Stock Price Command-line Predictor
Udacity Machine Learning Engineer Capstone Project

Renee Shiyan Liu

2020-05-21

Keywords: Deep Learning, Stock Price Prediction, LSTM, time series forecasting, Argparse command-line

1. Project Overview

This project builds an easy-to-use command-line stock predictor that can make predictions on a particular stock's prices on nultiple days in the future. The underlying model is a 2-layer LSTM neural network.

The goal is to help a user quickly make buysell decisions on a particular stock through the terminal.

2. Project Statement

Casino industry is really getting the worst impact due to coronavirus. As a casino mecca, Las Vegas is one of the hardest-hit place. Almost all casino stock prices crashed overnight. The silver lining is that this might give investors a once-in-a-lifetime opportunity to invest in these stocks once the Coronavirus is defeated or the lock-down is lifted.

The immediate question is what stocks should you have in a casino portfolio? Some stocks seem to rebound really well, and some are not. This project intends to choose some of the casino stocks that seem to shield the shock much better than others. After all, we small investors care about stability and long-term values more than anything else.

This project intends to use different algorithms to formulate a prediction model that can inform investors the stock prices for the next 7 days (7 is the default period). In addition, this resulting command-line app gives the user a recommendation of buy and sell on a particular day(s).

3. Dataset and inputs

Downloaded 6 publicly traded casino stocks from Yahoo finance. These time series data are daily prices include open, close, high, low, and adjusted closing price.

I used each time series independently to test the accuracy of the model. I also used them to fine tune the parameters because they are all casino stocks so they can be seen as a pool of training dataset.

This is similar to cross-validation when using non time series data.

4. Analysis

4.1. Data Pre-processing, Exploration, and Visualization

Since these are daily time series data with no missing values, there aren't much to be done about pre-processing.

The nature of this project task is univariate time series prediction, visualizing the patterns of these data series is the best way to explore the pattern of the data.

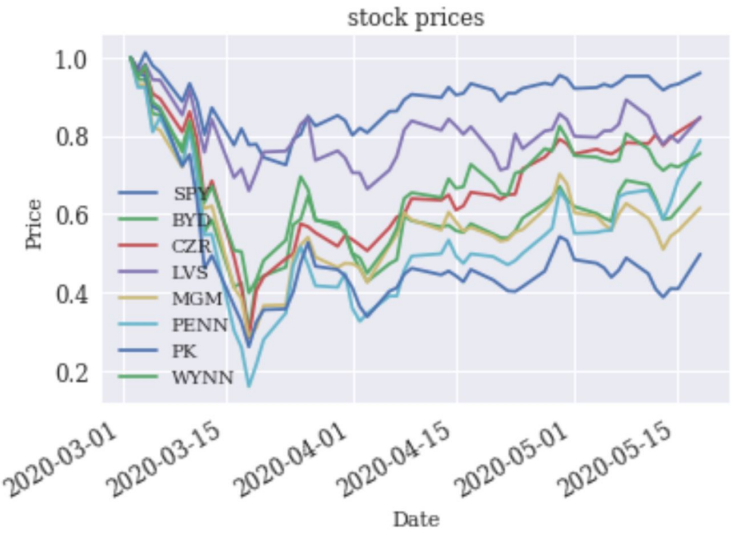

Figure 1: Stocks Price Plot

4.2. Benchmark model

I will use plain and simple time series analysis to be the benchmark model. Precisely, I will use multi-step time series prediction to predict the 7 days predictions. Then, compare the results with the LSTM model predictions later on.

4.3. Algorithms and Techniques

Three algorithms and/or techniques are used.

First, I used Moving Average as a benchmark prediction. This is traditionally used in time series forecast and serves as a good start to evaluate the trend in a time series. I used a rolling window length of 50. I plot the result below,

Figure 2: Moving Average with 50 rolling window

Figure 2 suggests that Moving Average prediction performs very bad because it only captures the nominal values of the past few days. This suggests that we should use a model that captures deeper information within the time series.

Then, I used Auto-ARIMA model that takes into account the error from past predictions. The chose parameters minimize the prediction errors. I plot the predictions and actual values in Figure 3.

ARIMA certainly improves because the the predicted values certainly follows the trend of the actual values.

