TWE are LSTM recurrent neural networks an improvement over ARIMA models at predicting stock market price fluctuations?

I. Introduction
    A. Introduction
II. Background
    A. Stock Market fluctuations
    B. ARIMA model and Neural Networks
III. Data Gathering
    A. Choosing and Extracting Stock Data
    B. Labeling and Preparing Stock Data
IV. LSTM model
    A. Model Outline
    B. Experiment
        1. Well-Performing Stocks
        2. Stable-Performing Stocks
        3. Poor-Performing Stocks
    C. Conclusion
V. ARIMA model
    A. Model Outline
    B. Experiment
    C. Conclusion
VI. Evaluation
    A. Comparing and Contrasting Accuracies
    B. Weighing Advantages of LSTM over ARIMA
VII. Conclusion
    A. Possible Improvements
    B. Other Problems

I. Introduction

Introduction

The New York Stock Exchange is the center for investors seeking to buy and sell stock shares. Generally, a successful investor looks to buy stocks at a low price and sell them at a higher price. However, various different factors can affect stock prices, making trends difficult to predict. Regardless of the stock market, the overarching need for time-series prediction has led to the use of models such as the ARIMA model, primarily used in statistics and economics. The rapid growth and development of computer science, especially in the field of artificial intelligence, has introduced neural networks to the world of information processing. Long Short-Term Memory networks, in particular, have shown promise in making accurate predictions based on time-series data. Whether this type of neural network poses a significant improvement over AutoRegressive Integrated Moving Average model is not concretely established.

In order to determine the best prediction model, this essay will reveal the findings from feeding stock data into either model, taking into account various factors in order to reach a conclusion. I will download historical stock data as .csv files, probably from Yahoo Finance, and feed these data into local ARIMA and LSTM models written in the Python programming language. I will discuss the varying difficulties, speeds, results, accuracies, scalabilities, and user-friendliness in implementing these models in order to determine the superior model. I will also briefly discuss the models' varying successes in other tasks. I will establish and develop an argument for the supremacy of one model over the other.

This research question is worthy of investigation as it would help determine the most worthy and effective method of stock market price prediction while providing a further understanding of each method's advantages. Observing these different prediction models would provide context for their implementation in the real world. This essay will attempt to reveal the best manner in which to approach the stock market with technology. It is important to determine the most promising model worthy of investment, to which improvements and innovations can and will be developed in the future.

II. Background

Stock Market Fluctuations

The stock market fluctuates by the second. Prices generally vary based on the simple principles of supply and demand. However, there are many other factors, some that aren't quantifiable, that play a role in determining stock prices. Remarkably, Princeton University professor Burton Malkiel claims that "a blindfolded monkey throwing darts at a newspaper's financial pages could select a portfolio that would do just as well as one carefully selected by experts." Malkiel essentially and accurately establishes that the stock market is really down to chance most of the time. It is

almost impossible to completely and effectively predict prices; there are simply too many variables at play. Regardless, tireless efforts have gone towards developing a functioning model that can accurately predict stock market prices.

Historical stock price records are known as time-series data. A statistical model used to interpret and develop predictions for a given time-series data is the ARIMA model. With the emergence of big data and metadata, various developments in computer science have led to the development of the LSTM model, also capable of predicting future points given a time-series dataset.

Autoregressive Integrated Moving Average Model and Neural Networks

The Autoregressive Integrated Moving Average Model (ARIMA) is a variation of the Autoregressive Moving Average Model. Autoregressive is used to describe a model that uses data from previous timesteps as inputs to a regression function and integration deals with finding the difference between each data value from another to achieve stationarity.

Neural networks are computing systems meant to vaguely resemble the human brain, hence the name. There are different types of neural networks, but recurrent neural networks are best suited to forecast time-series data due to them being tailored to chronological data. Long Short-Term Memory recurrent neural networks are a particular architecture of an RNN that allows cell states to last longer instead of vanishing like in traditional RNNs, essentially incorporating long term memory.

III. Gathering Data

Extracting Stock Data

Historical stock data from the past 10 years was readily available from the NASDAQ website, downloadable as .csv files. Two well-performing (Amazon and Tesla), two stable (Kellogg and Dannon), and two poorly performing (Transocean and Fluor) stocks were chosen and downloaded.

Labeling and Preparing Data

The close value of each stock for each day was chosen. Another viable option was to find the average between the high and low value for each day, but a significant advantage of using the close value was that this data is more accurate for predictions, and the average between high and low could be insignificant to predict if the range between the two was large.

IV. LSTM Model

Model Outline

An LSTM network is a type of recurrent neural network (RNN) where traditional nodes are replaced with LSTM cells, an innovation courtesy of Sepp Hochreiter, a german computer scientist. Neural Networks essentially take in an input matrix (i.e. [0,1,2]), and produce an output matrix (i.e. [3,4,5]). In a traditional RNN, the model ingests a matrix where one data point relies heavily on the previous data point, something true in time-series data like historical stock prices. The output when processing one data point is used for the next input through a hidden state, hence the name 'recurrent'.

First, a matrix is initialized with random values and multiplied by the input matrix. Based on how accurate the product is (the loss value)—which is calculated most commonly by the mean squared error function, an optimization function is applied, which basically uses calculus to figure out how the nodes affect the output, and then the random matrix can be modified to decrease the loss.

However, a shortcoming presents itself: basically, later nodes prove to be more influential than the earlier nodes, something known as the vanishing gradient problem; the RNN fails to learn long term dependencies. This is where LSTM networks come in. LSTM cells provide a cell state, which passes through all the LSTM cells, carrying data about the sequence as a whole. At each cell, there are input and output gates, determining which data gets added to the cell state and which data should be removed, thereby allowing for elongated memory based on short terms.

As aforementioned, the data must first be separated into training and testing data. 65% of the stock data is stored into the training data array and the rest into the test data array.

```
39          training_size = int(len(df1) * 0.65)
40          test_size = len(df1) - training_size
41          train_data, test_data = df1[0:training_size, :], df1[training_size:len(df1), :1]
```

Within the training data, smaller sets of data must be created. This is done through the create_dataset function, where our train_data array (that has 65% of the dataset) is passed through as an argument. The timestep defines the size of the smaller dataset. Basically, all of the close values within a range are added to dataX and the close value right after is added to dataY, which is then returned to us and stored in X_train and y_train.

