Project Report

Stock Prediction using Machine Learning

Team Members:

Jyoti Nikam (893453050)

Snehal Jadhav (893557538)

CPSC 531-01 (12877)

Spring, 2018

Professor: Dr. Chun-l Phillip Chen

Department of Computer Science

California State University, Fullerton

May 08, 2018

Table of Contents

Table of Figures

Abstract

Investment firms, hedge funds and even individuals have been using financial models to better understand market behavior and make profitable investments and trades. A wealth of information is available in the form of historical stock prices and company performance data, suitable for machine learning algorithms to process.

Investors make educated guesses by analyzing data. They'll read the news, study the company history, industry trends and other lots of data points that go into making a prediction. The prevailing theories is that stock prices are totally random and unpredictable but that raises the question why top firms like Morgan Stanley and Citigroup hire quantitative analysts to build predictive models. About 70% of all orders on Wall Street are now placed by software, we're now living in the age of the algorithm.

This project utilizes Deep Learning models, Long-Short Term Memory (LSTM) Neural Network algorithm, to predict stock prices. For data with timeframes recurrent neural networks (RNNs) come in handy but recent researches have shown that LSTM, networks are the most popular and useful variants of RNNs.

Chapter 1: Introduction

Background

Much of the hype surrounding neural networks is about image-based applications. However, Recurrent Neural Networks (RNNs) have been successfully used in recent years to predict future events in time series as well. RNNs have contributed to breakthroughs in a wide variety of fields centered around predicting sequences of events. In this piece, however, we'll demonstrate how one type of RNN, the Long Short-Term Memory (LSTM) network, can be used to predict even financial time series data—perhaps the most chaotic and difficult of all-time series.

Project Goal

The goal of this project is to accurately predict the future closing value of a given stock across a given period in the future. For this project I will use a Long Short-Term Memory networks – usually just called "LSTMs" to predict the closing price of the 1 S&P 500 using a dataset of past prices

- Explore stock prices.
- Implement basic model using linear regression.
- Implement LSTM using Keras library.
- Compare the results and submit the report.

Relevance and Significance

Share market is very uncertain and complicated to predict the stock prices as it depends on many factors. Some fundamental factors that drive stock prices:

1. Economic Strength of Market and Peers
2. Inflation
3. Substitutes
4. Incidental Transactions
5. Demographics
6. Trends
7. Liquidity

Stock exchanges are considered major players in financial sectors of many countries. Most

Stockbrokers, who execute stock trade, use technical, fundamental or time series analysis in trying to predict stock prices, so as to advise clients. However, these strategies do not usually guarantee good returns because they guide on trends and not the most likely price. It is therefore, necessary to explore improved methods of prediction. Developing a machine learning based stock predictive system is a non-trivial and challenging task. As there many research, and development are currently ongoing to find the predictive system that is more optimal and provides a better result.

Definition of Terms

- LSTM:

  Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. This is a behavior required in complex problem domains like machine translation, speech recognition, and more.

- S&P 500:

  This index tracks 500 large U.S. companies across a wide span of industries and sectors. The stocks in the S&P 500 represent roughly 70 percent of all the stocks that are publicly traded.

- Dow Jones:

  Named after Charles Dow, this index tracks the 30 largest U.S. companies. This means it represents “large-cap” companies, which is the industry term for “very big companies” like Johnson & Johnson, McDonald's, and Coca-Cola.

- NYSE Composite:

  The NYSE Composite is a stock market index covering all common stock listed on the New York Stock Exchange, including American depositary receipts, real estate investment trusts, tracking stocks, and foreign listings.

Chapter 2: Review of the Literature

Data Exploration

The data used in this project is retrieved from Google Finance API which may belong to stock market indexes: Dow Jones, NYSE Composite or S&P 500. You may select the time in years which results in a series of data points indexed in time order or a time series. Our goal is to predict the closing price for any given date after training. The prediction has to be made for Closing (Adjusted closing) price of the data. Since Google Finance already adjusts the closing prices for us, we just need to make prediction for "CLOSE" price.

The dataset is of the form:

Date	Open	High	Low	Close	Volume
30-Jun-17	943.99	945.00	929.61	929.68	2287662
29-Jun-17	951.35	951.66	929.60	937.82	3206674
28-Jun-17	950.66	963.24	936.16	961.01	2745568

Figure 1: Snapshot from local file 'Google.csv'

The mean, standard deviation, maximum and minimum of the data is represented as:

Feature	Open	High	Low	Close	Volume
Mean	382.5141	385.8720	378.7371	382.3502	4205707.8896
Std	213.4865	214.6022	212.08010	213.4359	3877483.0077
Max	1005.49	1008.61	1008.61	1004.28	41182889
Min	87.74	89.29	86.37	87.58	521141

Figure 2: Statistics of the dataset

We can infer from this dataset that date, high and low values are not important features of the data. As it does not matter at what was the highest prices of the stock for a day or what was the lowest trading prices. What matters is the opening price of the stock and closing prices of the stock. If at the end of the day we have higher closing prices than the opening prices that we have some profit otherwise we saw losses. Also, volume of share is important as a rising market should see rising volume, i.e., increasing price and decreasing volume show lack of interest, and this is a warning of a potential reversal. A price drop (or rise) on large volume is a stronger signal that something in the stock has fundamentally changed.

