



Algorithmic trading using LSTM-models for intraday stock predictions

Machine Learning (CSN-526)

Anuj Gupta 18114008

Rahul Atul Kumar Jain 18114061

Subhash Suryawanshi 18114076



Objective

- The objective of this project is to make a model which can better give daily return in intraday trading.
- In this project, We investigate LSTM models for return predictions on a portfolio of stocks.

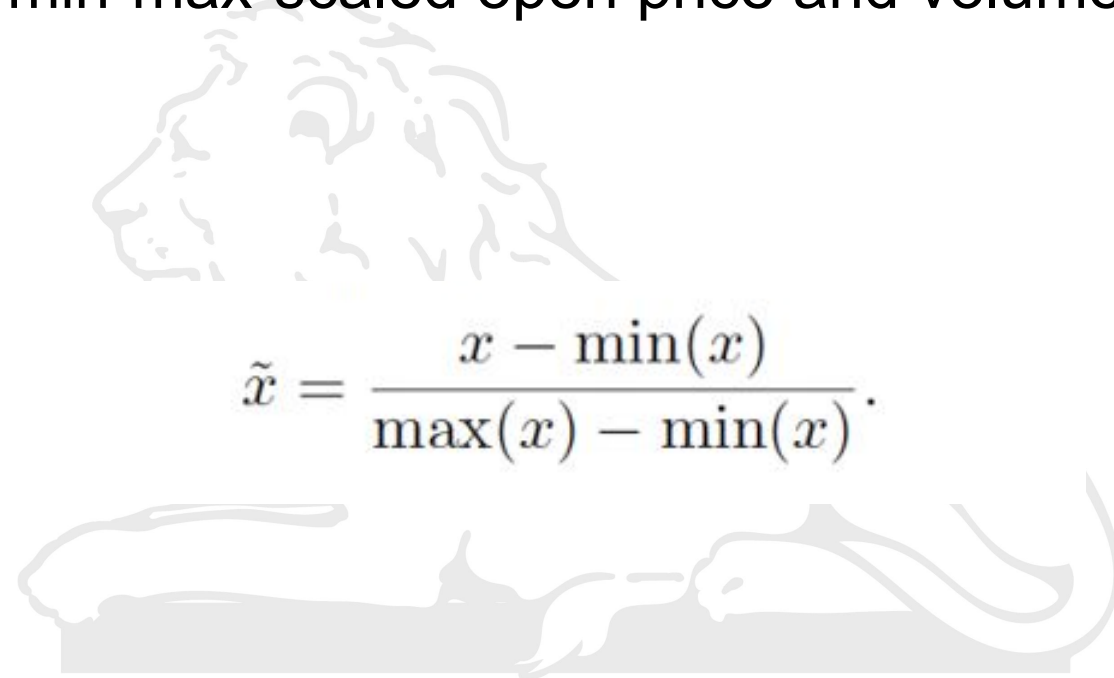


Data Set

- We used a Dataset between 9/11/2017 to 2/16/2018 of S and P stocks.
- In particular, we used a portfolio of 30 stocks.
- We wanted the stocks to correlated in order to improve the accuracy of the model. For that we used stocks in the information technology sector.
- Features include: Open, Close, High, Low, Volume.

Data preprocessing

We ended up using 4 ground features for training the model. These were previous returns, difference between high and low and min-max-scaled open price and volume.

A faint, stylized background illustration of a lion and a bull, likely representing the Ashoka Lion Capital and the Bull Capital, which are historical landmarks of IIT Roorkee.
$$\tilde{x} = \frac{x - \min(x)}{\max(x) - \min(x)}.$$

General Implementation Details

LSTM

- The key idea behind LSTM is that each layer has a memory, ability to forget not useful information, and remember beneficial information.
- LSTM is a recurrent neural network (RNN) architecture that REMEMBERS values over arbitrary intervals. LSTM is well-suited to classify, process and predict time series given time lags of unknown duration.

Techniques



The table below shows the architecture of our network.

lstm_1 (LSTM)	(None, 24, 4)	144	input_1[0][0]
lstm_2 (LSTM)	(None, 24, 4)	144	input_2[0][0]
lstm_3 (LSTM)	(None, 24, 4)	144	input_3[0][0]
lstm_4 (LSTM)	(None, 24, 4)	144	input_4[0][0]
lstm_5 (LSTM)	(None, 24, 4)	144	input_5[0][0]
lstm_6 (LSTM)	(None, 24, 4)	144	input_6[0][0]
lstm_7 (LSTM)	(None, 24, 4)	144	input_7[0][0]
lstm_8 (LSTM)	(None, 24, 4)	144	input_8[0][0]
lstm_9 (LSTM)	(None, 24, 4)	144	input_9[0][0]
lstm_10 (LSTM)	(None, 24, 4)	144	input_10[0][0]

