Amazon Stock Price Analysis and Prediction using Machine Learning

Ruben Solis

April 2023

Contents

List of Figures

1 Introduction

1.1 Objective:

The scope of this project is to build several deep learning algorithms based on RNN techniques that can predict future values of an indicator using Time-Series Forecasting methods in order to achieve the highest possible accuracy.

1.2 Significance:

Stock market prediction is the act of trying to determine the future value of company stock or other financial instruments traded on an exchange. The successful prediction of a stock's future price could yield significant profit. Moreover, applying the correct stock market prediction methods helps you know better about your entry and exit points. So often the traders either enter or exit the market at the wrong times which means they fail to capitalize on the full potential of making profits.

2 Data

2.1 Data Source:

The data used for this project may be found at the following link:
`https://www.kaggle.com/datasets/varpit94/amazon-stock-data`

2.2 Data Description:

This dataset provides the history of daily prices of Amazon stock (AMZN). All the column descriptions are provided. The currency is USD.

- `Open` = Price from the first transaction of a trading day
- `High` = Maximum price in a trading day
- `Low` = Minimum price in a trading day
- `Close` = Price from the last transaction of a trading day
- `Adj Close` = Closing price adjusted to reflect the value after accounting for any corporate actions
- `Volume` = Number of units traded in a day**

2.3 Wrangling and Cleaning:

The purpose of data wrangling and cleaning are:

- To ensure that all features are of the correct data type.
- To ensure that missing values are properly dealt with (imputation).
- To prepare the data set for exploratory data analysis and statistical analysis.

Our data set contains both numeric and date-time objects (specifically data are of `float64`, `int64`, and `datetime64`). The information is summarized below.

Column Name	Data Type
Date	datetime64
Open	float64
High	float64
Low	float64
Close	float64
Adj Close	float64
Volume	int64

2.4 Incorrect Value and Missing Value Imputation

This data set was very tidy from the get-go. For instance, there were no missing values in the dataset. Most of the features were the correct data type. We only had to change the `date` feature to a `datetime64` object and set it to our index. This was done so we can work with it as a time series.

3 Data Visualization and Analysis

3.1 Feature Description and Distribution

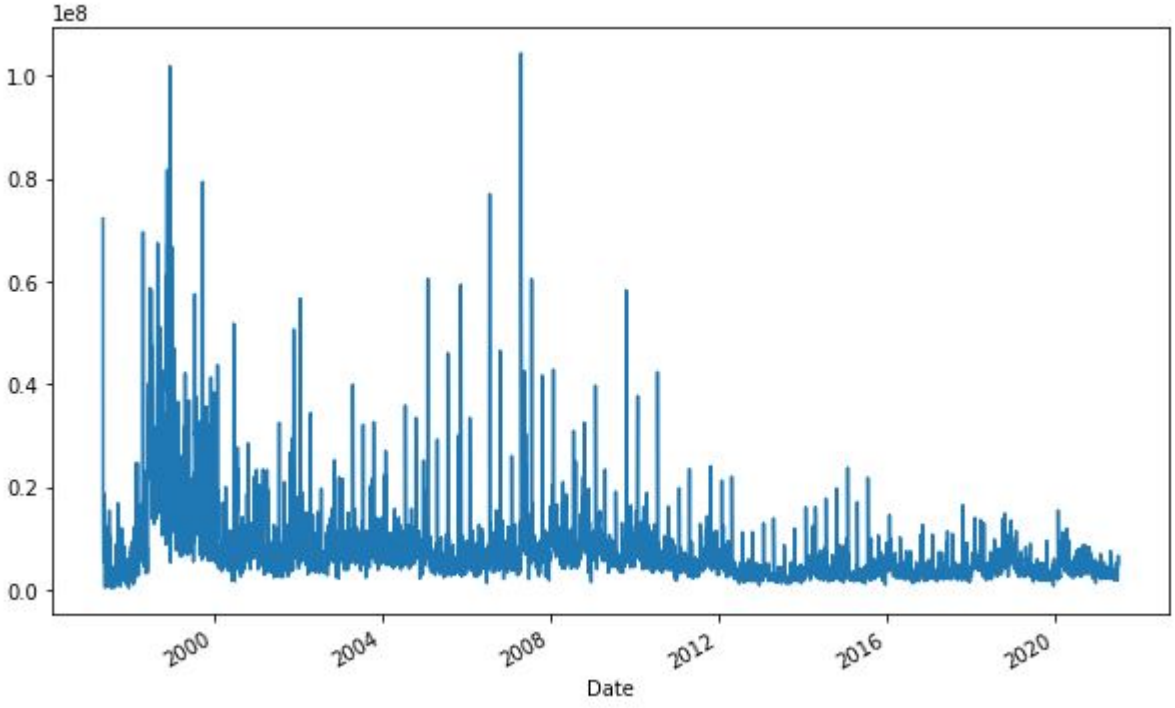

Figure 1: Volume of Units from 2000 to 2020

When we look at Figure 1, We can see a lot of peaks and density between 2000 and 2010. Let us look at how the other features are distributed.

