

Summary of Feasibility of Using Stacked LSTM Model for Stock Market Predictions





Data


Stock to be examined will be Tesla (NASDAQ: TSLA) over a 5 year evaluation

Data is copied from YahooFinance and updated to a Github page for easier access

1260 lines (1260 sloc) | 91.8 KB

RawBlame





Search this file...

1	Date	Open	High	Low	Close	Adj Close	Volume
2	2016-12-12	192.800003	194.419998	191.039993	192.429993	192.429993	2438876
3	2016-12-13	193.179993	201.279999	193.000000	198.149994	198.149994	6823884
4	2016-12-14	198.740005	203.000000	196.759995	198.690002	198.690002	4150927
5	2016-12-15	198.410004	200.740005	197.389999	197.580002	197.580002	3219567
6	2016-12-16	198.080002	202.589996	197.600006	202.490005	202.490005	3796889
7	2016-12-19	202.490005	204.449997	199.839996	202.729996	202.729996	3488071
8	2016-12-20	203.050003	209.000000	202.500000	208.789993	208.789993	4689071
9	2016-12-21	208.449997	212.229996	207.410004	207.699997	207.699997	5207622
10	2016-12-22	208.220001	209.990005	206.500000	208.449997	208.449997	3111108

Using ML to Predict TSLA Stock

It is known that in the financial/investment field that machine learning and deep learning can be used to make investment decisions.

LSTM models are known to be able to predict time series data, therefore I decided to create a stacked LSTM model and see how well it can perform on predicting stock market prices.

I will only be considering the daily closing value as my feature of interest; however for a more accurate model, internal performance metrics such as sales revenue, customer acquisition costs, and other such metrics can be used.

Design

Prepare Data

Auto Regression

Stacked LSTM Model

Evaluate

x1	x2	x3	x4	y
12	15	24	36	22
15	24	36	22	33
24	36	22	33	47
36	22	33	47	11
-	-	-	-	-
164	188	194	111	146
188	194	111	146	???

```
model = models.Sequential([
    layers.Input(shape=(100,1)),
    layers.LSTM(50, return_sequences=True),
    layers.LSTM(50, return_sequences=True, recurrent_dropout=0.2),
    layers.LSTM(50),
    layers.Dropout(0.2),
    layers.Dense(50),
    layers.Dense(1),
])
```

WARNING:tensorflow:Layer lstm_4 will not use cuDNN kernels since it doesn't

```
[ ] model.compile(
    loss = losses.MeanSquaredError(),
    optimizer = optimizers.Adam(),
    metrics = ['mean_squared_error'],
)
model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #

lstm_3 (LSTM)	(None, 100, 50)	10400
lstm_4 (LSTM)	(None, 100, 50)	20200
lstm_5 (LSTM)	(None, 50)	20200
dropout_1 (Dropout)	(None, 50)	0
dense_1 (Dense)	(None, 50)	2550
dense_2 (Dense)	(None, 1)	51

Total params: 53,401		
Trainable params: 53,401		
Non-trainable params: 0		

Design

Prepare Data

Data is preprocessed to use the first 80% as training set and last 20% as test set.

Next the data is normalized using SKLearns built in standard scalar.

```
[ ] # Split the data into training and test data
    train_size = int(len(df_close)*0.8)
    test_size = len(df_close)-train_size
    train_split,test_split = df_close[0:train_size], df_close[train_size:len(df_close)]

    # Apply Normalization to both
    #scaler = MinMaxScaler(feature_range=(0,1))
    scaler = StandardScaler()
    train_data, test_data = scaler.fit_transform(np.array(train_split).reshape(-1,1)), scaler.fit_transform(np.array(test_split).reshape(-1,1))
```

Design

Auto Regression

Data needs to be further processed before feeding into the model.

Method to convert the time-series data to a useable form was taken from Machine Learning Mastery.

