Machine Learning Engineer Nanodegree

Capstone Project,

Sasi Kiran Adapa,

Sunday, February 17th, 2019

I. Definition

Project Overview

Predicting the stock prices falls under the category of investment and trading, of finance market. Predicting how the stock market will perform is one of the most difficult things to do. There are so many factors involved in the prediction – physical factors vs. psychological, rational and irrational behavior, etc. All these aspects combine to make share prices volatile and very difficult to predict with a high degree of accuracy.

Stock market prediction is the act of trying to determine the future value of a company stock or other financial instrument traded on an exchange. The successful prediction of a stock's future price could yield significant profit. The efficient-market hypothesis suggests that stock prices reflect all currently available information and any price changes that are not based on newly revealed information thus are inherently unpredictable. Others disagree and those with this viewpoint possess myriad methods and technologies which purportedly allow them to gain future price information.

Thousands of companies use software to predict the movement in the stock market in order to aid their investing decisions. Primitive predicting algorithms such as a time-series, linear regression can be done with a time series prediction by leveraging python packages like scikit-learn.

Personal Motivation:

I am always interested in Finance field, from this project I am able to analyze the stock market.

The data that we are going to use for this problem is Time series data or sequential data. The Dataset we are going to use in this project is S&P 500 stock data. The dataset is available in the Kaggle site a reference link is provided below. Which contains all companies' historical data for past 5 years. The dataset contains 619K rows and 7 columns

https://www.kaggle.com/camnugent/sandp500

Here is the reference that I found while researching for this Problem

https://ieeexplore.ieee.org/document/6740438

A simple TensorFlow model is used here to predict the stock prices.

https://medium.com/mlreview/a-simple-deep-learning-model-for-stock-price-prediction-using-tensorflow-30505541d877

Problem Statement

The Main Aim of this project is to Predict the stock prices of a particular-company

using the stock market dataset. In this project I am going to using use LSTM networks to predict stock prices for a good accuracy. LSTMs are very powerful in sequence prediction problems because they're able to store past information. This is important in our case because the previous price of a stock is crucial in predicting its future price. It's important to note that there are always other factors that affect the prices of stocks, such as the political atmosphere and the market. However, we won't focus on those factors for this Project.

Metrics

I choose my evaluation metric as Mean Square Error (or) Mean Sqaure Deviation. The MSE is a measure of the quality of an estimator—it is always non-negative, and values closer to zero are better. In regression analysis, the term *mean squared error* is sometimes used to refer to the unbiased estimate of error variance: the residual sum of squares divided by the number of degrees of freedom.

II. Analysis

Data Exploration

The Dataset we are going to use in this project is S&P 500 stock data. The dataset is available in the Kaggle site a reference link is provided below. Which contains all companies' historical data for past 5 years. The dataset contains 619K rows and 7 columns

https://www.kaggle.com/camnugent/sandp500

Contents

The main contents of data are

- Date: The date on which we are predicting the stock
- Open: The value at which the stock is opened.
- High: Maximum value of the stock
- Low: Minimum value of the stock
- Close: The value at which the stock is closed
- Volume: The volume of stocks bought
- Name: The name of the company

The dataset contains stock prices of 505 companies

A sample of data is Mentioned below.

	date	open	high	low	close	volume	Name
0	2013-02-08	15.07	15.12	14.63	14.75	8407500	AAL
1	2013-02-11	14.89	15.01	14.26	14.46	8882000	AAL
2	2013-02-12	14.45	14.51	14.10	14.27	8126000	AAL
3	2013-02-13	14.30	14.94	14.25	14.66	10259500	AAL
4	2013-02-14	14.94	14.96	13.16	13.99	31879900	AAL

There are a set of Unique values in the Data they are mentioned below.

date 1259

open 49715

high 81499

low 82354

close 51151

volume 586441

Name 505

dtype: int64

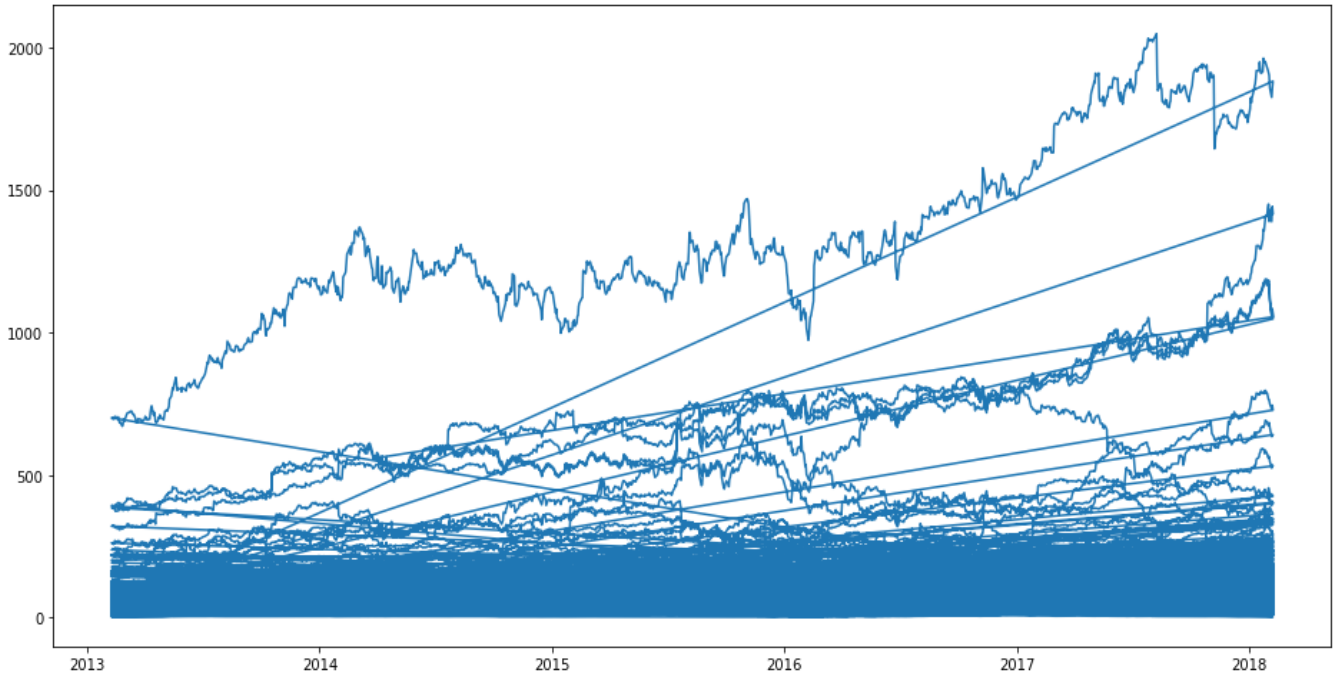
There are some missing values in the data, they are mentioned below

