

x

-

(<http://play.google.com/store/apps/details?id=com.analyticsvidhya.android>)



(<https://www.analyticsvidhya.com/blog/>)



([https://datahack.analyticsvidhya.com/contest/american-](https://datahack.analyticsvidhya.com/contest/american-express-amexpert-2018/?utm_source=AVtopblogBanner)

[express-amexpert-2018/?utm\\_source=AVtopblogBanner](https://datahack.analyticsvidhya.com/contest/american-express-amexpert-2018/?utm_source=AVtopblogBanner))

[DEEP LEARNING \(HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/CATEGORY/DEEP-LEARNING/\)](https://www.analyticsvidhya.com/blog/category/deep-learning/)

[MACHINE LEARNING \(HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/CATEGORY/MACHINE-LEARNING/\)](https://www.analyticsvidhya.com/blog/category/machine-learning/)

[PYTHON \(HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/CATEGORY/PYTHON-2/\)](https://www.analyticsvidhya.com/blog/category/python-2/)

[TIME SERIES \(HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/CATEGORY/TIME-SERIES/\)](https://www.analyticsvidhya.com/blog/category/time-series/)

## Stock Prices Prediction Using Machine Learning and Deep Learning Techniques (with Python codes)

[AISHWARYA SINGH \(HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/AUTHOR/AISHWARYASINGH/\)](https://www.analyticsvidhya.com/blog/author/aishwaryasingh/), OCTOBER 25, 2018



**Bussiness only prices**

[amazon.in/business](https://amazon.in/business)

### Introduction

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. physhological, rational and irrational behaviour, etc. All these aspects combine to make share prices volatile and very difficult to predict with a high degree of accuracy. ^

Can we use machine learning as a game changer in this domain? Using features like the latest announcements about an organization, their quarterly revenue results, etc., machine learning techniques have the potential to unearth patterns and insights we didn't see before, and these can be used to make unerringly accurate predictions.



In this article, we will work with historical data about the stock prices of a publicly listed company. We will implement a mix of machine learning algorithms to predict the future stock price of this company, starting with simple algorithms like averaging and linear regression, and then moving on to advanced techniques like Auto ARIMA and LSTM.

The core idea behind this article is to showcase how these algorithms are implemented, so I will briefly describe the technique and provide relevant links to brush up on the concepts as and when necessary. In case you're a newcomer to the world of time series, I suggest going through the following articles first:

- [A comprehensive beginner's guide to create a Time Series Forecast](https://www.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes-python/) (<https://www.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes-python/>)
- [A Complete Tutorial on Time Series Modeling](https://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/) (<https://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/>)

## Table of Contents

1. Understanding the Problem Statement
2. Moving Average
3. Linear Regression
4. k-Nearest Neighbors
5. Auto ARIMA
6. Prophet
7. Long Short Term Memory (LSTM)

## Understanding the Problem Statement

We'll dive into the implementation part of this article soon, but first it's important to establish what we're aiming to solve. Broadly, stock market analysis is divided into two parts – Fundamental Analysis and Technical Analysis.

- Fundamental Analysis involves analyzing the company's future profitability on the basis of its current business environment and financial performance.
- Technical Analysis, on the other hand, includes reading the charts and using statistical figures to identify the trends in the stock market.

As you might have guessed, our focus will be on the technical analysis part. We'll be using a dataset from Quandl (<https://www.quandl.com/>) (you can find historical data for various stocks here) and for this particular project, I have used the data for 'Tata Global Beverages' (<https://www.quandl.com/data/NSE/TATAGLOBAL-Tata-Global-Beverages-Limited>). Time to dive in!

We will first load the dataset and define the target variable for the problem:

```

#import packages
import pandas as pd
import numpy as np

#to plot within notebook
import matplotlib.pyplot as plt
%matplotlib inline

#setting figure size
from matplotlib.pylab import rcParams
rcParams['figure.figsize'] = 20,10

#for normalizing data
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))

#read the file
df = pd.read_csv('NSE-TATAGLOBAL(1).csv')

#print the head
df.head()

```

	Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
0	2018-10-08	208.00	222.25	206.85	216.00	215.15	4642146.0	10062.83
1	2018-10-05	217.00	218.60	205.90	210.25	209.20	3519515.0	7407.06
2	2018-10-04	223.50	227.80	216.15	217.25	218.20	1728786.0	3815.79
3	2018-10-03	230.00	237.50	225.75	226.45	227.60	1708590.0	3960.27
4	2018-10-01	234.55	234.60	221.05	230.30	230.90	1534749.0	3486.05

There are multiple variables in the dataset – date, open, high, low, last, close, total\_trade\_quantity, and turnover.

- The columns *Open* and *Close* represent the starting and final price at which the stock is traded on a particular day.
- *High*, *Low* and *Last* represent the maximum, minimum, and last price of the share for the day.

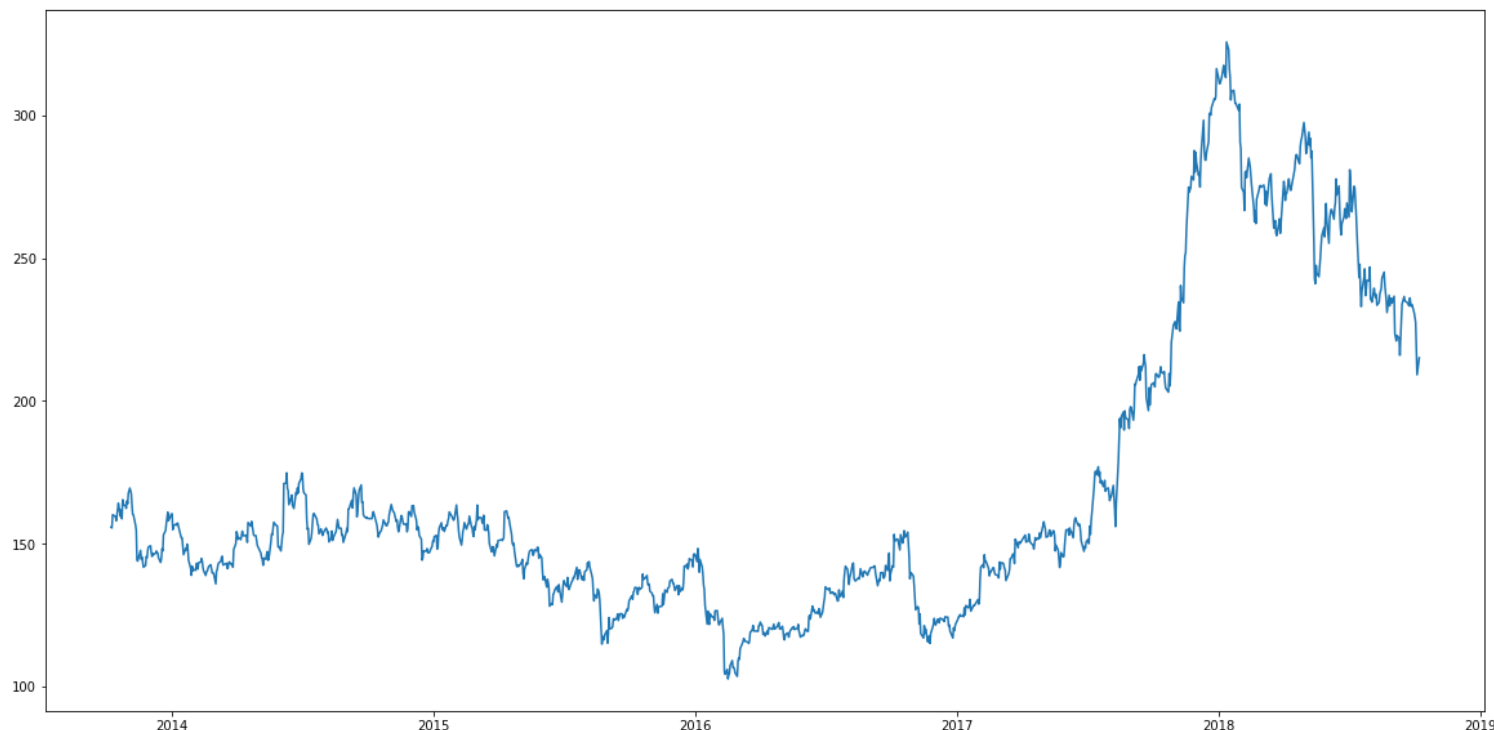
- *Total Trade Quantity* is the number of shares bought or sold in the day and *Turnover (Lacs)* is the turnover of the particular company on a given date.

