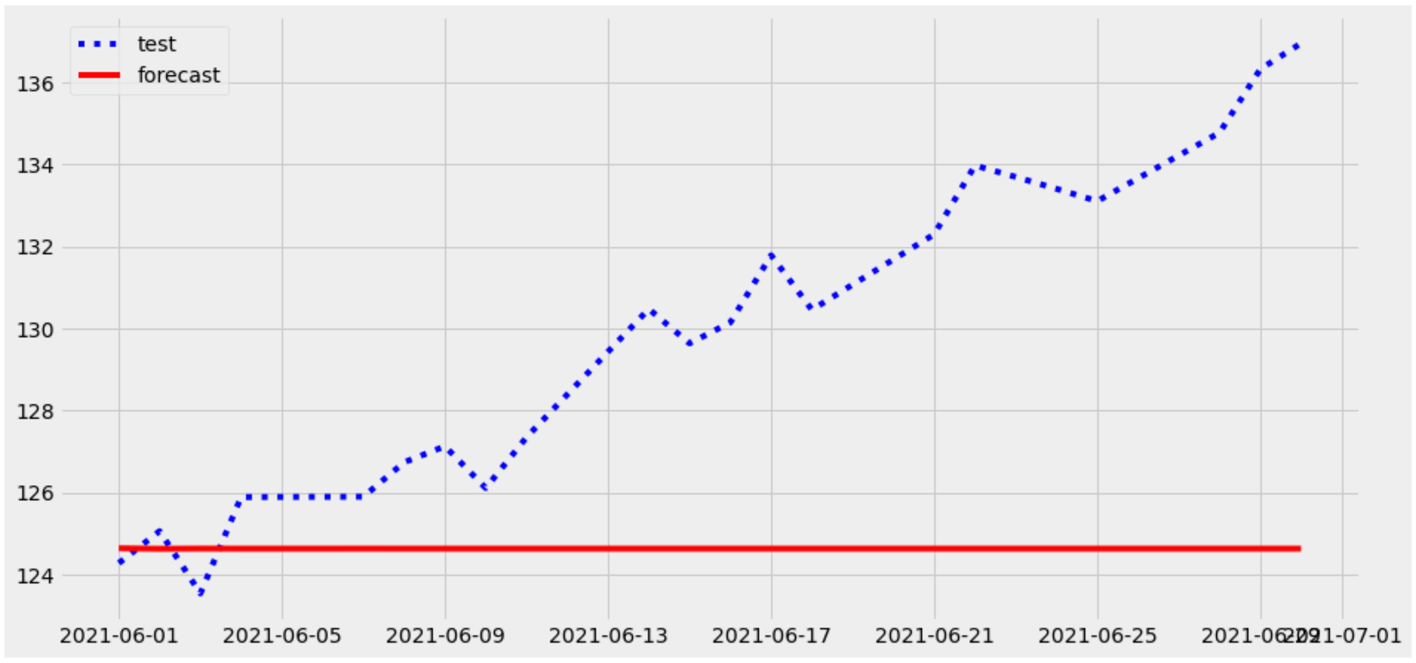


ARIMA_2

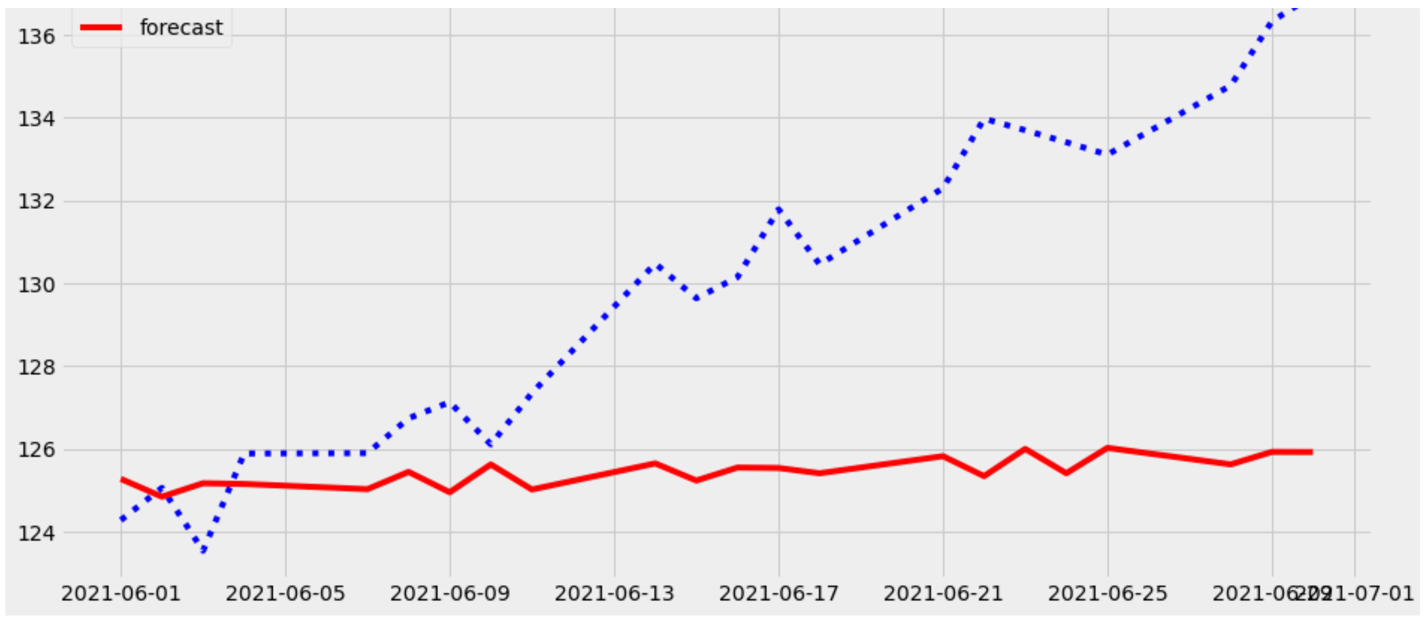




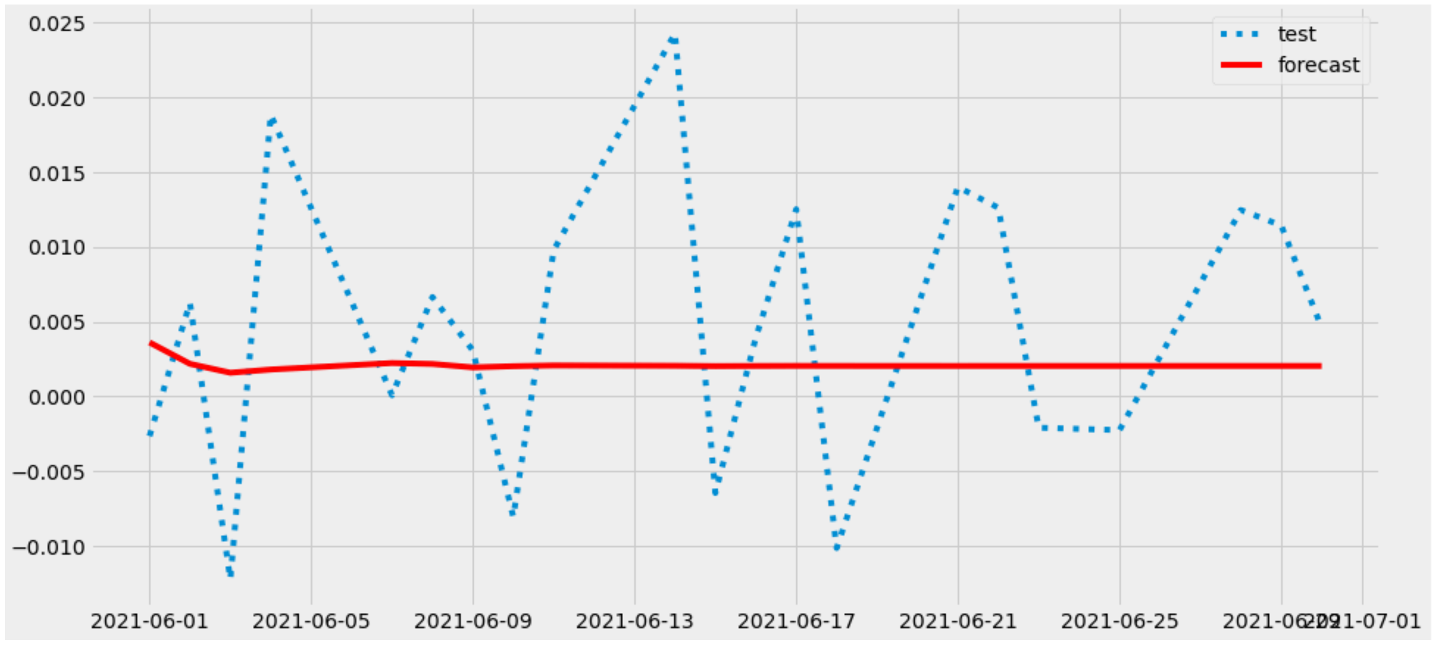
ARIMA_3



ARIMA_4

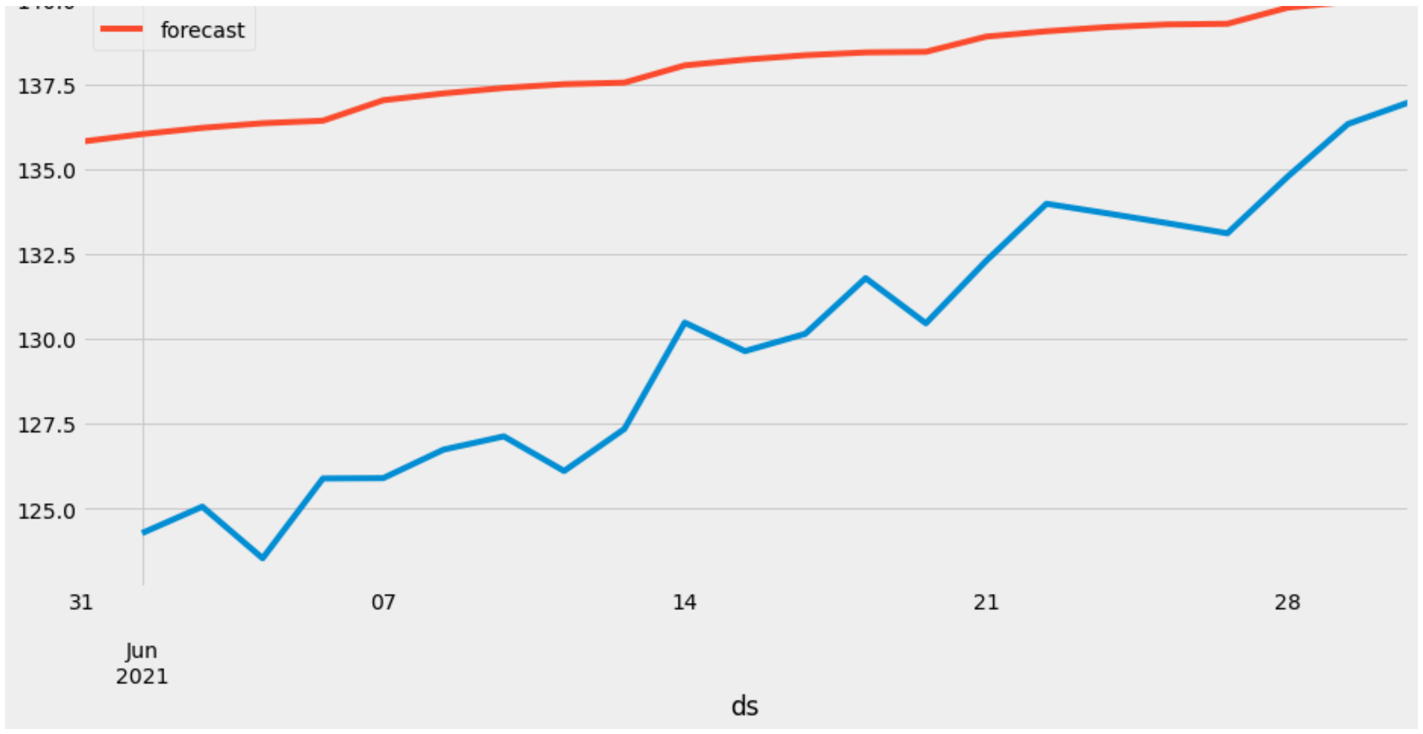