Therefore, we have removed Date, High and low features from data set at preprocessing step. The mean, standard deviation, maximum and minimum of the preprocessed data was found to be following:

	Mean	Std	Max	Min
Open	0.3212	0.23261	1.0	0.0
Close	0.3215	0.2328	1.0	0.0
Volume	0.09061	0.0953	1.0	0.0

Figure 3: Statistics of the processed dataset

Exploratory Visualization

To visualize the data, we have used matplotlib library. We have plotted Closing 7 stock price of the data with the no of items (no of days) available. Following is the snapshot of the plotted data:

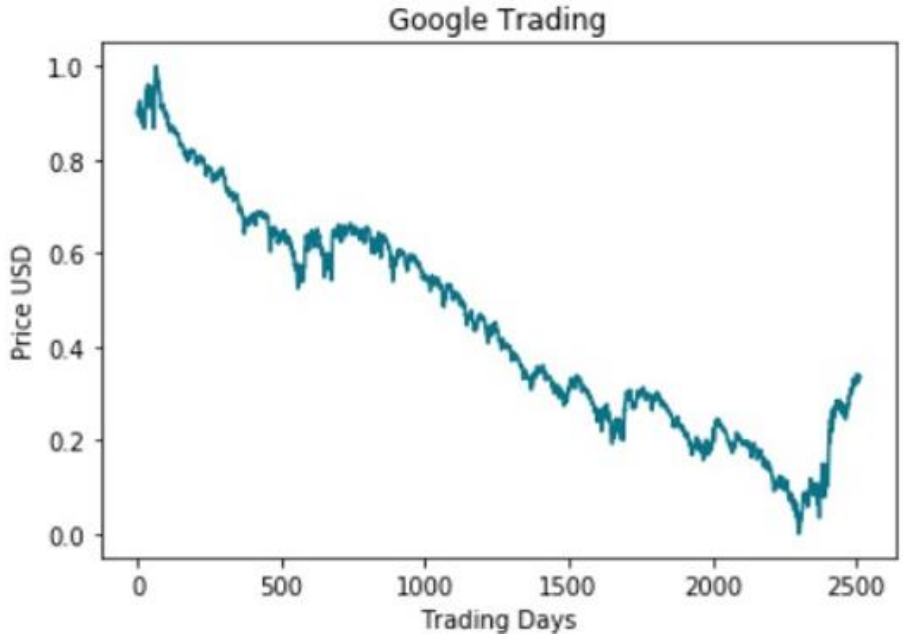

Figure 4: Visualizing closing stock price of data

Algorithms and Techniques

The goal of this project is to study time-series data and explore as many options as possible to accurately predict the Stock Price. Through the research, we came to know about Recurrent Neural Nets (RNN) which are used specifically for sequence 8 and pattern learning. As they are networks with loops in them, allowing information to persist and thus ability to memorize the data accurately. But Recurrent Neural Nets have vanishing Gradient descent problem which does not allow it to learn from past data as was expected. The remedy of this problem was solved in Long-Short Term Memory Networks, usually referred as LSTMs. These are a special kind of RNN, capable of learning long-term dependencies. In addition to adjusting the architecture of the Neural Network, the following full set of parameters can be tuned to optimize the prediction model:

- Input Parameters
    - Preprocessing and Normalization

- Neural Network Architecture
    - Number of Layers (how many layers of nodes in the model)
    - Number of Nodes (how many nodes per layer)
- Training Parameters
    - Training / Test Split (how much of dataset to train versus test model on)
    - Validation Sets (kept constant at 0.05% of training sets)
    - Batch Size (how many time steps to include during a single training step)
    - Optimizer Function (which function to optimize by minimizing error)
    - Epochs (how many times to run through the training process)

Benchmark Model

For this project, we have used a Linear Regression model as its primary benchmark. As one of the goals is to understand the relative performance and implementation differences of machine learning versus deep learning models. This Linear Regressor was used for error rate comparison MSE and RMSE utilizing the same dataset as the deep learning models.

