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 move on to advanced techniques like Auto ARIMA and LSTM.

The core idea behind this article is to showcase how these algorithms are implemented. 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
- A Complete Tutorial on Time Series Modeling
- Free Course: Time Series Forecasting using Python

Table of Contents

1.
    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 (you can find historical data for various stocks here) and for this particular project, I have used the data for 'Tata Global Beverages'. Time to dive in!

Note: Here is the dataset I used for the code: Download

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')
```

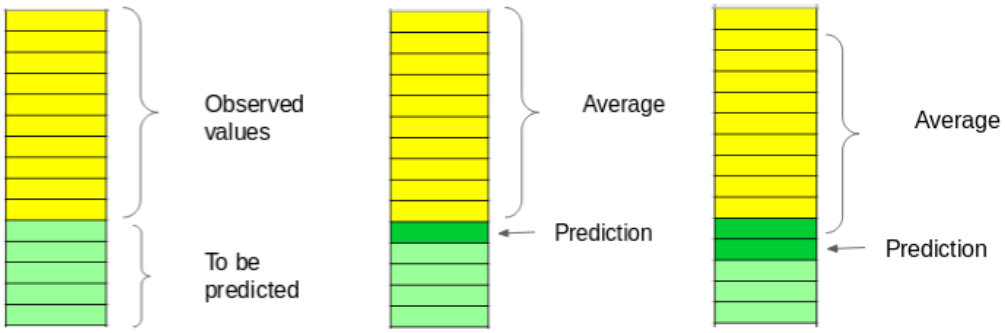
In the upcoming sections, we will explore these variables and use different techniques to predict the daily closing price of the stock.

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.

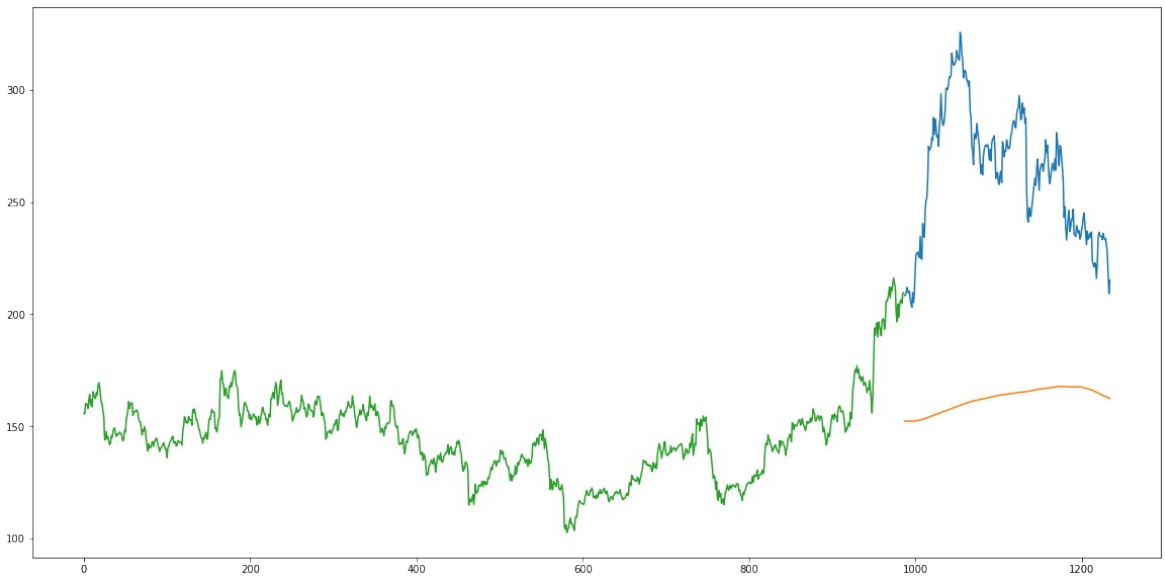
```
#make predictions
preds = []
for i in range(0,248):
    a = train['Close'][len(train)-248+i:].sum() + sum(preds)
    b = a/248