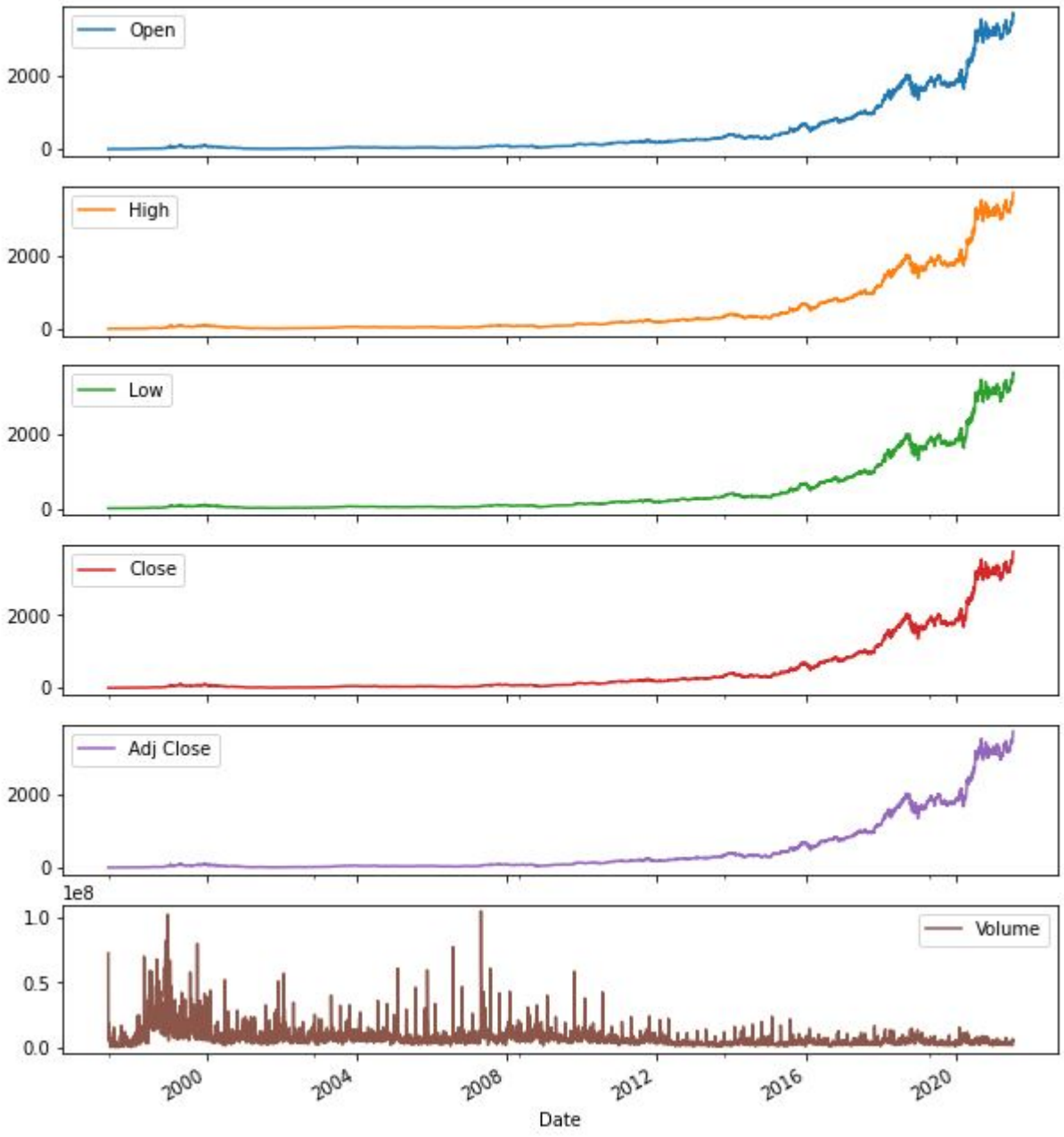

Figure 2: Feature Distribution

The shape of the curve for `Open`, `Close`, `High`, and `Low` data have the same shape. Only the `Volume` has a different shape.

3.2 Seasonality

Resampling for months or weeks and making bar plots is another widely used method of finding seasonality. Here we create a bar plot of monthly data for the 2020 year.

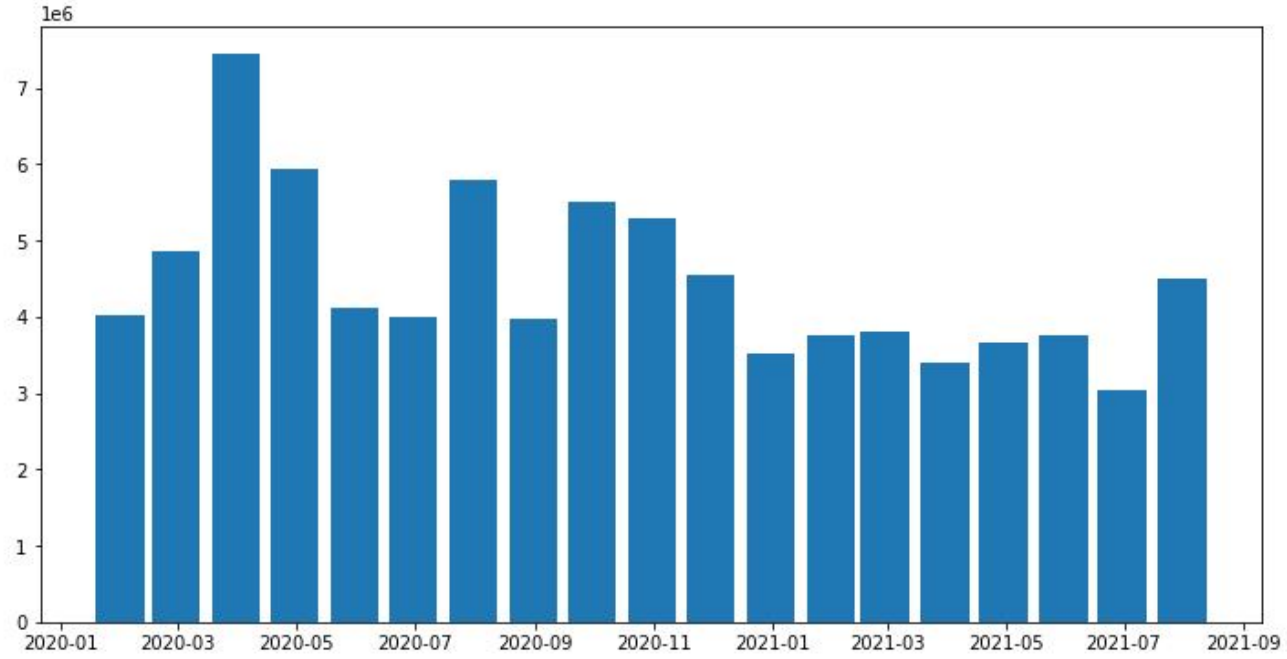

Figure 3: Bar Plot of Monthly Data in 2020

Each bar represents a month. A huge spike in April 2020. Otherwise, there is monthly seasonality after 2020 ends.

