Stocks Prediction

Data Science Nanodegree
December 28th 2021
Lama Alharbi

Table of Contents

Definition

Project Overview

In recent years, different technological applications were utilized by the financial sector. One of these applications is machine learning. Stocks market is considered volatile. It's hard to predict the prices since multiple factors contribute to the prices changes such as political, social and economic factors.

This project aims to follow a data science approach in predicting the prices of stocks. The project consists of two parts. The first part is a Jupyter notebook that demonstrate two models and utilizes the data of Tesla and Amazon. The second part is a web app where the user can enter any valid company ticker name to train the model on the data and show the results as a chart.

Problem Statement

The goal is to develop a robust model to predict stocks prices. the tasks involved are:

1-Fetch the stocks prices data and preprocess it.

2-Do a exploratory data analysis to understand the patterns.

3-Model the date through using two algorithms:Long Short Term Memory and AutoRegressive Integrated Moving Average.

4-Compare results through metrics discussed in the next section.

5-Deploy the optimal model on a web app.

The final application will fetch the data according to the user input then will train the model and display the results for the user in the web page.

Metrics

A metric is a function that is used to judge the performance of your model. The metric used to evaluate the performance of both algorithms is Root Mean Sqaure Error(RMSE) Where y' denotes the predicted value and y denotes the actual value. The number n refers to the total number of values in the test set.This metric is used since the problem is a time series prediction. One advantage of RMSE is that it's in the same unit as the forecast variable, Moreover the outliers have a huge effect on the result so it will be easier to discover them. The lower the value the better the performance.[1]

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

Fig1: RMSE formula

Analysis

Data Exploration

Dataset used are fetched from Yahoo finance through a python library called "yfinance".
Two compaines were used to demonstrate the model.
1-Tesla, Inc. designs, develops, manufactures, leases, and sells electric vehicles, and energy generation and storage systems in the United States, China, and internationally. The company operates in two segments, Automotive, and Energy Generation and Storage.[2]
2-Amazon.com, Inc. engages in the retail sale of consumer products and subscriptions in North America and internationally. The company operates through three segments: North America, International, and Amazon Web Services (AWS). It sells merchandise and content purchased for resale from third-party sellers through physical and online stores.[3]
The features included are:
Ticker:The company symbol.
Close:The closing recorded price of the equity symbol on the date.
High:The highest market price of the equity symbol on the date.
Low:The lowest recorded market price of the equity symbol on the date.
Open: The opening market price of the equity symbol on the date.
Volume:it measures the number of a stock's shares that are traded on a stock exchange in a day or a period of time .
Adj Close:the closing price after adjustments for all applicable splits and dividend distributions.[4]

The data doesn’t contain duplicated or missed records and the targeted feature for this project is the adjusted closing price.

Data visualization

the following chart shows the Tesla and Amazon stocks prices from January 2017 until December 2021.

