Stock Prediction Models Using Deep Learning

Sirish Prabakar, Dane Jew

Stock Prediction Models Using Deep Learning

Abstract

Deep Learning models can be used to predict time dependent outcomes. This can be demonstrated by using the RNN and CNN models to predict the closing price of stocks.

Problem Statement

The goal of this project is to create a timebased stock price prediction tool using deep learning models. This system should be able to predict the price of a stock based on the past 7 days.

Methodology

The models to be used in this project are a Fully Connected Neural Network (FCNN), a Recurrent Neural Network (RNN) and a Convolution Neural Network (CNN). The RNN model will be implemented using the LSTM library from Keras.

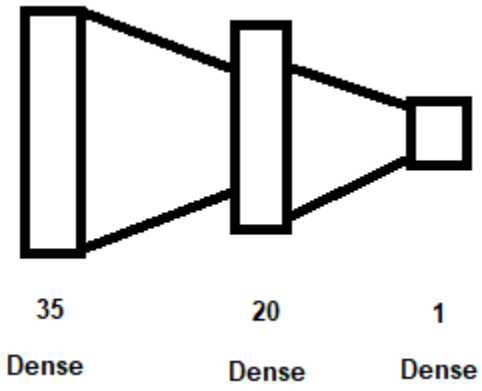
Dataset

The dataset is a csv file downloaded from a Google drive network share. The records are a list of stock prices, each with the following 5 attributes: Open, High, Low, Volume, and the Close. Each row represents a successive trading day for this particular stock. This data will need to be preprocessed, meaning the inputs will need to be reshaped depending on the model.

In the FCNN model there is no provision for time. The input is represented by the 5 features: "open", "high", "low", "volume" and "close". The last attribute, "close" can be used as the target variable. In order to represent 7 consecutive days the input needs to be "flattened" to an input of 7 x 5 or 35 features. The shape of the data will ultimately be a 1 x 35 array.

Figure 1

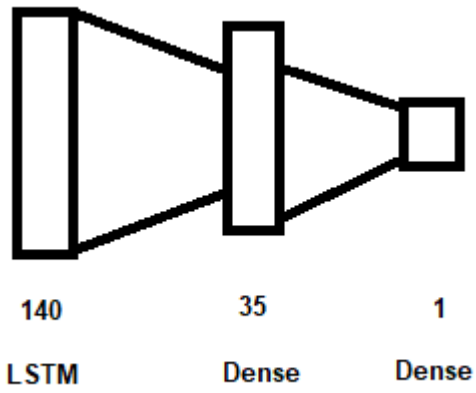
FCNN Model



In the RNN model, the data needs to be in the form of vectors of 7 x 5 element arrays. Each successive vector needs to represent a moving window to reflect consecutive days of trading. For instance the first day of trading would be a vector of v[0 .. 6] while the next day would be v [1 .. 7]. The target variable for day one is y[7]["open"]. In addition the input x needs to be a 3-dimensional array in order to utilize the LSTM library.

Figure 2

LSTM (RNN) Model



The CNN model requires the dataset to be viewed as a vector of images. This is similar to the data formatted for the LSTM / RNN model except the dimensions does not have to be in 7x5 vectors.

Experimental Results and Analysis

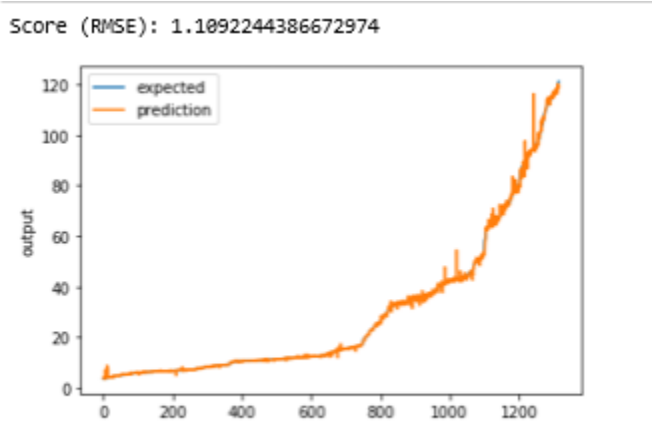
Fully-Connected Neural Network (FCNN).

The result of the FCNN is very close between the predicted price versus the actual price. The RMSE score is close to 1.11.

The best parameter to use of activation is "relu". In trying the other options for activation, "tanh" and "sigmoid" didn't turn out to be a good options since both activation functions caused the F-1 score to drop.

Figure 3

FCNN Lift Chart



RNN Using LSTM.