Another important thing to note is that the market is closed on weekends and public holidays. Notice the above table again, some date values are missing – 2/10/2018, 6/10/2018, 7/10/2018. Of these dates, 2nd is a national holiday while 6th and 7th fall on a weekend.

The profit or loss calculation is usually determined by the closing price of a stock for the day, hence we will consider the closing price as the target variable. Let's plot the target variable to understand how it's shaping up in our data:

```
#setting index as date
df['Date'] = pd.to_datetime(df.Date,format='%Y-%m-%d')
df.index = df['Date']

#plot
plt.figure(figsize=(16,8))
plt.plot(df['Close'], label='Close Price history')
```



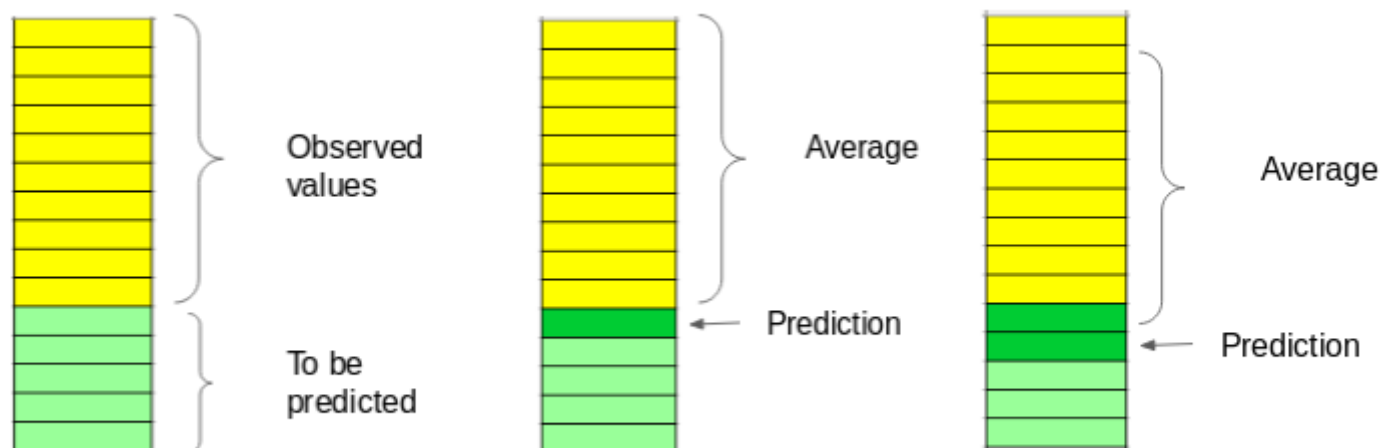
In the upcoming sections, we will explore these variables and use different techniques to predict the daily closing price of the stock.

## Moving Average

### Introduction

'Average' is easily one of the most common things we use in our day-to-day lives. For instance, calculating the average marks to determine overall performance, or finding the average temperature of the past few days to get an idea about today's temperature – these all are routine tasks we do on a regular basis. So this is a good starting point to use on our dataset for making predictions.

The predicted closing price for each day will be the average of a set of previously observed values. Instead of using the simple average, we will be using the moving average technique which uses the latest set of values for each prediction. In other words, for each subsequent step, the predicted values are taken into consideration while removing the oldest observed value from the set. Here is a simple figure that will help you understand this with more clarity.



We will implement this technique on our dataset. The first step is to create a dataframe that contains only the *Date* and *Close* price columns, then split it into train and validation sets to verify our predictions.

### Implementation

```
#creating dataframe with date and the target variable
data = df.sort_index(ascending=True, axis=0)
new_data = pd.DataFrame(index=range(0,len(df)),columns=['Date', 'Close'])

for i in range(0,len(data)):
    new_data['Date'][i] = data['Date'][i]
    new_data['Close'][i] = data['Close'][i]
```

While splitting the data into train and validation, we cannot use random splitting since that will destroy the time component. So here I have set the last year's data into validation and the 4 years' data before that into train.

```
#splitting into train and validation
train = new_data[:987]
valid = new_data[987:]
```

```
new_data.shape, train.shape, valid.shape
((1235, 2), (987, 2), (248, 2))
```

```
train['Date'].min(), train['Date'].max(), valid['Date'].min(), valid['Date'].max()

(Timestamp('2013-10-08 00:00:00'),
Timestamp('2017-10-06 00:00:00'),
Timestamp('2017-10-09 00:00:00'),
Timestamp('2018-10-08 00:00:00'))
```

The next step is to create predictions for the validation set and check the RMSE using the actual values.

```
#make predictions
preds = []
for i in range(0,248):
    a = train['Close'][len(train)-248+i:].sum() + sum(preds)
    b = a/248
    preds.append(b)
```

## Results

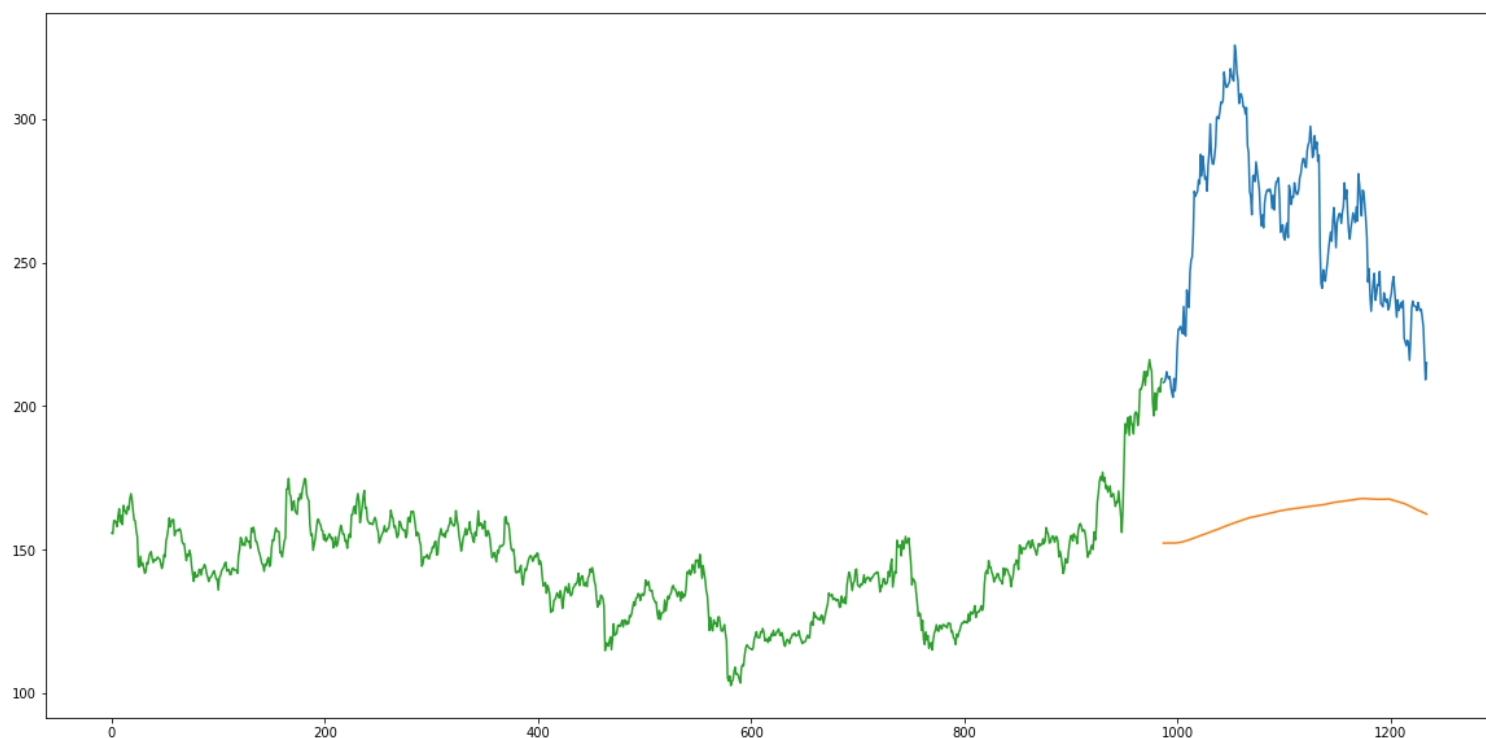
```
#calculate rmse
rms=np.sqrt(np.mean(np.power((np.array(valid['Close'])-preds),2)))
rms
```

```
104.51415465984348
```

Just checking the RMSE does not help us in understanding how the model performed. Let's visualize this to get a more intuitive understanding. So here is a plot of the predicted values along with the actual values.

```
#plot
valid['Predictions'] = 0
valid['Predictions'] = preds
plt.plot(train['Close'])
plt.plot(valid[['Close', 'Predictions']])
```





## Inference

The RMSE value is close to 105 but the results are not very promising (as you can gather from the plot). The predicted values are of the same range as the observed values in the train set (there is an increasing trend initially and then a slow decrease).