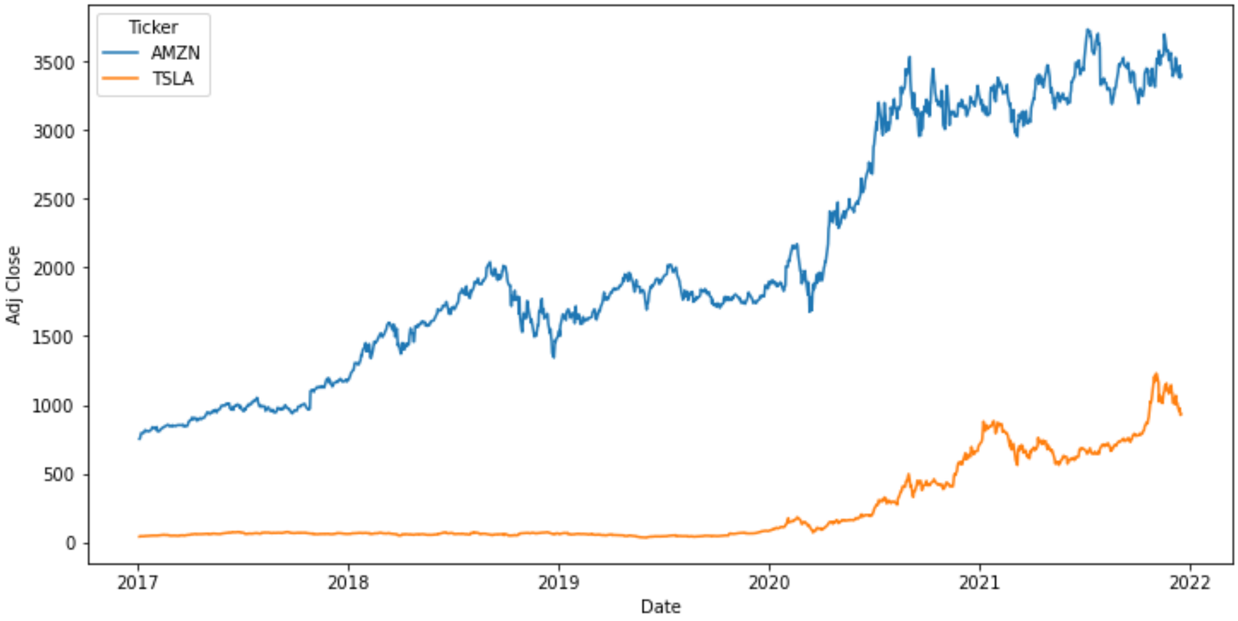

fig2: stocks prices for Tesla and Amazon from 2017 until 2021

As it can be seen both companies’ values have increased as time passes by. For Amazon there has been a significant drop when the pandemic started but it went up again by mid 2020. Amazon stocks volatility is higher than Tesla's. As for the starting prices, it was around 600USD for Amazon while it was around 50USD for Tesla. the current price for Amazon is approximately 3200USD while it's 700USD for Tesla.

In order to understand the features better. As it can be seen all features are linearly correlated except for the volume. This could indicate that models such ARIMA and linear regression won’t perform well since dependent variables are correlated.