3.3 Resampling and Rolling

Resampling is very common in time-series data. Most of the time resampling is done to a lower frequency. Though resampling of higher frequency is also necessary, especially for modeling purposes. Not so much for data analysis purposes. In the `Volume` data we are working on right now, we can observe some big spikes here and there. These types of spikes are not helpful for data analysis or for modeling. Normally to smooth out the spikes, resampling to a lower frequency and rolling is very helpful.

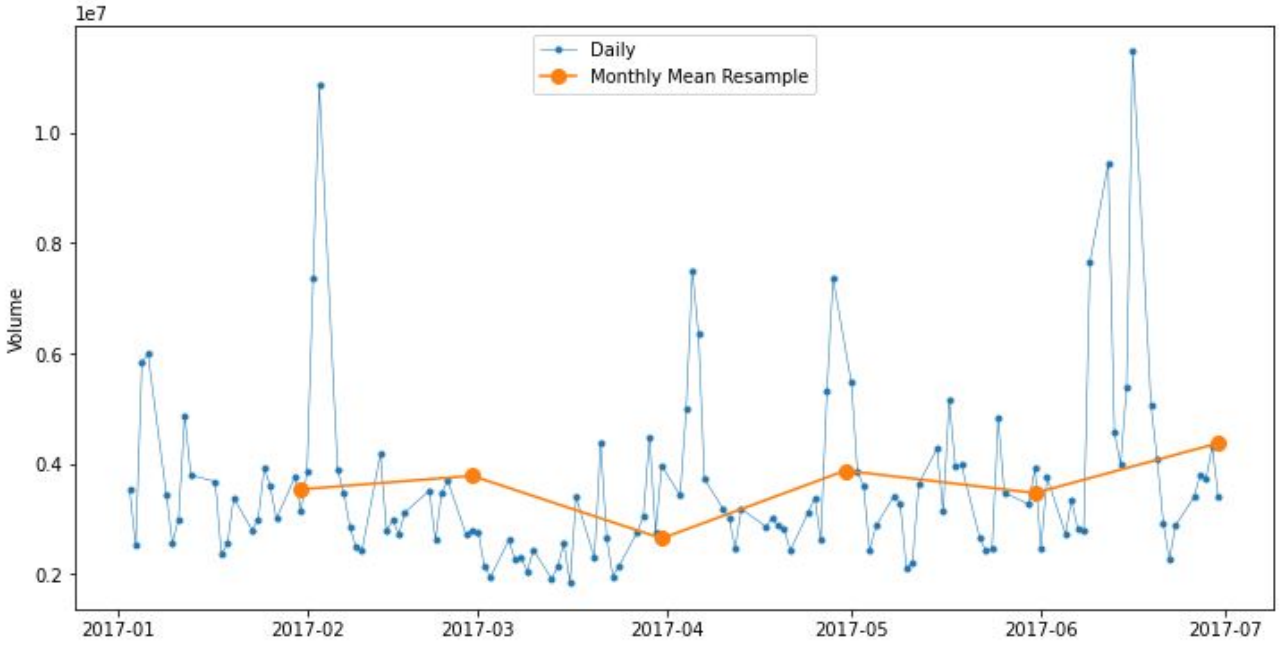

Figure 4: Monthly Mean Resample

3.3.1 Weekly Resampling

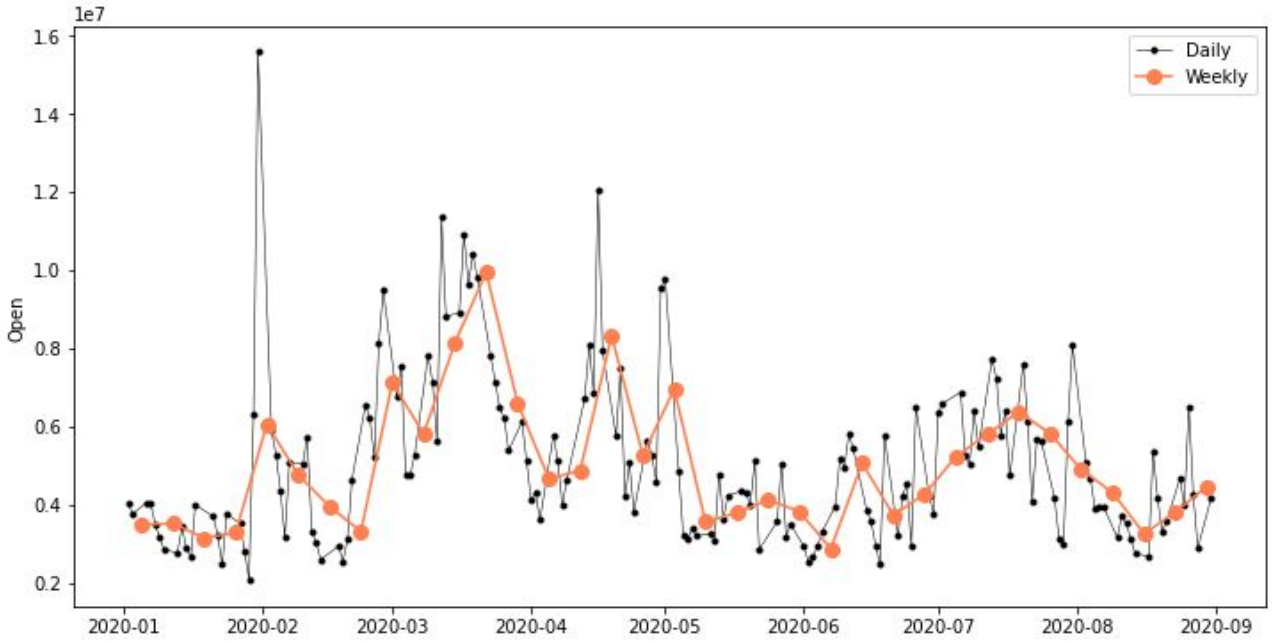

Figure 5: Weekly Resampling

3.4 Rolling

Rolling is a useful method of smoothing out the curve. It works by taking the average of a specified amount of data. For instance, if I would like a 7-day rolling, it gives us the 7-day average data.