```
[ ] def create_dataset(dataset, look_back=1):  
    dataX, dataY = [], []  
    for i in range(len(dataset)-look_back-1):  
        a = dataset[i:(i+look_back), 0]  
        dataX.append(a)  
        dataY.append(dataset[i + look_back, 0])  
    return np.array(dataX), np.array(dataY)
```

x1	x2	x3	x4	y
12	15	24	36	22
15	24	36	22	33
24	36	22	33	47
36	22	33	47	11
-	-	-	-	-
164	188	194	111	146
188	194	111	146	???

Design

LSTM Model

The model consists of 3 LSTM layers, with 2 dropout layers to assist in reducing overfitting, followed by 2 dense layers. The model is then compiled using the Adam optimizer with Mean Squared Error as the loss metric.

```
[ ] model = models.Sequential([
    layers.Input(shape=(100,1)),
    layers.LSTM(50, return_sequences=True),
    layers.LSTM(50, return_sequences=True, recurrent_dropout=0.2),
    layers.LSTM(50),
    layers.Dropout(0.2),
    layers.Dense(50),
    layers.Dense(1),
])
```

```
[ ] model.compile(
    loss = losses.MeanSquaredError(),
    optimizer = optimizers.Adam(),
    metrics = ['mean_squared_error'],
)
model.summary()
```

Design

Evaluate Model

After training, we evaluate the model using Mean Squared Error and Root Mean Squared Error.

The test set had a MSE of 0.14 and RMSE of 0.37.

These are acceptable values for MSE and RMSE, which means that while the model can somewhat predict stock movement, it would require further refinement before being used in real-life investments.

```
[ ] import math
_, mse_train = model.evaluate(x_train, y_train)
print("MSE: %.2f RMSE: %.2f" % (mse_train, math.sqrt(mse_train)))
```

```
29/29 [=====] - 2s 38ms/step - loss: 0.1061 - mean_squared_error: 0.1061
MSE: 0.11 RMSE: 0.33
```

```
[ ] _, mse_test = model.evaluate(x_test, y_test)
print("MSE: %.2f RMSE: %.2f" % (mse_test, math.sqrt(mse_test)))
```

```
5/5 [=====] - 0s 43ms/step - loss: 0.1369 - mean_squared_error: 0.1369
MSE: 0.14 RMSE: 0.37
```


Summary of Evaluation

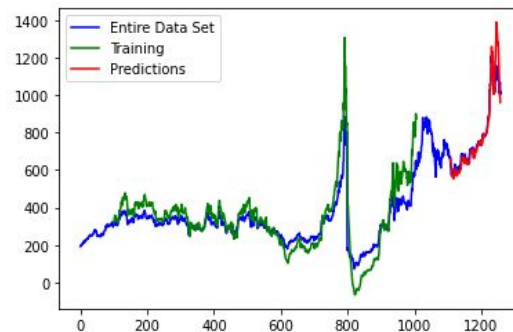
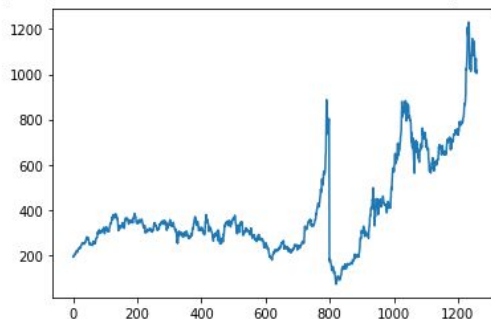
```
[ ] import math
_, mse_train = model.evaluate(x_train, y_train)
print("MSE: %.2f RMSE: %.2f" % (mse_train, math.sqrt(mse_train)))
```

```
29/29 [=====] - 2s 38ms/step - loss: 0.1061 - mean_squared_error: 0.1061
MSE: 0.11 RMSE: 0.33
```

```
[ ] _, mse_test = model.evaluate(x_test, y_test)
print("MSE: %.2f RMSE: %.2f" % (mse_test, math.sqrt(mse_test)))
```

```
5/5 [=====] - 0s 43ms/step - loss: 0.1369 - mean_squared_error: 0.1369
MSE: 0.14 RMSE: 0.37
```

[<matplotlib.lines.Line2D at 0x7fefbd3c5e10>]



Citations:

<https://machinelearningmastery.com/time-series-prediction-with-deep-learning-in-python-with-keras/>

<https://machinelearningmastery.com/reshape-input-data-long-short-term-memory-networks-keras/>

<https://finance.yahoo.com/>