Stock Market Prediction

Using ARIMA & CNN

Arnav Desai - 202018031
MSc Data Science, DAIICT
Gandhinagar

Aneri Joshi - 202018032
MSc Data Science, DAIICT
Gandhinagar

Jalaj Mehta - 20218033
MSc Data Science, DAIICT
Gandhinagar

Anjali Jain - 202018036
MSc Data Science, DAIICT
Gandhinagar

Darshan Jain - 202018043
MSc Data Science, DAIICT
Gandhinagar

Pankti Fadia - 202018045
MSc Data Science, DAIICT
Gandhinagar

Aayush Ramrakhyani - 202018046
MSc Data Science, DAIICT
Gandhinagar

ABSTRACT :

Our project aims to forecast/predict the stock price in terms of the actual value and the predicted value. For this, we have tried implementing the Arima and CNN approach to predict the stock prices.

In this project, we are using both these approaches and observing the variations in results given by both these models which we will discuss in this report.

1. Introduction

Stock market price data is generated in huge volume and it changes every second. Stock market is a complex and challenging system where people will either gain money or lose their entire life savings. Stock price prediction is very important as it is used by most of the business people as well as common people. People will either gain money or lose their entire life savings in stock market activity. It is a chaos system. Building an accurate model is difficult as variation in price depends on multiple factors such as news, social media data, fundamentals, production of the company, government bonds, historical price and country's economics. Analysis of financial time series and prediction of future stock price values and future stock price movements have been an active area of research over a long period of time.

In this project we proposed ARIMA Model and a suite of Deep Learning based Regression Model(CNN) for the purpose of forecasting Tesla index values.

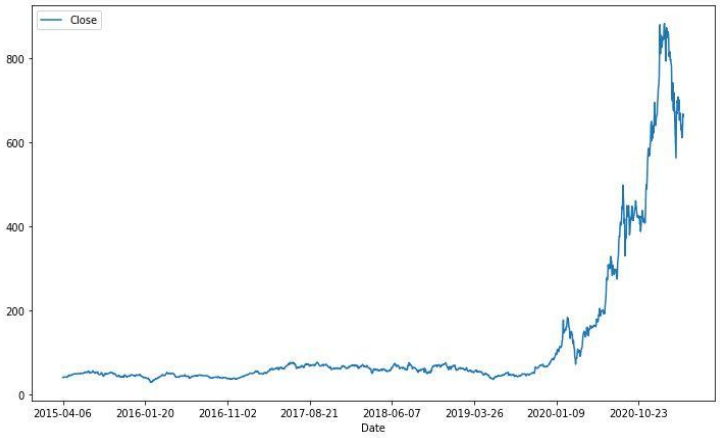
2. Dataset :

To build these models the historical values of the Tesla stock index for the period 6th April, 2015 to 1st April, 2021 have been used as the training and testing records. These Records were downloaded in the form of Comma Separated Values(CSV) from Yahoo! Finance Website

The following attributes constituted the daily records of Tesla index values :

1. Date
2. Open
3. High
4. Low
5. Close
6. Adj. Closed
7. Volume.

	Date	Open	High	Low	Close	Adj Close	Volume
0	2015-04-06	39.599998	41.549999	39.500000	40.619999	40.619999	62279000
1	2015-04-07	40.501999	41.012001	40.228001	40.650002	40.650002	21739500
2	2015-04-08	41.639999	42.180000	41.174000	41.534000	41.534000	31515500
3	2015-04-09	41.686001	42.074001	41.223999	42.018002	42.018002	19001000
4	2015-04-10	41.970001	42.330002	41.799999	42.180000	42.180000	20338500
...	...	...	...	...	...	...	...
1505	2021-03-26	641.869995	643.820007	599.890015	618.710022	618.710022	33778400
1506	2021-03-29	615.640015	616.479980	596.020020	611.289978	611.289978	28637000
1507	2021-03-30	601.750000	637.659973	591.010010	635.619995	635.619995	39432400
1508	2021-03-31	646.619995	672.000000	641.109985	667.929993	667.929993	33337300
1509	2021-04-01	688.369995	692.419983	659.419983	661.750000	661.750000	35206300



3. ARIMA :

An Auto-Regressive Integrated Moving Average, or ARIMA, is a statistical analysis model that uses time series data to either better understand the data set or to predict future trends. An autoregressive integrated moving average model is a form of regression analysis that gauges the strength of one dependent variable relative to other changing variables. The model's goal is to predict future securities or financial market moves by examining the differences between values in the series instead of actual values.

An ARIMA model can be understood by outlining each of its components as follows:

Auto-Regression (AR) :
Refers to a model that shows a changing variable that regresses on its own lagged, or prior, values.