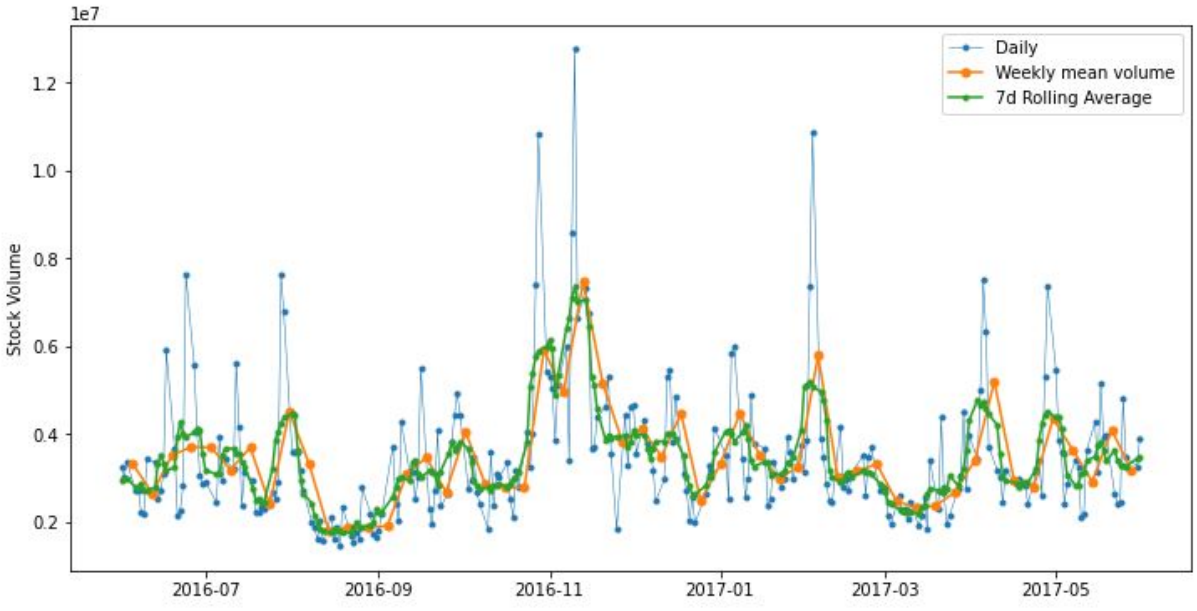

Figure 6: Seven Day Rolling Average

As we can see a 7 day rolling average is a bit smoother than the weekly average.

3.5 Plotting the Change

3.5.1 Shift

The shift function works by shifting the data before or after the specified amount of time. It will shift the data by one day by default. That means you will get the previous day's data. In financial data, like this one, it is helpful to see the previous day's data and today's data side by side.

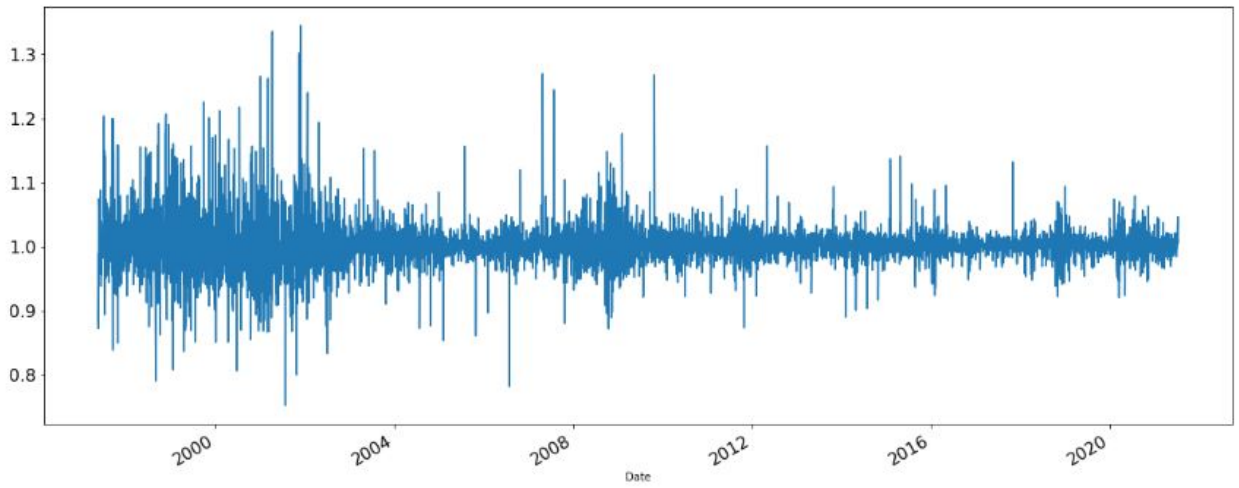

Figure 7: Shift

In the figure above, we use the `div()` method to help fill up the missing data. For instance, `df.div(6)` will divide each element in the data frame by 6. However, here we used `df.Close.shift()`. As a result, each element of df will be divided by each element of `df.Close.shift()`. We do this to avoid the null values that are created by the 'shift()' operation.