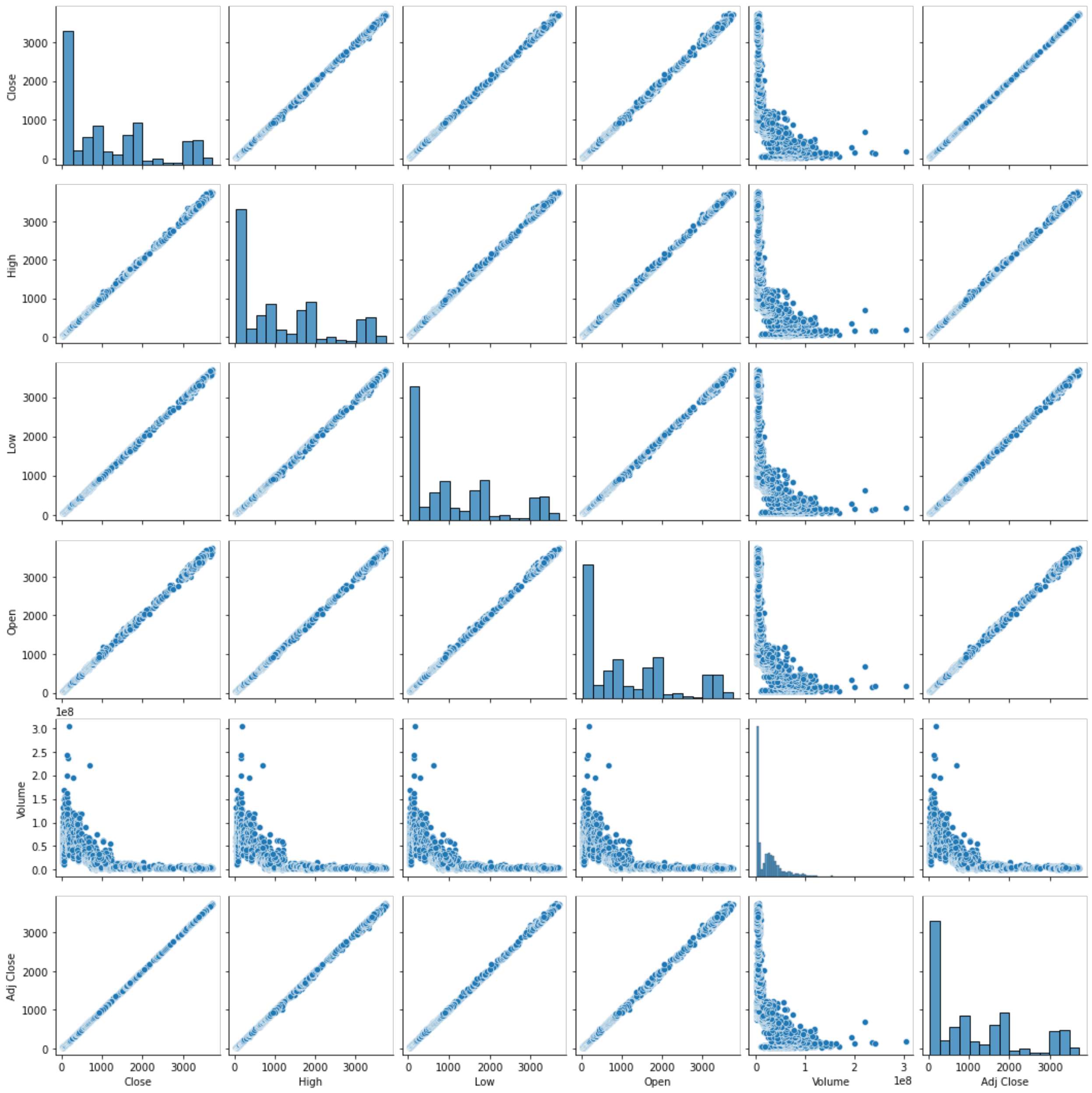

Fig3: Data features correlation plot

Methodology

Data Preprocessing

The preprocessing for ARIMA included splitting the data into train and test sets and checking if the data was stationary or not. A stationary time series data means that the statistical features such as mean, variance and autocorrelation are constant over time. If the data is stationary, it means that it’s easy to predict the future values. Stationarizing the data is a crucial step to fit ARIMA model. This is done through the differencing factor which will be discussed in the implementation part. In order to check if the data is stationary or not two tests can be done: Augmented Dickey Fuller Test (ADF) and Kwiatkowski-Phillips-Schmidt-Shin Test (KPSS). In this project ADF is done. Hypotheses for the test are:

-Null Hypothesis: there is a unit root hence the time series is not stationary

-Alterative Hypothesis: there is no unit root hence the time series is stationary

The test provides the following information: P-value, value of the test statistics and the critical values. If the p-value obtained is less than or equal 0.05 the null hypothesis is rejected. If the p-value obtained is more than 0.05 then it means the fail to reject the null hypothesis. [5]

For Amazon data, the p-value is 0.84 and the test statistic is greater than the critical values the null hypothesis won't be rejected hence the data is non stationary.

As for Tesla data, the p-value is 0.97 and the test statistic is greater than the critical values the null hypothesis won't be rejected hence the data is non stationary.

As for long short-term memory model, the preprocessing steps are normalization, train-test splitting and reshaping the data into three dimensional shape. To prepare the input for the neural network. the data need to be normalized using a scaler, the scaler used is MinMaxScaler in order to obtain values between the range given. Which is between the 0 and 1. Scaling is used to reduce the complexity of the input so the neural network will learn faster, and the accuracy will be better than normal input. After normalizing the data, the input need to be reshaped into a three dimensional data frame which is done using Numpy library. The three dimensions are the following:

1-Samples: each record is considered a sample so it’s the number of records in the data.

2-Time steps: the number of time-steps that are in the sequence for this project 60 was chosen.

3-Features: the number of features to train on, for this project the targeted feature is adjusted closing price.

Implementation

ARIMA

ARIMA, short for ‘Auto Regressive Integrated Moving Average’. Are a class of models to forecast time series data that is stationary. If the data is not stationary a method called differencing is needed. Differencing is a method to transform a time series data from non-stationary to stationary through removing the series dependance of time.