So for example, if you had the numbers 1-50 and the timestep was 4, numbers 1-4 would be added to dataX (stored in X_train), and number 5 would be added to dataY (stored in y_train). Then 2-5 would be added to dataX, and 6 to dataY, and then 3-6 to dataX and 7 to dataY, and so on until we reach 50. The time-series data needs to be separated into smaller series of data in order to fit the model. There are also arrays for testing that are used for validation when training the model, as seen on line 46.

```
137  def create_dataset(dataset, timestep = 1):
138      dataX, dataY = [], []
139      for i in range(len(dataset)-timestep-1):
140          a = dataset[i:(i+timestep),0]
141          dataX.append(a)
142          dataY.append(dataset[i + timestep,0])
143      return np.array(dataX), np.array(dataY)
```

create_dataset function that returns two arrays

```
45          X_train, y_train = create_dataset(train_data, time_step)
46          X_test, ytest = create_dataset(test_data, time_step)
47
48          X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
49          X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)
```

dataX and dataY stored in X_train and y_train and reshaped

Next the stacked LSTM model is outlined. No dropout was added, as weekends and public holidays are already omitted from historical stock price data.

```
52          model.add(LSTM(50, return_sequences=True, input_shape=(100,1)))
53          model.add(LSTM(50, return_sequences=True))
54          model.add(LSTM(50))
55          model.add(Dense(1))
```

After the stacked LSTM model is defined, the loss function and optimizer function are defined in the compile method. The arrays with the training values are passed through as x and y parameters, and testing values are passed through as validation data.

```
58          model.compile(loss='mean_squared_error', optimizer='adam')
59          model.fit(X_train, y_train, validation_data=(X_test,ytest),epochs=self.num_epochs, batch_size=64, verbose=1)
```

Experiment

For each stock, the LSTM model was trained with 10 years' worth of data (11 Oct 2010 - 04 Sept 2020). The LSTM model was then made to forecast the next 30 days of stock price movement (8 September 2020 - 19 October 2020). The orange lines represent the model's forecasted prices and the blue lines represent the actual stock prices.

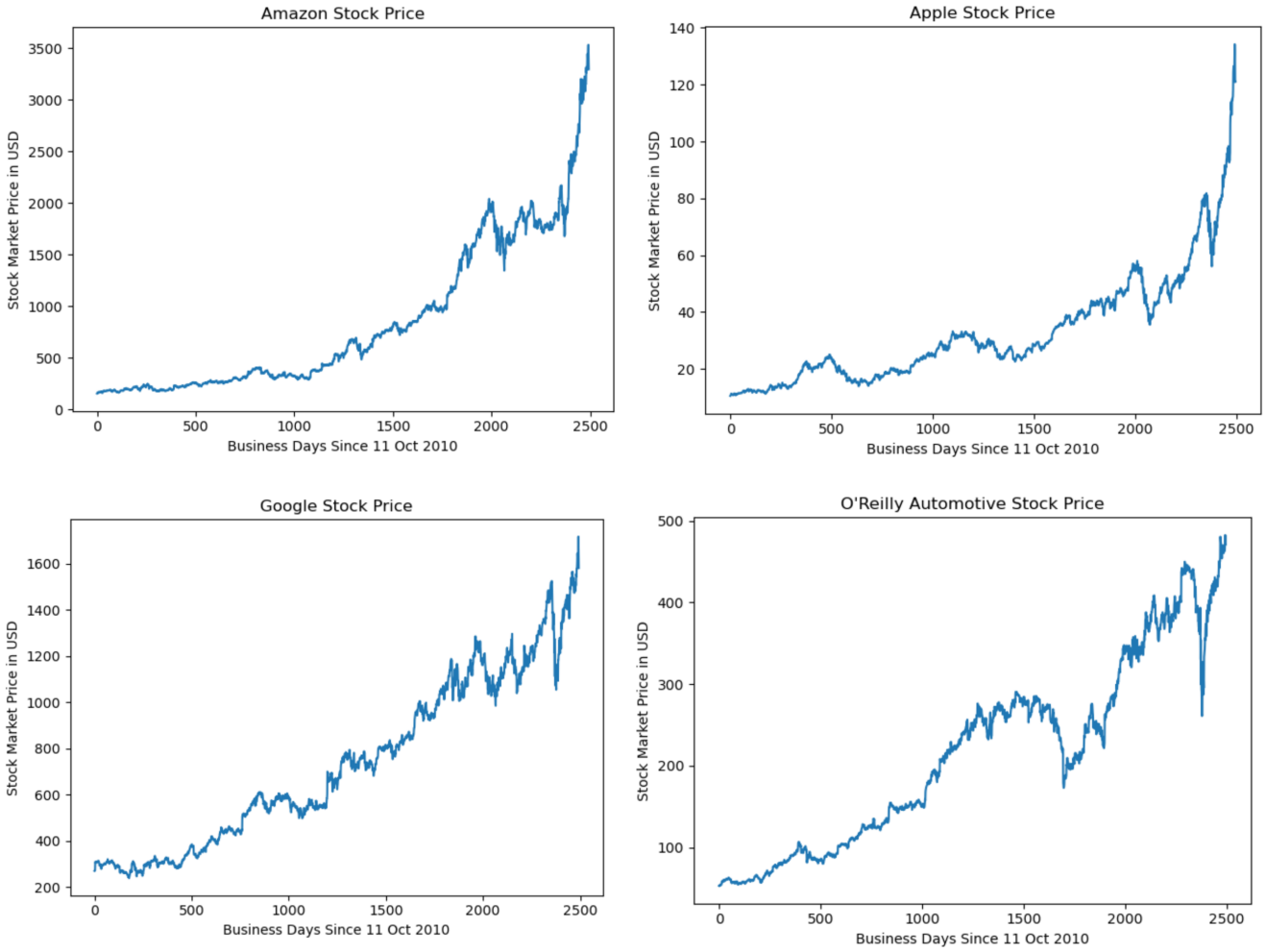
The accuracy was calculated using the formula below. The absolute value of the difference between the actual value A and the predicted value P is divided by the actual value to find the error of the prediction relative to the actual value, subtracted from 1 to find the complementary accuracy, and multiplied by 100 to yield an accuracy percentage.

$$\text{Accuracy} = \left(1 - \frac{|At - Pt|}{At}\right)(100)$$

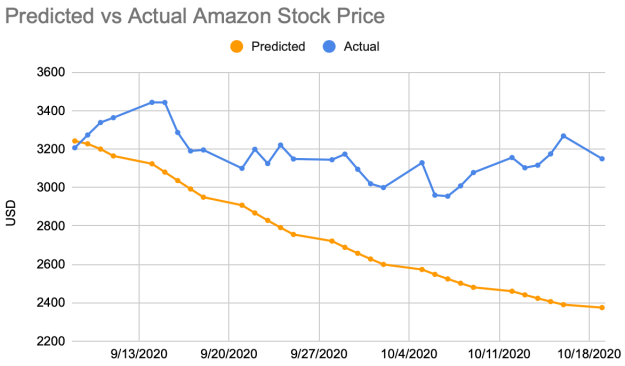
Training the LSTM model for 100 epochs took an estimated 25 minutes per stock, along with about 5 minutes for price predictions and graphing.