In the next section, we will look at two commonly used machine learning techniques – Linear Regression and kNN, and see how they perform on our stock market data.

## Linear Regression

### Introduction

The most basic machine learning algorithm that can be implemented on this data is linear regression. The linear regression model returns an equation that determines the relationship between the independent variables and the dependent variable.

The equation for linear regression can be written as:

$$Y = \theta_1 X_1 + \theta_2 X_2 + \dots \theta_n X_n$$

Here,  $x_1, x_2, \dots, x_n$  represent the independent variables while the coefficients  $\theta_1, \theta_2, \dots, \theta_n$  represent the weights. You can refer to the following article to study linear regression in more detail:

- [A comprehensive beginners guide for Linear, Ridge and Lasso Regression](https://www.analyticsvidhya.com/blog/2017/06/a-comprehensive-guide-for-linear-ridge-and-lasso-regression/)  
(<https://www.analyticsvidhya.com/blog/2017/06/a-comprehensive-guide-for-linear-ridge-and-lasso-regression/>).

For our problem statement, we do not have a set of independent variables. We have only the dates instead. Let us use the date column to extract features like – day, month, year, mon/fri etc. and then fit a linear regression model.

## Implementation

We will first sort the dataset in ascending order and then create a separate dataset so that any new feature created does not affect the original data.

```
#setting index as date values
df['Date'] = pd.to_datetime(df.Date,format='%Y-%m-%d')
df.index = df['Date']

#sorting
data = df.sort_index(ascending=True, axis=0)

#creating a separate dataset
new_data = pd.DataFrame(index=range(0,len(df)),columns=['Date', 'Close'])

for i in range(0,len(data)):
    new_data['Date'][i] = data['Date'][i]
    new_data['Close'][i] = data['Close'][i]
```

```
#create features
from fastai.structured import add_datepart
add_datepart(new_data, 'Date')
new_data.drop('Elapsed', axis=1, inplace=True) #elapsed will be the time stamp
```

This creates features such as:

'Year', 'Month', 'Week', 'Day', 'Dayofweek', 'Dayofyear', 'Is\_month\_end', 'Is\_month\_start', 'Is\_quarter\_end', 'Is\_quarter\_start', 'Is\_year\_end', and 'Is\_year\_start'.

*Note: I have used add\_datepart from fastai library. If you do not have it installed, you can simply use the command **pip install fastai**. Otherwise, you can create these feature using simple for loops in python. I have shown an example below.*

Apart from this, we can add our own set of features that we believe would be relevant for the predictions. For instance, my hypothesis is that the first and last days of the week could potentially affect the closing price of the stock far more than the other days. So I have created a feature that identifies whether a given day is Monday/Friday or Tuesday/Wednesday/Thursday. This can be done using the following lines of code:

```
new_data['mon_fri'] = 0
for i in range(0,len(new_data)):
    if (new_data['Dayofweek'][i] == 0 or new_data['Dayofweek'][i] == 4):
        new_data['mon_fri'][i] = 1
    else:
        new_data['mon_fri'][i] = 0
```

If the day of week is equal to 0 or 4, the column value will be 1, otherwise 0. Similarly, you can create multiple features. *If you have some ideas for features that can be helpful in predicting stock price, please share in the comment section.*

We will now split the data into train and validation sets to check the performance of the model.

```
#split into train and validation
train = new_data[:987]
valid = new_data[987:]

x_train = train.drop('Close', axis=1)
y_train = train['Close']
x_valid = valid.drop('Close', axis=1)
y_valid = valid['Close']

#implement linear regression
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x_train,y_train)
```

## Results

```
#make predictions and find the rmse
preds = model.predict(x_valid)
rms=np.sqrt(np.mean(np.power((np.array(y_valid)-np.array(preds)),2)))
rms
```

```
121.16291596523156
```

The RMSE value is higher than the previous technique, which clearly shows that linear regression has performed poorly. Let's look at the plot and understand why linear regression has not done well:

```
#plot
valid['Predictions'] = 0
valid['Predictions'] = preds

valid.index = new_data[987:].index
train.index = new_data[:987].index

plt.plot(train['Close'])
plt.plot(valid[['Close', 'Predictions']])
```



## Inference

Linear regression is a simple technique and quite easy to interpret, but there are a few obvious disadvantages. One problem in using regression algorithms is that the model overfits to the date and month column. Instead of taking into account the previous values from the point of prediction, the model will consider the value from the same *date* a month ago, or the same *date/month* a year ago.

As seen from the plot above, for January 2016 and January 2017, there was a drop in the stock price. The model has predicted the same for January 2018. A linear regression technique can perform well for problems such as Big Mart sales (<https://datahack.analyticsvidhya.com/contest/practice-problem-big-mart-sales-iii/>) where the

independent features are useful for determining the target value.

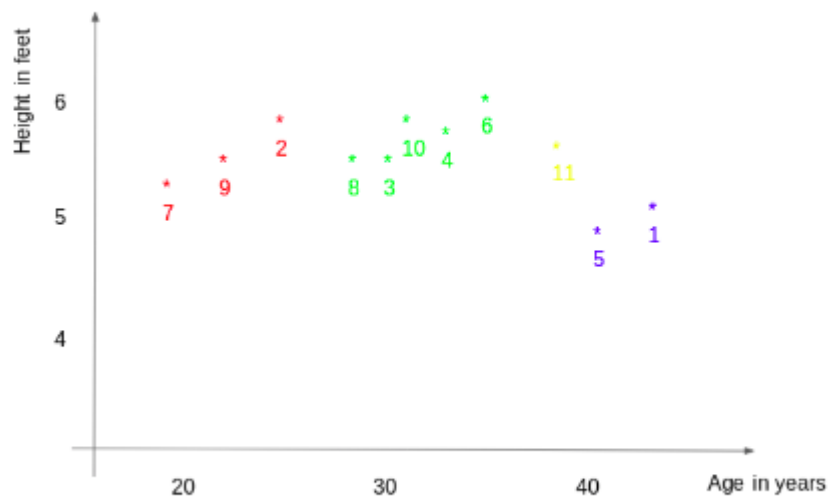
## k-Nearest Neighbours

### Introduction

Another interesting ML algorithm that one can use here is kNN (k nearest neighbours). Based on the independent variables, kNN finds the similarity between new data points and old data points. Let me explain this with a simple example.

Consider the height and age for 11 people. On the basis of given features ('Age' and 'Height'), the table can be represented in a graphical format as shown below:

ID	Age	Height	Weight
1	45	5	77
2	26	5.11	47
3	30	5.6	55
4	34	5.9	59
5	40	4.8	72
6	36	5.8	60
7	19	5.3	40
8	28	5.8	60
9	23	5.5	45
10	32	5.6	58
11	38	5.5	?



To determine the weight for ID #11, kNN considers the weight of the nearest neighbors of this ID. The weight of ID #11 is predicted to be the average of its neighbors. If we consider three neighbors ( $k=3$ ) for now, the weight for ID#11 would be  $= (77+72+60)/3 = 69.66$  kg.