    preds.append(b)
```

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:

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

Here, x_1, x_2,….x_n represent the independent variables while the coefficients θ_1, θ_2, …. θ_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.

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 where the independent features are useful for determining the target value.

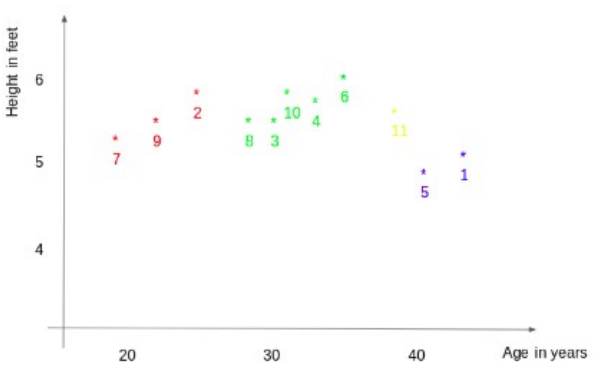
k-Nearest Neighbours

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 it's neighbors. If we consider three neighbours (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
- A Practical Introduction to K-Nearest Neighbors Algorithm for Regression

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:

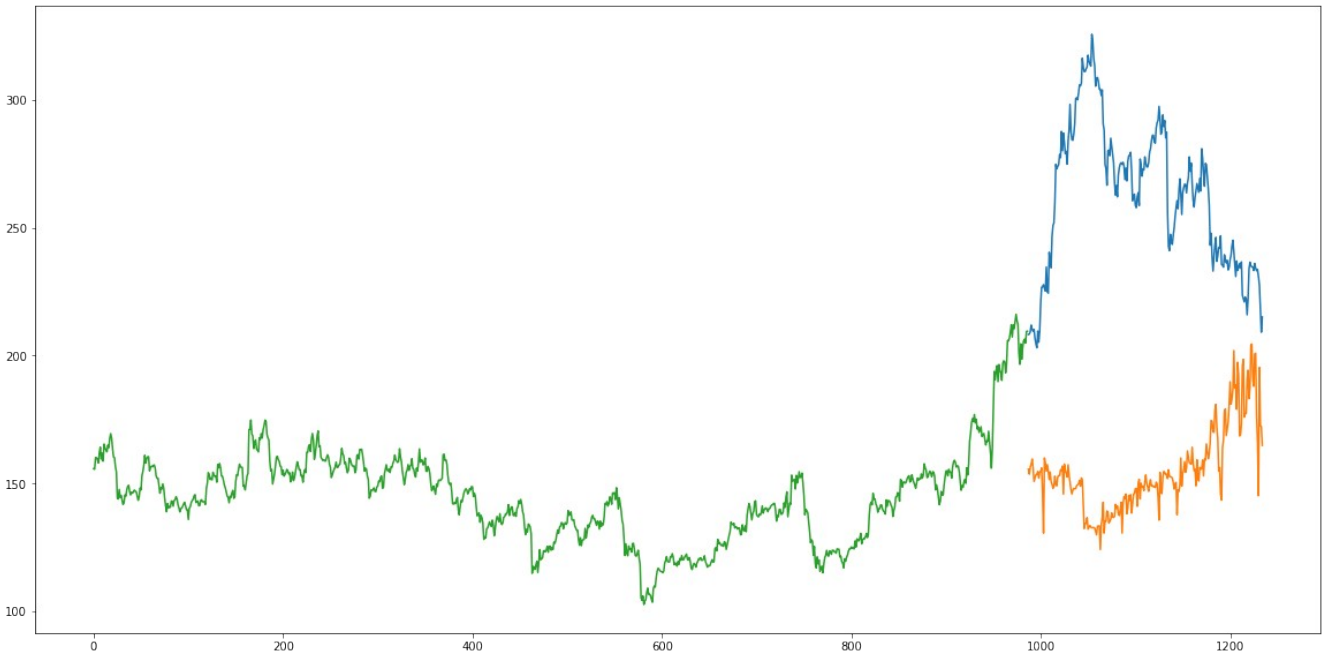
```
#scaling data
x_train_scaled = scaler.fit_transform(x_train)
x_train = pd.DataFrame(x_train_scaled)
x_valid_scaled = scaler.fit_transform(x_valid)
x_valid = pd.DataFrame(x_valid_scaled)

#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

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.