In order to fit the ARIMA model obtaining the optimal hyperparameters is needed. This could be done through multiple ways either by checking autocorrelation function (ACF) and partial autocorrelation (PACF) plots and identify the AR and MA terms through them or through the use of a function called “auto_arima” from statsmodels library in python.

The hyperparameters of ARIMA model are:

p: is the number of autoregressive terms.

q: is the number of lagged forecast errors in the prediction equation.

d: is the number of nonseasonal differences needed for stationarity.

For Amazon data the optimal ARIMA order is p:1, q:1, d:1 as for Tesla data the optimal ARIMA order is p:3 ,q:2 , d:0.

To implement the model, 80% of the data was used as training data while 20% of the data was used a testing data. As for the RMSE the difference between the forecasted data and the testing data was calculated.

LSTM

LSTM is a is an advanced RNN, a sequential network, that allows information to persist. It is capable of handling the vanishing gradient problem faced by RNN. A recurrent neural network is also known as RNN is used for persistent memory.[6]

After normalizing and reshaping the data, the model can be fitted. To build the LSTM model 4 layers were implemented: two LSTM layers and two hidden/dense layers. A dense layer is a fully connected neural network layer which means every neuron is connected to its preceding layer neurons. It’s used to change the dimensions of the vector.[7] As for LSTM layer it learns long-term dependencies between time steps in sequential data.

In order to compile the model, Adam optimizer is used instead of classical optimizer to update the network weights. Adam combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems.[8] After that the model is fitted with the size of the batch being 1 and the epochs are 20. An epoch is when the dataset is passed forward backward in the neural network. As for batch size is the total number of training samples present in a single batch.

Refinement

As it was discussed before, for ARIMA model the optimal order for the AR and MA terms were calculated using the auto_arima function. Despite having the optimal order due to its limitations with long term prediction, the model performance reached its limit.

As for the LSTM model, epochs were increased to 20 epochs to let the neural network re-learn through seeing the training data 20 times, As the epoch is increased the better the results in terms of loss and accuracy as it can be seen from the following charts.