ID	Height	Age	Weight
1	5	45	77
5	4.8	40	72
6	5.8	36	60

For a detailed understanding of kNN, you can refer to the following articles:

- [Introduction to k-Nearest Neighbors: Simplified](https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/)  
(<https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/>)
- [A Practical Introduction to K-Nearest Neighbors Algorithm for Regression](https://www.analyticsvidhya.com/blog/2018/08/k-nearest-neighbor-introduction-regression-python/)  
(<https://www.analyticsvidhya.com/blog/2018/08/k-nearest-neighbor-introduction-regression-python/>)

## Implementation

```
#importing libraries
from sklearn import neighbors
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))
```

Using the same train and validation set from the last section:

```
#scaling data
x_train_scaled = scaler.fit_transform(x_train)
x_train = pd.DataFrame(x_train_scaled)
x_valid_scaled = scaler.fit_transform(x_valid)
x_valid = pd.DataFrame(x_valid_scaled)

#using gridsearch to find the best parameter
params = {'n_neighbors':[2,3,4,5,6,7,8,9]}
knn = neighbors.KNeighborsRegressor()
model = GridSearchCV(knn, params, cv=5)

#fit the model and make predictions
model.fit(x_train,y_train)
preds = model.predict(x_valid)
```

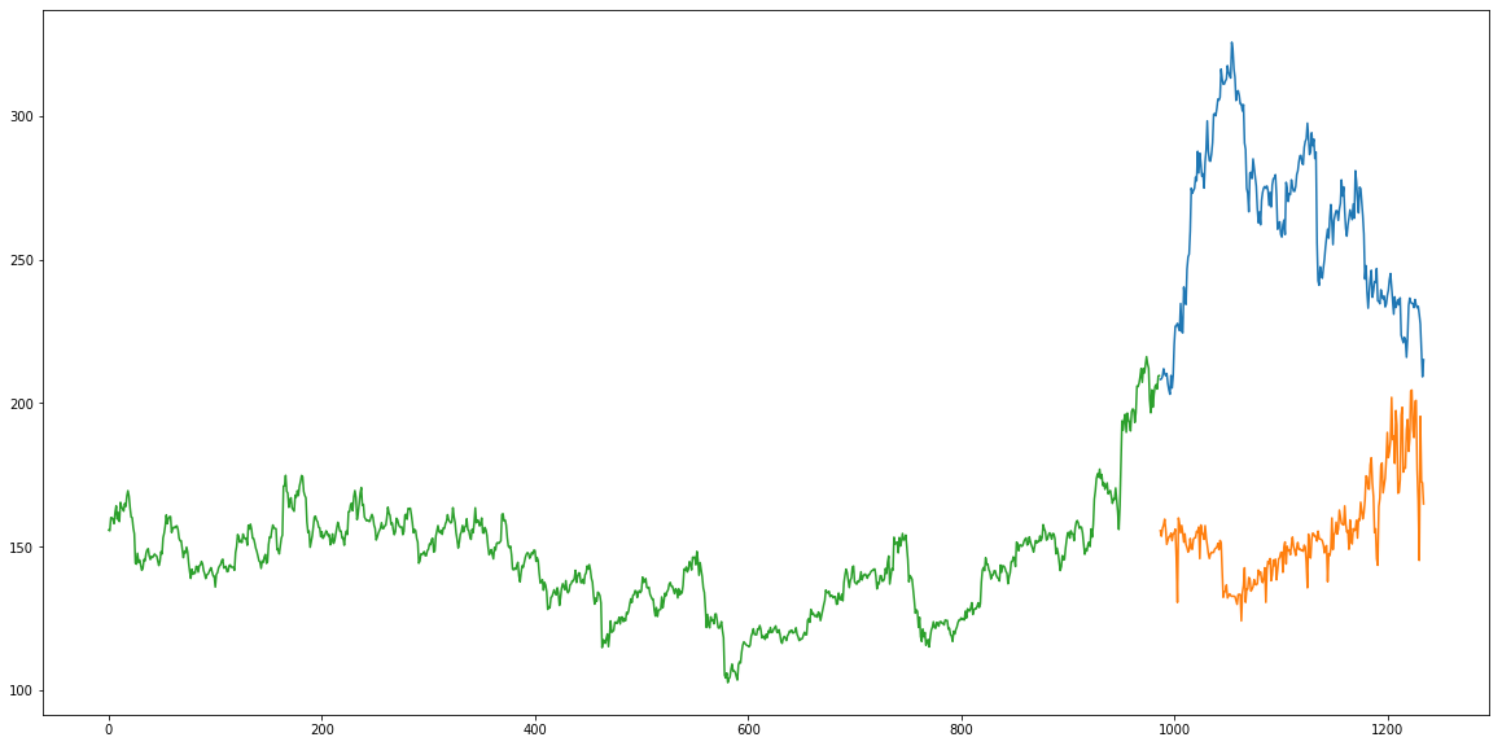
## Results

```
#rmse
rms=np.sqrt(np.mean(np.power((np.array(y_valid)-np.array(preds)),2)))
rms
```

```
115.17086550026721
```

There is not a huge difference in the RMSE value, but a plot for the predicted and actual values should provide a more clear understanding.

```
#plot
valid['Predictions'] = 0
valid['Predictions'] = preds
plt.plot(valid[['Close', 'Predictions']])
plt.plot(train['Close'])
```



## Inference



The RMSE value is almost similar to the linear regression model and the plot shows the same pattern. Like linear regression, kNN also identified a drop in January 2018 since that has been the pattern for the past years. We can safely say that regression algorithms have not performed well on this dataset.

Let's go ahead and look at some time series forecasting techniques to find out how they perform when faced with this stock prices prediction challenge.

## Auto ARIMA

### Introduction

ARIMA is a very popular statistical method for time series forecasting. ARIMA models take into account the past values to predict the future values. There are three important parameters in ARIMA:

- p (past values used for forecasting the next value)
- q (past forecast errors used to predict the future values)
- d (order of differencing)

Parameter tuning for ARIMA consumes a lot of time. So we will use auto ARIMA which automatically selects the best combination of (p,q,d) that provides the least error. To read more about how auto ARIMA works, refer to this article:

- [Build High Performance Time Series Models using Auto ARIMA](https://www.analyticsvidhya.com/blog/2018/08/auto-arima-time-series-modeling-python-r/)  
(<https://www.analyticsvidhya.com/blog/2018/08/auto-arima-time-series-modeling-python-r/>).

### Implementation

```
from pyramid.arima import auto_arima

data = df.sort_index(ascending=True, axis=0)

train = data[:987]
valid = data[987:]

training = train['Close']
validation = valid['Close']

model = auto_arima(training, start_p=1, start_q=1,max_p=3, max_q=3, m=12,start_P=0, seasonal=True,d=
1, D=1, trace=True,error_action='ignore',suppress_warnings=True)
model.fit(training)

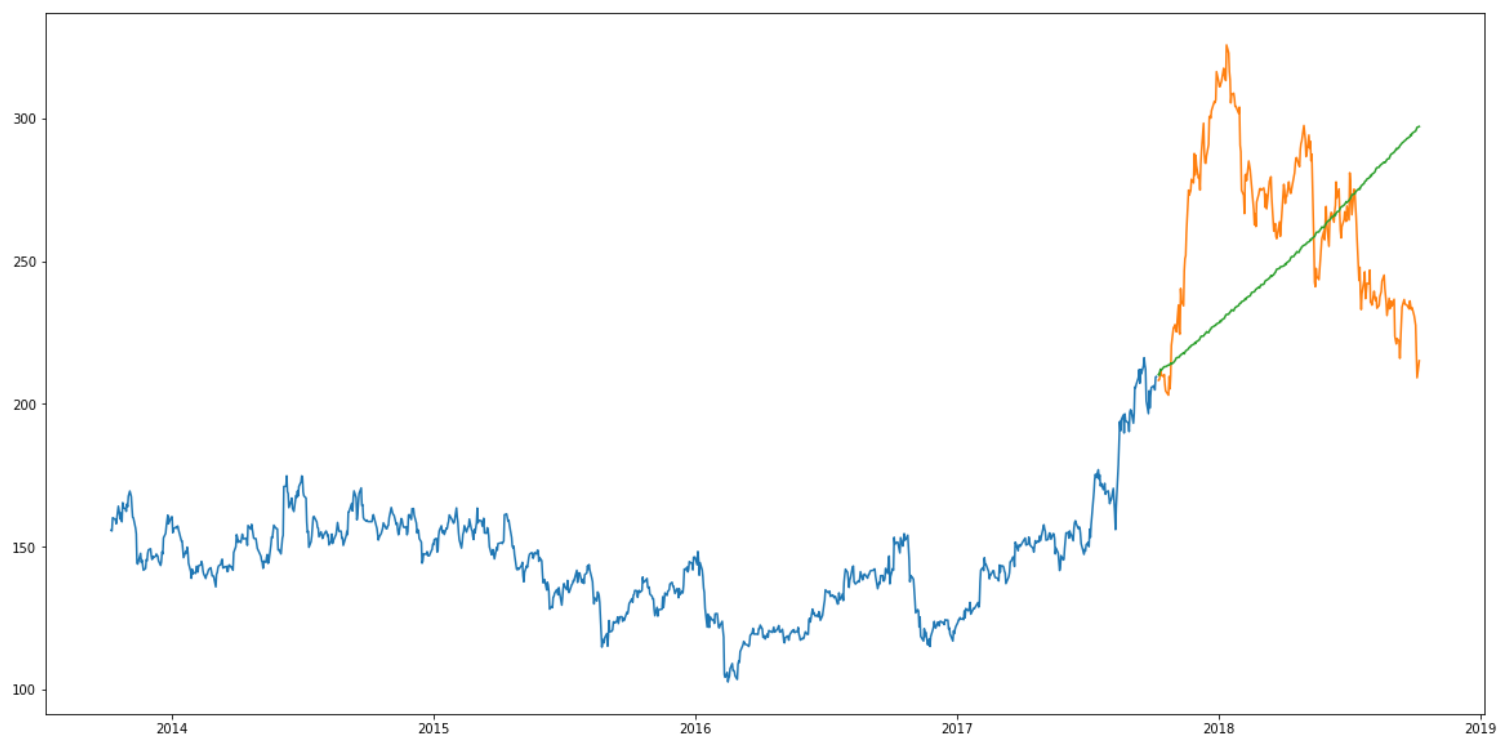
forecast = model.predict(n_periods=248)
forecast = pd.DataFrame(forecast,index = valid.index,columns=['Prediction'])
```