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 it's 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

For now, let us implement LSTM as a black box and check it's 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! 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

How to build Ensemble Models in machine learning? (with code in R)

February 15, 2017

In "Machine Learning"

12 Frequently Asked Questions on Deep Learning (with their answers)!

May 21, 2018

In "Deep Learning"

Tags : auto arima, KNN, linear regression, LSTM, Moving average, prophet, python, Stock market analysis, Stock prediction, Time Series, Time Series Forecasting

Next Article

A Computer Vision Approach to Hand Gesture Recognition

Previous Article

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

Aishwarya Singh

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.

90 Comments

- *James Verdant*

  October 25, 2018 at 6:53 pm

  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?

  Reply

  - *Aishwarya Singh*

    October 26, 2018 at 12:08 pm

    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

    Reply

    - *Teru*

      October 26, 2018 at 6:39 pm

      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 mothods, any of methods would produce a good result, I guess.

      Reply

      - *Aishwarya Singh*

October 26, 2018 at 7:46 pm

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!

Reply

- *Jay*

  October 25, 2018 at 11:54 pm

  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

  Reply

  - *Aishwarya Singh*

    October 26, 2018 at 11:49 am

    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

    Reply

- *Pankaj*

  October 26, 2018 at 10:26 am

  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.

  Reply

  - *Aishwarya Singh*

    October 26, 2018 at 11:39 am

    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.

    Reply

  - *Vishal*

    November 13, 2018 at 3:33 pm

    Pankaj use below code
    from pandas import Timestamp

    Reply

    -  *Aishwarya Singh*

      November 13, 2018 at 4:51 pm

Hi Vishal,

I believe Pankaj has accidentally pasted the result of the code, along with the command. Thus, importing Timestamp would not solve the issue.

Reply

-  *Joseph*

  March 4, 2019 at 5:13 pm

  It actually solved it!

  Reply

- *Zarief Marzuki Lao*

  October 26, 2018 at 3:35 pm

  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

Reply

- *Aishwarya Singh*

  October 26, 2018 at 4:42 pm

  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.

  Reply

- *rohit*

  October 26, 2018 at 10:25 pm

  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”””””

  Reply

  - *Aishwarya Singh*

    October 29, 2018 at 10:58 am

    Hi rohit,

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

Reply

- *Nick*

  January 18, 2019 at 10:11 pm

  Try running
  pip install pandas

  in a command line

  Reply

- *Shan*

  October 27, 2018 at 1:55 am

  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

  Reply

- *Jay*

  October 27, 2018 at 3:02 am

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

Reply

- *Aishwarya Singh*

  October 27, 2018 at 11:15 am

  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.

  Reply

- *Roberto*

  October 27, 2018 at 8:30 pm

  Hi,

  Nice article.

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

  Any idea?

  Reply

  - *Aishwarya Singh*

October 31, 2018 at 12:22 pm

Hi Roberto,

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

Reply

- *Shreyansh Sharma*

  May 9, 2019 at 9:21 pm

  Please use the following code as fastai package has been changed:
  from fastai.tabular import add_datepart

  Reply

- *Jingmiao*

  October 29, 2018 at 1:47 am

  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!!!

  Reply

  - *Aishwarya Singh*

    October 29, 2018 at 11:19 am

    Really glad you liked the article. Thank you!

    Reply

- *Vedamurthy KB*

  October 31, 2018 at 4:09 pm

  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

Reply

- *Aishwarya Singh*

  October 31, 2018 at 5:12 pm

  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 (dropped a mail)

  Reply

- *Ravi Shankar*

  October 31, 2018 at 4:51 pm

  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

  Reply

- *Aishwarya Singh*

  October 31, 2018 at 5:08 pm

  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.

  Reply

  - *Shahnawaz*

    November 8, 2018 at 5:55 pm

    This was a really useful tutorial covering different techniques.Could you please detail how we can predict in future as you mentioned(10 days in future).Which area exactly we need to change?
    Will it be this part of the code?
    #predicting 246 values, using past 60 from the train data

    Reply

    - *Aishwarya Singh*

      November 12, 2018 at 4:08 pm

      Yes exactly. You need to create a validation set with only 10 rows as input to the LSTM model.

      Reply

- *Moobi*

  November 11, 2018 at 10:41 pm

  Its nice tutorial, thanks. I need to know how can i predict just tomorrow's price?

  Reply

  - *Aishwarya Singh*

November 12, 2018 at 4:05 pm

Hi,

To make predictions only for the next day, the validation set should have only 1 row (with past 60 values).