Integrated (I) :
Represents the differencing of raw observations to allow for the time series to become stationary, i.e., data values are replaced by the difference between the data values and the previous values.

Moving Average (MA) :
Incorporates the dependency between an observation and a residual error from a moving average model applied to lagged observations.

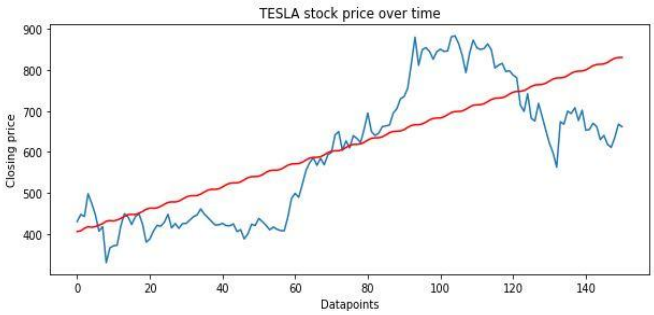
Each component functions as a parameter with a standard notation. For ARIMA models, a standard notation would be ARIMA with p, d, and q, where integer values substitute for the parameters to indicate the type of ARIMA model used. The parameters can be defined as:

p : The number of lag observations in the model; also known as the lag order.
d : The number of times that the raw observations are different; also known as the degree of differencing.
q : The size of the moving average window;also known as the order of the moving average.

3.1 Auto arima :

Initially , we took the whole dataset i.e. from 6th April 2015 to 1st April 2021 . In arima , it is all trial and error based AR-I-MA values which are p,d and q values. Therefore we implemented AUTO-ARIMA. AUTO-ARIMA tries multiple parameters for ARIMA and checks for the best parameters. The best parameter we got from auto arima is (6,2,9).

Predictions of auto arima are shown below :



3.2 Evaluation :

- Mean Squared Error : **1067.868843206765**
- r2_score : **0.5906**

3.3 Result and Observations :

The Arima model didn't give us the expected accuracy as it is an old and outdated model. After this, many researchers implemented various techniques and models for better accuracy.

4. Convolutional Neural Network (CNN) :

To make our forecasting framework more robust and accurate, we implemented convolutional neural networks (CNNs) for forecasting time series index values.

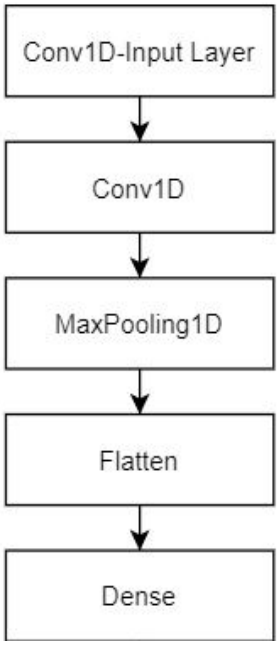
A CNN consists of two major processing layers – the convolutional layers and the pooling layers. The convolutional layers are used for reading the inputs either in the form of a two-dimensional image or as a sequence of one-dimensional data. The results of the reading are projected into a filter map that represents the interpretation of the input. The pooling layers operate on the extracted feature maps and derive the most essential features by averaging (average pool) or max computing (max pooling) operations.For extracting deep features from the input sequence, the convolution and the pooling layers may be repeated multiple times. The output from the last pooling layer is sent to a one or more dense layer(s) for extensive learning from the input data.

In this model also , we took the whole dataset i.e. from 6th April 2015 to 1st April 2021.

4.1 Data processing :

- We performed Feature Engineering and we kept only close features of the dataset.
- We used MinMaxScaler and scaled our dataset to numbers between 0 and 1.
- CNN needs the data to be in some specific format. So we created the training data in 60 timestamps and converted it into a NumPy array.

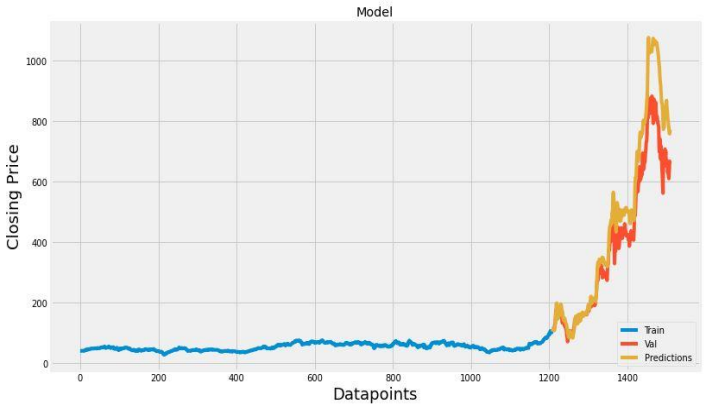
4.2 Model Architecture :