We next plot the change for 2001.

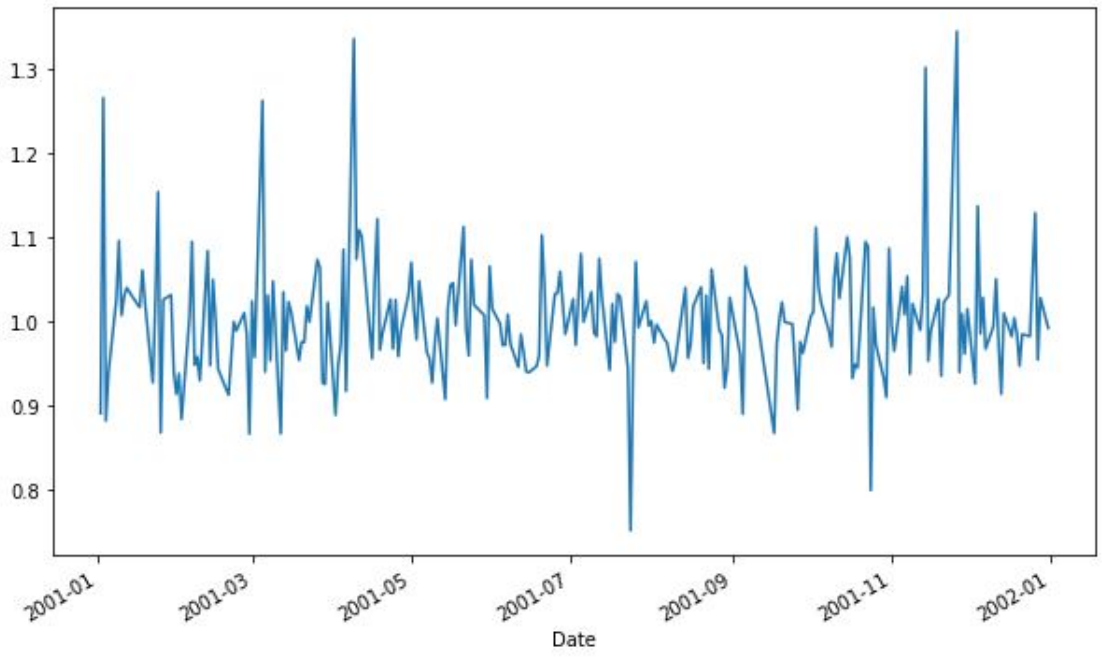

Figure 8: 2001 Change

3.6 Percent Change

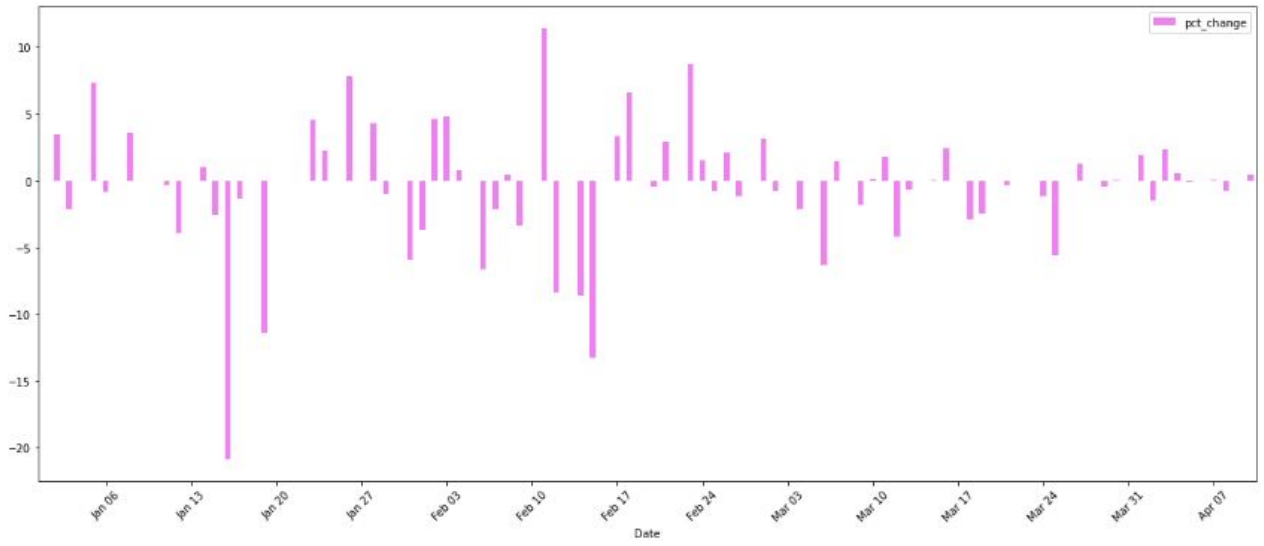

Figure 9: Percent Change

We can clearly see the percentage change in the data.

3.7 Differencing

Differencing takes the difference in values of a specified distance. It is a popular method to remove the trend in the data. The trend is not good for forecasting or modeling.