	date	open	high	low	close	volume	Name
82949	2017-07-26	NaN	NaN	NaN	69.0842	3	BHF
165734	2015-07-17	NaN	88.76	88.24	88.7200	2056819	DHR
165857	2016-01-12	NaN	NaN	NaN	88.5500	0	DHR
205076	2015-07-17	NaN	48.49	47.85	47.9200	1246786	ES
239832	2016-07-01	NaN	NaN	NaN	49.5400	0	FTV
434379	2015-07-17	NaN	47.31	46.83	46.9900	1229513	O
434502	2016-01-12	NaN	NaN	NaN	52.4300	0	O
478594	2015-06-09	NaN	NaN	NaN	526.0900	12135	REGN
558213	2016-04-07	NaN	NaN	NaN	41.5600	0	UA
581906	2015-05-12	NaN	NaN	NaN	124.0800	569747	VRTX
598236	2015-06-26	NaN	NaN	NaN	61.9000	100	WRK

These are the Data abnormalities present in the data we should take care of them particularly in accordance with missing values.

Exploratory Visualization

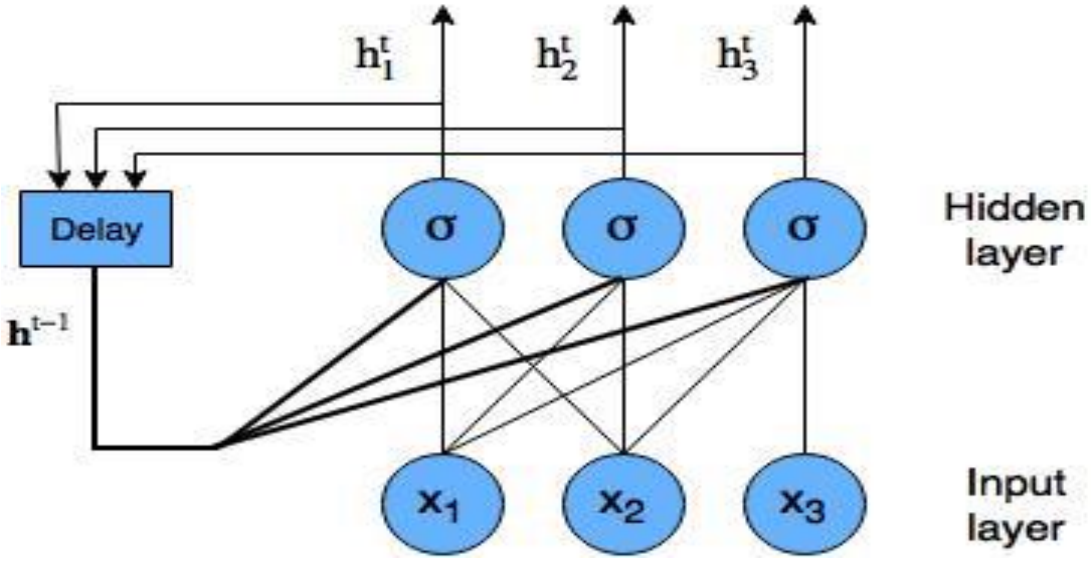
We have visualized the closing stocks of all the companies at a time the plot that we obtained is as follows.



We will be using a companies' stock to train the model and make predictions out of it.

Algorithms and Techniques

For this project I want to use going to use LSTM networks to predict stock prices. A LSTM network is a kind of recurrent neural network. A recurrent neural network is a neural network that attempts to model time or sequence dependent behavior – such as language, stock prices, electricity demand and so on. This is performed by feeding back the output of a neural network layer at time *t* to the input of the same network layer at time *t + 1*.



Recurrent neural network diagram with nodes shown

Limitations of RNNs

Recurrent Neural Networks work just fine when we are dealing with short-term dependencies. This is a well explained example regarding rnn's

The colour of the sky is _____.

RNNs turn out to be quite effective. This is because this problem has nothing to do with the context of the statement. The RNN need not remember what was said before this, or what was its meaning, all they need to know is that in most cases the sky is blue. Thus, the prediction would be:

The colour of the sky is blue.