## Results

```
rms=np.sqrt(np.mean(np.power((np.array(valid['Close'])-np.array(forecast['Prediction'])),2)))
rms
```

```
44.954584993246954
```

```
#plot
plt.plot(train['Close'])
plt.plot(valid['Close'])
plt.plot(forecast['Prediction'])
```



## Inference

As we saw earlier, an auto ARIMA model uses past data to understand the pattern in the time series. Using these values, the model captured an increasing trend in the series. Although the predictions using this technique are far better than that of the previously implemented machine learning models, these predictions are still not close to the real values.

As its evident from the plot, the model has captured a trend in the series, but does not focus on the seasonal part. In the next section, we will implement a time series model that takes both trend and seasonality of a series into account.

## Prophet

### Introduction

There are a number of time series techniques that can be implemented on the stock prediction dataset, but most of these techniques require a lot of data preprocessing before fitting the model. Prophet, designed and pioneered by Facebook, is a time series forecasting library that requires no data preprocessing and is extremely simple to implement. The input for Prophet is a dataframe with two columns: date and target (ds and y).

Prophet tries to capture the seasonality in the past data and works well when the dataset is large. Here is an interesting article that explains Prophet in a simple and intuitive manner:

- [Generate Quick and Accurate Time Series Forecasts using Facebook's Prophet](https://www.analyticsvidhya.com/blog/2018/05/generate-accurate-forecasts-facebook-prophet-python-r/)  
(<https://www.analyticsvidhya.com/blog/2018/05/generate-accurate-forecasts-facebook-prophet-python-r/>).

## Implementation

```
#importing prophet
from fbprophet import Prophet

#creating dataframe
new_data = pd.DataFrame(index=range(0,len(df)),columns=['Date', 'Close'])

for i in range(0,len(data)):
    new_data['Date'][i] = data['Date'][i]
    new_data['Close'][i] = data['Close'][i]

new_data['Date'] = pd.to_datetime(new_data.Date,format='%Y-%m-%d')
new_data.index = new_data['Date']

#preparing data
new_data.rename(columns={'Close': 'y', 'Date': 'ds'}, inplace=True)

#train and validation
train = new_data[:987]
valid = new_data[987:]

#fit the model
model = Prophet()
model.fit(train)

#predictions
close_prices = model.make_future_dataframe(periods=len(valid))
forecast = model.predict(close_prices)
```

## Results

```
#rmse
forecast_valid = forecast['yhat'][987:]
rms=np.sqrt(np.mean(np.power((np.array(valid['y'])-np.array(forecast_valid)),2)))
rms
```

57.494461930575149

```
#plot
valid['Predictions'] = 0
valid['Predictions'] = forecast_valid.values

plt.plot(train['y'])
plt.plot(valid[['y', 'Predictions']])
```



## Inference

Prophet (like most time series forecasting techniques) tries to capture the trend and seasonality from past data. This model usually performs well on time series datasets, but fails to live up to its reputation in this case.

As it turns out, stock prices do not have a particular trend or seasonality. It highly depends on what is currently going on in the market and thus the prices rise and fall. Hence forecasting techniques like ARIMA, SARIMA and Prophet would not show good results for this particular problem.

Let us go ahead and try another advanced technique – Long Short Term Memory (LSTM).

## Long Short Term Memory (LSTM)

### Introduction

LSTMs are widely used for sequence prediction problems and have proven to be extremely effective. The reason they work so well is because LSTM is able to store past information that is important, and forget the information that is not. LSTM has three gates:

- **The input gate:** The input gate adds information to the cell state
- **The forget gate:** It removes the information that is no longer required by the model
- **The output gate:** Output Gate at LSTM selects the information to be shown as output

For a more detailed understanding of LSTM and its architecture, you can go through the below article:

- [Introduction to Long Short Term Memory\\_\(https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/\)](https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/)

For now, let us implement LSTM as a black box and check its performance on our particular data.

### Implementation

```
#importing required libraries
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, Dropout, LSTM

#creating dataframe
data = df.sort_index(ascending=True, axis=0)
new_data = pd.DataFrame(index=range(0,len(df)),columns=['Date', 'Close'])
for i in range(0,len(data)):
    new_data['Date'][i] = data['Date'][i]
    new_data['Close'][i] = data['Close'][i]

#setting index
new_data.index = new_data.Date
new_data.drop('Date', axis=1, inplace=True)

#creating train and test sets
dataset = new_data.values

train = dataset[0:987,:]
valid = dataset[987:,:]

#converting dataset into x_train and y_train
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(dataset)

x_train, y_train = [], []
for i in range(60,len(train)):
    x_train.append(scaled_data[i-60:i,0])
    y_train.append(scaled_data[i,0])
x_train, y_train = np.array(x_train), np.array(y_train)

x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))
```

```
# create and fit the LSTM network

model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(x_train.shape[1],1)))
model.add(LSTM(units=50))
model.add(Dense(1))

model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(x_train, y_train, epochs=1, batch_size=1, verbose=2)

#predicting 246 values, using past 60 from the train data
inputs = new_data[len(new_data) - len(valid) - 60:].values
inputs = inputs.reshape(-1,1)
inputs = scaler.transform(inputs)

X_test = []
for i in range(60,inputs.shape[0]):
    X_test.append(inputs[i-60:i,0])
X_test = np.array(X_test)

X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
closing_price = model.predict(X_test)
closing_price = scaler.inverse_transform(closing_price)
```