forecast
136
134
132
130
128
126
124
2021-06-01
2021-06-05
2021-06-09
2021-06-13
2021-06-17
2021-06-21
2021-06-25
2021-06-29
2021-07-01

VAR

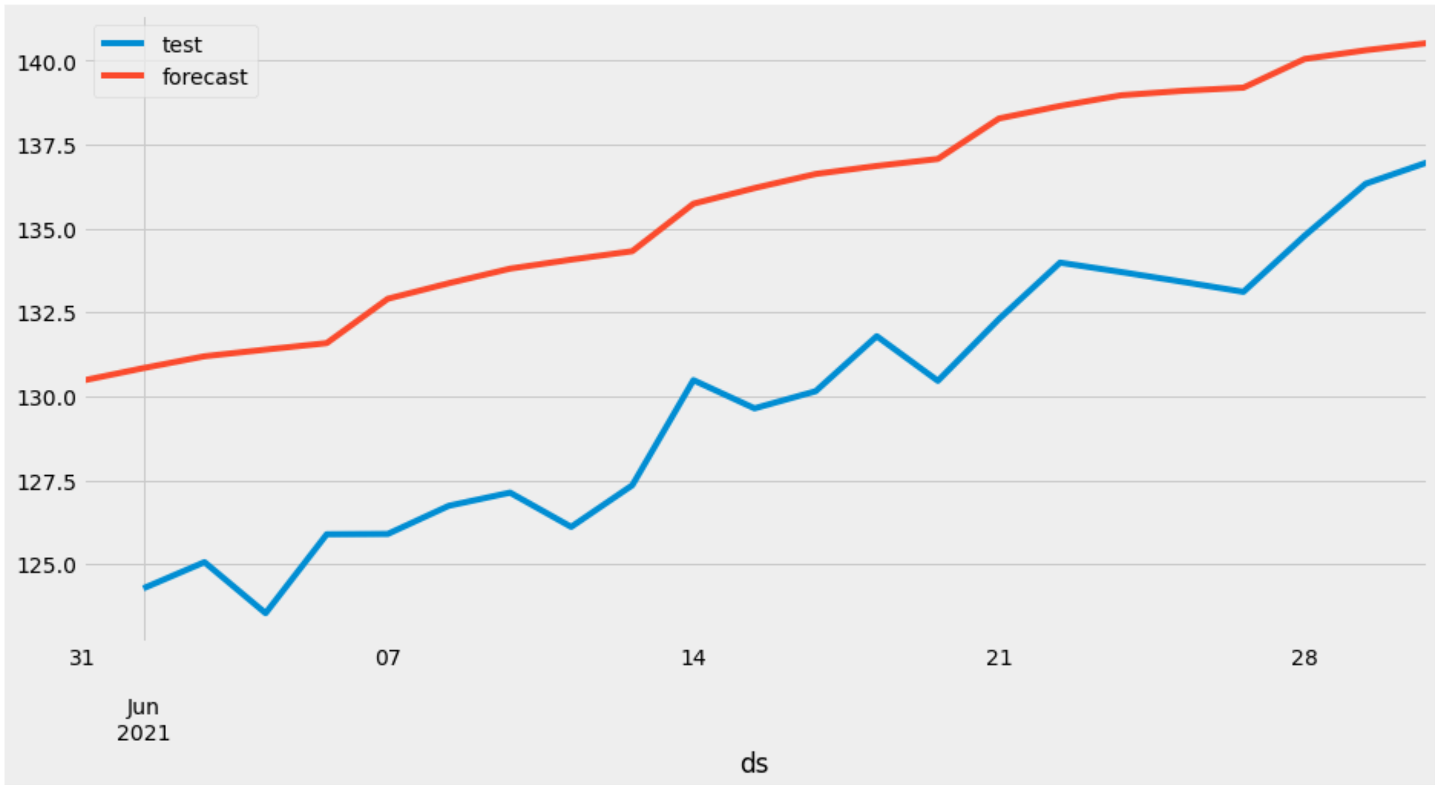

test
forecast
0.025
0.020
0.015
0.010
0.005
0.000
−0.005
−0.010
2021-06-01
2021-06-05
2021-06-09
2021-06-13
2021-06-17
2021-06-21
2021-06-25
2021-06-29
2021-07-01