It can't predict large sentences this is because the RNN needs to remember this context. The relevant information may be separated from the point where it is

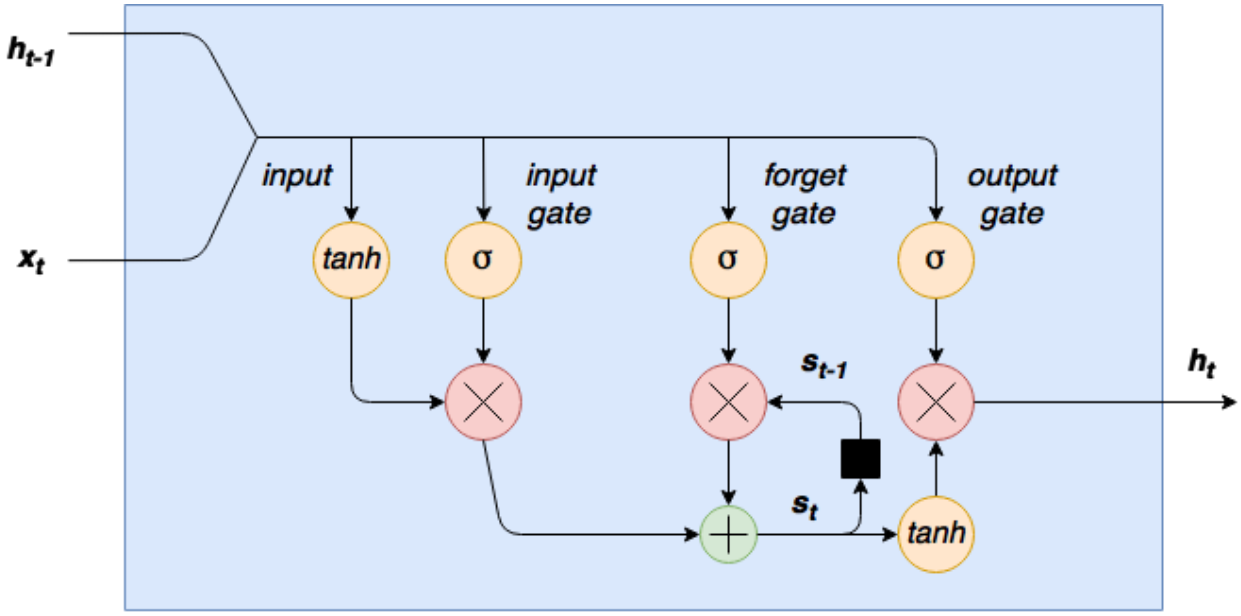
needed, by a huge load of irrelevant data. This is where a Recurrent Neural Network fails!

The reason behind this is the problem of Vanishing Gradient. In order to understand this, you'll need to have some knowledge about how a feed-forward neural network learns. We know that for a conventional feed-forward neural network, the weight updating that is applied on a particular layer is a multiple of the learning rate, the error term from the previous layer and the input to that layer. Thus, the error term for a particular layer is somewhere a product of all previous layers' errors. When dealing with activation functions like the sigmoid function, the small values of its derivatives (occurring in the error function) gets multiplied multiple times as we move towards the starting layers. As a result of this, the gradient almost vanishes as we move towards the starting layers, and it becomes difficult to train these layers.

This is the reason why we choose LSTM networks over rnn's

LSTM networks

An LSTM network is a recurrent neural network that has LSTM cell blocks in place of our standard neural network layers. These cells have various components called the input gate, the forget gate and the output gate – these will be explained more fully later. Here is a graphical representation of the LSTM cell:

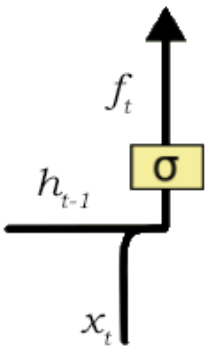


Architecture of LSTM

1. Forget Gate
2. Input Gate
3. Output Gate

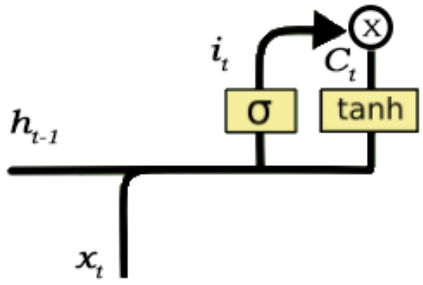
Forget Gate

A forget gate is responsible for removing information from the cell state. The information that is no longer required for the LSTM to understand things or the information that is of less importance is removed via multiplication of a filter. This is required for optimizing the performance of the LSTM network.



Input Gate

The input gate is responsible for the addition of information to the cell state.



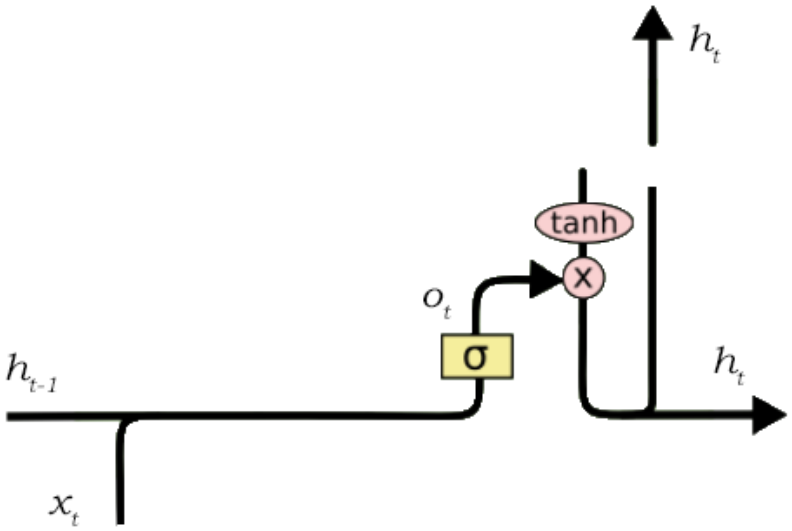
Regulating what values need to be added to the cell state by involving a sigmoid function. This is basically very similar to the forget gate and acts as a filter for all the information from h_t-1 and x_t.

1. Creating a vector containing all possible values that can be added (as perceived from h_t-1 and x_t) to the cell state. This is done using the **tanh** function, which outputs values from -1 to +1.
2. Multiplying the value of the regulatory filter (the sigmoid gate) to the created vector (the tanh function) and then adding this useful information to the cell state via addition operation.