Well-Performing: Amazon, Apple, Google, O'Reilly Automotive

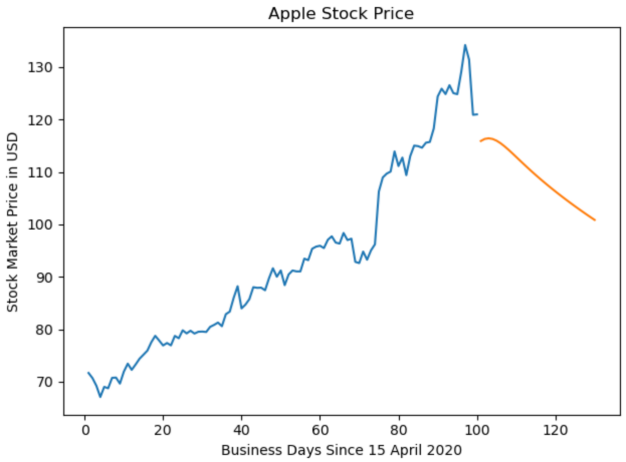
Over the past 10 years, Amazon, Apple, Google, and O'Reilly Automotive stocks have been performing relatively well.

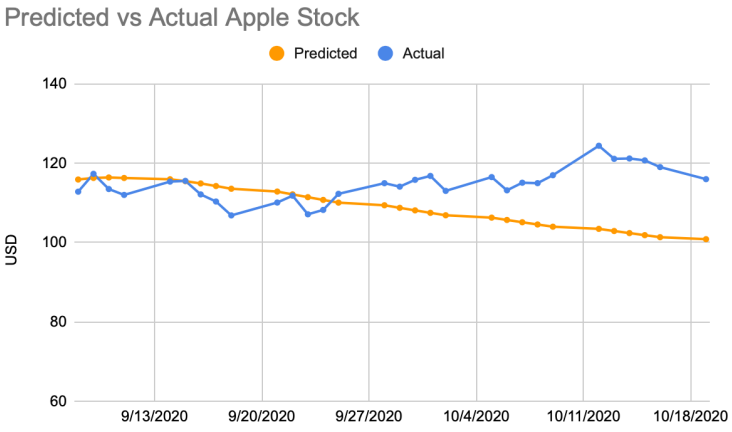


The LSTM model accurately predicted a general decrease in Amazon stock value from September 8, 2020 to October 19, 2020. However, there was an actual decrease of 57.37 USD, whereas the model predicted a decrease of 867.56 USD, a remarkable difference.

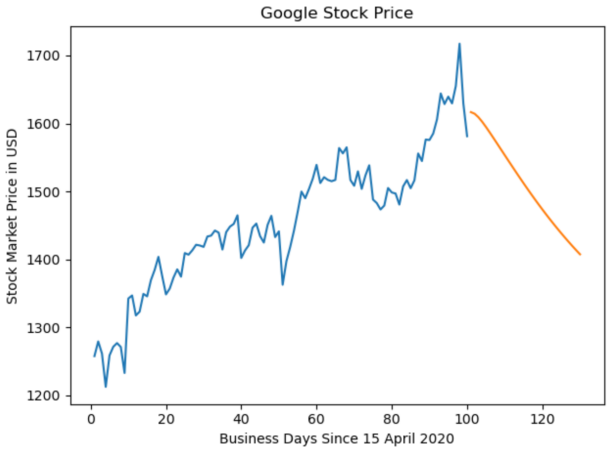


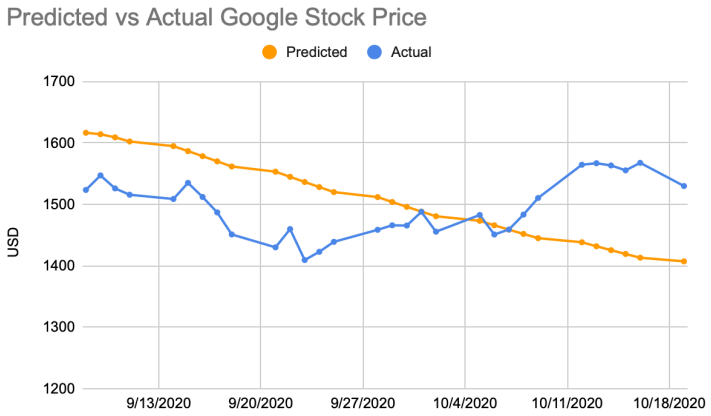
While the average accuracy for the 30-day Amazon stock price forecast was 86.71%, it is important to note that for the first 10 business days (8 September 2020 - 21 September 2020), the average accuracy was 93.99%, for the second set of 10 business days the average accuracy was 86.74%, and for the last set of 10 business days the average accuracy was 79.39%. This trend highlights the LSTM model's greater performance at forecasting less further into the future.



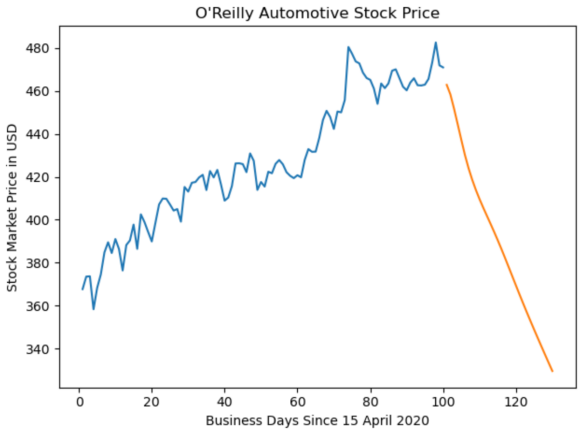


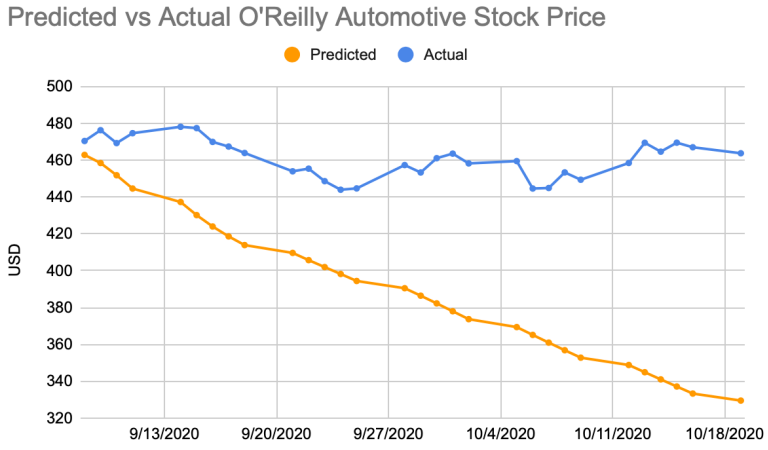
Apple stock price did experience a drop in price initially, but recovered and rose to 115.98 USD in the latter half of the 30 forecasted days, something the LSTM model failed to predict. Once again, the average prediction accuracy for the first 10 days was significantly higher than the rest (95.30% for the 2nd 10 days and 87.37% for the third) at 97.46%