Predicted results:

Figure 5: Visualizing predicted results of benchmark model

Chapter 3: Methodology

Data Analysis and Modelling

We are using Google Finance Client API for retrieving the raw stock data related to each index. Googlefinance.client is a python client library for google finance API. It fetches current or historical stocks information from Google Finance. We are fetching the data of three most popular indexes:

- Dow Jones
- NYSE Composite
- S&P 500

Data preprocessing

Acquiring and preprocessing the data for this project occurs in following sequence, much of which has been modularized into the preprocess.py file for importing and use across all notebooks:

1. Request the data from the Google Finance Python API and save it in **google.csv** file in the following format:

Date	Open	High	Low	Close	Volume
30-Jun-17	943.99	945.00	929.61	929.68	2287662
29-Jun-17	951.35	951.66	929.60	937.82	3206674
28-Jun-17	950.66	963.24	936.16	961.01	2745568

Figure 6: Raw data format

2. Normalize the data using **MinMaxScalar** helper function from Scikit-Learn and store it in **google_preprocessed.csv** file for future reusability.

Item	Open	Close	Volume
0	0.012051	0.015141	0.377248
1	0.014198	0.010658	0.325644
2	0.009894	0.010112	0.189820
3	0.010874	0.007407	0.242701

Figure 7: Normalize the raw data

3. Split the dataset into the training and test datasets for linear regression model.
4. Split the dataset into the training and test datasets for LSTM model.

Software and Libraries

- Python 2.7

  Python is an interpreted high-level programming language for general-purpose programming.

- NumPy

  NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

- Pandas

  Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

- Keras

  Keras is an open source neural network library written in Python. It can run on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or MXNet.

- TensorFlow

  TensorFlow is an open-source software library for dataflow programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks.

- Jupyter Notebook

  The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

Technical Approach

1. Acquire the data for preprocessing.
2. Implement following steps for prediction:

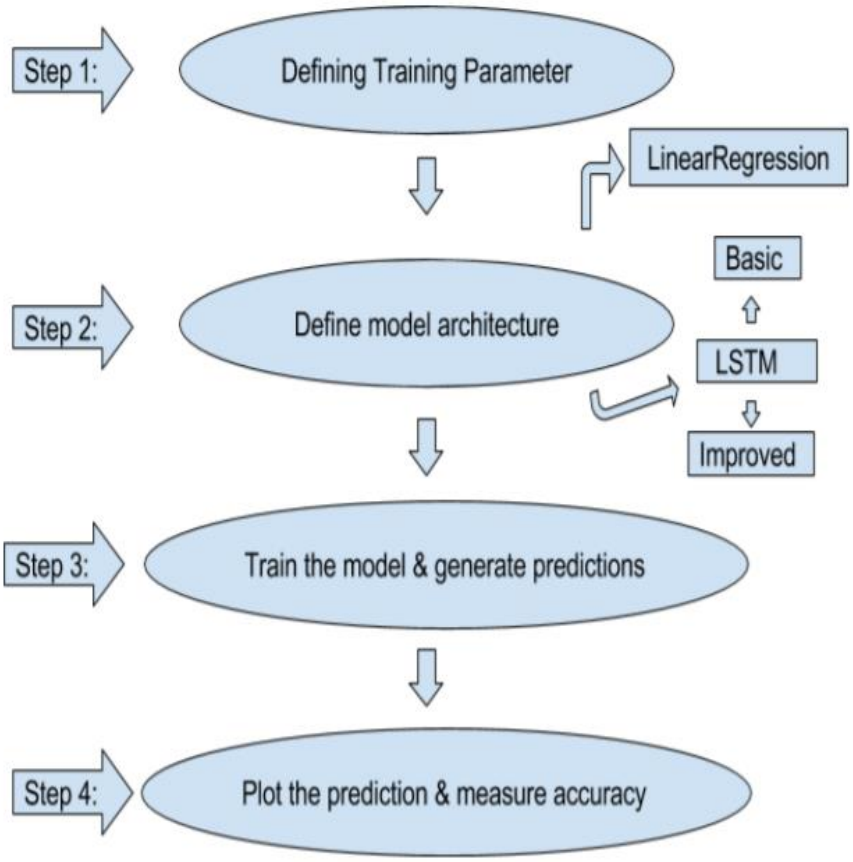

Figure 8: The repeatable implementation process

Proposed Methodology

This project is implemented through the Keras/Tensor Flow library using LSTM Neural Networks. Development workflow follows the below sequence:

1. Set Up Infrastructure
    - Python Notebook
    - Incorporate required Libraries (Keras, Tensor flow, Pandas, Matplotlib, Sklearn, Numpy)
2. Prepare Dataset
    - Incorporate data of the selected Index.
    - Process the requested data into Pandas Data frame.
    - Develop function for normalizing data.
    - Dataset will be used with 80/20 split on training and test data across all models.
3. Develop Benchmark Model
    - Set up basic Linear Regression model with Scikit-Learn
    - Calibrate parameters
4. Develop Basic LSTM Model
    - Set up basic LSTM model with Keras utilizing parameters from Benchmark Model.
5. Improve LSTM Model
    - Develop, document, and compare results using additional labels for the LSMT model.
6. Document and Visualize Results
    - Plot Actual, Benchmark Predicted Values, and LSTM Predicted Values per time series.
    - Analyze and describe results for report.