Reply

- *Sanat Mishra*

  January 15, 2019 at 12:04 am

  can you please tell where we need to make change in code…and what change?
  suppose i have csv with data from 1st jan 2001 till today (15th jan 2019), how will i predict value for tomorrow (i.e. 16th jan 2019)

  Reply

  - *Aishwarya Singh*

    March 5, 2019 at 3:55 pm

    Hi Sanat,

    When you define the input data for your LSTM model, make the input to be a single row with past 60 days data. The model has been trained such that it looks at the past 60 days data to predict the 61st day.

    Reply

- *Pabitra*

  November 12, 2018 at 7:20 pm

  What is the difference between last and closing price?

  Reply

  - *Aishwarya Singh*

    November 13, 2018 at 10:59 am

    The difference is not significant. I plotted the two variables and they overlapped each other.

    Reply

- *Miguel*

  November 13, 2018 at 12:07 am

  Hi, thanks for the article.

  I got this error:

  rms=np.sqrt(np.mean(np.power((np.array(valid[‘Close’])-preds),2)))
  ValueError: operands could not be broadcast together with shapes (1076,) (248,)

  Can you help me on this?

  Reply

  - *Aishwarya Singh*

    November 13, 2018 at 10:53 am

    Hi,

    Looks like the validation set and predictions have different lengths. Can you please share your notebook with me? I have sent you a mail on the email ID you provided.

    Reply

  - *Isaac*

    November 30, 2018 at 12:54 am

    I ran into the same error as ValueError: operands could not be broadcast together with shapes (1088,) (248,). Guidance towards resolution would be appreciated.

    Reply

    - *Aishwarya Singh*

      December 5, 2018 at 11:56 am

      Could you please share the notebook Isaac?

      Reply

- *Alex*

  November 24, 2018 at 3:12 pm

Thank you so much for the code, you inspire me a lot. I applied your algorithm for my example and it it works fine. Now i have to make it predict the price for the next 5 years, do you know how to achieve that? I should use all the data without splitting into test and train for training and somehow generate new dates in that array, then predict the value for them. I tried to modify your code but i couldn't figure it out. I'm new to ML and it's really hard to understand those functions and classes.

Thank you very much in advance

Reply

- *Aishwarya Singh*

  December 5, 2018 at 11:44 am

  Hi Alex,

  Currently the model is trained to look at the recent past data and make predictions for the next day. To make predictions for 5 years in future, we'll first have to change the way we train model here, and we'll need a, much bigger dataset as well.

  Reply

- *Shabbir Anaswala*

  December 5, 2018 at 12:18 pm

  Hi,

  I am getting this error

  x_train_scaled = scaler.fit_transform(x_train)
  Traceback (most recent call last):

  File "", line 1, in
  x_train_scaled = scaler.fit_transform(x_train)

  File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py", line 517, in fit_transform
  return self.fit(X, **fit_params).transform(X)

  File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\preprocessing\data.py", line 308, in fit
  return self.partial_fit(X, y)

  File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\preprocessing\data.py", line 334, in partial_fit
  estimator=self, dtype=FLOAT_DTYPES)

File “C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py”, line 433, in check_array
array = np.array(array, dtype=dtype, order=order, copy=copy)

TypeError: float() argument must be a string or a number, not ‘Timestamp’

Reply

-  *Aishwarya Singh*

  December 5, 2018 at 12:26 pm

  Hi Shabbir,

  The data you are trying to scale has the “Date” column as well. Please follow the code from the start. If you have extracted features from the date column, you can drop this column and then go ahead with the implementation.

  Reply

  -  *Shabbir Anaswala*

    December 5, 2018 at 1:23 pm

    Thanks Aishwarya,
    I am ruuning this code in spyder

    while trying to import Arima I am getting below error.

    from pyramid.arima import auto_arima
    Traceback (most recent call last):

    File “”, line 1, in
    from pyramid.arima import auto_arima

    ModuleNotFoundError: No module named ‘pyramid’

    Please let me know if there are any work around for this?

    Reply

-  *Aishwarya Singh*

  March 5, 2019 at 1:14 pm

  Hi Shabbir,

  The current code I have written is in python 3 using jupyter notebook

  Reply

-  *Muralidharan*

  February 6, 2019 at 1:10 pm

  I too got the same error as Shabbir but i am unable to resolve it.You have explained that you have extracted features from date column and you can drop this column.Can you tell me how can I drop this column?