Once this three-step process is done with, we ensure that only that information is added to the cell state that is *important* and is not *redundant.*

Output Gate

This job of selecting useful information from the current cell state and showing it out as an output is done via the output gate.

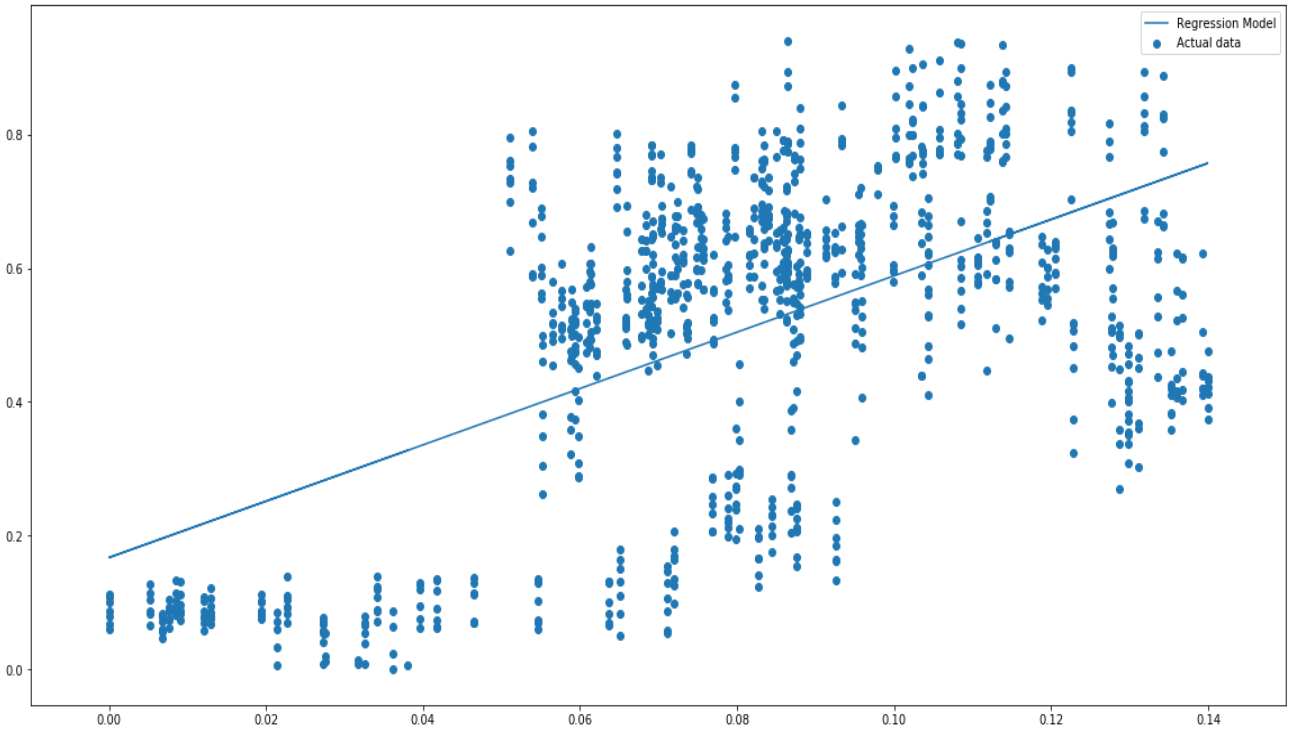


The functioning of an output gate can again be broken down to three steps:

1. Creating a vector after applying **tanh** function to the cell state, thereby scaling the values to the range -1 to +1.
2. Making a filter using the values of h_t-1 and x_t, such that it can regulate the values that need to be output from the vector created above. This filter again employs a sigmoid function.
3. Multiplying the value of this regulatory filter to the vector created in step 1, and sending it out as a output and also to the hidden state of the next cell.

Benchmark

I am going to use Linear Regression as my bench mark model. By using LSTM's, I will try to improve my accuracy. I have Plotted the bench mark model for this sequential data



The score for the linear regression model is 0.3 which is really-low

III. Methodology

Data Preprocessing

In order to build the model, we must do some pre-processing for a better prediction rate and accuracy.

In this problem we are going to use MinMaxScalar from sklearn package and apply on for a stock.

Implementation

I have implemented the LSTM network over the given data with the mentioned specifications

1. single hidden layer
2. loss function mse
3. optimizer Adam.

Adam

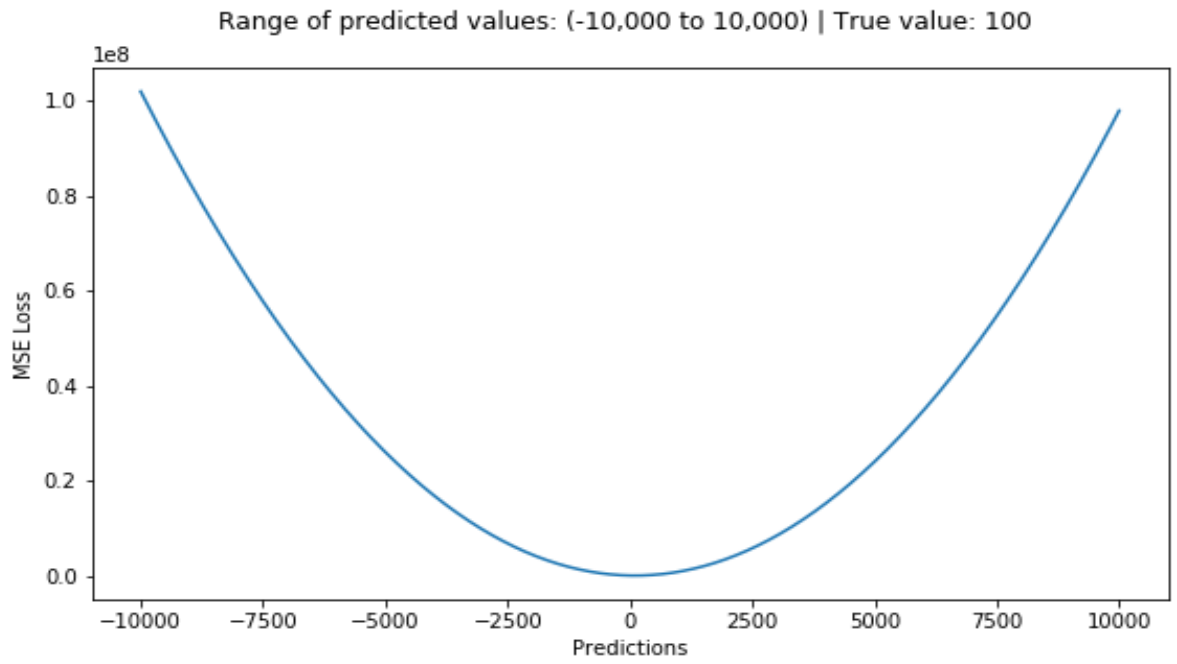
Adam is an optimization algorithm that can used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data.

