Stanford University / Spring 2021

Predicting stock market price applying Deep Learning

Utkarsh Mittal
SUNet ID: mittal12
mittal12@stanford.edu

Abstract

Recurrent Neural network (RNN) has been used to achieve the state-of-the-art performance and to classify the key decision variable by a trader to buy, sell, or hold considering different financial parameters and daily stock prices. Decision variables (buy, sell, or hold) are built on the probability of up/down variation of the stock prices in the next seven trading days. The data set includes daily S&P 500 adjusted closing prices from 2010 onwards and leading/lagging indicators namely rolling simple moving averages, volatility in the stock price, daily returns, and daily trading volume.

1. Introduction

Profit is the objective of any business and when played with an edge it will grow faster. Stock market is not an exception rather sky may be the limit for the traders in the stock market. Gaining an edge should not come from being privy to stock market news prior to be known by anybody. The news aired has been heard and analyzed by thousands of people including by their grandma / grandpa. To make money it is necessary to take decisions consistently based on logics i.e. one must bring precisions in his decisions and avoid subjectivity in their outcome. In trading timing of the decision is very important i.e. when to enter/exit(buy/sell) a stock. This project explores various algorithms to scraps the real time data from internet and to come up with a winning strategy.

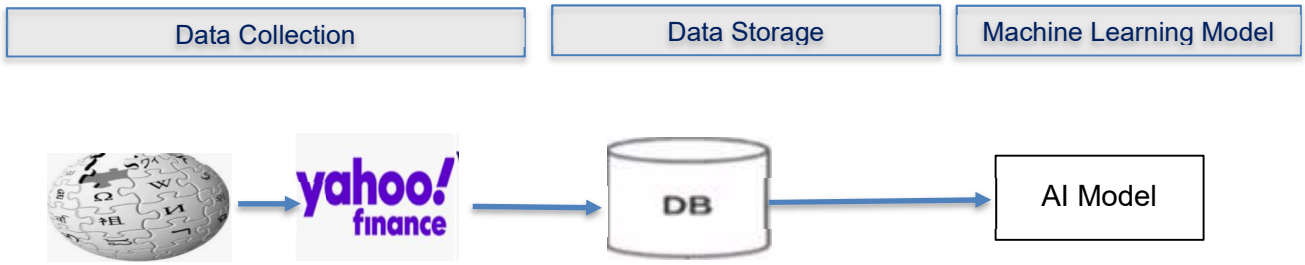
In this project, an attempt has been made to predict the price movement of all S&P 500 stocks and come up with an objective of right decision of buy, sell, or hold the stocks. Forty features have been identified to predict the movement in stock prices. More specific, classic features like Adjusted close Volume are coupled with trend features with different rolling windows namely simple moving averages, relative strength index and volatility have been introduced to develop a robust stock market prediction model.

2. Related Work

A plethora of research have been done using feature-based approaches or methods to extract a set of features that represent time series patterns. [6] Promising results have been obtained using Bag of words, Bag of features, Word extraction for classification of time series. [7] Ensemble algorithms also has produced promising results with time series classification problems. However, these methods failed to capture the complex non-linear nature of the stock market. [5] Lot of research being done on the theme of long-term investment using financial ratios and metrics. [1][2] Some research papers are focused on historical stock price and simple trend strategies.

3. Data set and features

3.1 Dataset -The dataset contains pricing data and the volume data for all the stocks in S&P 500 since 2010 until 2019. Web-scraping on **S&P 500 wiki page** was performed to collect the ticker symbols.



3.2 Features- The dataset consists of forty features which includes the data features extracted from the web for example Adjusted closing price, Trading volume. Several trend features were created like RSI and some based on moving averages. In addition to that, some lagged features related to volatility to predict the price movement of the stock in next seven days.

3.3 Preprocessing – Each stock data is standardized separately on the training sample of that stock. So that whole data has zero mean and unit variance. Processed data is then fed into the model to predict the target variable.

$$\hat{X}_i = \frac{X_i - X_{\text{train}}}{S_{X_{\text{train}}}}$$

3.4 Training/test split – Dataset collected from yahoo finance since 2010 and included 2000+ daily samples of each stock. The data is split among training/test sets into 90/10.