We have used expanding window, another way of transformation. It keeps adding the cumulative. For example, if you add an expanding function to the 'High' column first element remains the same. The second element becomes cumulative of the first and second elements, the third element becomes cumulative of the first, second, and third elements, and so on. You can use aggregate functions like mean, median, standard deviation, etc. on it too

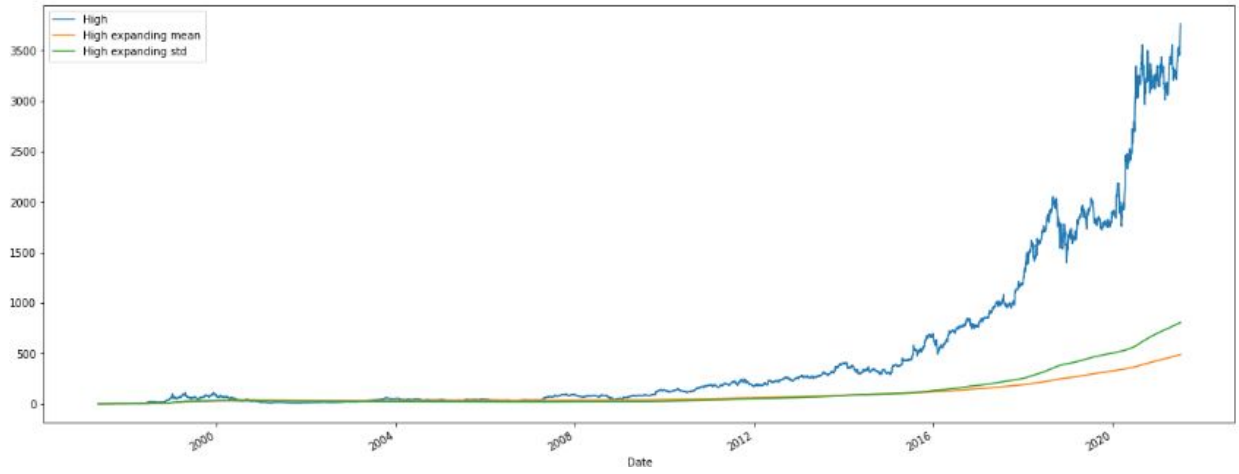

Figure 10: Differencing

3.8 Decomposition

Decomposition will show the observations and these three elements in the same plot:

- Trend: Consistent upward or downward slope of a time series.
- Seasonality: Clear periodic pattern of a time series.
- Noise: Outliers or missing values.

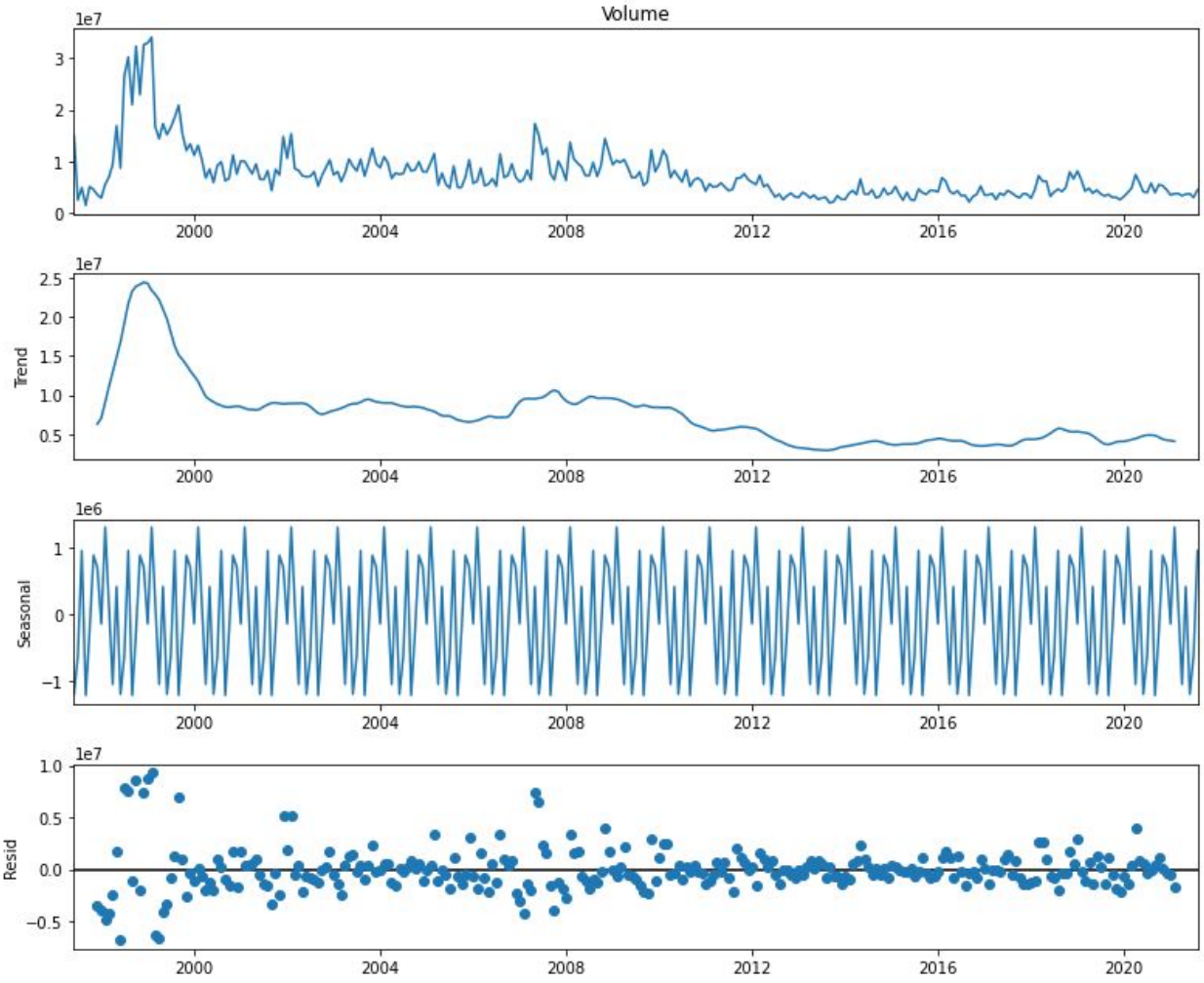

Figure 11: Decomposition

We will now move to the modeling phase of the report.