MSE

Mean Square Error (MSE) is the most commonly used regression loss function. MSE is the sum of squared distances between our target variable and predicted values.

$$MSE = \frac{\sum_{i=1}^{n} (y_i - y_i^p)^2}{n}$$

Below is a plot of an MSE function where the true target value is 100, and the predicted values range between -10,000 to 10,000. The MSE loss (Y-axis) reaches its minimum value at prediction (X-axis) = 100. The range is 0 to ∞.



Plot of MSE Loss (Y-axis) vs. Predictions (X-axis)

We have defined a function Named `processData(data, lb)`

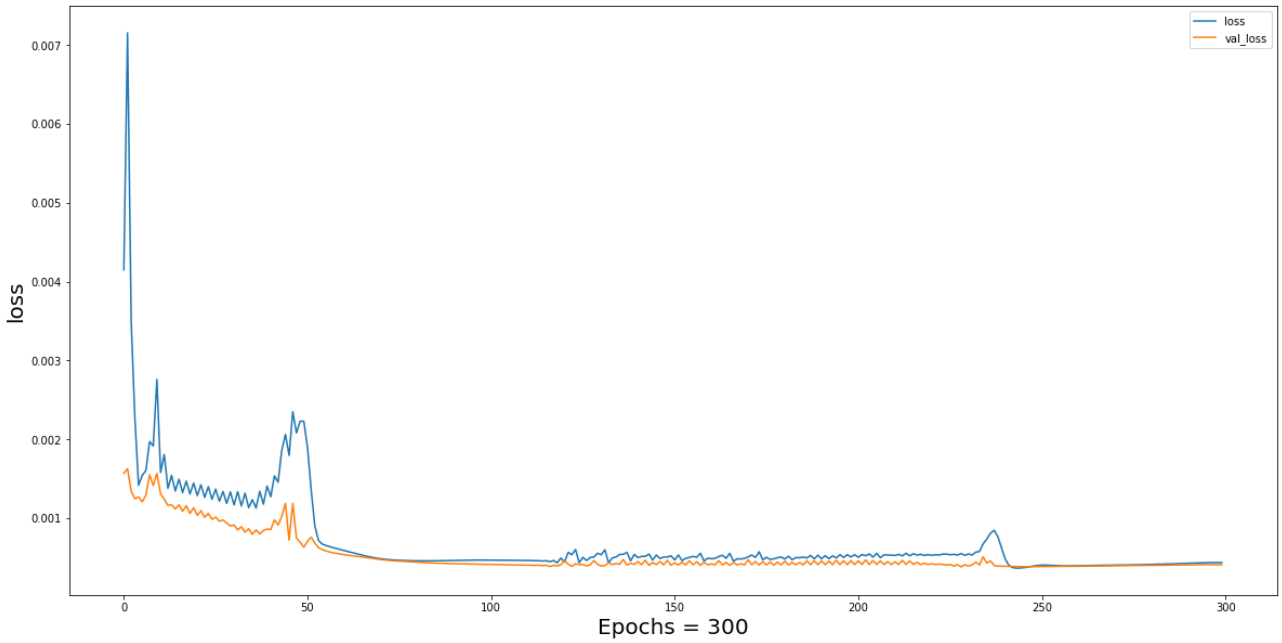
`Data:` Takes the stocks data of a particular company