fb_1

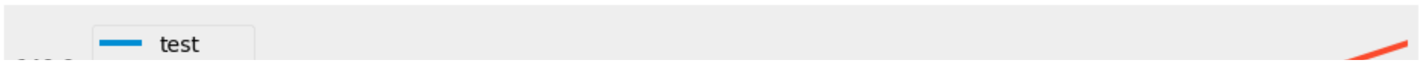
test
140.0

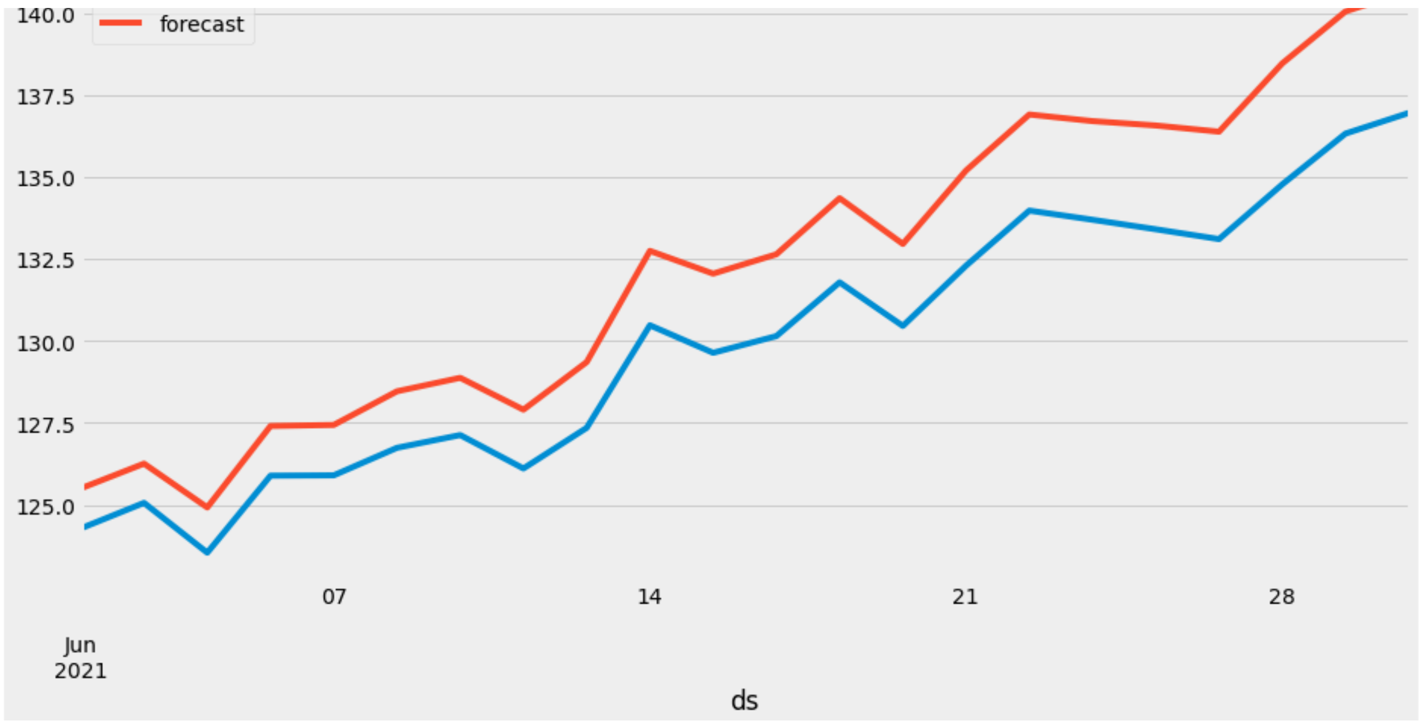

forecast
137.5
135.0
132.5
130.0
127.5
125.0
31
07
14
21
28
Jun
2021
ds

fb_2


test
forecast
140.0
137.5
135.0
132.5
130.0
127.5
125.0
31
07
14
21
28
Jun
2021
ds