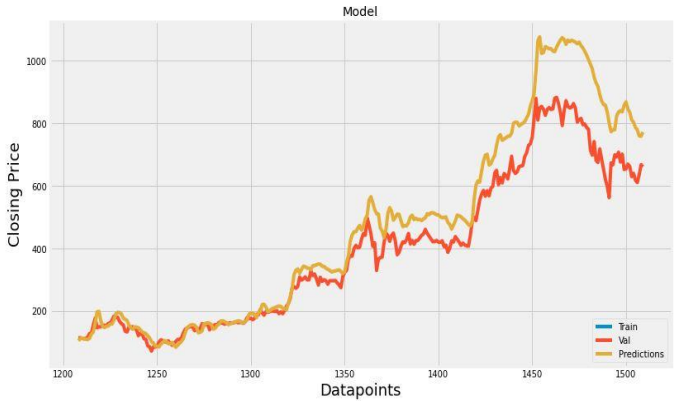


The Model is trained using the records in the training dataset and then forecasting the testing values which is the closing price. Our dataset is splitted into 80% training data and 20% testing data. The CNN model consists of a convolution layer that extracts feature maps from the input data with a kernel size. The convolution layer enables the network to read the input data in time-steps with each reading resulting in the extraction of features. The subsampling layer following the convolutional layer performs a max-pooling operation thereby reducing the size

of the feature maps. The output of the subsampling layer is then converted into a one-dimensional vector and then interpreted by a flatten layer (which is also a fully-connected layer) before the dense layer predicts the closing values. The Rectified Linear Unit (ReLU) function has been used in the convolution and fully-connected layer. The performance of the layers is optimized by using the ADAM version of the stochastic gradient descent algorithm. We trained the model using 20 epochs. For measuring the level of accuracy in forecasting of the model,we have used root mean square error (RMSE) as the metric.

4.3 Model Predictions :





	Close	Predictions
1208	113.912003	105.348541
1209	114.440002	111.492676
1210	112.963997	113.110298
1211	111.603996	112.470863
1212	113.379997	110.158150
...	...	...
1505	618.710022	787.528870
1506	611.289978	779.729370
1507	635.619995	760.513428
1508	667.929993	759.129456
1509	661.750000	772.420410

4.4 Evaluation :

- Root mean square error - **69.73**
- R2_score - **0.8143**

4.5 Result and Observations :

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_4 (Conv1D)            (None, 56, 128)           768
_________________________________________________________________
conv1d_5 (Conv1D)            (None, 52, 128)           82048
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 26, 128)           0
_________________________________________________________________
dense_2 (Dense)              (None, 26, 1)             129
=================================================================
Total params: 82,945
Trainable params: 82,945
Non-trainable params: 0
_________________________________________________________________
None
```

So first we tried without using a flatten layer.The output was in two dimensions, but it should be in one dimension. So in order to convert the pooled feature to a single column we added a flatten

layer to the neural network. So, after adding the flatten layer, the result is shown below.

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_14 (Conv1D)           (None, 56, 128)           768
_________________________________________________________________
conv1d_15 (Conv1D)           (None, 52, 128)           82048
_________________________________________________________________
max_pooling1d_7 (MaxPooling1 (None, 26, 128)           0
_________________________________________________________________
flatten_5 (Flatten)          (None, 3328)              0
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 3329
=================================================================
Total params: 86,145
Trainable params: 86,145
Non-trainable params: 0
_________________________________________________________________
None
```

Also, we tried training with different epoch values .So to avoid overfitting and underfitting we decided to go with 20 epochs as we got a good r2_score.

5. Conclusion :

So, here in this project we proposed two different approaches for the prediction of Tesla stock price index values which includes ARIMA and CNN models. The CNN model gave us good results compared to ARIMA.

We would like to discuss the challenges faced during the implementation of neural networks:

We were unaware at the start of the semester how the neural network works? We knew that this project is going to be interesting in terms of experiencing, implementing and learning some new concepts. As we started researching neural networks we came to know that it is most commonly used for processing image data. It is a deep learning technique most commonly applied to process pixel data.

But as you have given us the task with numerical data, there might be some techniques to implement it with temporal data as well. So here in this project we have tried implementing it and we concluded that CNN approach was better suited for our project.

6. References :

https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv1D
https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense
https://www.tensorflow.org/api_docs/python/tf/keras/layers/Flatten
https://www.tensorflow.org/api_docs/python/tf/keras/layers/MaxPool1D
https://www.tensorflow.org/api_docs/python/tf/keras/layers/ReLU