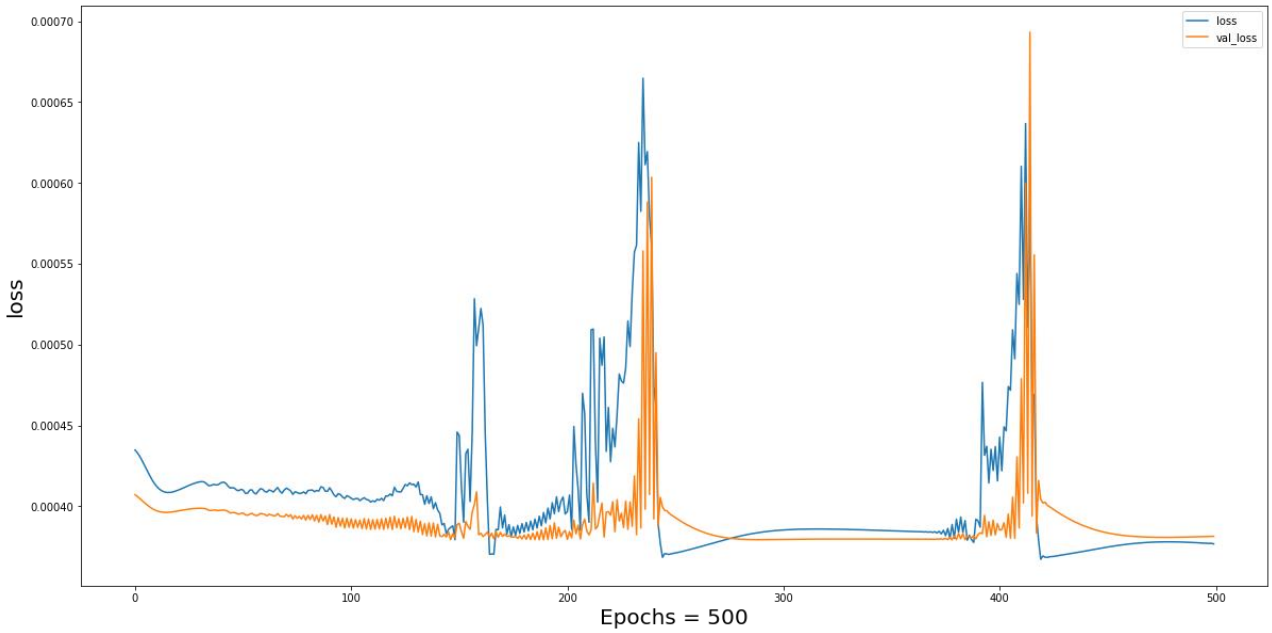
`lb:` Takes an integer value

Refinement

Using trial and error method we can find number of epochs best suitable for the model iterations.

Here I have tried two version of epochs they are 300 and 500, the figures are given below



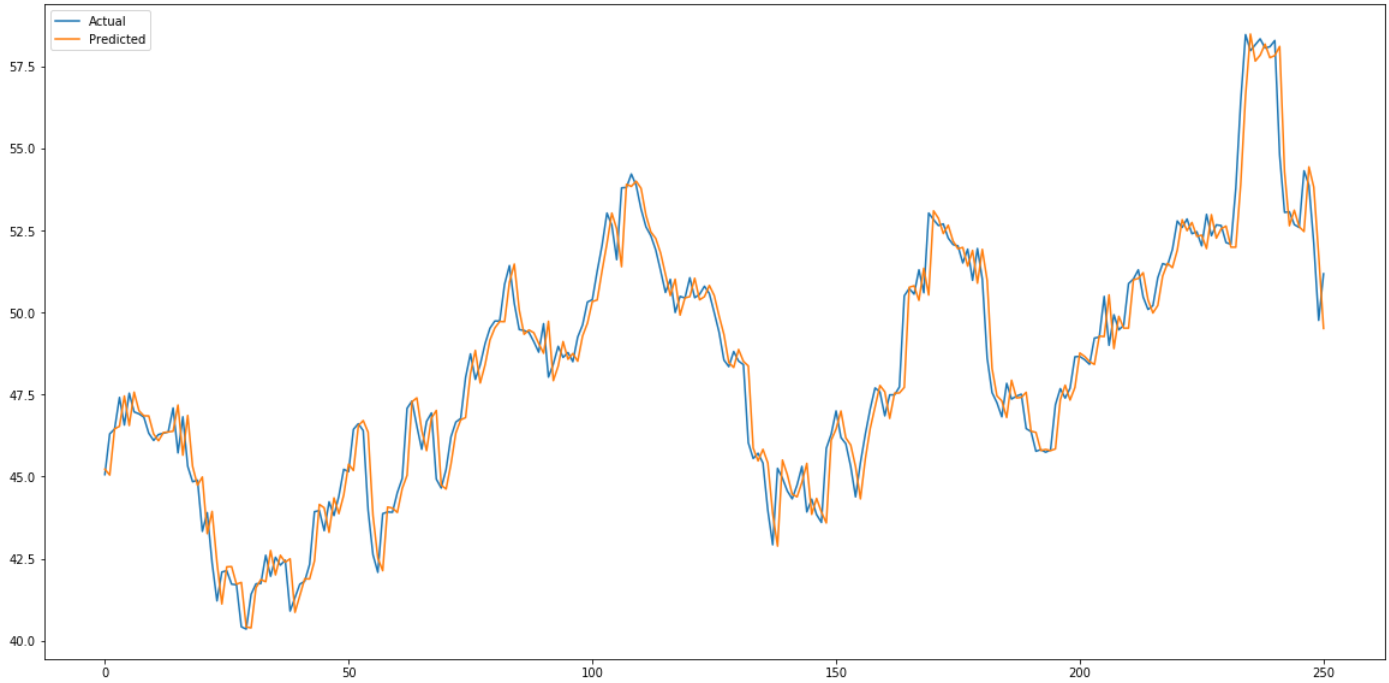


It is evident that epochs=300 best suits for the model.

IV. Results

Model Evaluation and Validation

The model that we have built is doing well with the test data, here is the plot that we get using the model.