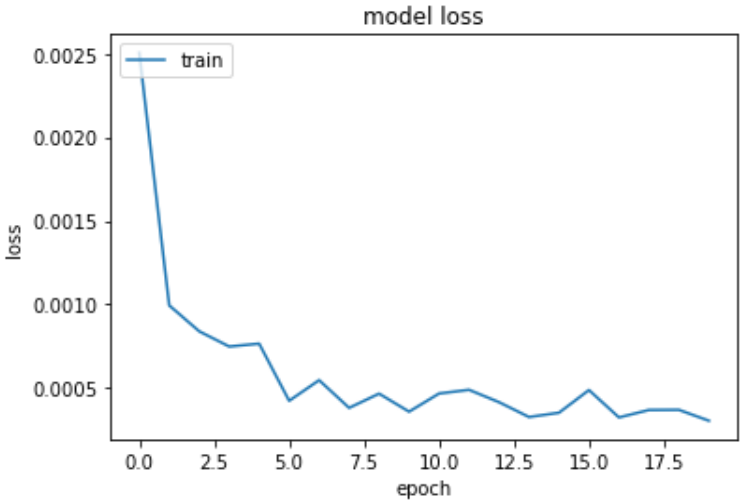

Fig4:Model loss for Amazon Data

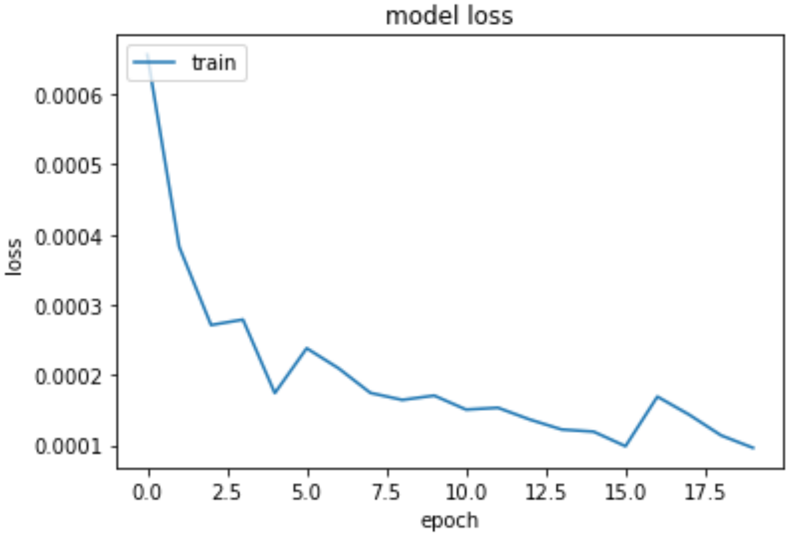

Fig5:Model loss for Tesla Data

Results

Model Evaluation and Validation

During implementation, a test set was created to validate the results.
As it can be seen from the following charts, ARIMA model performance isn't good. the difference between the test set and the predicted values is huge.

