Stock Prediction

Multivariate Time Series Forecasting with LSTM

Jesse King, Ryan Little, James Park, Keshia Vereda

Overview

- The team used Machine Learning (ML) to predict stock market information.
- The model selected looked at stock data quarterly and daily data to help forecast opening prices to help make the buying process easier.
- Compared stocks from different sectors to see the impact of other influential factors (ex. COVID, politics) had an impact on the accuracy of the model.

Libraries used:

- Python Pandas
- Python MatPlotLib
- Python NumPy
- Sci-Kit Learn
- Keras
- TensorFlow
- HTML / CSS / BootStrap
- Web Scraping (Finnhub, Yahoo, and Financial Model Prep)

LSTM vs Traditional Models

- Traditional machine learning models use input features like samples and examples to learn which lacks a time dimension feature
- Time-series forecasting models predict future values with values that are constantly changing
- LSTM was chosen because of its layering format (each mini-batch is passed and is sent to the next layer the “forward pass” but the results are preserved incase a “backward pass is needed.”)

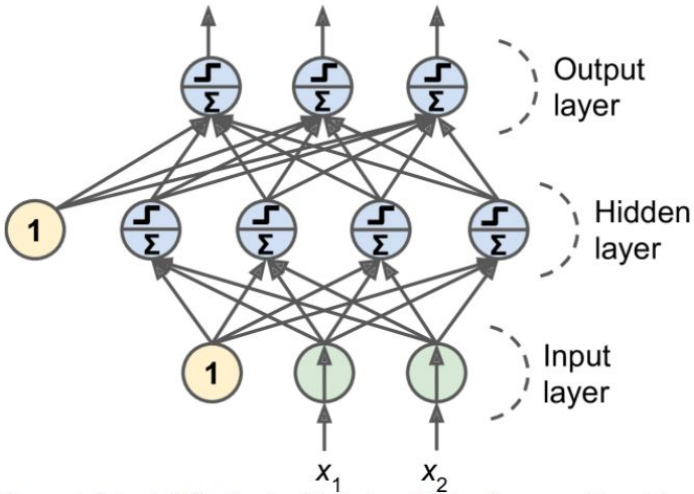

Figure 10-7. Architecture of a Multilayer Perceptron with two inputs, one hidden layer of four neurons, and three output neurons (the bias neurons are shown here, but usually they are implicit)

Data pre-processing

- Data pulled from Financial Modeling Prep, Yahoo and Finhub
  - Used Quarterly and Daily and found that Quarterly was harder to use for prediction purposes
  - Best: Financial Modeling Prep
- Data must be in float format in order to be processed into a matrix
- One feature vs Multiple features:
  - Forecasting improves with additional features
  - Features : Open, High, Low, Close, Volume, RSI, market price, volatility index
- Standard scaler was used for each type of variables
  - Independent (High, Low, Close, Volume, RSI, market price, volatility index)
  - Dependent (open prices)

Training a Multivariate LSTM -

Hyperparameters used:

- 2 LSTM layers
- 1 Dropout layer at 25% (regularization technique to prevent overfitting)
- Dense (1 output)
- Optimizer used: Adam
  - Mix of AdaGrad + RMSprop
  - Used to calculate individual learning rates for each parameter

Our Model:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 90, 64)            20992
_________________________________________________________________
lstm_1 (LSTM)                (None, 10)                3000
_________________________________________________________________
dropout (Dropout)            (None, 10)                0
_________________________________________________________________
dense (Dense)                (None, 1)                 11
=================================================================
Total params: 24,003
Trainable params: 24,003
Non-trainable params: 0
```

Testing and Validating

- Validation Split : 20% of our data before shuffling for validation
- Validation Error is computed at the end of each epoch
- Training Error is computed during each epoch
- Training loss and validation loss decrease with each pass

```
4/4 [==============================] - 1s 140ms/step - loss: 0.0548 - val_loss: 1.062
Epoch 45/50
4/4 [==============================] - 1s 145ms/step - loss: 0.0434 - val_loss: 1.037
Epoch 46/50
4/4 [==============================] - 1s 208ms/step - loss: 0.0401 - val_loss: 1.039
Epoch 47/50
4/4 [==============================] - 1s 201ms/step - loss: 0.0387 - val_loss: 1.016
Epoch 48/50
4/4 [==============================] - 1s 207ms/step - loss: 0.0405 - val_loss: 0.991
Epoch 49/50
4/4 [==============================] - 1s 209ms/step - loss: 0.0355 - val_loss: 1.013
Epoch 50/50
4/4 [==============================] - 1s 199ms/step - loss: 0.0379 - val_loss: 0.985
```

```
model.evaluate(X_train, y_train)
```

```
36/36 [==============================] - 0s 8ms/step - loss: 0.2077

0.20769251883029938
```

Forecasting:

Demo site:

https://mrryanlittle.github.io

References:

DataSeries. *Data-Preprocessing with Python.* Ajay, Muktha Sai. Sep. 29, 2020.
https://medium.com/dataseries/data-preprocessing-with-python-3914d3e9dd30

Financial Modeling Prep. https://financialmodelingprep.com/developer

Multivariate Time Series Prediction with LSTM and Multiple features (Predict Google Stock Price)
https://www.youtube.com/watch?v=gSYiKKoREFI

O'Reilly. *Hands-on Machine learning with Scikit-Learn, Keras & TensorFlow.* Aurelien Geron 2nd Edition

Towards Data Science. *In 12 Minutes: Stocks Analysis with Pandas and SciKit-Learn.* Tatan, Vincent. May 26, 2019.
https://towardsdatascience.com/in-12-minutes-stocks-analysis-with-pandas-and-scikit-learn-a8d8a7b50ee7

Towards Data Science. *Machine Learning for Stock Prediction, A Quantitative approach.* Malato, Gianluca. Feb. 27, 2020.
https://towardsdatascience.com/machine-learning-for-stock-prediction-a-quantitative-approach-4ca98c0bfb8c

Towards Data Science. *What Happened When I tried Market Prediction with Machine Learning.* Bowling, Jason. Nov. 22, 2019.
https://towardsdatascience.com/what-happened-when-i-tried-market-prediction-with-machine-learning-4108610b3422

Yahoo! Finance. http://finance.yahoo.com