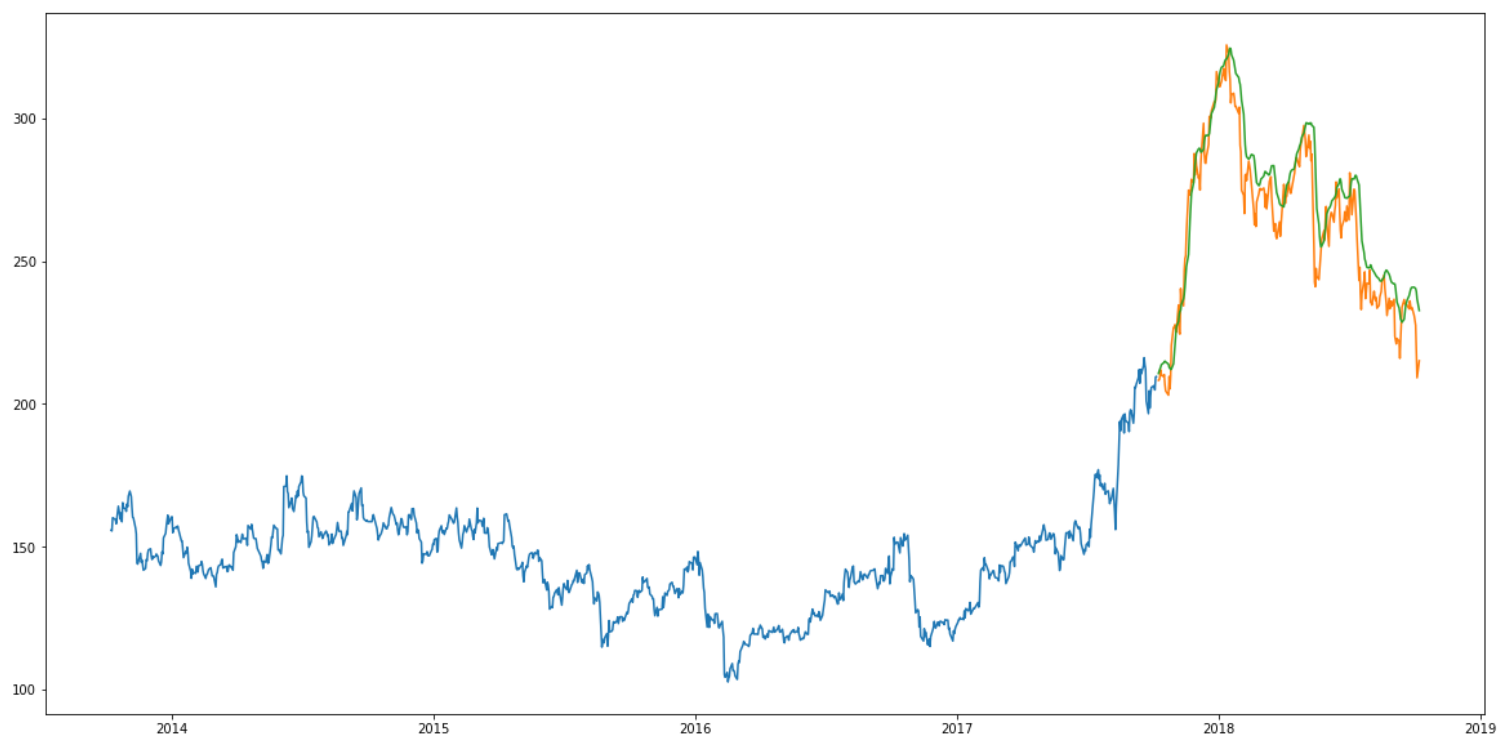
## Results

```
rms=np.sqrt(np.mean(np.power((valid-closing_price),2)))
rms
```

```
11.772259608962642
```



```
#for plotting
train = new_data[:987]
valid = new_data[987:]
valid['Predictions'] = closing_price
plt.plot(train['Close'])
plt.plot(valid[['Close', 'Predictions']])
```



## Inference

Wow! LSTM has easily outshone any algorithm we saw so far. The LSTM model can be tuned for various parameters such as changing the number of LSTM layers, adding dropout value or increasing the number of epochs. But are the predictions from LSTM enough to identify whether the stock price will increase or decrease? Certainly not!

As I mentioned at the start of the article, stock price is affected by the news about the company and other factors like demonetization or merger/demerger of the companies. There are certain intangible factors as well which can often be impossible to predict beforehand.

## End Notes

Time series forecasting is a very intriguing field to work with, as I have realized during my time writing these articles. There is a perception in the community that it's a complex field, and while there is a grain of truth in there, it's not so difficult once you get the hang of the basic techniques.


I am interested in finding out how LSTM works on a different kind of time series problem and encourage you to try it out on your own as well. If you have any questions, feel free to connect with me in the comments section below.


You can also read this article on Analytics Vidhya's Android APP




[\(/play.google.com/store/apps/details?id=com.analyticsvidhya.android&utm\\_source=blog\\_article&utm\\_campaign=blog&pcampaignid=MKT-Other-global-all-co-prtnr-py-PartBadge-Mar2515-1\)](https://play.google.com/store/apps/details?id=com.analyticsvidhya.android&utm_source=blog_article&utm_campaign=blog&pcampaignid=MKT-Other-global-all-co-prtnr-py-PartBadge-Mar2515-1)


### Share this:


 (<https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/?share=linkedin&nb=1>)


 (<https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/?share=facebook&nb=1>)

289

 (<https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/?share=google-plus-1&nb=1>)

 (<https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/?share=twitter&nb=1>)

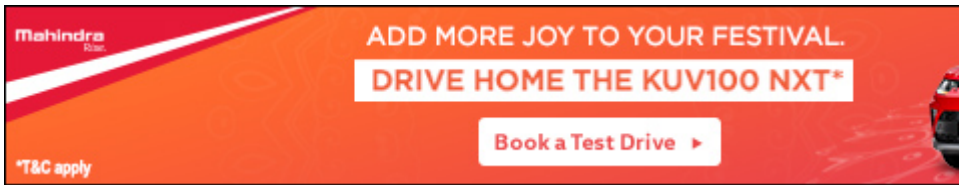
 (<https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/?share=pocket&nb=1>)

 (<https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/?share=reddit&nb=1>)

---

### Like this:

Loading...



TAGS : [AUTO ARIMA \(HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/AUTO-ARIMA/\)](https://www.analyticsvidhya.com/blog/tag/auto-arma/), [KNN \(HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/KNN/\)](https://www.analyticsvidhya.com/blog/tag/knn/), [LINEAR-REGRESSION \(HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/LINEAR-REGRESSION/\)](https://www.analyticsvidhya.com/blog/tag/linear-regression/), [LSTM \(HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/LSTM/\)](https://www.analyticsvidhya.com/blog/tag/lstm/), [MOVING AVERAGE \(HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/MOVING-AVERAGE/\)](https://www.analyticsvidhya.com/blog/tag/moving-average/), [PROPHET \(HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/PROPHET/\)](https://www.analyticsvidhya.com/blog/tag/prophet/), [PYTHON \(HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/PYTHON/\)](https://www.analyticsvidhya.com/blog/tag/python/), [STOCK MARKET ANALYSIS \(HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/STOCK-MARKET-ANALYSIS/\)](https://www.analyticsvidhya.com/blog/tag/stock-market-analysis/), [STOCK PREDICTION \(HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/STOCK-PREDICTION/\)](https://www.analyticsvidhya.com/blog/tag/stock-prediction/), [TIME SERIES \(HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/TIME-SERIES/\)](https://www.analyticsvidhya.com/blog/tag/time-series/), [TIME SERIES FORECASTING \(HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/TIME-SERIES-FORECASTING/\)](https://www.analyticsvidhya.com/blog/tag/time-series-forecasting/)

NEXT ARTICLE

### **A Computer Vision Approach to Hand Gesture Recognition**

(<https://www.analyticsvidhya.com/blog/2018/10/computer-vision-approach-hand-gesture-recognition/>)

...

PREVIOUS ARTICLE

### **An Introductory Guide to Deep Learning and Neural Networks (Notes from deeplearning.ai Course #1)**

(<https://www.analyticsvidhya.com/blog/2018/10/introduction-neural-networks-deep-learning/>)



(<https://www.analyticsvidhya.com/blog/author/aishwaryasingh/>)

**Aishwarya Singh**

**(<https://www.analyticsvidhya.com/blog/author/aishwaryasingh/>)**

An avid reader and blogger who loves exploring the endless world of data science and artificial intelligence. Fascinated by the limitless applications of ML and AI; eager to learn and discover the depths of data science.

## RELATED ARTICLES

**KUNAL JAIN** ([HTTPS://WWW.ANALYTICS...](https://www.analyticsvidhya.com/blog/2016/02/quick-insights-india-analytics-and-big-data-salary-report-2016/))



(<https://www.analyticsvidhya.com/blog/2016/02/quick-insights-india-analytics-and-big-data-salary-report-2016/>)

**Quick Insights: India Analytics and Big Data Salary Report 2016**  
(<https://www.analyticsvidhya.com/blog/2016/02/quick-insights-india-analytics-and-big-data-salary-report-2016/>)



(<https://www.analyticsvidhya.com/blog/2016/07/12-winning-tips-to-clinch-win-data-science-competitions/>)

**12 Winning Tips to Clinch Your First Win in Data Science Competitions**  
(<https://www.analyticsvidhya.com/blog/2016/07/12-winning-tips-to-clinch-win-data-science-competitions/>)

**JAMES VERDANT**

**GUEST BLOG** ([HTTPS://WWW.ANALYTIC...](https://www.analyticsvidhya.com/blog/2016/10/complete-study-of-factors-contributing-to-air-pollution/))



(<https://www.analyticsvidhya.com/blog/2016/10/complete-study-of-factors-contributing-to-air-pollution/>)

**Complete Study of Factors Contributing to Air Pollution**  
(<https://www.analyticsvidhya.com/blog/2016/10/complete-study-of-factors-contributing-to-air-pollution/>)