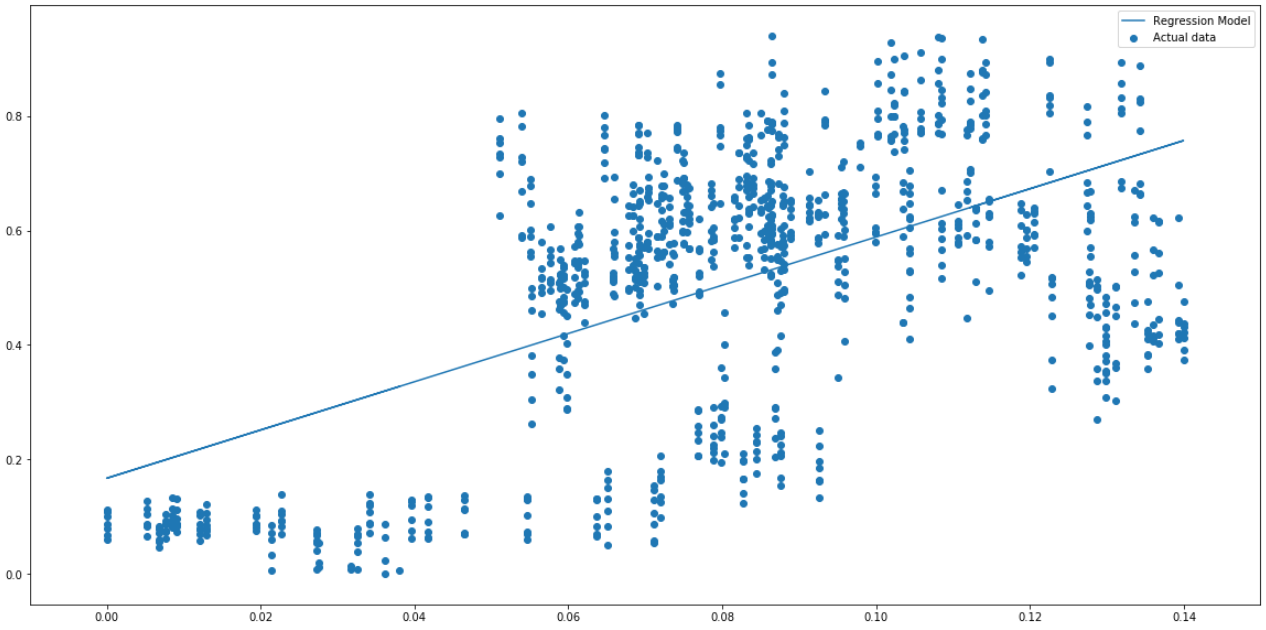


As we can see the model that we have built did really well in unseen data, the predicted stock prizes are nearly equal to actual prices.

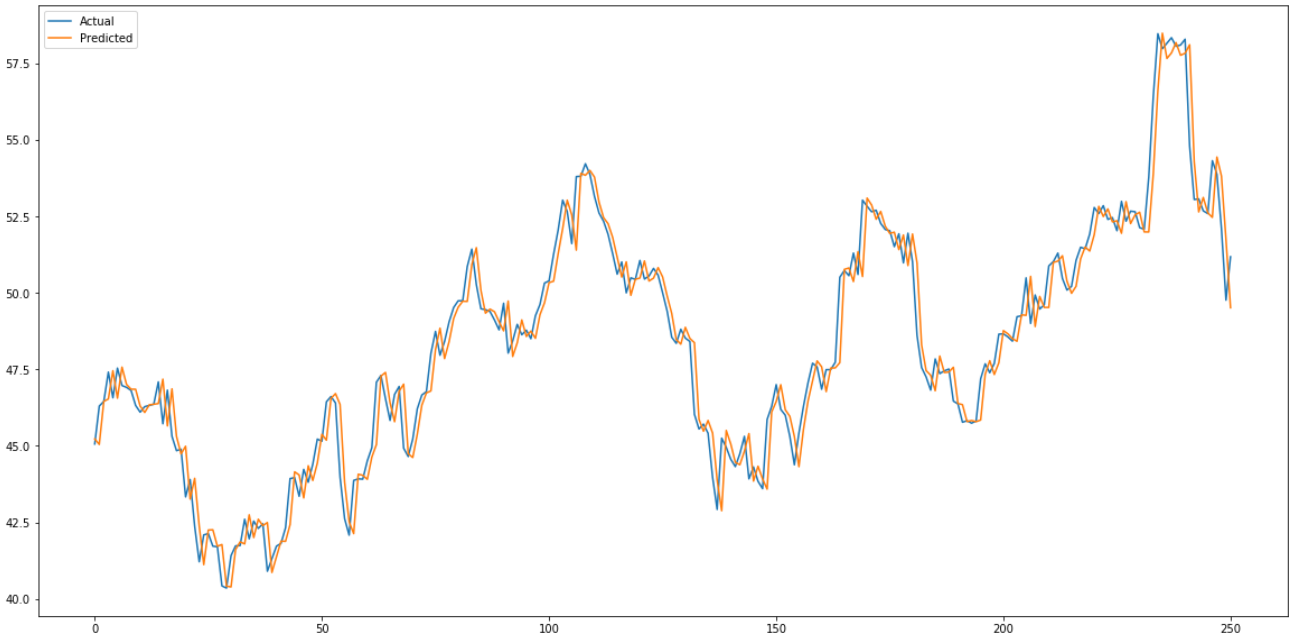
The model is robust and reliable. The results from the model can be trusted.

Justification

The benchmark model did very poorly with the data. The lstm model that we have built that did well when compared with the test data. This model's results are accurate then the benchmark model. When we compare the results of benchmark model with the lstm model. It is evident that lstm model did really amazing.