4 Model Development and Selection

Before we split the data, we used the `MinMaxScaler` function which transforms the data into a given range, in our case between zero and one. Then we went ahead to split the data.

4.1 Data Modeling

For our modeling process, we elected to use Recurrent Neural Networks (RNNs) for modeling sequential data and a specific subset of sequential data—time-series data. It would be beneficial to discuss the components of RNNs briefly.

4.1.1 RNNs and Sequential Data

What makes sequences unique, from other data types, is that elements in a sequence appear in a certain order, and are not independent of each other. In other words, typical machine learning algorithms for supervised learning assume that the input data is Independent and Identically Distributed (IID). For example. if we have n data samples, $x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(n)}$ the order in which we use the data for training our machine learning algorithm does not matter. But this assumption is not valid anymore when we deal with sequences-by definition, order matters

4.1.2 Representing Sequence

The following figure provides an example of time-series data where both x's and y's naturally follow the order according to their time axis; therefore, both x's and y's are sequences:

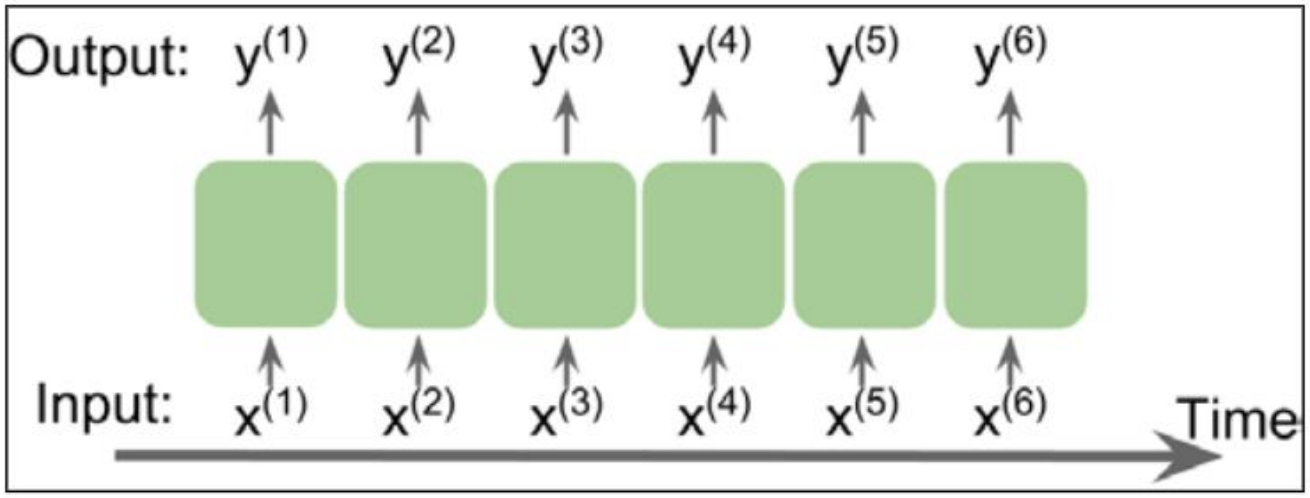

Figure 12: Sequential Data

The standard neural network models, such as Multilayer Perceptron (MLPs) and Convolution Neural Networks (CNNs), are not capable of handling the order of input samples. In other words, one can say that such models do not have a memory of the past seen samples. For instance, the samples are passed through the feed-forward and back-propagation steps and the weights are updated independent of the order in which the sample is processed.

RNNs, by contrast, are designed for modeling sequences and are capable of remembering past information and processing new events accordingly.

4.1.3 Different Categories of Sequential Modeling

The figure below shows the various types of sequential modeling.