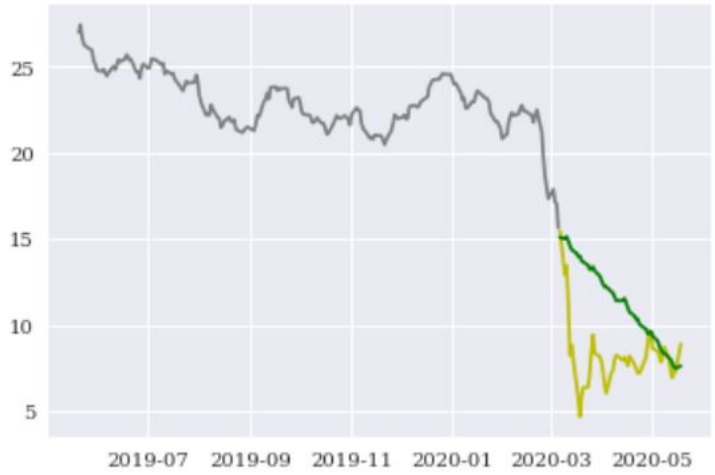

Figure 3: Auto ARIMA Model Prediction

5. Methodology

5.1. Pipeline Overview

I decided to use LSTM as my final model choice. This is because LSTM model is naturally suited to this task, where complex past information needs to be retained to make future predictions.

I used Keras to build up a simple 2-layer LSTM architecture for this task.

5.2. Preprocessing & Implementation

Following Keras documentation, I reshaped the feature data accordingly. This mainly involves adding dimensions to input arrays. Specifically, LSTM model needs 3-dimensional features that of the shape (n, t, d), where n, t, d refer to number of samples, time steps, and number of features respectively.

Then I used MinMax Scaler to preprocess the data as this is a required step in fitting Keras LSTM model. One thing that is worth of mentioning is that this same scaler is needed to pre-process any test data.

Then I split the dataset into a train and test datasets using 80-20 rule.

The implementation is straightforward once the input data is in the right shape and the model is correctly defined. I manually I will present the model architecture here for your reference.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 10, 32)            4352
_________________________________________________________________
lstm_2 (LSTM)                (None, 16)                3136
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 17
=================================================================
Total params: 7,505
Trainable params: 7,505
Non-trainable params: 0
_________________________________________________________________
```

Figure 4: Model Architecture - The Classifier Layer

This architecture is suitable to this particular prediction task because we need the model to capture the past information in a sequential order and LSTM is designed to do that. Figure 5 below will show visually how well it performs.

6. Results

The green line shows the predicted prices for the actual prices (in red). We see that it is surprisingly accurate. In this case, the prediction seems to be a bit more optimistic than the actual prices. However, this model beats every other traditional time series techniques.

7. Conclusion

To recapture the pipeline for this project,

1. In the development phase, I tried different time series approaches including auto ARIMA model, moving average prediction, and neural network approach.
2. Root Mean Squared Error (rmse) is used as the model metric.
3. Using visualization to help decide on which model is best suited to this prediction task.
4. Build a 2-layer LSTM model to output multiple-step price predictions.

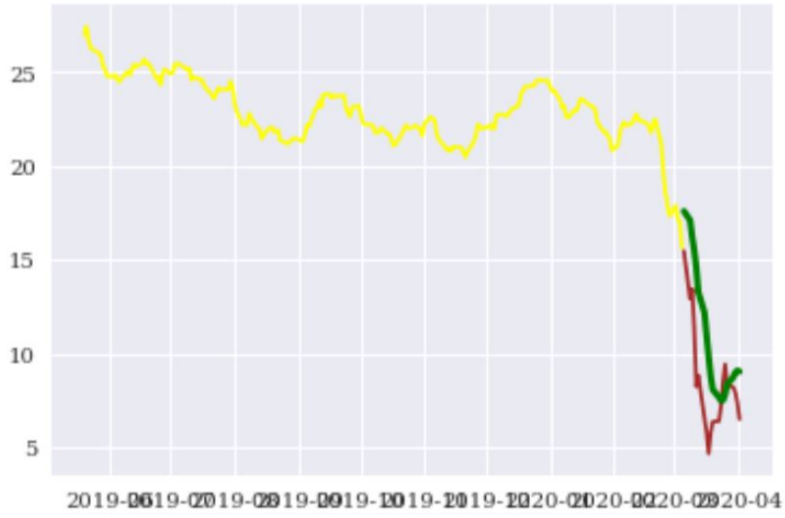

Figure 5: Model Architecture - The Classifier Layer

5. Validate the model and display it visually.
6. Fine Tune a set of hyperparameters which include epochs and timesteps.
7. Built a command line interface script that take in user inputs and output price predictions.
8. Output recommended day(s) of buys and sells directly from the command line.

What I found particularly difficult about this project is to make the correct input dimension for the LSTM model use. The other part I need to work more on is to understand the data better so that I can find the right way to design the model architecture.

8. Refinement

This project has a lot of potentials to be improved upon. A list of refinement I plan to do in the future includes,

- Tune the LSTM model by changing hyperparameters.
- Try out different architectures.

- Try to hook the command script with Yahoo finance API. This can make the process much more automatic. The user doesn't have to download the data anymore.
- Try to more rigorous on the choices of timesteps when training LSTM model.

9. References

- Classification Metrics
- How to choose time series prediction metrics
- how to develp time series model
- Time Series Forcasting performance
- LSTM fundamentals
- LSTM helpful tutorial