Transforming Education Transforming India

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Stock Market Analysis and Prediction On AI

NAME OF THE CANDIDATES

Kousik Snai (11804893) (Roll No: 45)

Praveen Kumar Singh (11805361) (Roll No: 65)

Sunny Raj Hans (11805373) (Roll No:40)

Pankaj Kumar Bind (11805362) (Roll No: 50)

SECTION: K18UW

SUBMITTED TO:

ANKITA WADHAWAN

Abstract:

In the world of finance, stock trading is one of the most important activities. Professional traders have developed a variety of analysis methods such as fundamental analysis, technical analysis, quantitative analysis, and so on. Such analytically methods make use of different sources ranging from news to price data, but they all aim at predicting the company's future stock prices so they can make educated decisions on their trading.

In recent years, the increasing prominence of machine learning in various industries have enlightened many traders to apply machine learning techniques to the field, and some of them have produced quite promising results. In this paper, we will focus on short-term price prediction on general stock using time series data of stock price.

Introduction:

There have been numerous attempt to predict stock price with Machine Learning. The focus of each research project varies a lot in three ways. (1) The targeting price change can be near-term (less than a minute), short-term (tomorrow to a few days later), and long-term (months later). (2) The set of stocks can be limited to less than 10 particular stock, to stocks in a particular industry, to generally all stocks. (3) The predictors used can range from global news and economy trend, to particular characteristics of the company, to purely time series data of stock price.

Based on the works we find, more progress has been made in predicting near-term [1] and long-term price changes [2]. In particular, long-term prediction has achieved over 70 percent accuracy when only considering limited number of stocks or sticks in a particular industry [3]. These well working models often rely on particular information of the company [4], which is often hard to access by general public and doesn't work with short-term prediction.

We decided to focus our project on the domain that currently has the worst prediction accuracy: short-term price prediction on general stock using purely time series data of stock price. Most researches in this domain have only found models with around 50 to 60 percent accuracy. And they often work only for classification [5]. After close investigation, we noticed that some papers that claim to have better result used problematic metrics to evaluate the model [6][7]. Although the graphs or numbers might look fancy, the actual model provides limited meaningful guidance for trading. (One example would be trying to predict whether the price tomorrow is increased or decreased from several days ago.)

Our preliminary model suggests that naively passing in time series data of stock price works no better than random guessing, so we applied complicated models to pre-process the time series data before running ML models. We didn't find any works that combined these financial technical indicators with machine learning algorithms like we did. And we got better results than other papers we found in this particular problem domain.

Literature Review:

We use Alpha Vantage API [8] to access the time series data of 82 randomly chosen stocks traded at NYSE. The API provides access to intraday time series, daily time series, weekly time series and monthly time series data. Since the domain of our problem is short-term prediction, we decided to proceed with daily time series data, which includes daily open price, daily high price, daily low price, daily close price and the daily volume.

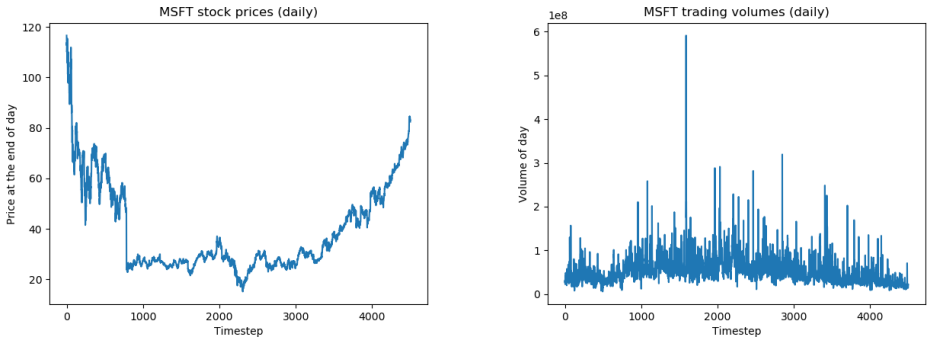


Figure 1: Daily prices and trading volume of Stock "MSFT"

Then we consulted several investment education websites. We carefully selected 11 models of processing time series data and get 17 technical indicators out of the models. The models are selected based on their popularity and reviews on the investment education websites.

Now we put together the stock price at the end of the day at the beginning of each period, the volume of the same day, and the technical indicators above and we have 260k data samples from 82 stocks. Together with the daily trading volume and the price, we have 19 predictors in total.

Proposed Methodology:

The problem we want to tackle is hard, so we started with a simplified problem: predicting whether the prices will increase or decrease in the next *n* days, using the stock prices and volumes in the past *mdays*. For this classification problem, we implemented Logistic Regression, Bayesian Network, Simple Neural Network, and SVM with rbf kernel in the sklearn[9] library and ran them on the prices of one specific stock named "MSFT".