For Google, the LSTM model forecasted a decrease to ~1400 USD, a projected 12.95% decrease of its initial value. The stock price actually did reach this low on September 23, but recovered steadily to yield a 0.42% increase in value after the 30 days. As for accuracy, the model enjoyed its sharpest predictions in the 2nd set of 10 days at 96.14%. This, however, was heavily undermined by its failure to predict a rise in value towards the latter third of the forecasted month. Regardless, the average percentage accuracy for the 30 days was a relatively impressive 94.83%



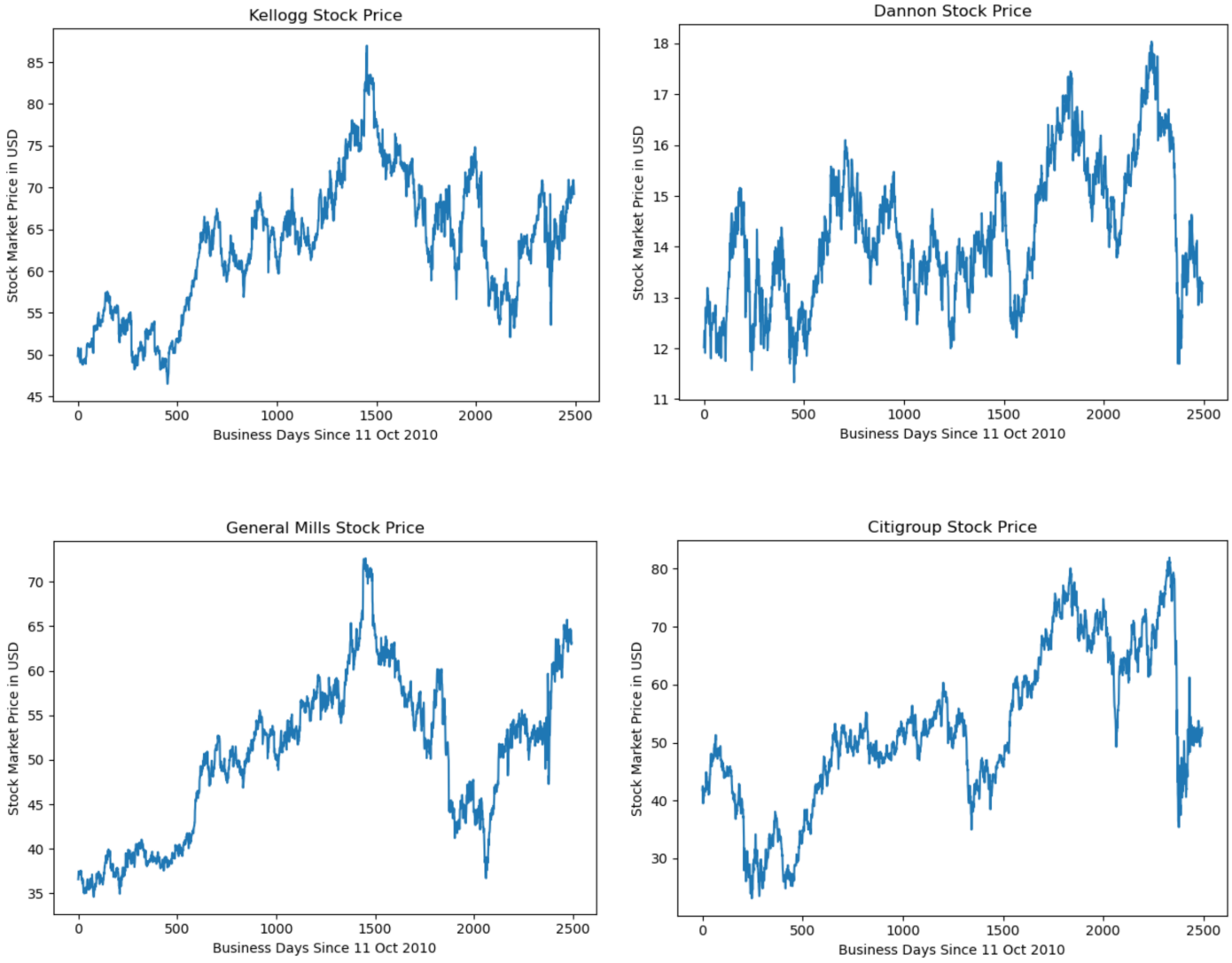


The LSTM model forecasted a very sharp decrease in O'Reilly stock to 329.52 USD, a price range not seen since April 2020. While the price did initially decrease towards the beginning, reflected by the sound average percentage accuracy of the first 10 days of 92.54%, the price eventually stabilized and actually rose a bit, resulting in the model's poor average percentage of 75.77% for the last 10 days. Once again, the model became less accurate as time wore on in its forecasts for well-performing stocks.

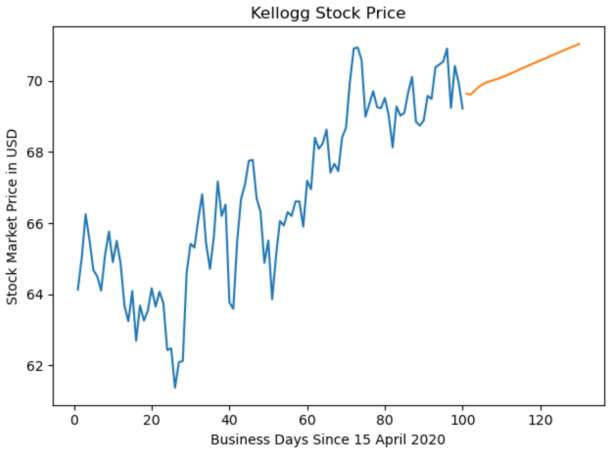
Well-performing stocks generally experienced impressive prediction accuracies by the LSTM model—at least for the first 10 days in each forecast. Afterwards, predictions tended to falter in accuracy, either due to a failure to foresee defining trends in the prices or due to excessively exotic predictions.