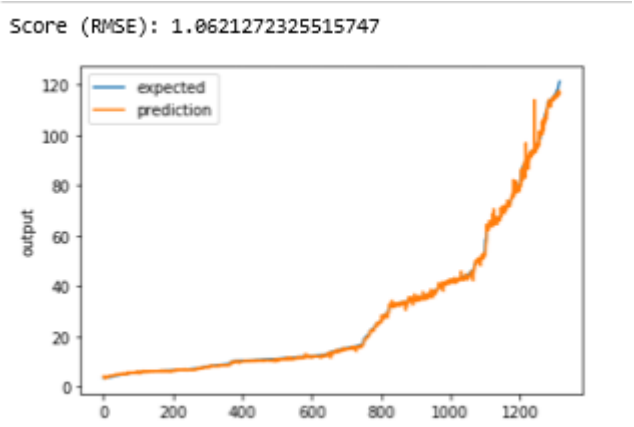
I comparison to the FCNN, the LSTM model appears to be more accurate. The RMSE score for the LSTM is 1.06. Only one LSTM layer was used with this model with an input size of 7x5.

Changing the activation functions and other parameters had minimal effect on F-1 score. Increasing the number of layers and nodes resulted in very small increase(negligible) in F-1 score for this dataset. When the layers and number of nodes were minimal, this dataset gave us almost a perfect F-1 score. So by increasing number of layer/nodes did not have a noticeable impact on the F-1 score. The decision was made to set the input layer 20 neurons and intermediate layer having 10 neurons.

The optimizer was set to "adam", since "sgd" was not a good option. F1 score obtained by "sgd" keeping other parameters same, caused the F-1 score to drop.

Figure 4

RNN (LSTM) Lift Chart



Convolution Neural Network (CCN).

A variety of pixel dimensions was used, but the best one was (4385,1,7,5).

When the kernel size was set to (3,3) and the kernel numbers was set to (64,128), the test loss became 3996.435 with a final RMSE of 63.217. When the kernel size was decreased to 2,2, the test loss changed to 4359.245 and RMSE to 66.024. When the number of kernels decreased, the best score of the test loss turned out to be 3901.449 and RSME to 62.461.

Task Division and Project Reflection

The work breakdown is as follow in the table below:

Coding

	Work Pct
Dane Jew	50%
Sirish Prabakar	50%

Documentation

	Work Pct
Dane Jew	50%
Sirish Prabakar	50%

Challenges

The biggest challenge as always is preprocessing the data. This requires analysis of the data and knowledge and familiarity of the tools, such as Numpy. It is especially difficult to format the data for the LSTM and CNN models.

In implementing LSTM, was unable to employ checkpoint since it generated an error, so it was excluded.

The cuda toolkit and cudNN that was included with the anaconda installation were not compatible with the latest version of tensorflow. This prevented us from converging our CNN model. For this reason we had to uninstall Cuda and cudNN and re- install the latest version in the correct directory. This was time consuming as well as confusing and resulted us in spending a lot of time researching to overcome this problem.

The CNN model uses 4D array as an input and this required us to not only visualize its representation in 4D but also put it down in the correct form of code. In other words we had to find the best way to present this non image data as an image while still giving us the highest possible accuracy. This was challenging but helped us understand in deep about the physical layout of such high dimensional arrays.

Additional Features

1) In the project, you predict [Close] of a day based on the last 7 days' data. Can you find the best N value (number of the days we should consider in the past) that yields the most accurate model

Answer:

After trying different values for number of days N between 4 to 14, it seems that 10 days produces the best results. This is only if you set the LSTM layer to 30, hidden layer to 20, and the early stopping "patience" parameter to 12. The number of epochs was set to 1000. The RSME turned out to be 0.96.

4) In the shared Google drive folder, you can find the stock price for the following companies.

- o Royal Dutch Shell
- o Apple
- o Google
- o JPMorgan

Based on your observations from the project, build a good model for stock price prediction. Show RMSE and regression lift chart.:

Answer:

The model that seems to perform the best for stock prediction of individual companies seem to be the FCNN model. The layers are "dense", 30, 20, 1. Early stopping patience is set to 7 with epochs of 1000.

In running the tests, both Google2, Royal Dutch Shell and JPMorgan seems to be very accurate, as they both generated RMSE's of 9.3,0.801 and 0.797 respectively. Apple on the other hand did not perform as well, it generated RMSE's of 24.572 and. For whatever reason these stocks seem to cause the model to overfit drastically.

2 more additional features are used. Details about these can be found in Additional features.txt.