(<https://www.analyticsvidhya.com/blog/2016/02/analytics-big-data-salary-report-2016/>)

**India Exclusive: Analytics and Big Data Salary Report 2016**  
(<https://www.analyticsvidhya.com/blog/2016/02/analytics-big-data-salary-report-2016/>)

**DISHASHREE GUPTA** ([HTTPS://WWW.AN...](https://www.analyticsvidhya.com/blog/2017/10/fundamentals-deep-learning-activation-functions-when-to-use-them/))



(<https://www.analyticsvidhya.com/blog/2017/10/fundamentals-deep-learning-activation-functions-when-to-use-them/>)

**Fundamentals of Deep Learning – Activation Functions and When to Use Them?**  
(<https://www.analyticsvidhya.com/blog/2017/10/fundamentals-deep-learning-activation-functions-when-to-use-them/>)



(<https://www.analyticsvidhya.com/blog/2017/08/skilltest-logistic-regression/>)

**30 Questions to test your understanding of Logistic Regression**  
(<https://www.analyticsvidhya.com/blog/2017/08/skilltest-logistic-regression/>)

Reply

October 25, 2018 at 6:53 pm (<https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/#comment-155456>)

Isn't the LSTM model using your "validation" data as part of its modeling to generate its predictions since it only goes back 60 days. Your other techniques are only using the "training" data and don't have the benefit of looking back 60 days from the target prediction day. Is this a fair comparison?



**AISHWARYA SINGH**

[Reply](#)

October 26, 2018 at 12:08 pm (<https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/#comment-155466>)

Hi James,

The idea isn't to compare the techniques but to see what works best for stock market predictions. Certainly for this problem LSTM works well, while for other problems, other techniques might perform better. We can add a lookback component with LSTM is an added advantage



**TERU**

[Reply](#)

October 26, 2018 at 6:39 pm (<https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/#comment-155472>)

I think that you cannot say LSTM works well because what it actually does is to predict one day ahead based on the recent 60 days. In other words, the model just go over all the validation data daily basis to predict "tomorrow". This is a totally different prediction scheme from the other prediction methods, which have to predict the entire validation data points without seeing any of information in the validation data. If you use the "daily basis prediction" scheme for other methods, any of methods would produce a good result, I guess.



**AISHWARYA SINGH**

[Reply](#)

October 26, 2018 at 7:46 pm (<https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/#comment-155473>)

Hi Teru,

When James first pointed out, I started looking at how can I use validation in other models (its simpler with LSTM). For ARIMA and PROPHET, the input can only be a univariate series so we can make prediction for one day, change the training set (add that day's value) after each prediction and retrain before predicting for the next

day. For moving average and regression it should be comparatively easier. If you both have any suggestions as to how can I update train data with each day's values, do share your ideas, I am really interested in finding out how the results turn out to be!

**JAY**[Reply](#)

October 25, 2018 at 11:54 pm (<https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/#comment-155460>)

Getting index error –

---

IndexError Traceback (most recent call last)

in

1 #Results

—> 2 rms=np.sqrt(np.mean(np.power((np.array(valid['Close']) – np.array(valid['Predictions'])),2)))

3 rms

IndexError: only integers, slices (':'), ellipsis ('...'), numpy.newaxis ('None') and integer or boolean arrays are valid indices

**AISHWARYA SINGH**[Reply](#)

October 26, 2018 at 11:49 am (<https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/#comment-155464>)

Hi Jay,

Please use the following command before calculating rmse `valid['Predictions'] = 0`

`valid['Predictions'] = closing_price` . I have updated the same in the article

**PANKAJ**[Reply](#)

October 26, 2018 at 10:26 am (<https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/#comment-155461>)

After running the following codes-

`train['Date'].min(), train['Date'].max(), valid['Date'].min(), valid['Date'].max()`

```
(Timestamp('2013-10-08 00:00:00'),  
Timestamp('2017-10-06 00:00:00'),  
Timestamp('2017-10-09 00:00:00'),  
Timestamp('2018-10-08 00:00:00'))
```

I am getting the following error :  
name 'Timestamp' is not defined

Please help.



**AISHWARYA SINGH**

[Reply](#)

October 26, 2018 at 11:39 am (<https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/#comment-155463>)

Hi Pankaj,

The command is only

`train['Date'].min(), train['Date'].max(), valid['Date'].min(), valid['Date'].max()` , the timestamp is the result I got by running the above command.



**ZARIEF MARZUKI LAO**

[Reply](#)

October 26, 2018 at 3:35 pm (<https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/#comment-155468>)

Hi Aishwarya,

Just curious. LSTM works just TOO well !!

Is splitting dataset to train & valid step carry out after the normalizing step ?!

i.e.

```
#converting dataset into x_train and y_train  
scaler = MinMaxScaler(feature_range=(0, 1))  
scaled_data = scaler.fit_transform(dataset)
```

then

```
dataset = new_data
```

```
train = dataset[:987]
valid = dataset[987:]
```

Then this

```
x_train, y_train = [], []
for i in range(60, len(train)): # <- replace dataset with train ?!
    x_train.append(scaled_data[i-60:i, 0])
    y_train.append(scaled_data[i, 0])
x_train, y_train = np.array(x_train), np.array(y_train)
```

Guide me on this.

Thanks



**AISHWARYA SINGH**

[Reply](#)

October 26, 2018 at 4:42 pm (<https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/#comment-155470>)

Hi Zarief,

Yes, the train and test set are created after scaling the data using the for loop :

```
x_train, y_train = [], []
for i in range(60, len(train)):
    x_train.append(scaled_data[i-60:i, 0]) #we have used the scaled data here
    y_train.append(scaled_data[i, 0])
x_train, y_train = np.array(x_train), np.array(y_train)
```

Secondly, the command `dataset = new_data.values` will be before scaling the data, as shown in the article, since `dataset` is used for scaling and hence must be defined before.



**ROHIT**

[Reply](#)

October 26, 2018 at 10:25 pm (<https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/#comment-155474>)

hi



I'm getting below error...

```
""">>> #import packages
>>> import pandas as pd
```

Traceback (most recent call last):

File "", line 1, in

import pandas as pd

ImportError: No module named pandas"""



**AISHWARYA SINGH**

[Reply](#)

October 29, 2018 at 10:58 am (<https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/#comment-155488>)

Hi rohit,

Is the issue resolved? Have you worked with pandas previously?



**SHAN**

[Reply](#)

October 27, 2018 at 1:55 am (<https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/#comment-155476>)

Hi..

Thanks for nicely elaborating LSTM implementation in the article.

However, in LSTM rms part if you can guide, as I am getting the following error :

```
valid['Predictions'] = 0.0
valid['Predictions'] = closing_price
rms=np.sqrt(np.mean(np.power((np.array(valid['Close'])-np.array(valid['Predictions'])),2)))
rms
```

#####

IndexError Traceback (most recent call last)

```
in ()
--> 1 valid['Predictions'] = 0.0
2 valid['Predictions'] = closing_price
3 rms=np.sqrt(np.mean(np.power((np.array(valid['Close'])-np.array(valid['Predictions'])),2)))
4 rms
```

IndexError: only integers, slices (`:`), ellipsis (`...`), numpy.newaxis (`None`) and integer or boolean arrays are valid indices

**JAY**[Reply](#)

October 27, 2018 at 3:02 am (<https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/#comment-155478>)

Still same error –

---

IndexError Traceback (most recent call last)

in

—> 1 valid['Predictions'] = closing\_price

IndexError: only integers, slices (`:`), ellipsis (`...`), numpy.newaxis (`None`) and integer or boolean arrays are valid indices

And also why are we doing valid['Predictions'] = 0 and valid['Predictions'] = closing\_price instead of valid['Predictions'] = closing\_price

**AISHWARYA SINGH**[Reply](#)

October 27, 2018 at 11:15 am (<https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/#comment-155480>)

Yes you can skip the line but it will still show an error because the index hasn't been defined. Have you followed the code from the start? Please add the following code lines and check if it works

```
new_data.index = data['Date'] #considering the date column has been set to datetime format
```

```
valid.index = new_data[987:].index
```

```
train.index = new_data[:987].index
```

Let me know if this works. Other wise share the notebook you are working on and I will look into it.

**ROBERTO**[Reply](#)

October 27, 2018 at 8:30 pm (<https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/#comment-155483>)

Hi,

Nice article.

I have installed fastai but I am getting the following error:  
ModuleNotFoundError: No module named 'fastai.structured'

Any idea?