Benchmark model

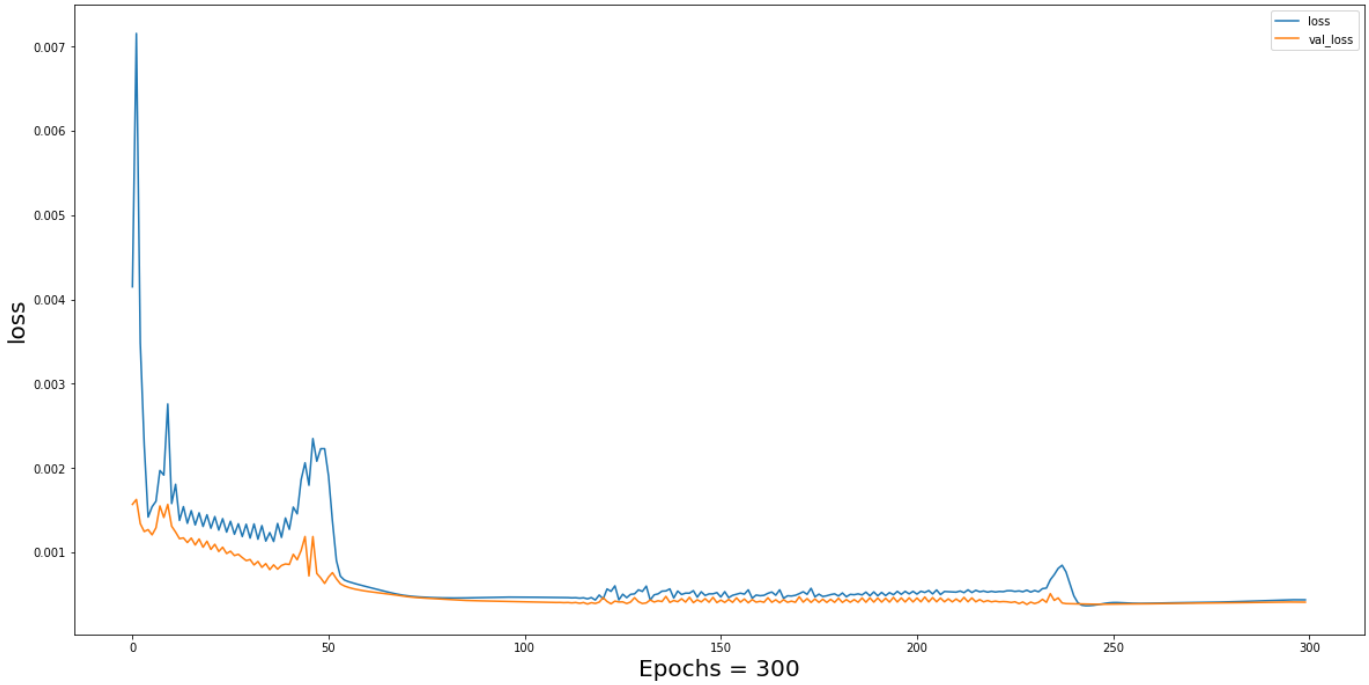


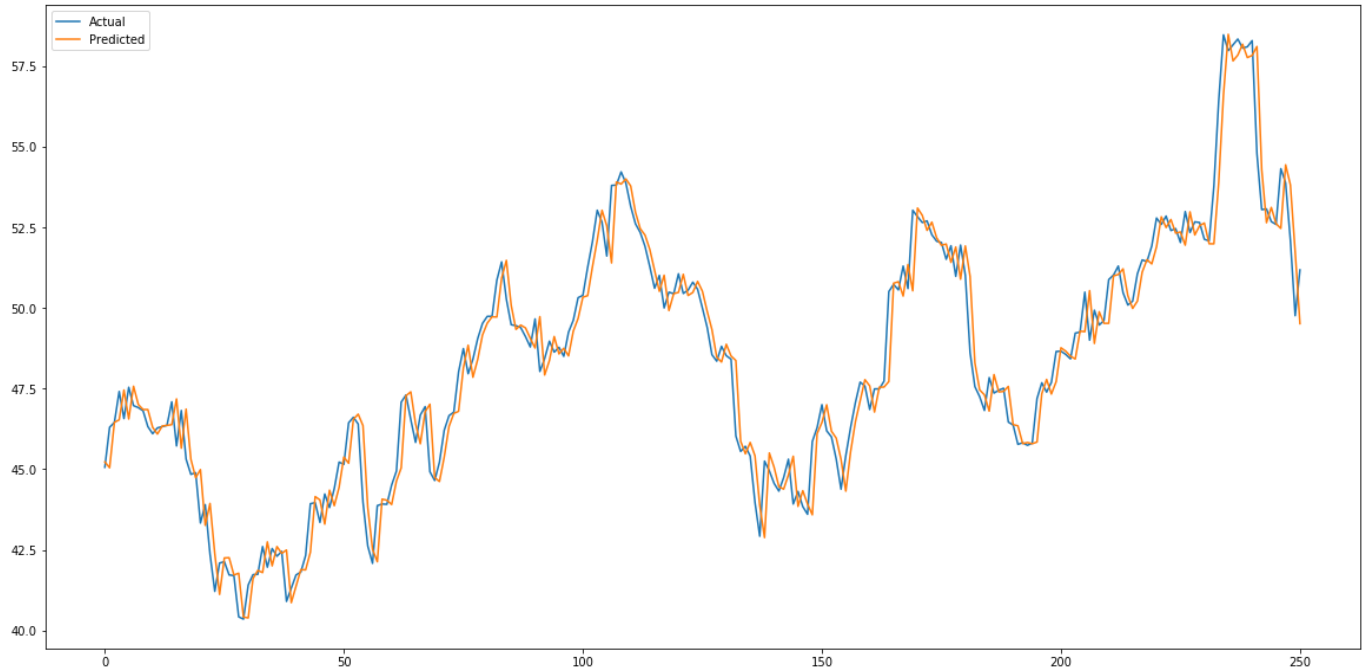
Lstm model

V. Conclusion

Free-Form Visualization

I have already used the relevant plots in the analysis part. I thoroughly analyzed and discussed about these features of data.





These are the Plots that we get when plot the related attributes.

Reflection

I took problem because I am always interested in finance field. It is a bit hard for me in the beginning but as I have learned how to solve these kinds of problems.

I thought how we can predict the stocks prices, but after so much of research I found out that lstm's can be used to solve this problem. I know the difficulties in this problem using the lstm network. The model I built is perfect fit for this problem it reached my expectations.

Improvement

I used LSTM with a single layer, for this problem, in order to improve we can use lstm with a greater number of layers, but to this problem as simple as possible I used a single layer of lstm for large data sets we can use more than one layer of lstm network