Result & Discussion:

Preliminary Experiments:

Before we approach the actual problem we want to tackle, we start with a simplified question: can we actually predict the prices trends? We implemented several models available in the sklearn package and the following are our results on the stock 'MFTS':

(m, n)	(20, 5)	(20, 3)	(10, 3)	(10, 1)	(5, 1)
Logistic Regression	51.26%	51.04%	52.37%	47.97%	48.08%
Bayesian Network	50.84%	50.97%	48.52%	47.09%	46.87%
Simple Neural Network	47.06%	45.93%	44.66%	42.14%	42.83%
SVM with rbf kernel	46.22%	44.79%	43.38%	41.33%	41.01%

Table 1: Test error rates for different models

Linear Regression with Technical Indicators:

Our preliminary results were sufficient to show that past prices by itself is not a good predictor for the problem we would like to tackle, so we introduced more predictors into the problem. After doing research [10][11] we found out that in the field of trading, people often use technical indicators for making decisions, so we decide to include them in our model. We chose the 13 most used indicators in the field and together with the stock prices we have a total of 19 predictors for time-step. Using these 19 predictors, we attempt to predict the price change for the next n days. From the results of our preliminary findings, we decided to start with linear regression. For the first metric we discussed in the methods section, we see that the test loss oscillates between 0.2 and 0.8 for n = 1 with different random partitions of the data, while the training loss stays around 0.5 to 0.6. This suggests that over fitting is not an issue for our model and the model generalizes quite well. For the second metric, we achieved a correct prediction rate as high as 61.5% when n = 1, and around 55% for $n \geq 10$. Figure 2 is a diagram of the correct classification rate. We see that the results are consistent with the results of our preliminary findings, and compared to those results we get more improvements on the prediction for when n is big, but not as much improvement for smaller values of n.

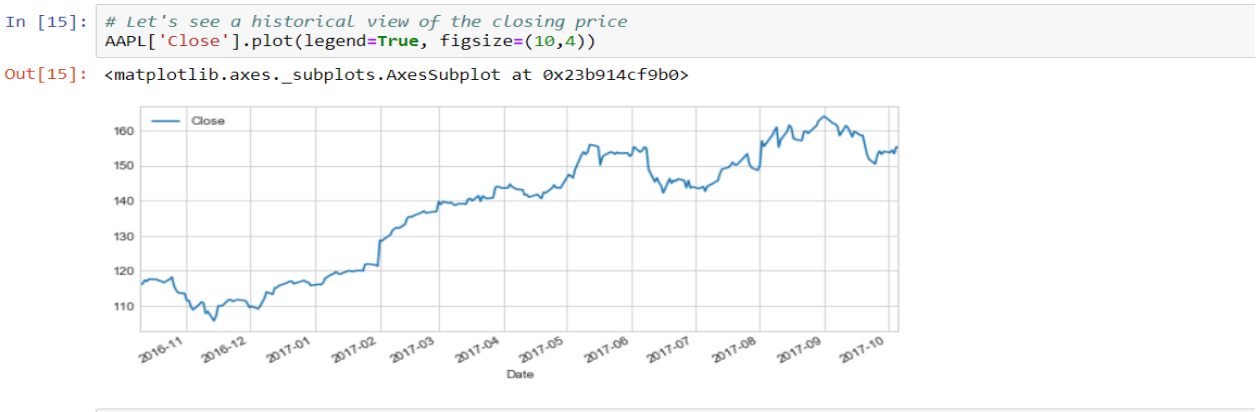
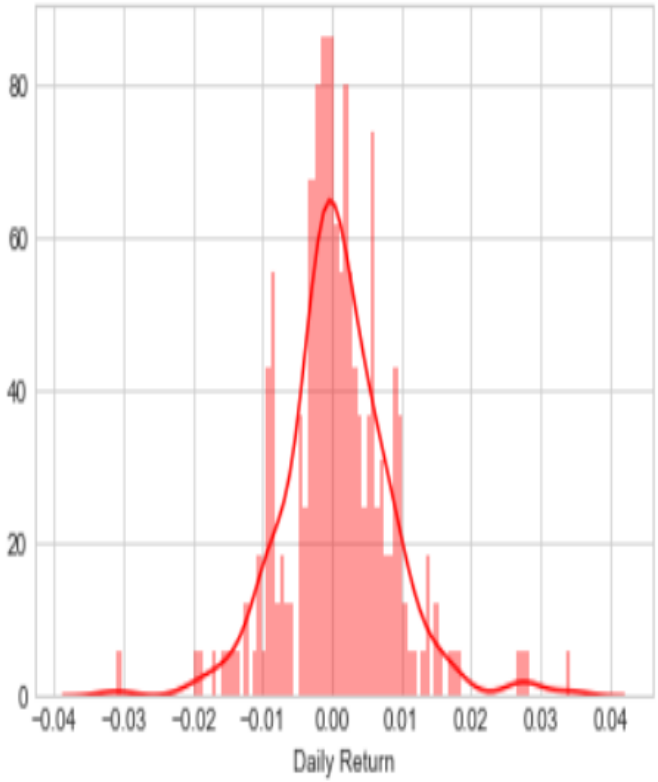