**AISHWARYA SINGH**

[Reply](#)

October 31, 2018 at 12:22 pm (<https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/#comment-155508>)

Hi Roberto,

Directly clone it from here : <https://github.com/fastai/fastai> (<https://github.com/fastai/fastai>) . Let me know if you still face an issue.



**JINGMIAO**

[Reply](#)

October 29, 2018 at 1:47 am (<https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/#comment-155486>)

Hello AISHWARYA,  
I dont know what's your motivation to spend such a long time to write this blog  
But thank you soooooo much!!!  
Appreciate your time for both the words and codes (easy to follow) !!!  
Such a great work!!!



**AISHWARYA SINGH**

[Reply](#)

October 29, 2018 at 11:19 am (<https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/#comment-155489>)

Really glad you liked the article. Thank you!



**VEDAMURTHY KB**

[Reply](#)

October 31, 2018 at 4:09 pm (<https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/#comment-155510>)

```
new_data.index=data['Date']
valid.index=new_data[987:].index
train.index=new_data[:987].index
```

gives

AttributeError Traceback (most recent call last)

in ()

1 new\_data.index=data['Date']

—> 2 valid.index=new\_data[987:].index

3 train.index=new\_data[:987].index

AttributeError: 'numpy.ndarray' object has no attribute 'index'

```
valid['Predictions'] = 0
```

```
valid['Predictions'] = closing_price
```

gives

IndexError Traceback (most recent call last)

in ()

—> 1 valid['Predictions'] = 0

2 valid['Predictions'] = closing\_price

IndexError: only integers, slices ('::'), ellipsis ('...'), numpy.newaxis ('None') and integer or boolean arrays are valid indices

V good article. I am getting above errors. Kindly help solving it



**AISHWARYA SINGH**

[Reply](#)

October 31, 2018 at 5:12 pm (<https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/#comment-155513>)

Hi,

Please print the validation head and see if the index values are actually the dates or just numbers. If they are numbers, change the index to dates. Please share the screenshot here or via mail at [aishwarya@analyticsvidhya.com](mailto:aishwarya@analyticsvidhya.com) (<mailto:aishwarya@analyticsvidhya.com>).



**RAVI SHANKAR**

[Reply](#)

[October 31, 2018 at 4:51 pm \(https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/#comment-155511\)](https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/#comment-155511)

Hi,

Thanks for putting the efforts in writing the article.

If you believe LSTM model works this well, try buying few shares of Tata Global beverages and let us know the returns on the same. I guess, you would understand the concept of over-fit.

Thanks,  
Ravi



**AISHWARYA SINGH**

[Reply](#)

[October 31, 2018 at 5:08 pm \(https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/#comment-155512\)](https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/#comment-155512)

Hi Ravi,

I actually did finally train my model on the complete data and predicted for next 10 days (and checked against the results for the week). The first 2 predictions weren't exactly good but next 3 were (didn't check the remaining). Secondly, I agree that machine learning models aren't the only thing one can trust, years of experience & awareness about what's happening in the market can beat any ml/dl model when it comes to stock predictions. I wanted to explore this domain and I have learnt more while working on this dataset than I did while writing my previous articles.

## LEAVE A REPLY

Your email address will not be published.

Comment

Name (required)

Email (required)

Website

☐ Notify me of new posts by email.

**SUBMIT COMMENT**



(<https://www.analyticsvidhya.com/datahack-summit-2018/?>

utm\_source=AVbannerdhslong)



## POPULAR POSTS

---

24 Ultimate Data Science Projects To Boost Your Knowledge and Skills (& can be accessed freely)  
(<https://www.analyticsvidhya.com/blog/2018/05/24-ultimate-data-science-projects-to-boost-your-knowledge-and-skills/>)

A Complete Tutorial to Learn Data Science with Python from Scratch  
(<https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/>)

Essentials of Machine Learning Algorithms (with Python and R Codes)  
(<https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/>)



Understanding Support Vector Machine algorithm from examples (along with code)

(<https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/>)

7 Types of Regression Techniques you should know!

(<https://www.analyticsvidhya.com/blog/2015/08/comprehensive-guide-regression/>)

6 Easy Steps to Learn Naive Bayes Algorithm (with codes in Python and R)

(<https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/>)

A comprehensive beginner's guide to create a Time Series Forecast (with Codes in Python)

(<https://www.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes-python/>)

Introduction to k-Nearest Neighbors: Simplified (with implementation in Python)

(<https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/>)



## RECENT POSTS

---

**A Practical Implementation of the Faster R-CNN Algorithm for Object Detection (Part 2 – with Python codes)** (<https://www.analyticsvidhya.com/blog/2018/11/implementation-faster-r-cnn-python-object-detection/>)

NOVEMBER 4, 2018

**Top 5 Machine Learning GitHub Repositories & Reddit Discussions (October 2018)**  
(<https://www.analyticsvidhya.com/blog/2018/11/best-machine-learning-github-repositories-reddit-threads-october-2018/>)

NOVEMBER 1, 2018

**An Introduction to Text Summarization using the TextRank Algorithm (with Python implementation)**  
(<https://www.analyticsvidhya.com/blog/2018/11/introduction-text-summarization-textrank-python/>)

NOVEMBER 1, 2018

**DataHack Radio #13: Data Science and AI in the Oil & Gas Industry with Yogendra Pandey, Ph.D.**  
(<https://www.analyticsvidhya.com/blog/2018/10/datahack-radio-podcast-oil-gas-ai/>)

OCTOBER 29, 2018



([http://www.edvancer.in/certified-data-scientist-with-python-course?](http://www.edvancer.in/certified-data-scientist-with-python-course?utm_source=AV&utm_medium=AVads&utm_campaign=AVadsnonfc&utm_content=pythonavad)

[utm\\_source=AV&utm\\_medium=AVads&utm\\_campaign=AVadsnonfc&utm\\_content=pythonavad](http://www.edvancer.in/certified-data-scientist-with-python-course?utm_source=AV&utm_medium=AVads&utm_campaign=AVadsnonfc&utm_content=pythonavad))



([https://trainings.analyticsvidhya.com/courses/course-](https://trainings.analyticsvidhya.com/courses/course-v1:AnalyticsVidhya+DS101+2018T2/about?utm_source=AVStickybanner1)

[v1:AnalyticsVidhya+DS101+2018T2/about?utm\\_source=AVStickybanner1](https://trainings.analyticsvidhya.com/courses/course-v1:AnalyticsVidhya+DS101+2018T2/about?utm_source=AVStickybanner1))

Analytics Vidhya

Learn everything about analytics

Analytics Vidhya

Learn everything about analytics

Analytics Vidhya

Learn everything about analytics

ANALYTICS VIDHYA

About Us

<http://www.analyticsvidhya.com/about>

Our team

<https://www.analyticsvidhya.com/team/>

Blog

<http://www.analyticsvidhya.com/blog/>

Hackathons

<https://trainings.analyticsvidhya.com/courses/course-hackathons>

Discussions

<https://datahack.analyticsvidhya.com/discussions>

Apply Jobs

<https://www.analyticsvidhya.com/career/>

Leaderboard

<https://datahack.analyticsvidhya.com/leaderboard/>

Contact Us

<https://www.analyticsvidhya.com/contact/>

DATA SCIENTISTS

Post Jobs

<https://www.analyticsvidhya.com/corporate/>

<https://www.analyticsvidhya.com/blog/>

<https://trainings.analyticsvidhya.com/courses/course-hackathons>

<https://datahack.analyticsvidhya.com/discussions>

<https://www.analyticsvidhya.com/career/>

<https://datahack.analyticsvidhya.com/leaderboard/>

<https://www.analyticsvidhya.com/contact/>

COMPANIES

Advertising

<https://www.analyticsvidhya.com/contact/>

Reach Us

<https://www.analyticsvidhya.com/contact/>

JOIN OUR COMMUNITY :

f

<https://www.facebook.com/AnalyticsVidhya/>

3017 Followers

t

<https://twitter.com/AnalyticsVidhya>

7513 Followers

G+

<https://plus.google.com/AnalyticsVidhya>

3017 Followers

in

<https://www.linkedin.com/company/AnalyticsVidhya>

7513 Followers

Subscribe to emailer

>

v1:AnalyticsVidhya+CVDL101+CVDL101s T1/about?

utm\_source=CV101AVBlogBanner&utm\_medium=StickyBanner2utm\_campaign=CV101s T1/about?