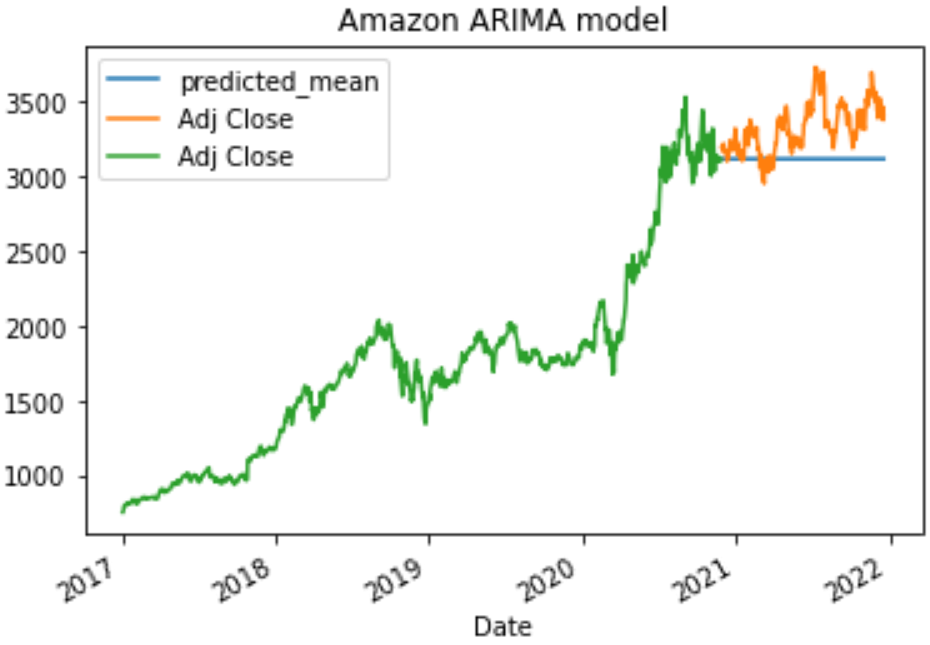

Fig6:Amazon ARIMA model validation

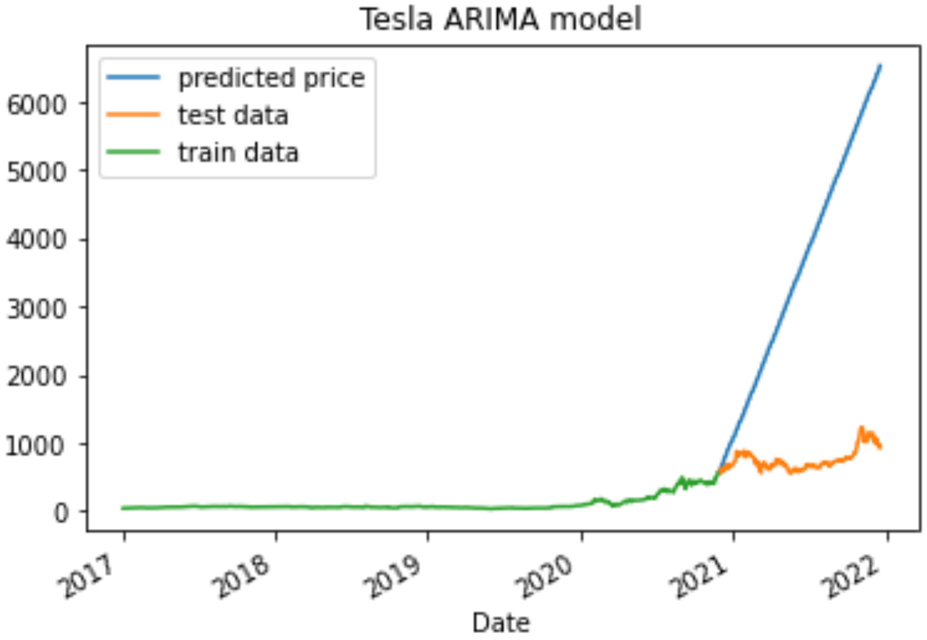

Fig7:Tesla ARIMA model validation

As it can be seen from the following charts, LSTM model performance is good, the test set values and the predicted values are very close.