Figure 2: Correct Classification rate for linear regressio

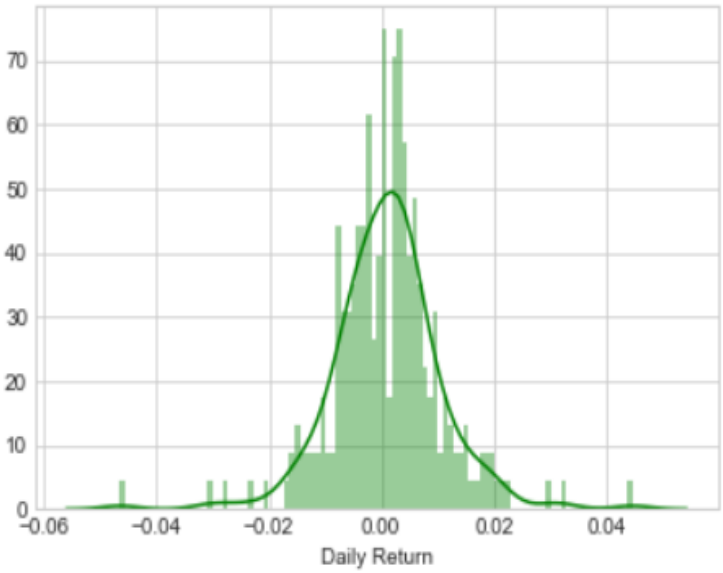
Besides plain linear regression, we also implemented ridge regression and lasso regression to justify our selection of the predictors. An output of the coefficients of ridge regression reveals that no coefficient has dramatically decreased. This suggests that all our predictors are significant, which is again justified by comparing the test error rates listed in Figure 3.

```
n [10]: # Note the use of dropna() here, otherwise the NaN values can't be read by seaborn
        sns.distplot(JNJ['Daily Return'].dropna(),bins=100,color='R')

ut[10]: <matplotlib.axes._subplots.AxesSubplot at 0x1df64345668>
```



```
In [15]: sns.distplot(WMT['Daily Return'].dropna(),bins=100,color='G')
Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x1df65aaf470>
```




```
In [16]: (WMT['Daily Return'].dropna()).quantile(0.05)
```

Support Vector Regression:

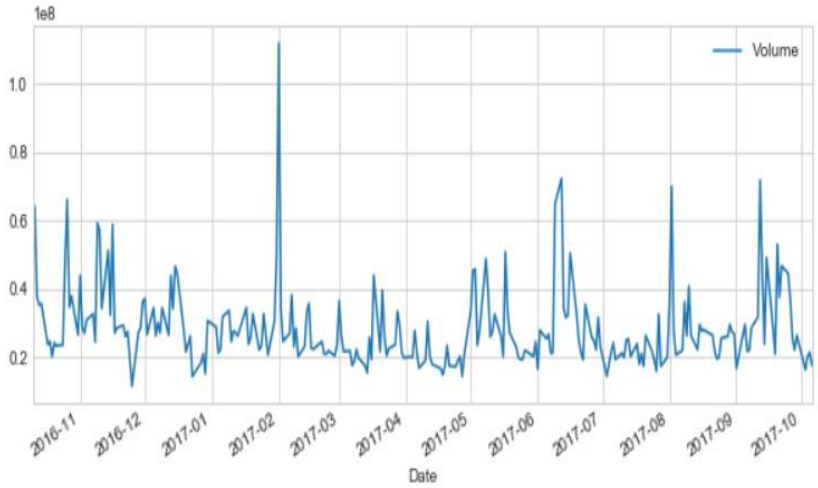
From our preliminary findings, we know that SVM with a Gaussian kernel performs well, so we decided to implement support vector regression on the data to see whether we can get a better result than those of linear regression. However, in the implementation we were forced to work with a smaller training set due to the nature of the model: Since Gaussian kernel maps our predictors into an infinite dimensional feature vector, we are forced to apply the kernel trick, which leads to a space and time complexity of O(n 2 samples). Working with 200k+ samples is therefore not an option. In our experiment, we randomly select 5% of our training data as the data we use for the regression and we produced the following error rates shown in Figure 4. Unlike linear regression, we see that the error rates oscillates since we choose a smaller training set, leading to more variance in our fitted model. a common trait, however, is that we see the same pattern as n increases, since the correct rate drops around n = 20. Generally, we see that we can achieve a correct rate of 69.5% for the highests and 68.5% for the lowests, which is a big improvement on linear regression, and also superior to many similar projects. [5][6][7]