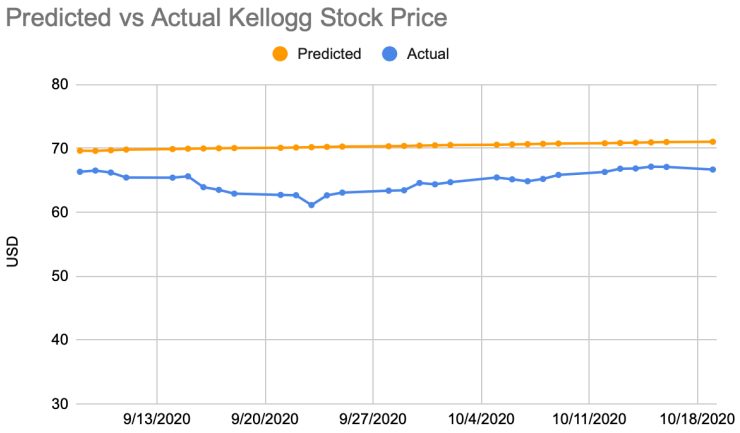
Stable-Performing: Kellogg, Dannon, General Mills, Citigroup

Kellogg, Dannon, General Mills, and Citigroup have enjoyed stable stock performance over the past 10 years

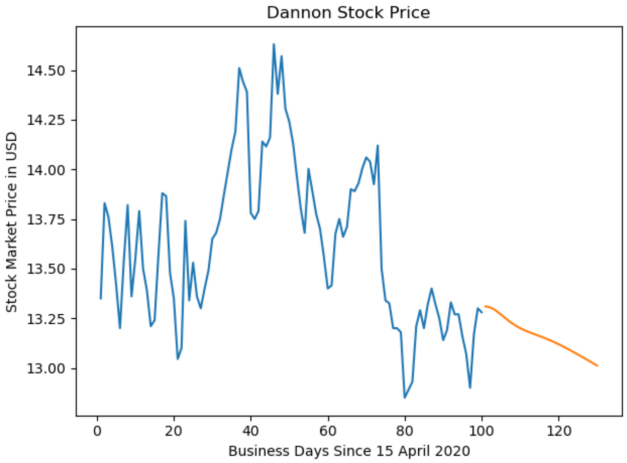


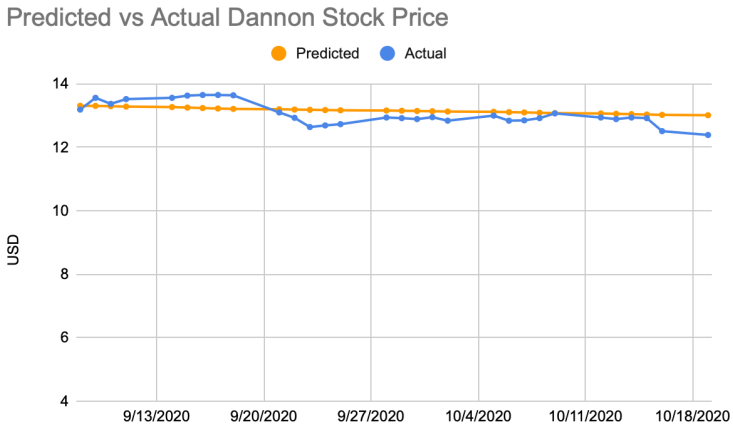
On average, the model demonstrated an accuracy of 91.49%. The first 10 forecasted days were predicted with an average accuracy of 92.22%, the second 10 with an average accuracy of 89.25%, and the last 10 days with an average accuracy of 92.99%. Overall, the pattern did not resemble the pattern in accuracy as seen in well-performing stocks. The LSTM model demonstrated a higher and more consistent prediction accuracy for the stable-performing stock that is Kellogg.



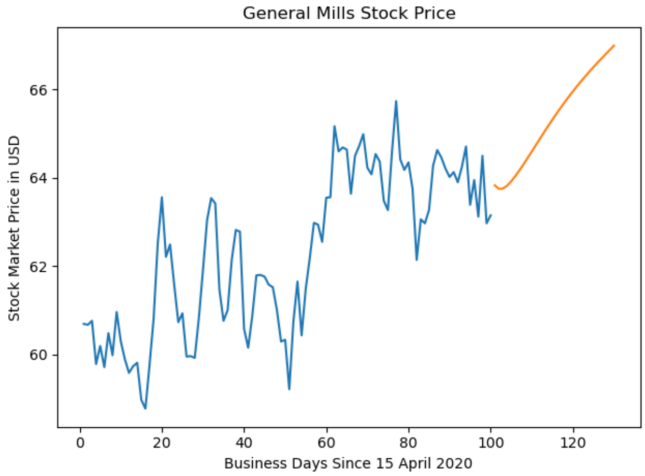


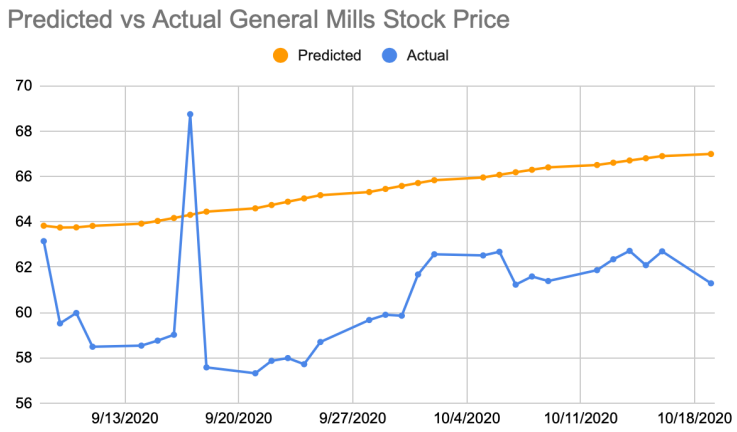
The model failed to predict a slight trough in the trend. However, the model accurately predicted a great stability in price, as opposed to the large increases/decreases forecasted for other differently performing stocks, as reflected by the aforementioned greater average accuracy (91.49%).