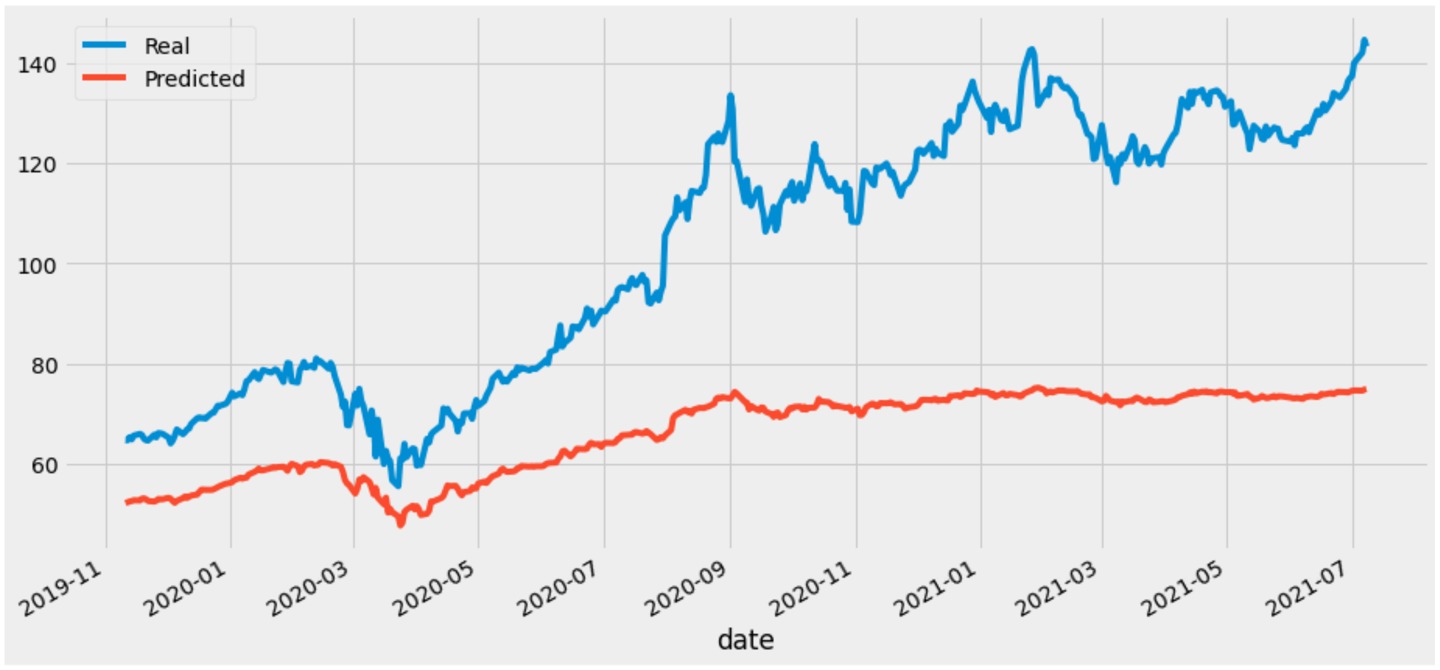
fb_3


test



forecast
140.0
137.5
135.0
132.5
130.0
127.5
125.0
07
14
21
28
Jun
2021
ds

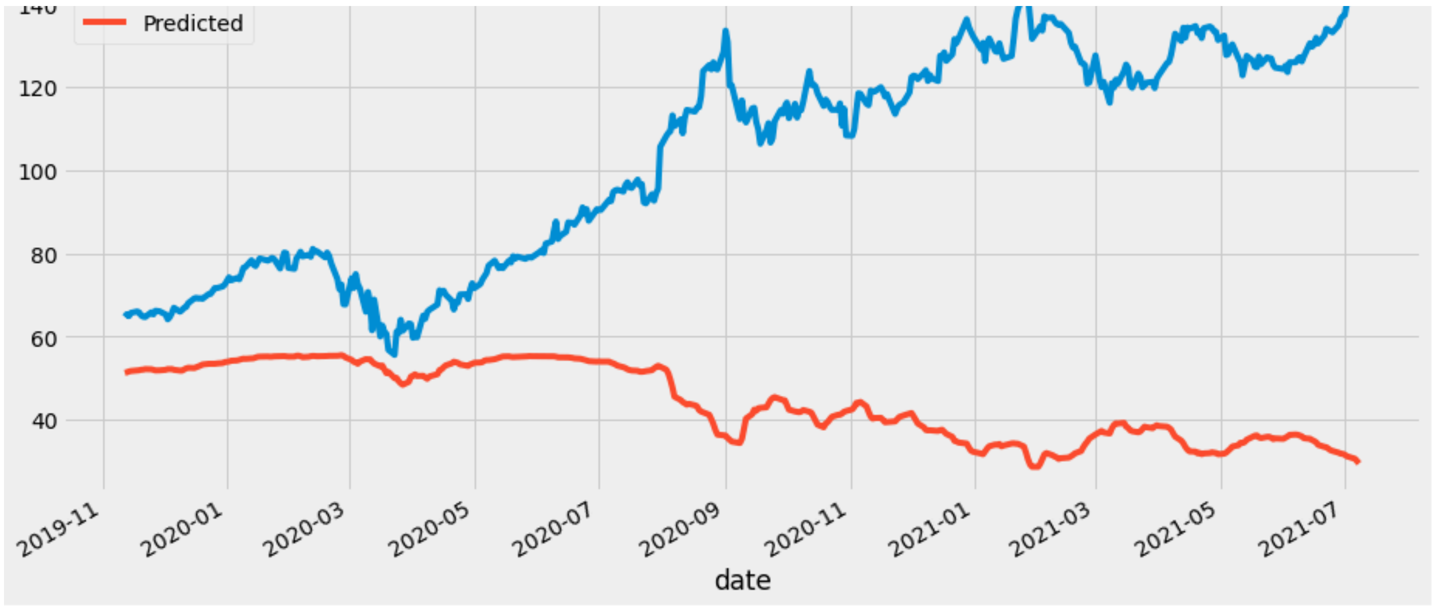
LSTM_1



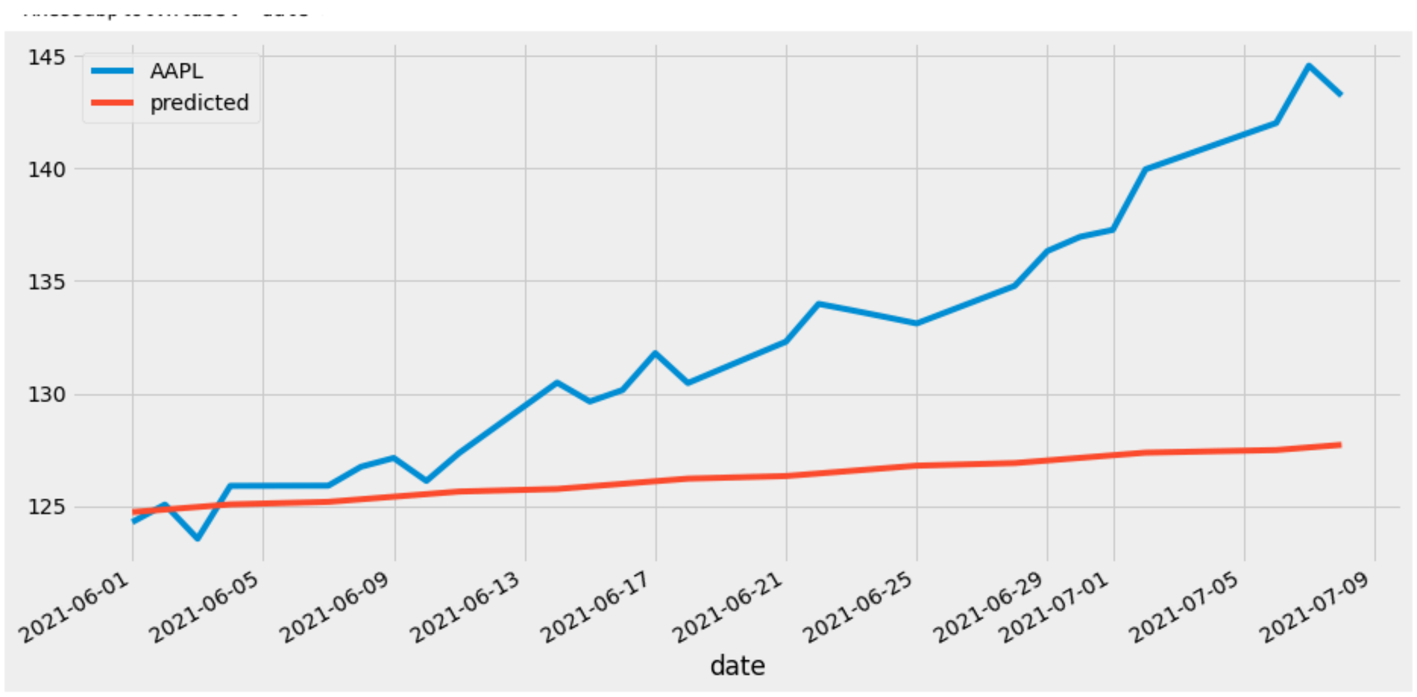
Real
Predicted
140
120
100
80
60
2019-11
2020-01
2020-03
2020-05
2020-07
2020-09
2020-11
2021-01
2021-03
2021-05
2021-07
date

LSTM_2

Real



LR



When comparing the results of MAPE we get the following:

```
fb_3        0.018307
```

```
ARIMA_1    0.023564
ARIMA_4    0.035751
ARIMA_3    0.041156
LR         0.043986
ARIMA_2    0.045235
fb_2       0.046379
fb_1       0.063526
LSTM_1     0.334533
LSTM_2     0.517857
VAR_1      2.260753
```

And for RMSE

```
VAR_1       0.009686
fb_3        2.533085
ARIMA_1     3.531296
ARIMA_4     5.835365
ARIMA_2     6.030342
fb_2        6.092310
ARIMA_3     6.635800
LR          7.673165
fb_1        8.624225
LSTM_1     40.996503
LSTM_2     68.123026
```

MAPE is a better option in our case when we are comparing since it gives us an estimation on how accurate the forecast model is (since it measure the accuracy as a percentage) and provides a good measure on the forecast error.