  Reply

  -  *Aishwarya Singh*

    February 7, 2019 at 11:00 am

    to drop a column, use the code
    `df.drop([column_name], axis=1, inplace=True)`

    Reply

-  *Ricky*

December 20, 2018 at 11:46 pm

Hello AISHWARYA SINGH,
Nice article…I had been working FOREX data to use seasonality to predict the next days direction for many weeks and your code under the FastAi part gave me an idea on how to go about it. Thanks for a great article.

Have you tried predicting the stock data based bulls and bears only, using classification?
I used np.log(df[‘close’]/df.[‘close’].shift(1)) to find the returns and if its negative I use “np.where” to assign -1 to it and if its positive I assign 1 to it. All I want to predict is if tomorrow would be 1 or -1.

I have not been able to get a 54% accuracy from that model.

a 60% would be very profitable when automated.

would you want to see my attempts?

Reply

-  *Aishwarya Singh*

  March 5, 2019 at 3:18 pm

  Hi Ricky,

  That’s a very interesting idea. Which algorithm did you use ?

  Reply

-  *shubham sharma*

  December 28, 2018 at 5:41 pm

  Thanks for sharing valuable information. a worth reading blog. I try this algorithm for my example and it works excellent.

  Reply

-  *Aishwarya Singh*

  December 31, 2018 at 12:14 pm

  Really glad you liked it. Thanks Shubham!

  Reply

-  *ice*

  December 31, 2018 at 8:59 am

  Hi Asihwarya,

  As far as i understand, the model takes in 60 days of real data to predict the next day's value in LSTM. I wonder how the results will be like if you take a predicted value to predict the next value instead. This will allow us to predict say 2 years of data for long term trading.

  Reply

  -  *Aishwarya Singh*

    December 31, 2018 at 12:12 pm

    If you do have the real time data, it'd be preferable to use that instead since you'll get more accurate results. Otherwise definitely using the predicted values would be beneficial.

    Reply

-  *Jwala*

  January 2, 2019 at 9:24 pm

  Hi Aishwarya,

  I am not able to download the dataset getting empty CSV file with header. Could you please help me.

  Reply

  -  *Aishwarya Singh*

    January 3, 2019 at 11:07 am

    Hi,

    I sent you the dataset via mail. Please check.

    Reply

    -  *Jwala*

      January 3, 2019 at 1:02 pm

      Thank you.

      It would be great help, if you can give me one real time business use case for Financial forecasting for stock prices. Thanks

      Reply

-  *Run Ma*

  January 3, 2019 at 5:44 pm

  I am also not able to download the test data (NSE-TATAGLOBAL(1).csv), could you send me? thanks!

  Reply

  -  *Aishwarya Singh*

    January 4, 2019 at 10:56 am

    Could you share your email id please?

    Reply

-  *Chris*

  January 4, 2019 at 2:03 am

  Could you send me the full working code, i cant seem to get it to work

  Reply

  -  *Aishwarya Singh*

    January 4, 2019 at 11:05 am

    I have sent you a mail.

Reply

-  *Doaa*

  January 6, 2019 at 7:59 am

  Hi AISHWARYA SINGH,
  thanks a lot for your great article .

  Reply

  -  *Aishwarya Singh*

    January 8, 2019 at 10:45 am

    Glad you liked it Doaa

    Reply

-  *chao*

  January 6, 2019 at 12:26 pm

  could you share the full code, I have strong interest in time series analysis

  Reply

  -  *Aishwarya Singh*

    January 15, 2019 at 12:57 pm

Hi chao,

The code is shared within the article itself.

Reply

-  *Robert*

  January 12, 2019 at 1:08 am

  Hi Aishwarya,
  The link provided in this article seems to be a lot smaller (min date is 3-2-2017) where in your article is spoken about year 2013…

  Could you please help me providing te Original CSV file?

  Thanx!

  Reply

  -  *Aishwarya Singh*

    March 5, 2019 at 5:33 pm

    Hi Robert,

    Shared the dataset via mail.

