Machine Learning Engineer Nanodegree

Capstone Project Report – Stock Price Prediction

Wei Feng
April 1, 2018

1 Definition

1.1 Project Overview

Stock price prediction is important for individuals, investment firms and hedge funds to make decisions in stock exchanges and portfolio management. Accurate stock price prediction is the key for investors to maximize their return.

There are basically two steams of analyses in stock price prediction: fundamental analysis and technical analysis (Xu, 2012). Fundamental analysis focuses on the entire stock market as a whole or determines a company's stock price based on underlying factors that affects the company's actual value; technical analysis tries to predict a stock price only based on its past trend (usually time series analysis and machine learning techniques).

This project focuses on technical analysis of stock price because it is free of individual company's information collection, screening and processing and it is more transferable to other stocks. The study focuses on short-term (1-day) stock price prediction, which enables investors to make daily purchase or sell decisions. This project does not intend to provide optimization recommendations for investors on portfolio management.

Facebook stock is selected for this project. Facebook and NASDAQ adjusted closing price data between February 4, 2016 and February 4, 2018 were used in this analysis.

1.2 Problem Statement

The goal of this project is to predict the end-of-day Facebook stock adjusted closing price (at the beginning of every day) based on historical adjusted closing prices of Facebook and NASDAQ.

Instead of predicting the exact adjusted closing price (continuous variable), a binary dependent variable is created – It is 1 if the end-of-day closing price is 1% higher than the closing price of the previous day; and 0 otherwise. Based on this binary prediction

result, a simple daily trading strategy can be used – buy the stock at the beginning-of-day and sell it at the end-of-day if it is predicted to increase at least 1% at the end-of-day; and not buy the stock otherwise.

Because stock data is time series data, the training and test data sets have to be split chronologically so that no future information is used in prediction (cross validation data is also split chronologically). The data will be split into training (339 days between 2016-04-29 and 2017-08-31) and test (83 days between 2017-09-01 and 2017-12-29) data sets. Four supervised machine learning algorithms (Logistic Regression, Support Vector Machine, Gradient Boosting Machine and XGBoost) will be developed based on the training data, and they will be tested and compared on the test data.

1.3 Metrics

The performance of the four algorithms will be evaluated based on four key metrics on the test data: accuracy, precision, recall and net gain. Net gain will be the primary metric to compare the four algorithms because the ultimate goal of this project is to maximize net gain based on the prediction results. High accuracy may not necessarily guarantees a high net gain. In an overall increase environment, high precision and high recall may lead to a high net gain, however, due to the stochastic nature of stock price changes, it is not easy to determine the weight between precision and recall, and therefore F-score is not used. When one model does not dominates the other model (both precision and recall are higher), net gain is a better metric to compare the models in this case.

Net gain is defined as:

$$\sum_{i=1}^{83} x_i(p_i - p_{i-1}) \tag{1}$$

$$x_i = \begin{cases} 1, \ if \ \frac{\hat{p}_i}{p_{i-1}} - 1 \geq 0.01 \\ 0, \ if \ \frac{\hat{p}_i}{y_{i-1}} - 1 < 0.01 \end{cases} \tag{2}$$

p_i is the closing price on day i, $x_i = 1$ if the predicted closing price of day i ($\hat{p}_i$) is at least 1% higher than the closing price of day $i-1$ (p_{i-1}), which triggers a buy action at the beginning of day i at price p_{i-1} and sell it at the end of day at price p_i, and the daily gain is $p_i - p_{i-1}$; $x_i = 0$ means no action and therefore no gain. Note that this net gain metric is based on the assumption that the stock can be purchased at the exact previous day's closing price and sold at the exact end-of-day closing price.

Accuracy, precision and recall are standard performance metrics defined as:

Accuracy = (Ture Positive + True Negative) / (True Positive + True Negative + False Positive + False Negative) *(3)*

Precision = True Positive / (True Positive + False Positive) *(4)*

Recall = True Positive / (True Positive + False Negative) *(5)*

2 Analysis

2.1 Data Exploration and Visualization

Two years' (February 4, 2016 – February 4, 2018) data for FB and NASDAQ (benchmark) stocks are downloaded from Yahoo! Finance, the data include the following basic information for each stock: Date, Open (price), High (price), Low (price), Close (price adjusted for splits), Adj Close (price adjusted for both splits and dividends) and Volume. Therefore, Adj Close will be used in this project. These data will be used in the stock price prediction model training as well as cross-validation and testing.