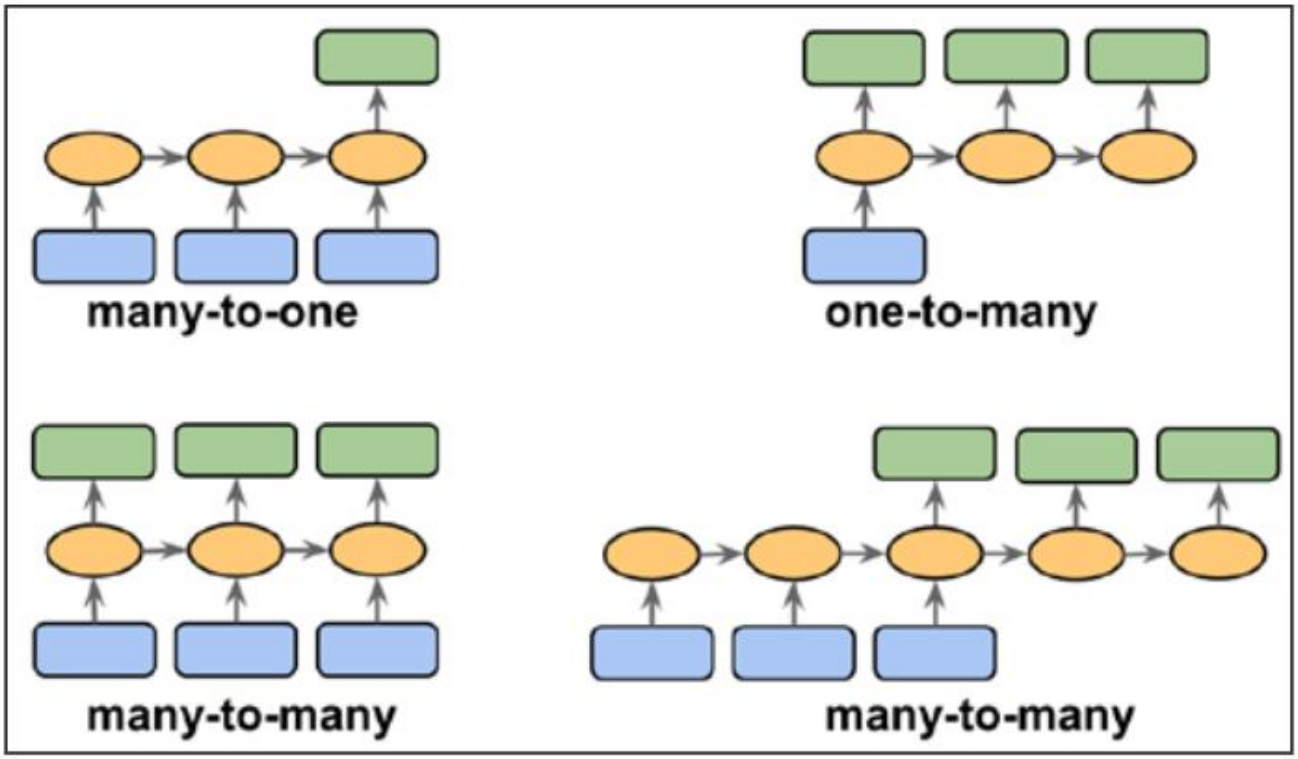

Figure 13: Sequential Model Types

- **Many-to-one**: The input data is a sequence, but the output is a fixed-size vector, not a sequence. For example, in sentiment analysis, the input is text-based and the output is a class label.
- **One-to-many**: The input data is in standard format, not a sequence, but the output is a sequence. An example of this category is image captioning—the input is an image; the output is an English phrase.
- **Many-to-many**: Both the input and output arrays are sequences. This category can be further divided based on whether the input and output are synchronized or not. An example of a synchronized many-to-many modeling task is video classification, where each frame in a video is labeled. An example of a delayed many-to-many would be translating one language into another. For example, an entire English sentence must be read and processed by a machine before producing its translation into German.

4.1.4 Challenges

Backpropagation through time, or BPTT, introduces some new challenges. Since $\frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(\mathbf{k})}}$ in the computing gradients of a loss function, the so-called vanishing or exploding gradient problem arises.

In practice, there are two solutions to this problem:

- Truncated backpropagation through time (TBPTT)
- Long short-term memory (LSTM)

We elected to go with LSTM since it has been more successful in modeling long-range sequences by overcoming the vanishing gradient problem.

4.1.5 LTSM

LSTMs were first introduced to overcome the vanishing gradient problem (*Long Short-Term Memory*, *S. Hochreiter*, and *J. Schmidhuber*, *Neural Computation*, 9(8): 1735-1780, 1997). The building block of an LSTM is a memory cell, which essentially represents the hidden layer.

For example, in each memory cell, there is a recurrent edge that has the desirable weight $w = 1$, as we discussed previously, to overcome the vanishing and exploding gradient problems. This brief explanation will have to suffice since the details of LSTM are beyond the scope of this report.

4.2 Performance Evaluation

In order to evaluate the performance of our model on a given data set, we need some way to measure how well its predictions actually match the observed data. In other words, we need to quantify the extent to which the predicted response value for a given observation is close to the true response value for that observation. We will look at the mean-squared error, mean absolute error, root-mean-squared error, and R^2 of our model.

4.2.1 Mean Squared Error

The mean squared error, given by

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2,$$

where $\hat{f}(x_i)$ is the prediction that $\hat{f}$ gives the ith observation. The MSE will be small if the predicted response is very small if the predicted responses are very close to the true responses, and will be large if, for some of the observations, the predicted and true responses differ substantially.

4.2.2 Root Mean Squared Error

The root-mean-squared error is given by

$$RMSE = \sqrt{MSE}$$

.

4.2.3 Mean Absolute Error

The mean absolute error is a measure of errors between paired observations expressing the same phenomenon. It is given by the following equation

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - x_i|,$$

wherr y_i is the prediction and x_i is the true value. Like the MSE, we desire this to be low for the reasons stated above.

4.2.4 R^2

The R^2 statistic provides an alternative measure of fit. It takes the form of a proportion of variances explained - and so it always takes on a value between 0 and 1, and is independent of the scale of Y.

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

where $RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ and $TSS = \sum_{i=1}^{n}(y_i - \bar{y})^2$.

We summarize the results below.

Model	**MSE**	**RSME**	**MAE**	$\mathbf{R^2}$
RNN	54104.765754421096	156.86053865508845	162.61549445475492	0.9256549474452863

5 Conclusion and Recommendations

In this report, we developed a model to predict future stock prices of Amazon based on time series data. Based on our findings, we have the following recommendations.

1. While the MSE, RSME and MAE are relatively high, we have very good R^2 score. With a score of approximately 92.566% , we can interpret this to mean that 92.566% of the variance in explained by our RNN.

2. As data sets become more and more readily available, it may prove to be beneficial to conduct this analysis again in the future on different time series data sets and see how well the model performs.

3. It may also be helpful to build different models with different parameters to determine which one performs best on a given data set.