Other Attempt:

Besides the models we discussed above, we also implemented a neural network with a hidden layer based on LSTM (long-term-short-term memory) and a dense layer for output. The model is widely used for various NLP tasks such as voice processing, and its incorporation of the concept of keeping memory of past data in a time series suggest that it could be useful for stock price prediction. [12]

j Unfortunately, however, our implementation of the model was not promising since we saw that the development MSE converges around the variance of the development set, with the converged model outputting predictions close to 0 and a correct prediction rate.

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x23b90cf27f0>



We can see that on Feb'2017 was the higher for AAPL stock being traded.

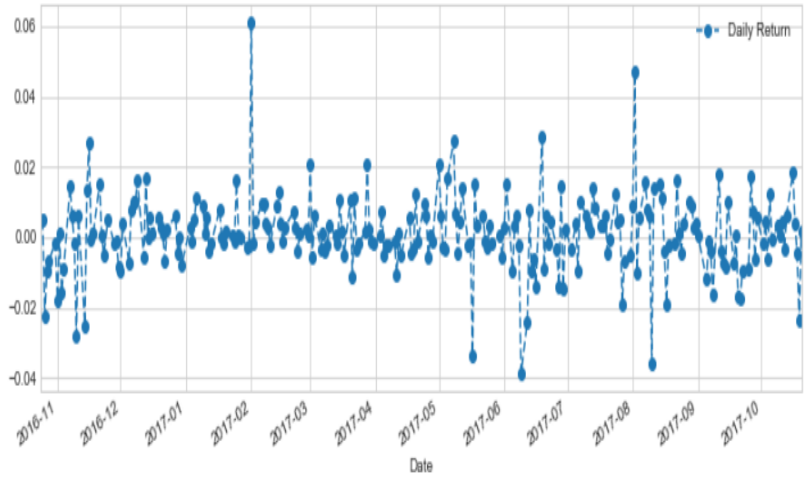
Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x12991f54278>

Great, now let's get an overall look at the average daily return using a histogram. By using seaborn to create both a histogram and kde plot on the same figure.

Conclusion :

Concluding from our results, we see that using only past price data technical indicators which are calculated based on past price data, we can achieve a correct prediction of the price trends for a shy of 70% of the time. Incorporating other factors such as news about the stock market and the companies will certainly improve the performance of our current model.

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x12991f54278>



Great, now let's get an overall look at the average daily return using a histogram. By using seaborn to create both a histogram and kde plot on the same figure.

Although a correct rate of 70% is not very high, we can still see that the results can be quite meaningful. Below is a chart of the predicted price change and actual price change for the next day:

Figure 5: Prediction versus Actual Price Change over 1 day

In practice, since small fluctuation might not by that meaningful, we are more interested in capturing the price trend when the actual price of the stock changes a lot (say more than $2). We see that we can in fact achieve a correct classification rate of 92.3% for when the price increases by more than $2, and 72.9% for when the price decrease for more than $2. For all the

```
In [20]: AAPL[['Close','MA for 10 days','MA for 20 days','MA for 50 days','MA for 100 days']].plot(subplots=False,figsize=(10,4))
Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x23b90900eb8>
```

Section 2 - Daily Return Analysis

models we use, as discussed in the experiments, the error rate for test set and error rate for training set has been really close. Thus overfitting shouldn't be a problem for us. On the other hand, we are more likely to take action when the predicted change of price is big (again say more than $2). When we predict that the price will decrease by more than $2, 97.3% of the time the actual price will decrease. Unfortunately we rarely predict that the price will increase by more than $2: we only predicted that the price will increase by more than $2 for 0.1% of the test data, so the sample is too small to produce a meaningful judgment. For future work, the first thing we would do is definitely find machines with higher computation power to run Support Vector Regression on the whole data set to improve the accuracy rate. At the same time, although our attempt with neural networks so far wasn't successful, there are sources [13] suggesting that artificial neural networks are useful for stock price predictions, so one of the future directions will be to implement more models base on neural networks.

Appendix:

Contributions:

Both team members were involved in all stage of the projects. The background research and analysis of related work was shared evenly. The processing of Data Set and selection of Features (Technical Indicators) was led by Alice Zheng. The implementation of Machine Learning Models was led by Jack Jin. The analysis of result was done by both members together.