    Reply

    -  *Bhanu*

      March 21, 2019 at 10:15 pm

      Can you please share the data set

Thank you

Reply

-  *Aishwarya Singh*

  March 25, 2019 at 7:51 pm

  Hi Bhanu, the link for dataset is in the article itself

  Reply

-  *Emerson Oliveira*

  January 20, 2019 at 7:16 am

  Why 987 in:

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

  Reply

  -  *Aishwarya Singh*

    January 22, 2019 at 6:33 pm

    Hi Emerson,

    I have used 4 years data for training and 1 year for testing. Splitting at 987 distributes the data in required format.

    Reply

-  *Emerson Oliveira*

  January 20, 2019 at 10:04 pm

  How can I solve this problem:

  ModuleNotFoundError: No module named ‘fastai.structured’

  Thanks

  Reply

  -  *Aishwarya Singh*

    January 22, 2019 at 6:53 pm

    Hey,

    Go to this link: https://github.com/fastai/fastai and clone/download. The error should be resolved.

    Reply

-  *Pramesh Bajracharya*

  January 22, 2019 at 4:00 pm

  Amazing Article. Very helpful . Thanks.

  Reply

-  *@run*

  January 27, 2019 at 7:33 pm

  Hey Aishwarya, your article is super helpful. Thanks a ton. Keep going!!!

  Reply

  -  *Aishwarya Singh*

    January 28, 2019 at 11:24 am

    Thanks Arun!

    Reply

-  *Doaa*

  January 29, 2019 at 7:32 am

  Hi Aishwarya,
  please can you tell me about books explain how to use LSTM in stock market in details
  thanks in advance

  Reply

-  *MikeMikeMike*

  January 30, 2019 at 4:15 am

Aishwaryai, I found your article very interesting. I'm not familiar with programing but I was able to follow the reason for each step in the process.

I do have a question.

You noted that there are many other factors that will ultimately affect the market. As someone who works in marketing and has to be plugged into social media I use many tools that tell me how a company is being perceived. Twitter and Facebook both offer lot of opportunity for social listening and the tools we use in advertising/marketing to measure take the temperature of the market are quite powerful.

Would it be possible to incorporate some machine learning to find patterns in the positive/negative sentiment measurements that are constantly active to help predict when stock values are going to change and in which direction? I feel like your article is using current and past market data but doesn't incorporate social perception.

http://www.aclweb.org/anthology/W18-3102

The study linked above only focuses on stock related posts made from verified accounts but didn't take public perception measurements into account in their data. With how connected to social media the world has become I wonder if combining the data you used in your article with data from social listening tools used by marketers could create predictive tools that would essentially game the system.

Reply

-  *smit*

  March 9, 2019 at 5:47 pm

  u can use sentimental analysis…..rest u can search yourself…
  ….u can watch sirajvideo….for more info

  Reply

-  *Muralidharan*

  February 11, 2019 at 8:42 am

Hi
How I can solve this error?

AttributeError: ‘DataFrame’ object has no attribute ‘Date’

Thanks

Reply

-  *Muralidharan*

  February 11, 2019 at 8:45 am

  Hi
  After dropping the date column(Linear Regression method), I am getting an error like this.

  AttributeError: ‘DataFrame’ object has no attribute ‘Date’

  How do I resolve this error?

  Reply

-  *Muralidharan*

  February 11, 2019 at 12:29 pm

  Hi

  Could please share the entire working code.

  Thanks

  Reply

-  *Justin*

March 22, 2019 at 12:04 am

Hi,

Thanks for the article. Regarding the following line: –

#predicting 246 values, using past 60 from the train data
inputs = new_data[len(new_data) – len(valid) – 60:].values

May I know what value obtained for : –
len(new_data) ?
len(valid) ?
inputs ?

Reply

-  *Aishwarya Singh*

  March 27, 2019 at 10:54 am

  Hi Justin,

  Valid and closing price length is 248, input is 308, and new data 1235 same as the len of df

  Reply

-  *Tobias Wang*

  March 24, 2019 at 6:35 am

  Hi Aishwarya,

  I was suspicious of your program, since it worked it little too well, so I played around with the program, and it after I tried to make it predict some future stocks (by making the validation set go to the future, and filling the rows with zeroes), and you can imagine my surprise when the prediction said the stocks would drop like a stone. I investigated further, and found that your data leaks in the LSTM implementation, and that’s why it works so well.