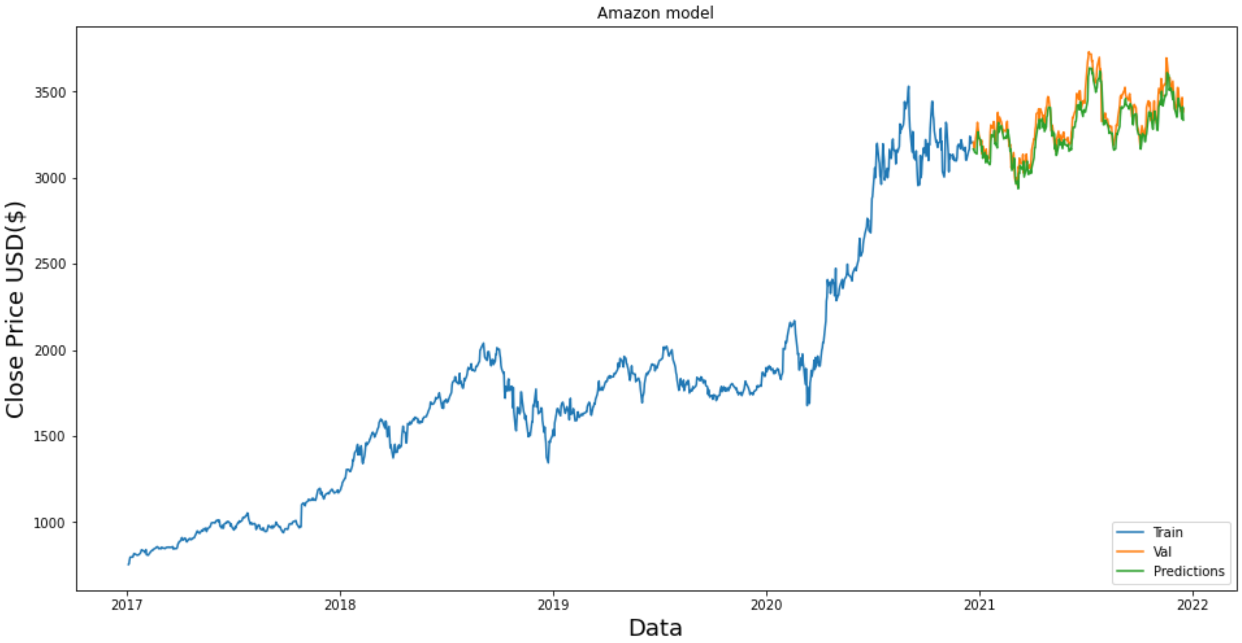

Fig8:Amazon LSTM model validation

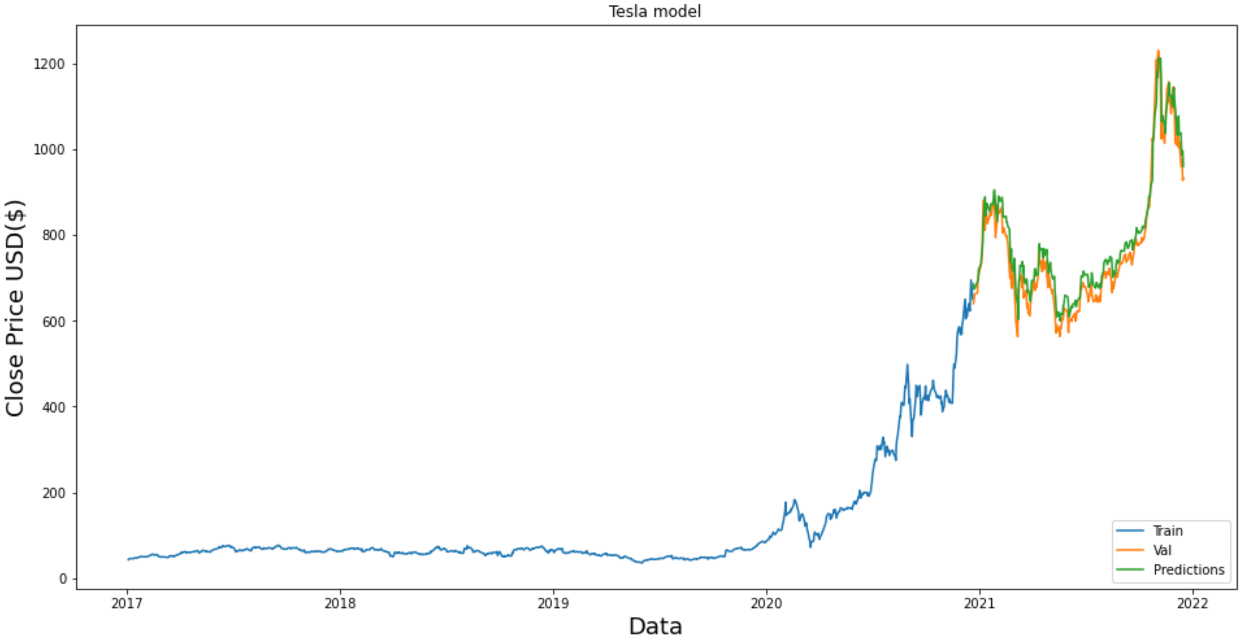

Fig9: Tesla LSTM model validation

The following table shows the RMSE for each model and data.

	Amazon	Tesla
ARIMA	211.51	2790
LSTM	47.69	25.36

Table1: RMSE for each model

Justification

As it was seen in the charts and the table, overall LSTM did better than ARIMA. This could be caused by multiple factors such as the linearity of the problem. ARIMA is a linear model hence fitting linear data is easier which is not the case in the data used. On the other hand, LSTM deals with non-linear problems well. Another factor would be that ARIMA is better for short-term prediction while LSTM can handle long-term prediction. Finally, LSTM is a neural network while ARIMA is a statistical model so the capability of the two models are quite different hence the difference in the results.

Conclusion

Reflection

Stock Prediction is not an easy task even with deep learning techniques due to the multiple factors that contribute in the price changes such as social and economical factors, even COVID had a significant effect on the prices as seen in the EDA. This project was a new experience for me despite its complexity since it deals with financial data which I've never dealt with before . It was a nice exposure on new algorithms such as ARIMA and LSTM since most of the algorithms I used before were linear regression and classification algorithms.

Improvement

It would be interesting to try and add more features generated from the technical analysis. Also, despite it being simple, linear regression could be used as another model to predict the stocks prices although as mentioned before, the problem is not linear hence models like linear regression won’t perform well on the data. Another implementation would be to try adding more layers to the neural network or even increase epochs to see if it will influence the loss and accuracy.

References & Acknowledgments

[1] https://analyticsindiamag.com/a-guide-to-different-evaluation-metrics-for-time-series-forecasting-models/

[2] https://finance.yahoo.com/quote/TSLA/profile?p=TSLA

[3] https://finance.yahoo.com/quote/AMZN/profile?p=AMZN

[4] https://www.kaggle.com/minatverma/nse-stocks-data

[5] https://www.sciencedirect.com/topics/economics-econometrics-and-finance/dickey-fuller-test

[6] https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/

[7] https://machinelearningknowledge.ai/keras-dense-layer-explained-for-beginners/

[8] https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/