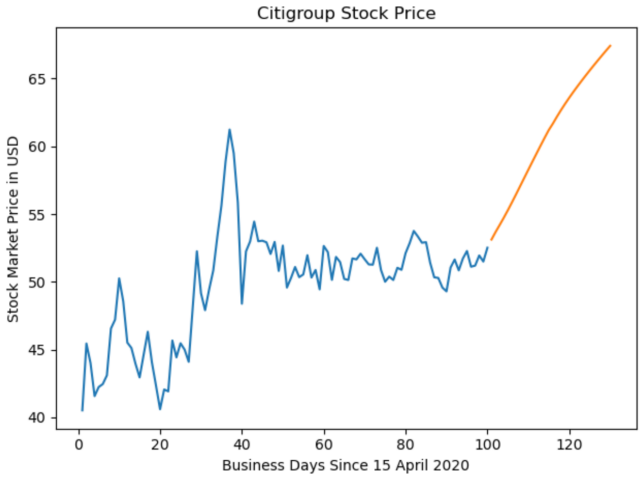


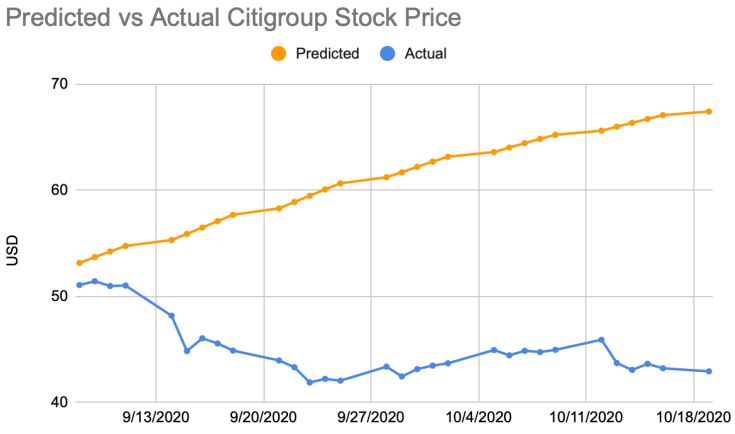
On average, the model demonstrated an average accuracy of 97.93% for the stable stock that is Dannon (DANOY). The average accuracy for the first 10 days was 98.00%, 97.63% for the second set of 10 days, and 98.15% for the last. The LSTM model again exhibited a higher and more consistent average forecast accuracy for the stable-performing stock. The LSTM model forecasted a decrease in price of 0.27 USD from September 4, 2020 to October 19, 2020; the price actually decreased by 0.89 USD.





The LSTM model exhibited consistent and mostly accurate forecasts. The actual stock price decreased to 57.32 USD and generally increased from there. The model accurately predicted the general increase, but was thrown off by the steep drop early on, culminating in a 91.72% average accuracy



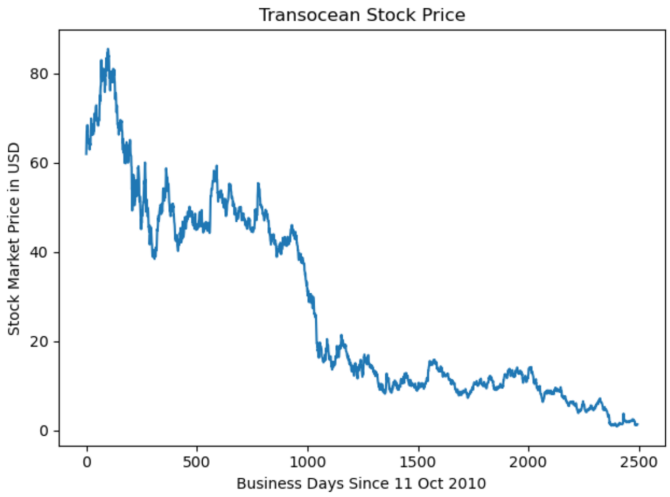


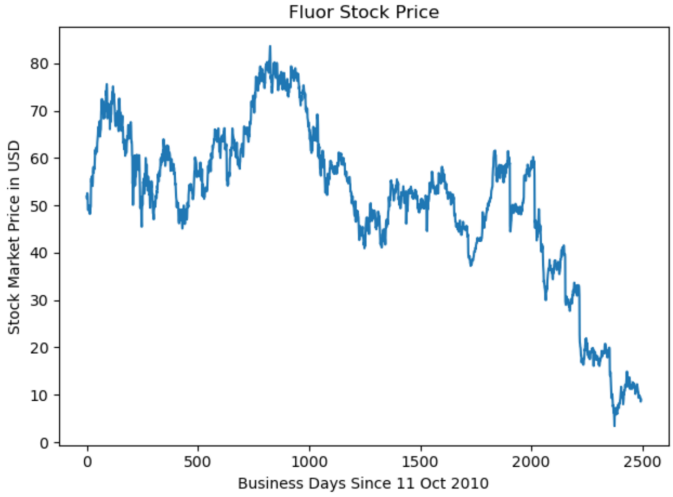
The LSTM model was completely inaccurate when predicting Citigroup's stock price for the 30 business days after September 4, 2020. Citigroup stock dropped by ~8 USD, while the LSTM model had a much more sunny outlook, predicting an increase of ~14 USD. There was at least an average accuracy of 82.90% for the first 10 days, but this is a mere consolation as the model performed terribly overall.

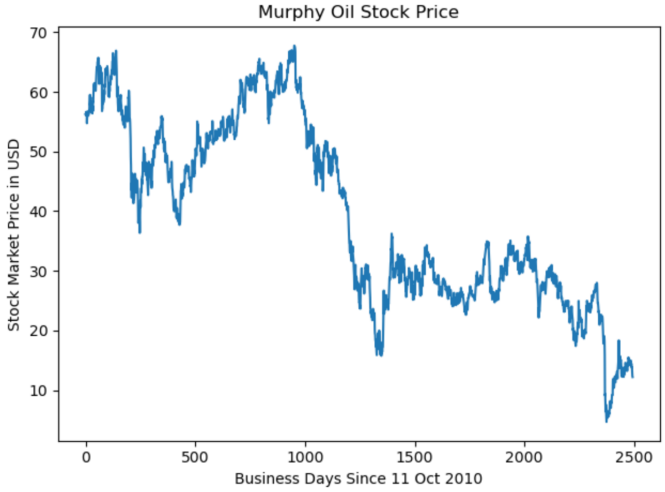
The LSTM model found its most accurate predictions when analyzing stable-performing stocks. Forecasts varied greatly, but were mostly exceptionally accurate barring Citigroup. Its prediction accuracies for stable stocks were much more consistent; the model didn't become as significantly inaccurate towards the end of the 30 days as it had become in the predictions for flourishing stocks.