  Reply

-  *Aishwarya Singh*

  March 25, 2019 at 7:44 pm

  Hi,

  Don’t fill the rows with zero! fill it with the predicted value. So you made a prediction for next day, use that to predict the third day. Even if you do not use the validation set as done here, use the predictions by your model. Giving it zero as input for the last 2-3 days, the model would understand that yesterday’s closing price was zero, and will show a drastic drop.

  Reply

  -  *Tobias Wang*

    March 25, 2019 at 8:49 pm

    I don’t think you understand. All the rows with zeroes are stored in the validation set, where it shouldn’t be seen at all. The last training data point is on March to 19th (I am using Google NASDAQ data), and the first few data points are actual stock values. How am I supposed to fill it with predicted values when I can’t make it predict? When I fill it with NaN’s, it just doesn’t predict.

    My point is that your LSTM implementation uses the data it is not supposed to uses, to predict the very same data. Your program is leaking data, and it’s kinda misleading for the reader of this article, since in reality this model has much worse accuracy than the one shown here.

    I think the leaking data can be attributed to the lines that use the MinMaxScaler, as this is a common cause of data leakage.

    Thank you

    Reply

-  *SurGyan*

  May 10, 2019 at 5:30 am

  Tobia,
  It was interesting observation you have, but you're missing her intuition. Here, even if you think there is a data leak from LSTM model, what you're missing is the RMSE values, hence it looks like overfitting or may be as u said it looks like data leak.
  If you forget LSTM for a while, even for any predictions, we try to predict the Y in the test data, if you tweak it by making zeroes, then u r probably missing the point here. We don't tweak the test (unseen) data. Try not to fill the test data manually but with your predictions.
  Regards,
  SurGyan.
  PS: Much appreciated Aishwarya, for the detailed explanation.

  Reply

-  *BMI*

  May 15, 2019 at 12:40 am

  I think it is allowed to use known data. Maybe "Short Term Memory" is meaning to use latest known data to help predicition. I think LSTM does this, and it does not use future data. Combination of learning and use latest known data is best way.

  Thanks for very good article.

  Reply

-  *rekha*

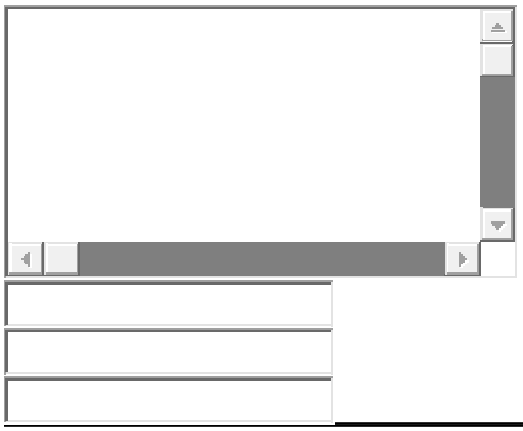
  May 17, 2019 at 11:09 pm

  hI,
  Which are the other sequence prediction time series problems where LSTM can be applied

  Reply

Leave A Reply

Your email address will not be published.



Notify me of new posts by email.

Join the nextgen data science ecosystem

- Get access to free courses on Analytics Vidhya
- Get free downloadable resource from Analytics Vidhya
- Save your articles
- Participate in hackathons and win prizes

Popular posts

- 24 Ultimate Data Science Projects To Boost Your Knowledge and Skills (& can be accessed freely)
- Commonly used Machine Learning Algorithms (with Python and R Codes)
- 7 Types of Regression Techniques you should know!
- A Complete Python Tutorial to Learn Data Science from Scratch
- Understanding Support Vector Machine algorithm from examples (along with code)
- Stock Prices Prediction Using Machine Learning and Deep Learning Techniques (with Python codes)
- Introduction to k-Nearest Neighbors: A powerful Machine Learning Algorithm(with implementation in Python)
- Complete Guide to Parameter Tuning in XGBoost with codes in Python

Recent Posts

The AI Comic: Z.A.I.N – Issue #2: Facial Recognition using Computer Vision

June 17, 2019

Build a Machine Learning Model in your Browser using TensorFlow.js and Python

June 14, 2019

An Introduction to the Powerful Bayes' Theorem for Data Science Professionals

June 13, 2019

Comprehensive Guide to Text Summarization using Deep Learning in Python

June 10, 2019