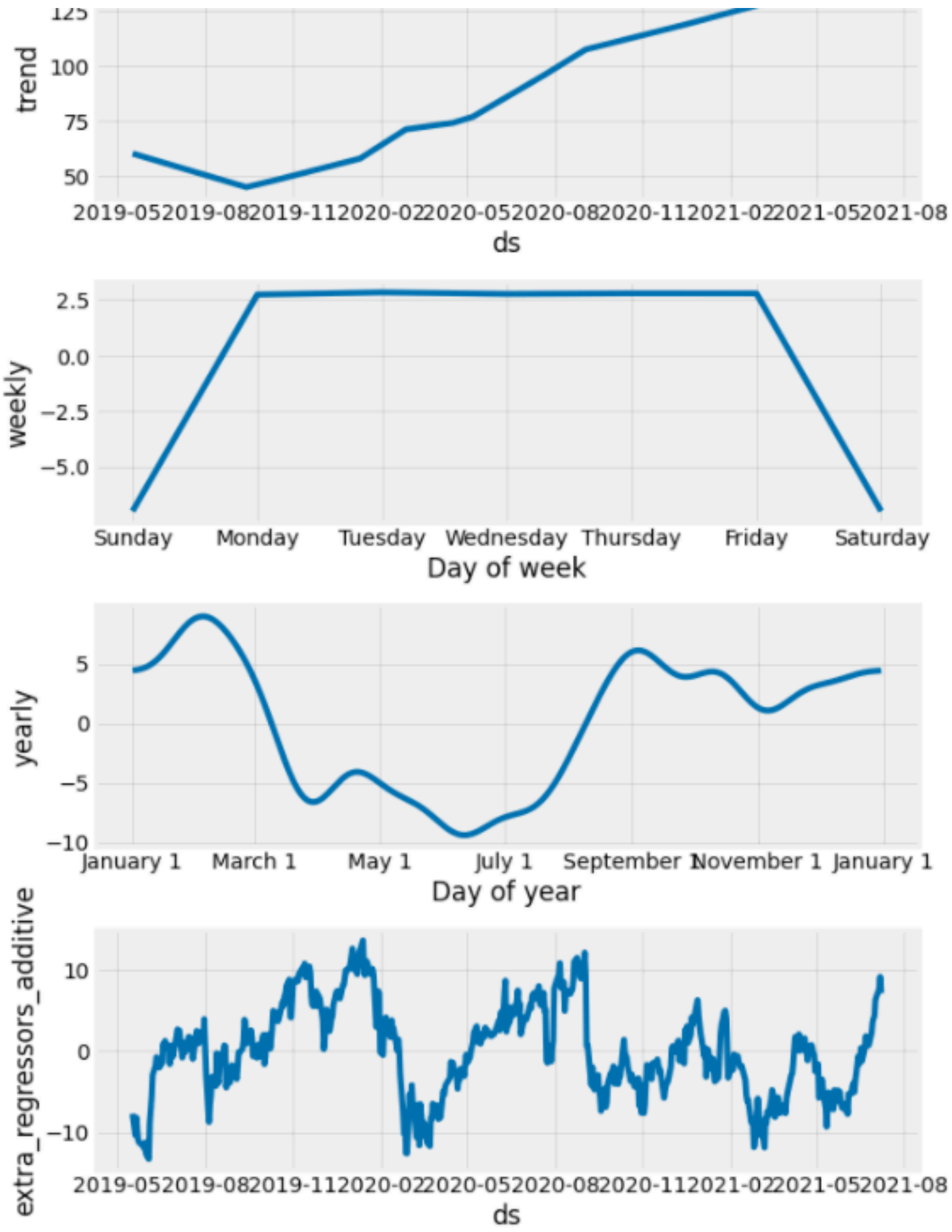
The Facebook Prophet library with Additional Regressor outperformed the rest of the models. Surprisingly the LSTM did not perform as expected. If we compare the statistical models (simple models) we also realize that ARIMA overall did do well given the four different scenarios. Machine Learning and Deep Learning generally do great with additional tuning, but if a simpler classical model does the job it would always be preferred.

Justification

Though we assumed LR (AR with lag 1) is benchmark model, we were comparing all the models in the end. There was an expectation initially that an LSTM will dominate in terms of predictive power, though there is always the fear of overfitting.

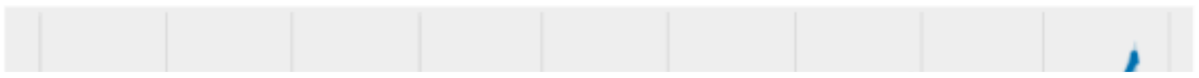
Facebook Prophet library had different options and we just trained it using default parameters. But overall the three iterations of the Prophet model outperformed LSTM.

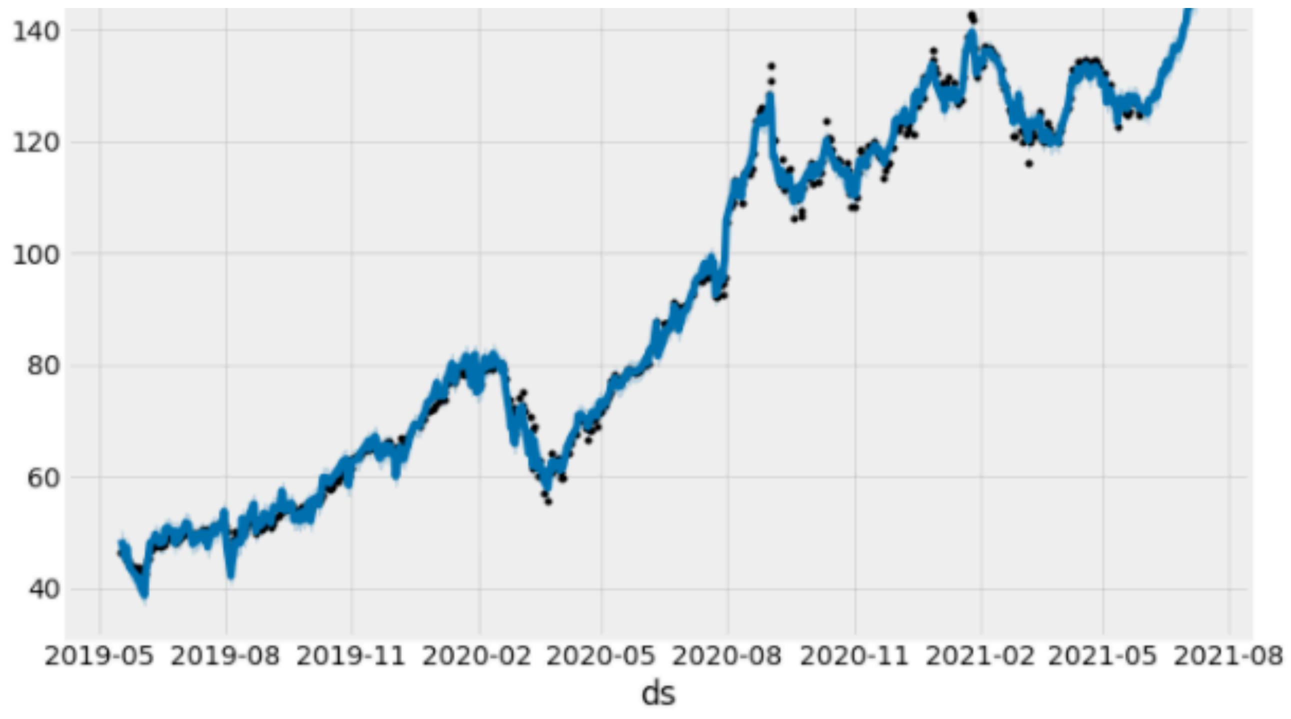
Prophet was also provides plenty of plotting capabilities to further analyze the model and its components. For example, Prophet will provide a components plot which graphically describes the model it has fit (example for the winning model `fb_3`)