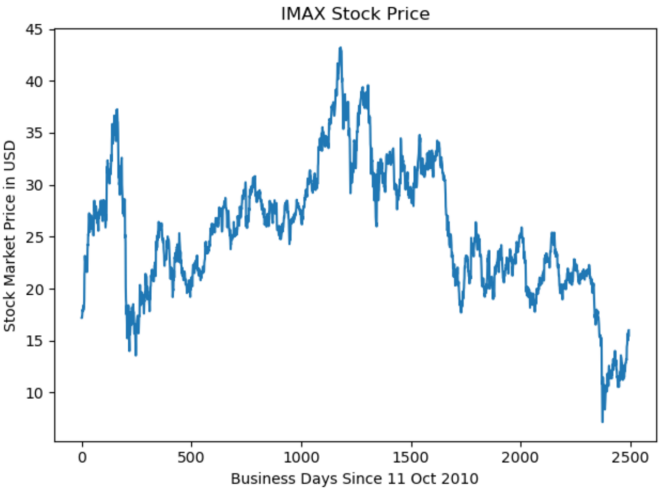
Poor-Performing: Transocean, Fluor

The LSTM model performed poorly when forecasting to pair stocks performing poorly themselves over the last 10 years, Transocean (RIG), Fluor (FLR).

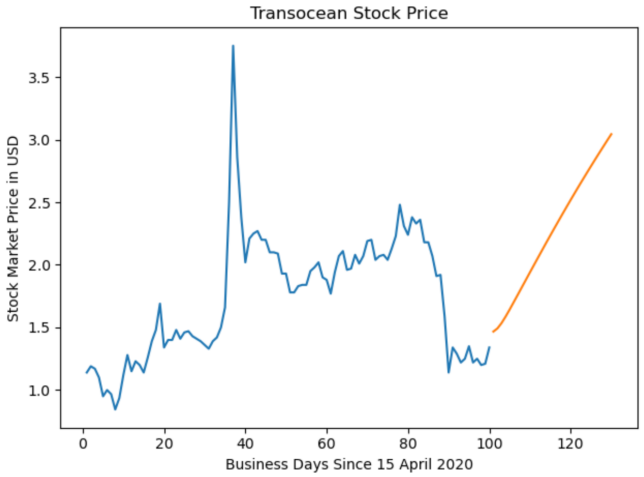


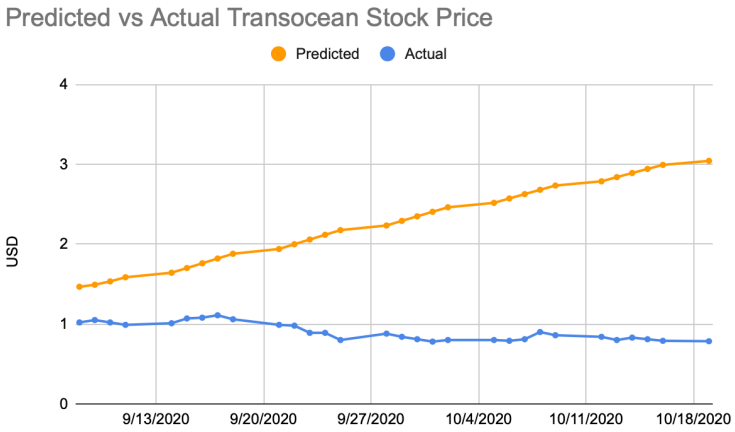




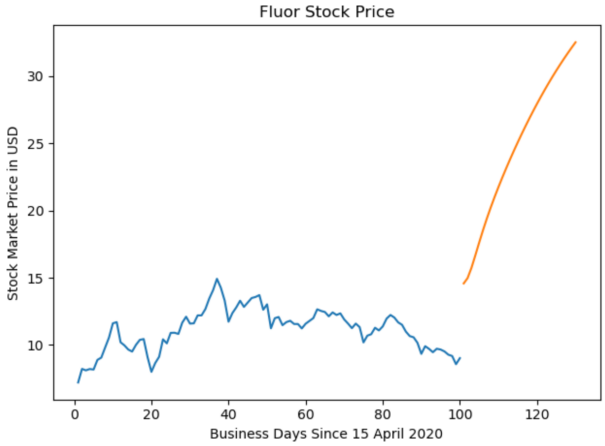


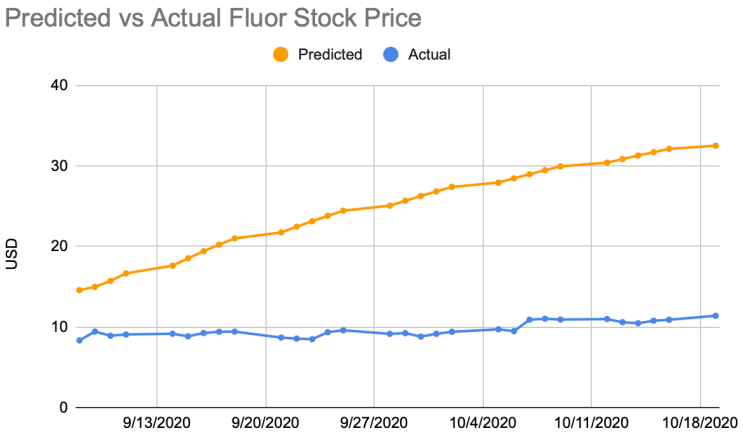
The model boldly predicted a sharp increase to ~3.0 USD, which Transocean had not seen since June 2020.



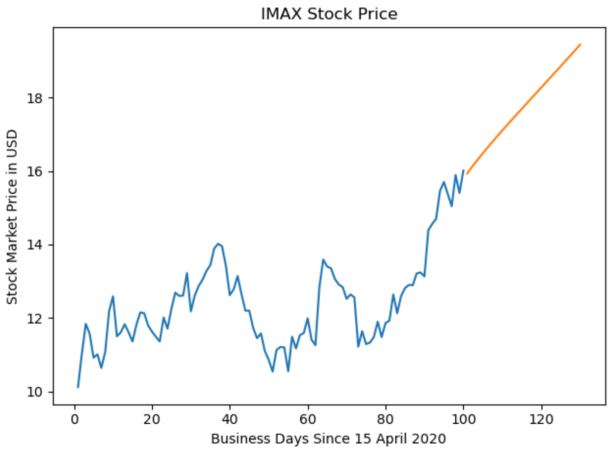


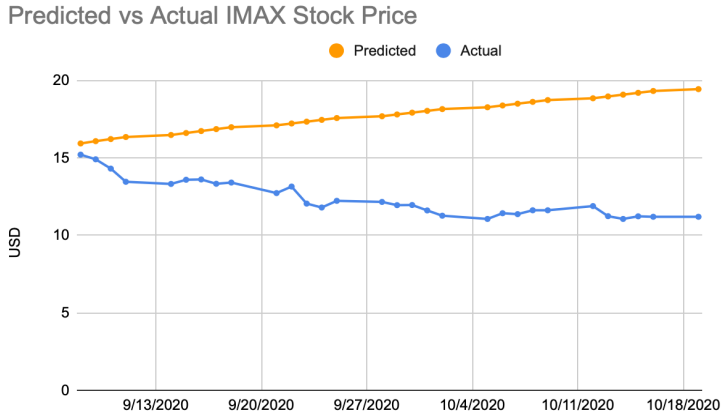
Transocean stock instead sunk down further to 0.79 USD, contrasting sharply with the LSTM model's projected increase to 3.04 USD. The same accuracy percentage calculation formula was applied, but yielded negative percentages for forecasts after September 21st due to the magnitude of the differences in proportion to the actual prices. These negative percentages were just switched to 0; they were not accurate at all. At least for the first 10 days, the LSTM model produced the positive percentage accuracy, although miniscule, of 38.13%