Chapter 4: Results and Discussion

Model Evaluation and Validation

With each model, we have refined and fined tune our predictions and have reduced mean squared error significantly.

1. For the first model: Linear Regression Model

   Train Score: 0.7877 MSE (0.8875 RMSE)

   Test Score: 0.51226856 MSE (0.71572939 RMSE)

Figure 9: Visualizing predicted results of linear regression model

2. For the second model: Basic Long-Short Term Memory Model

   Train Score: 0.03999493 MSE (0.19998733 RMSE)

   Test Score: 0.07217106 MSE (0.26864673 RMSE)

Figure 10: Visualizing predicted results of LSTM model

Refinement

For this project, we have worked on fine tuning parameters of LSTM to get better predictions. We did the improvement by testing and analyzing each parameter and then selecting the final value for each of them.

To improve LSTM, we have -

- Increased the number of hidden node from 100 to 128.
- Added Dropout of 0.2 at each layer of LSTM.
- Increased batch size from 1 to 512.
- Increased epochs from 1 to 20.
- Added verbose = 2.
- Made prediction with the batch size.
- Added a dense layer.

For our third and final model, using improved Long-Short Term memory model:

Train Score: 0.00049039 MSE (0.02214477 RMSE)

Test Score: 0.00070768 MSE (0.02660232 RMSE)

Figure 11: Visualizing predicted results of improved LSTM model

Justification

Comparing the benchmark model - Linear Regression to the final improved LSTM model, the Mean Squared Error improvement ranges from 0.51226856 MSE (0.71572939 RMSE) [Linear Regression Model] to 0.00070768 MSE (0.02660232 RMSE) [Improved LSTM]. This significant decrease in error rate clearly shows that our final model has surpassed the basic and benchmark model.

Chapter 5: Conclusions, Implications, and Recommendations

Conclusion

To recap, the process undertaken in this project:

- Set Up Infrastructure
    - iPython Notebook
    - Incorporate required Libraries (Keras, Tensor flow, Pandas, Matplotlib, Sklearn, Numpy)
    - Prepare Dataset.
    - Incorporate data of selected index.
    - Process the requested data into Pandas Dataframe.
    - Develop function for normalizing data.
    - Dataset used with a 80/20 split on training and test data across all models.
- Develop Benchmark Model
    - Set up basic Linear Regression model with Scikit-Learn.
    - Calibrate parameters
- Develop Basic LSTM Model
    - Set up basic LSTM model with Keras utilizing parameters from Benchmark Model.
- Improve LSTM Model
    - Develop, document, and compare results using additional labels for the LSMT model 5.
    - Document and Visualize Results.
- Plot Actual, Benchmark Predicted Values, and LSTM Predicted Values per time series/
- Analyze and describe results for report.

Implications

1. This application can be used by Investment firms, hedge funds, and even individuals to better understand market behavior and make profitable investments and trades.
2. This application may prove as a useful tool to understand relative performance and implementation differences of Linear Regression model versus LSTM model.

Recommendations

This project though predicts closing prices with very minimum Mean Squared Error, still there are many things that are lagging in this project.

Two of most important things are:

1. There is very less user interaction or interface provided in this project. A UI can be provided where user can check the value for future dates.
2. We can surely add more stock indexes in the list to make this project more comprehensive. We would definitely like to add these improvements to this project in future.

References

1. Predicting Stock Volume with LSTM. (n.d) Retrieved from https://sflscientific.com/data-science-blog/2017/2/10/predicting-stock-volume-with-lstm\

2. https:/ A simple deep learning model for stock price prediction using TensorFlow. (n.d) Retrieved from /medium.com/mlreview/a-simple-deep-learning-model-for-stock-price-prediction-using-tensorflow-30505541d877

3. Plain Stock Close-Price Prediction via LSTM. (n.d) Retrieved from https://isaacchanghau.github.io/2017/07/26/Plain-Stock-Close-Price-Prediction-via-LSTM-Initial-Exploration/

4. Stock Market Forecasting Using Machine Learning Algorithms Retrieved from http://cs229.stanford.edu/proj2012/ShenJiangZhang-StockMarketForecastingusingMachineLearningAlgorithms.pdf