Figure 5

Royal Dutch Shell Lift Chart

```
Score (RMSE): 0.8014251589775085
```

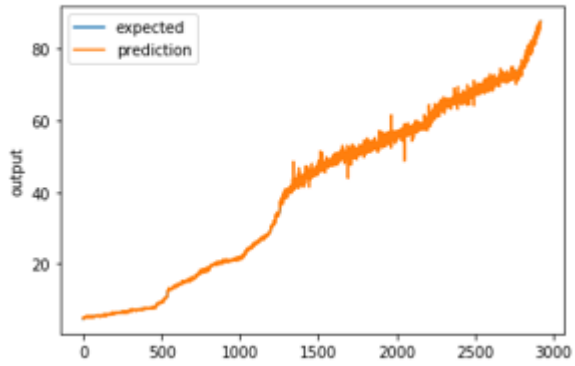


Figure 6

JPMorgan Lift Chart

```
Score (RMSE): 0.7970456480979919
```

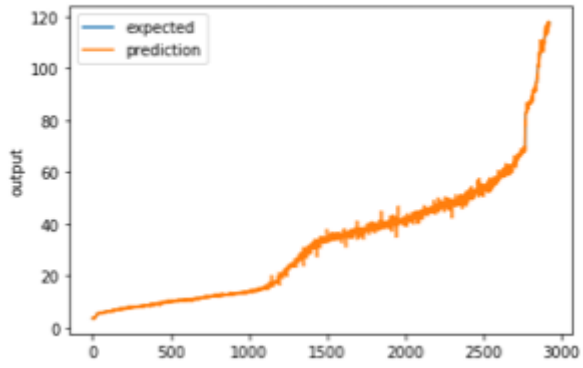


Figure 7

Apple Lift Chart

```
Score (RMSE): 24.5727596282959
```

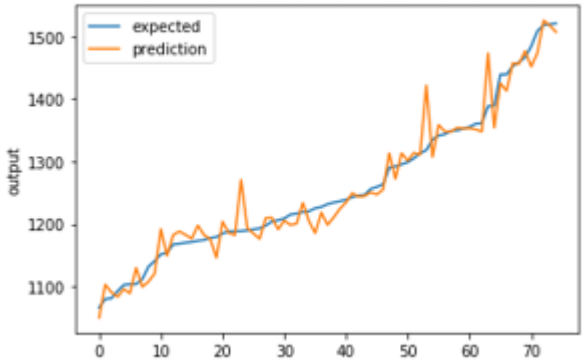


Figure 8

Google Lift Chart 2(3500 rows)

Score (RMSE): 9.386615753173828

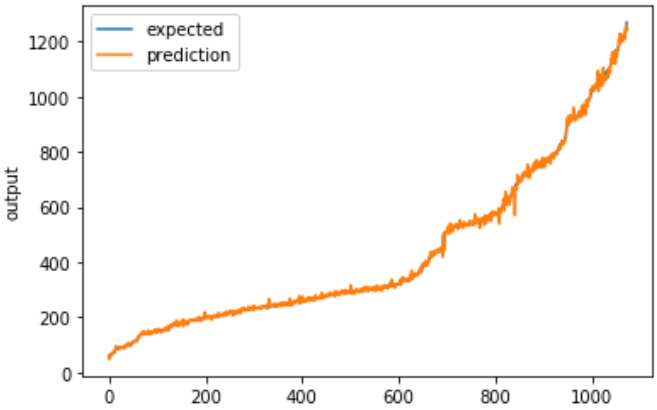
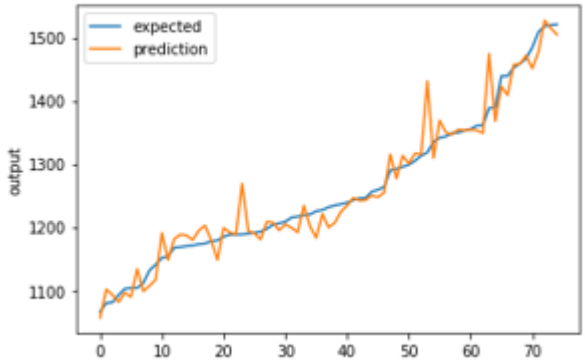


Figure 8

Google Lift Chart 1(250 rows)

```
Score (RMSE): 25.2864933013916
```



Conclusion

The Deep Learning models have a clear advantage to other models when predicting time dependent outcomes such as stock price predictions. RNN models, especially LSTM, are specifically designed to analyze data at multiple states. This is apparent in the way the dataset is formatted as input, as the attributes of the stock prices are represented as states. While tradition Neural Networks can also be used for the same task, it still views the data as a single state. The data for multiple days had to be flattened and treated as attributes of a single state entity. Deep Learning models ultimate have the advantage of allowing the creation of complex models beyond temporal and multi-state analysis.