3.5 Target Variable- The most important aspect of this algorithm is the target variable which is buy, hold and sell decision at a given point. The decision variable is a forward-looking leading indicator which is based on the 2% sudden movement in any direction within a week. If the stock price is majoritily up within a week and is greater than 2% threshold then it's a buying opportunity and vice versa.

3.6 Class Imbalance- Class imbalance is a dominant issue when it comes to classification problem whether it's a traditional machine learning model or deep neural network. Majority class dominates and make the model more biased towards that class. In this paper, dynamic weights based on the bin counts have been applied to nullify the impact of the dominant class.

3.7 Threshold Selection- Threshold selection is an iterative step. It can vary by stock to stock based on the volatility and the riskiness in the stock. Also, Optimal threshold will enable the balance between the conservative strategy vs aggressive trading recommendation. For example, sudden drop of 10% in stock price will be a much better opportunity compared to 2% drop within a span of seven days. But it can limit the no. of trading and there might be weeks of no trading. On the other hand, lower threshold will increase the no. of trading but can impact the returns as possibility of higher losses increases. This will be further optimized when we back test the model for returns calculations and creating the trading bot.

4. Methods

This report compares traditional machine learning models with plain vanilla deep neural network and different variation of CNN and RNN/LSTM models.

4.1 Base Model -Voting classifier between {Random forest, and SVM} combines the best outcome of two models have been used to get the better accuracy. Base model well on the training dataset but it was barely better than a coin toss on the test dataset. Delta between the training and test dataset was very high which cause the problem of overfitting the training dataset.

4.2 Vanilla DNN Model - MLPClassifier a form of neural network was applied to perform the task of classification. This model optimizes the log-loss function using LBFGS or stochastic gradient descent. Different iterations of hidden layer sizes (set the number of layers and the number of nodes in the Neural Network Classifier) and max iterations (number of epochs) were tried to get the better accuracy and overcome the problem of overfitting. Vanilla DNN model mitigated the impact of overfitting but the accuracy did not improve much on the test dataset.

4.3 Multi Dense CNN Model – This is a sequential model with three hidden layers. Hidden layers with Relu activation function. The final activation layer has one dense layer with Adam optimizer. Least accuracy among all models

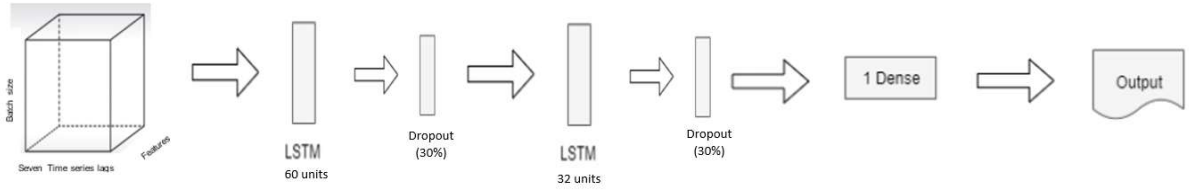
4.4 Multi Dense CNN Model with Dropout – This is an extension of 4.3 with five hidden layers and dropouts. With each hidden layer 20% dropout is added to avoid overfitting. No improvement in accuracy after applying dropout.

4.5 Multi Dense Layer CNN Model with L1 and L2 regularization – This is a variation of 4.4 to avoid overfitting. Instead of using dropout layers, L1 and L2 regularization were used. Similar results also witnessed here. No incremental improvement in accuracy.

4.6 Multi Dense Layer CNN Model with Dropout and different Optimization functions – This is an extension of 4.3 with different activation functions were compared for the accuracy. Different activation functions like SGD, Adam, Rmsprop were tried to increase the accuracy. SGD activation provided the highest accuracy among all CNN models.

4.7 Shallow RNN model with one hidden layer- This is a simple one LSTM layer with 32 hidden units. Input Training data is shuffled with buffer size of 300 and batches of 100 are created to pass through the initial LSTM layer. The final activation was a one-unit dense layer with a linear activation.

4.8 Dense RNN model with Dropout- The architecture for the deep LSTM consisted of two LSTM layers with 60 and 32 hidden units coupled with two dropout layer each with 30% dropout. Both the LSTM layers used Relu activation function. MAE is used as loss function.