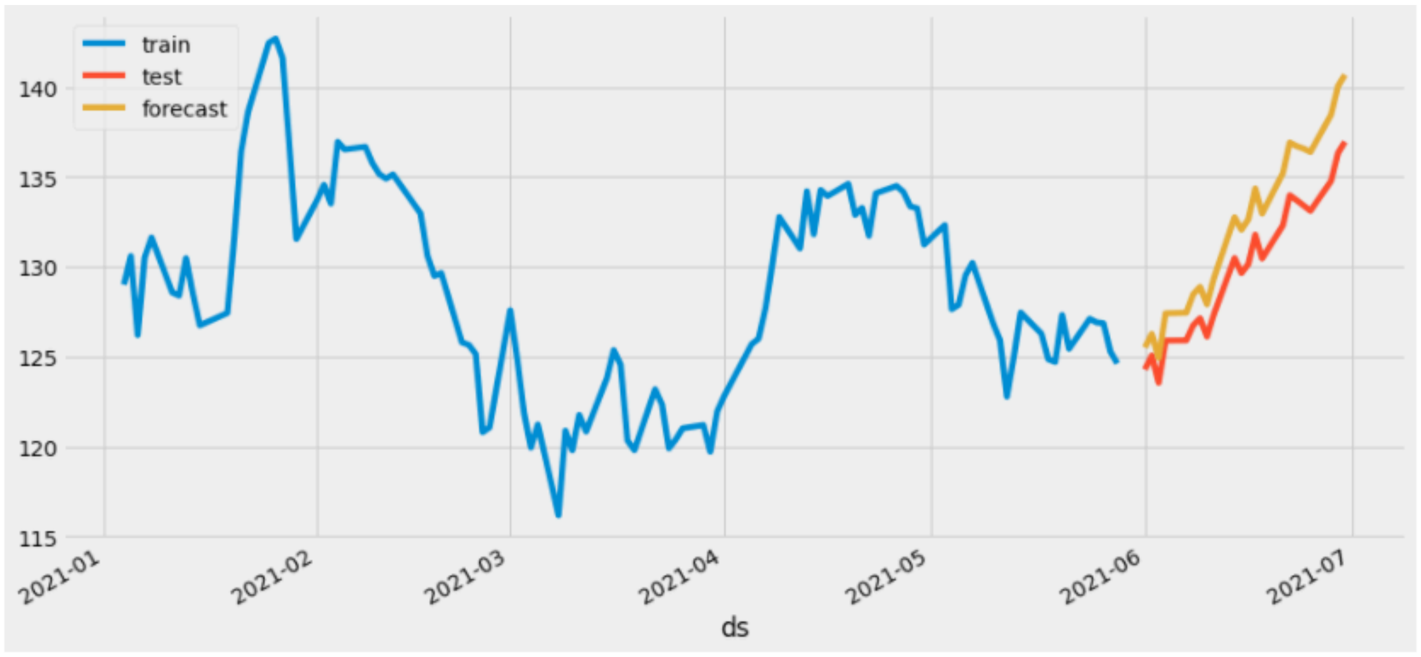
And forecasting within Prophet can be also be plotted using `plot(forecast)` method of the model

```
model.plot(forecast)
```





With a closer snapshot for 2021 we can see how good was Facebook Prophet at capturing the trend and predict the magnitude and direction of the stock price for AAPL



Next Steps

Next step would be to fine tune the Prophet model (optimize it). Possibly testing it on an index or other market indicators such as S&P500 in order to build a model we can use for other stocks.

LSTM has potential, and can be further enhanced and improved. With enough time, and resources there are many opportunities to   improve both Facebook Prophet and LSTM and compare them head to head.

Once a final polished and production ready model is established and ideal way to showcase the model would be using Streamlit for user interaction.

References

---

1. Bradley, C., The impact of COVID-19 on capital markets, one year in, McKinsey & Company, https://www.mckinsey.com/business-functions/strategy-and-corporate-finance/our-insights/the-impact-of-covid-19-on-capital-markets-one-year-in ↩

2. Nova A., Many are chasing the stock market by day trading in the pandemic. It could end badly, CNBC, https://www.cnbc.com/2020/09/21/many-people-turn-to-day-trading-in-pandemic-few-will-be-a-winners.html ↩

3. Monte Carlo method, Wikipedia, https://en.wikipedia.org/wiki/Monte_Carlo_method ↩

4. Autoregressive integrated moving average, Wikipedia, https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average ↩

5. Vector autoregressive, Wikepdia, https://en.wikipedia.org/wiki/Vector_autoregression ↩

6. Long short-term memory, Wikipedia, https://en.wikipedia.org/wiki/Long_short-term_memory ↩

7. Linear Regression Models, Duke University, https://people.duke.edu/~rnau/compare.htm ↩

8. Taylor, S. & Letham B., Prophet: forecasting at scale, Facebook Research, https://research.fb.com/blog/2017/02/prophet-forecasting-at-scale/ ↩