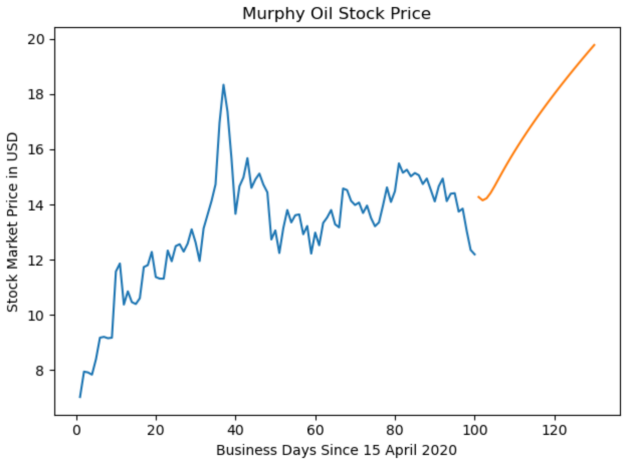


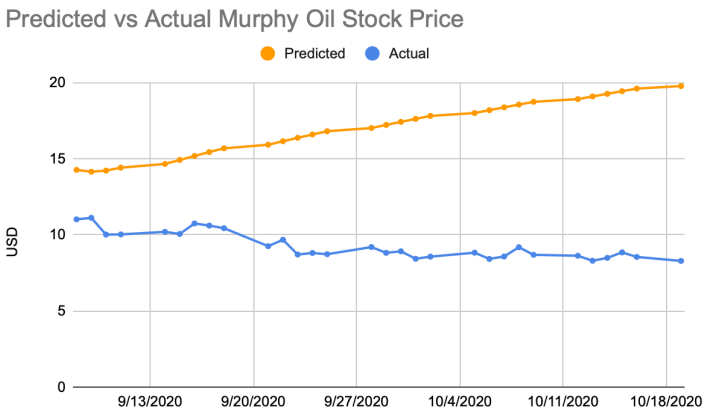
The LSTM model forecasted a strong revival of Fluor stock to 32.50 USD, a price not seen since May 2019. Indeed, a general increase was demonstrated from 8.33 USD to 11.39 USD, but still nowhere near the forecasted price, culminating in a 0% accuracy, indicating poor LSTM performance when handling poorly-performing stocks.





The LSTM model at least forecasted an absolute change of ~4 USD. However, IMAX stock price dropped by 4.01 USD, whereas the forecasted change was an increase of 3.50 USD. Again, the lines took completely different paths yielding a 54.73% average accuracy, a score that a blindfolded monkey could in fact realistically challenge.





Once again, the LSTM model failed miserably when attempting to predict the stock price of a miserably failing Murphy Oil. The trends took very different paths.

Conclusion

The Long Short Term Memory recurrent neural network model found itself long short of producing accurate forecasts for poorly performing stocks. This is most likely due to the fact that when there is next to 0 demand for something, its price will just keep decreasing, regardless of its historical value. The LSTM merely analyzes stock data as it would any time-series dataset, without taking into account supply and demand and such variables. In a world where stock prices depended solely on its previous values, the LSTM model would perhaps enjoy greater accuracy when predicting low-performance stocks.

Good	**1st 10 days**	**2nd 10 days**	**3rd 10 days**	**Average**
Amazon (NASDAQ: AMZN)	93.99	86.74	79.39	86.71
Apple (NASDAQ: AAPL)	97.46	95.3	87.37	93.38
O'Reilly (NASDAQ: ORLY)	92.54	85.41	75.77	84.57
Google (NASDAQ: GOOGL)	94.31	96.14	94.04	94.83
avg	94.58	90.90	84.14	89.87
Stable	**1st 10 days**	**2nd 10 days**	**3rd 10 days**	**Average**
Kellogg (NYSE: K)	92.22	89.25	92.99	91.49
Danone (OTCMKTS: DANOY)	98	97.63	98.15	97.93
General Mills (NYSE: GIS)	91.84	90.7	92.64	91.73
Citigroup (NYSE: C)	82.9	57.38	50.84	63.71
avg	91.24	83.74	83.655	86.21
Bad	**1st 10 days**	**2nd 10 days**	**3rd 10 days**	**Average**
Transocean (NYSE: RIG)	38.13	0	0	12.71
Murphy Oil (NYSE: MUR)	55.64	6.75	0	20.80
IMAX (NYSE: IMAX)	79.62	50.75	33.81	54.73
Fluor (NYSE: FLR)	0.57	0	0	0.19
avg	43.49	14.38	8.45	22.11

These shortcomings also present themselves in forecasts for good and stable stocks, but not to the extent that it is seen in forecasts for badly performing stocks. Overall, the LSTM model performed at its best when handling well-performing and stable stocks.

Moreover, more than just the previous day's value goes into determining a given day's price. LSTMs are actually already well suited to series analysis such as sentiment analysis, so a reasonable extension of this experiment could involve gathering news articles about respective stocks on according dates, evaluating the sentiments shown in the articles and whether they'd affect a stock's price negatively or positively and to what extent, and incorporating this information to more accurately predict the next day's price. The model of course can't predict how many or what kinds of articles will be written in the future, instead article analysis would have to be incorporated in real-time (once its impact based on sentiment is quantified). This could also allow for the artificial integration of other variables like lawsuits, executive position changes, PR stunts, etc., which would just be represented as a really good or bad article based on the result or implications. This is a significant advantage of LSTM models and recurrent neural networks over the ARIMA model; LSTM networks show more potential for growth while ARIMA models are strictly for time-series analysis.

LSTM RNNs are best tailored towards forecasting the values of well-performing and stable-performing stocks 10 days into the future, and are fairly inaccurate when analyzing low-performance stocks.

V. ARIMA Model