dropout_1 (Dropout)	(None, 24, 4)	0	lstm_1[0][0]
dropout_2 (Dropout)	(None, 24, 4)	0	lstm_2[0][0]
dropout_3 (Dropout)	(None, 24, 4)	0	lstm_3[0][0]
dropout_4 (Dropout)	(None, 24, 4)	0	lstm_4[0][0]
dropout_5 (Dropout)	(None, 24, 4)	0	lstm_5[0][0]
dropout_6 (Dropout)	(None, 24, 4)	0	lstm_6[0][0]
dropout_7 (Dropout)	(None, 24, 4)	0	lstm_7[0][0]
dropout_8 (Dropout)	(None, 24, 4)	0	lstm_8[0][0]
dropout_9 (Dropout)	(None, 24, 4)	0	lstm_9[0][0]
dropout_10 (Dropout)	(None, 24, 4)	0	lstm_10[0][0]
concatenate_1 (Concatenate)	(None, 24, 40)	0	dropout_1[0][0] dropout_2[0][0] dropout_3[0][0] dropout_4[0][0] dropout_5[0][0] dropout_6[0][0] dropout_7[0][0] dropout_8[0][0] dropout_9[0][0] dropout_10[0][0]
lstm_11 (LSTM)	(None, 10)	2040	concatenate_1[0][0]
dropout_11 (Dropout)	(None, 10)	0	lstm_11[0][0]
dense_1 (Dense)	(None, 10)	110	dropout_11[0][0]
Total params: 3,590			
Trainable params: 3,590			
Non-trainable params: 0			

Techniques

- The idea is that the first individual LSTM layer should remember individual dependencies for that stock and the second joint LSTM unit should remember joint dependencies.
- We train on a customized mean squared error that penalizes for getting the sign wrong.
- Hyper parameter tuning gave that learning rate = 0.05 , lag order = 37, LSTM units in first layer = 6.

Techniques



Number of LSTM units: 6

Lookback period: 37

Learning rate: 0.05

Average MSE: 4.747296338099813e-06

- If model predict a positive return, take long position. Else, take short position.
- We obtain very modest results in terms of standard metrics such as accuracy, mean and squared error.
- Our model perform better than the original project authors model in terms of accuracy compared on 10 stock in test data.

Original Results



Stock	MSE	Acc	Return	SR
ACN	3.782e-6	0.497	−1.560	−0.542e-2
AMAT	1.304e-5	0.493	−10.59	−2.271e-2
CDNS	9.580e-6	0.465	−19.79	−5.488e-2
IBM	3.532e-6	0.493	−9.408	−4.065e-2
INTU	4.508e-6	0.508	5.539	2.114e-2
LRCX	1.214e-5	0.477	−12.98	−2.979e-2
NTAP	1.979e-5	0.478	−4.568	−6.003e-2
VRSN	6.005e-6	0.515	8.589	2.782e-2
WU	6.867e-6	0.437	−4.337	−1.207e-2
XLNX	6.992e-6	0.478	−5.290	−1.493e-2

Table 4. Individual stock metrics using the market strategy.

Our Results



Stocks	MSE	Return	Accuracy
ACN	1.1210047957525346e-06	5.064670464981935	Accuracy: 0.5062111801242236
AMAT	6.762053338460228e-06	11.174851755532922	Accuracy: 0.5054347826086957
CDNS	1.3527416394618483e-06	7.905599043318334	Accuracy: 0.5372670807453417
IBM	5.701618800715873e-06	7.555271033228106	Accuracy: 0.5054347826086957
INTU	2.329144613580076e-06	6.807442393233787	Accuracy: 0.5178571428571429
LRCX	7.23974819622584e-06	10.36040468506152	Accuracy: 0.5124223602484472

Precision, Recall, F1 Score

Tags	ACN	AMAT	CDNS	IBM	INTU	LRCX	NTAP	VRSN	WU	XLNX
True positive	443	550	531	119	260	316	614	304	177	136
True Negative	193	101	65	532	407	344	49	340	451	525
False positive	441	548	626	93	223	287	570	326	250	148
False Negative	211	89	66	544	398	341	55	318	410	479
Precision	0.5011312217	0.5009107468	0.4589455488	0.5613207547	0.5383022774	0.5240464345	0.5185810811	0.4825396825	0.4145199063	0.4788732394
Recall	0.6773700306	0.8607198748	0.8894472362	0.1794871795	0.3951367781	0.4809741248	0.9177877429	0.4887459807	0.3015332198	0.2211382114
F1 score	0.5760728218	0.6332757628	0.6054732041	0.272	0.4557405784	0.5015873016	0.6627091203	0.4856230032	0.349112426	0.3025583982
						Average	0.4844152617		Minimum(F1)	0.272
										IBM