5. Experimentation/Results and Error Analysis

5.1 Model Results

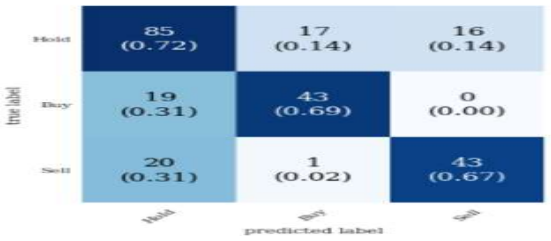
Model Name	Train Acc.	Test Acc.
Base Model (Voting Classifier)	87.5%	51.3%
Vanilla DNN Model (MLPClassifier)	61%	57.4%
Multi Dense CNN Model	25.1%	25.3%
Multi Dense CNN Model with Dropout	22.9%	25.27%
Multi Dense CNN Model with Reg	24.8%	25.2%
Multi Dense CNN Model with Dropout with diff. optimizers	61.2%	57.8%
Shallow RNN model with one hidden layer	76%	71%
Dense RNN model with Dropout	75.1%	54.5%

5.1 Model Evaluation and Hyperparameters tuning- The baseline model performed fairly good compared to the deep learning models but there is very delta between the training and test error which makes the model prone to overfitting the training dataset. All the CNN models performed worse, only exception was Multi dense model with diff optimizers. The test error and the training accuracy is close but model did slight better job then a single coin toss. The winning model is the Shallow RNN with one hidden layer where the training and test accuracy is comparably close, and model provides a winning chance of 71% given the volatility in the market and the stock.

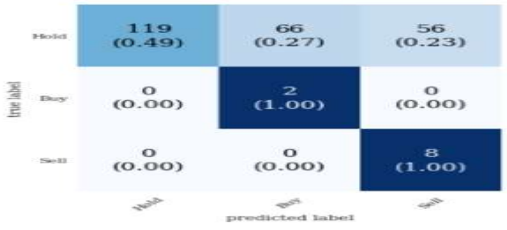
Shallow RNN with one hidden layer architectures used a regression loss function, specifically, mean absolute error (MAE), which is robust against outliers. We noted that smaller batches improved our results, so we select a batch size of 100. We selected the Rmsprop method as an optimizer after testing the Adam and stochastic gradient descent with default learning rate

Regression Plot of one stock

Shallow RNN Training and Validation Loss



Confusion Matrix Shallow RNN model



Confusion Matrix Shallow Base model

6. Conclusion and Future Work

In financial market both accuracy and returns are important. Lower accuracy generally leads to higher risk tending to bigger losses. However; traders are always look for higher returns with minimum risk. In this research a recommendation approach has been presented which provides more objectivity to the trader in terms of decision making. Different models have been compared and the novel approach of combining the regression loss function with classification has been presented. RNN/ LSTM based models consistently beat the traditional machine learning models and the CNN. It's worth mentioning that among all models, CNN based models performed worst when it comes to classify and predict the decision variables.

Future work entails more automated way of tuning the model using genetic algorithm to get the best optimized model and try different permutations of RNN model like multi-layer and multi-step model. Also, the model needs to be back tested to calculate the returns and the correlation between the stocks. Also, it may be considered to optimize the portfolio of the traders.

7. Contribution

Special thanks to my Teaching Assistant Prerna Khullar and Ayush Kanodia. Without their guidance and help this project wouldn't be successful.

8. Readings

[1] Brownlee J. Deep learning for time series forecasting: predict the future with MLPs, CNNs and LSTMs in Python. Machine Learning Mastery. 2018.
[2] Eapen J, Bein D, Verma A. Novel deep learning model with CNN and bi-directional LSTM for improved stock market index prediction. In: 2019 IEEE 9th annual computing and communication workshop and conference (CCWC). 2019. pp. 264–70.
[3] Fischer T, Krauss C. Deep learning with long short-term memory networks for financial market predictions. Eur J Oper Res. 2018;270(2)
[4] AlphaVantage, "Stock Time Series time series intraday," 2016.
[5] Bao, W.N., Yue, J. and Rao, Y. (2017) A Deep Learning Framework for Financial Time Series using Stacked Autoencoders and Long-Short Term Memory Plos one 12.
[6] A. Graves et al., Supervised Sequence Labelling with Recurrent Neural Networks. Springer, 2012, vol. 385
[7] J. Lines and A. Bagnall, "Time Series Classification with Ensembles of Elastic Distance Measures," Data Mining and Knowledge Discovery, vol. 29, no. 3, pp. 565–592, jun 2014.