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

$$\text{F1 Score} = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

ScreenShot Confusion Matrix



```
232 print('True positive:', TP)
233 print('True Negative:', TN)
234 print('False positive:', FP)
235 print('False Negative:', FN)
236 print('Precision:', TP/(TP+FP) )
237 print('Recall:', TP/TP+FP)
238 Precision= TP/(TP+FP)
239 Recall=TP/TP+FP
240 print('F1 Score:', 2*(Precision*Recall)/(Precision+Recall))
```

Code Snippet

```
===== ACN =====
Dummy MSE: 8.677873494102536e-07
MSE: 1.1210047957525346e-06
--
Predicted ones: 0.6863354
Real ones: 0.5077639751552795
True positive: 443
True Negative: 193
False positive: 441
False Negative: 211
Precision: 0.5011312217
Recall 0.6773700306
F1 score 0.5760728218
Accuracy: 0.5062111801242236
===== AMAT =====
Dummy MSE: 6.402671680502051e-06
MSE: 6.762053338460228e-06
--
Predicted ones: 0.85248446
Real ones: 0.49611801242236025
True positive: 550
True Negative: 101
False positive: 548
False Negative: 89
Precision: 0.5009107468
Recall 0.8607198748
F1 score 0.6332757628
Accuracy: 0.5054347826086957
```

Output

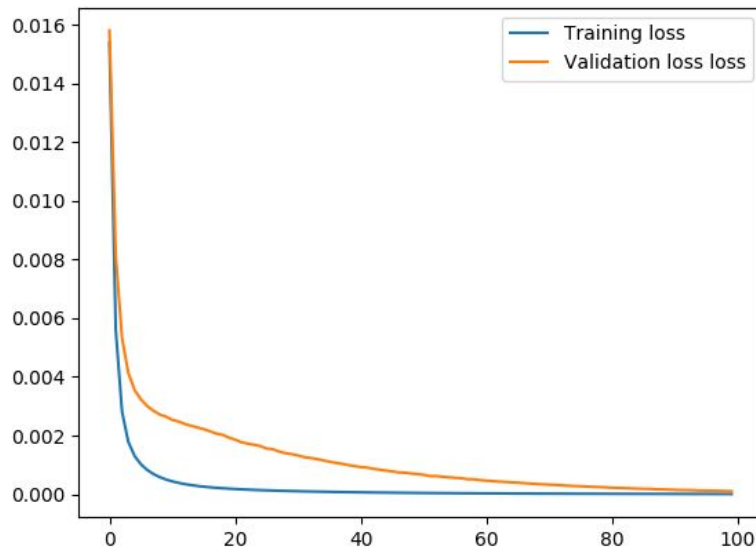
(8 more Stocks are there) **I I T ROORKEE** ■ ■ ■

Comparative Analysis

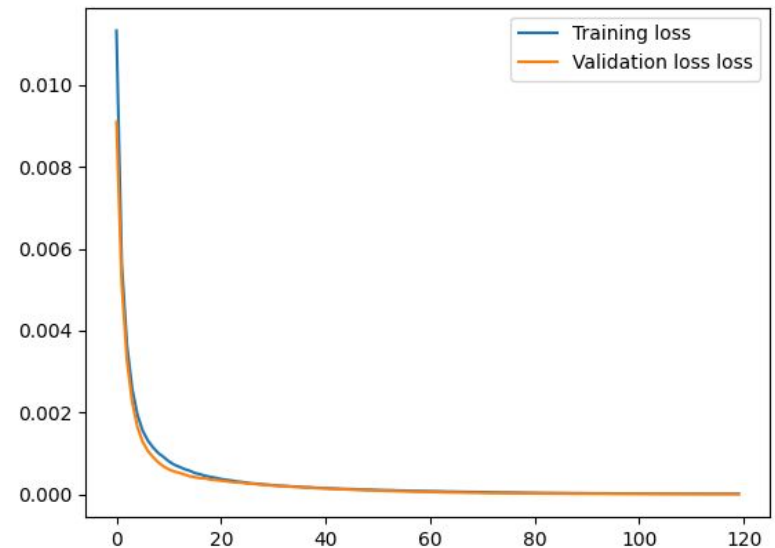
- On the test set, our model got positive return on 9 stocks out of 10 whereas original author's implementation got positive return on just 7 stock out of 10.
- Our MSE(average of the squared error) is in order of $x \cdot 10^{-6}$ whereas original MSE is order of $x \cdot 10^{-5}$, so our model is better here as well.
- We were constrained by inadequate resources, otherwise we could have experimented with training with more and more LSTM layers.

Training and Validation Loss

- Here Blue Line represents Training Loss.
- Here Yellow Line represents Validation Loss
- X-axis show number of epochs and
- Y-axis shows the two Losses.



Stanford model



Our model

Novel Contributions

- Our LSTM RNN-Network contains 10 LSTM layers which makes in way deeper than author's network architecture (5 LSTM layers).
- We trained the LSTM model on comparably higher dataset, roughly double the dataset then what the authors used.
- We used Random Hyperparameter tuning whereas authors used deterministic parameter tuning.

ThankYou