The trend of these stocks over the two-year period is shown in Figure 1:

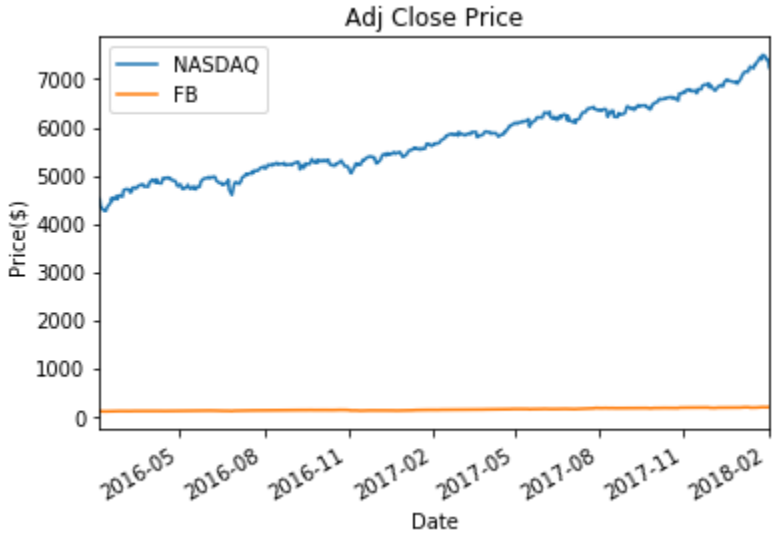


Figure 1 Adjusted Closing Price

Because stocks have different scales, prices are normalized by $\frac{p_i}{p_0} - 1$ as shown in Figure 2:

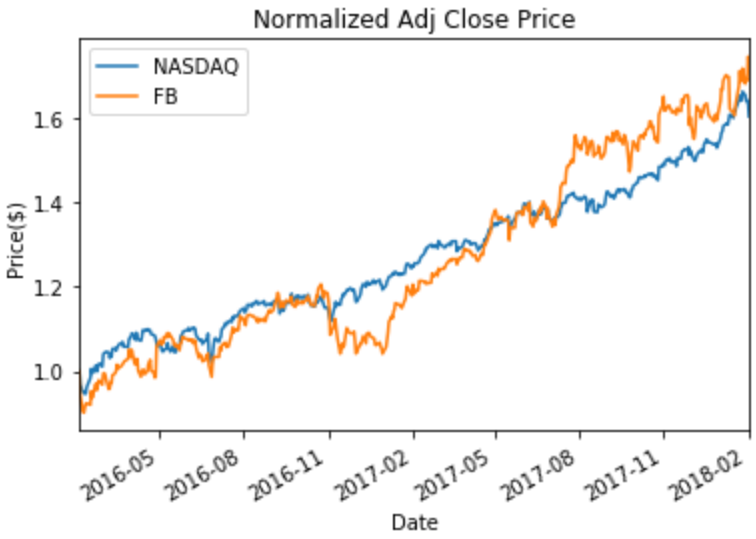


Figure 2 Normalized Adjusted Closing Price

NASDAQ has a close-to-linear increase over the past two years while each stock has a more volatile increase. The plot also show that there are no abnormal data points. It is also shown that there is no missing data points (as shown in the Jupyter notebook: df.isnull().values.ravel().sum() = 0). Therefore, the data is clean and complete and is ready for use. From now on, the normalized adjusted closing price will be used and for simplicity, stock price refers to the normalized adjusted closing stock price

To make stock predictions, the following features are inspired by the Udacity course "Machine Learning for Trading" by Tucker Balch.

- 'price - rm': previous day price – previous 20-day price rolling mean
- 'price - ub': previous day price – upper bound of the previous 20-day price Bollinger band
- 'price - lb': previous day price – lower bound of the previous 20-day price Bollinger band
- 'sharp ratio': sharp ratio of the previous 60-day price
- 'daily return': percent change of previous day price compared to the price of the day before
- 'rm momentum': the momentum of the previous 5-day price rolling mean
- 'market – market_rm': previous day market price (NASDAQ) – previous 20-day market price rolling mean

'price - rm' is created because a 20-day rolling mean is more stable and may represent the general trend of the future, therefore, if the price is higher than the rolling mean, it might be more likely to go down than go up. The daily trend of this variable is shown in Figure 3:

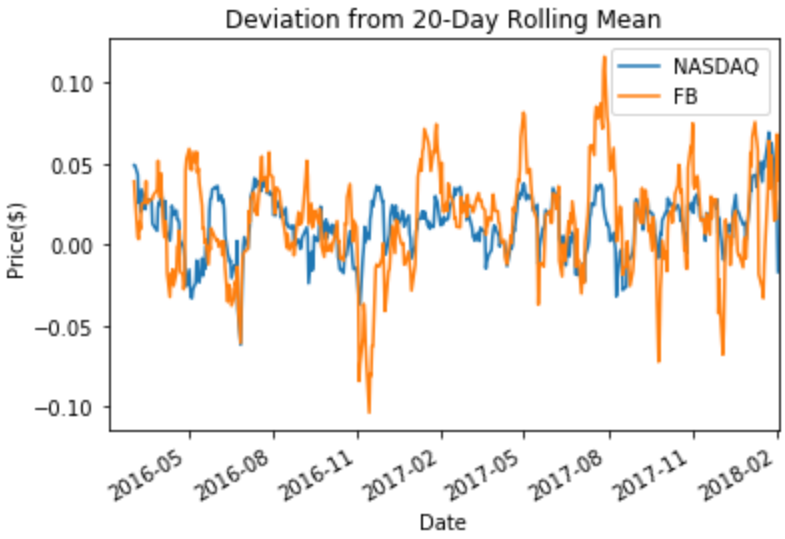
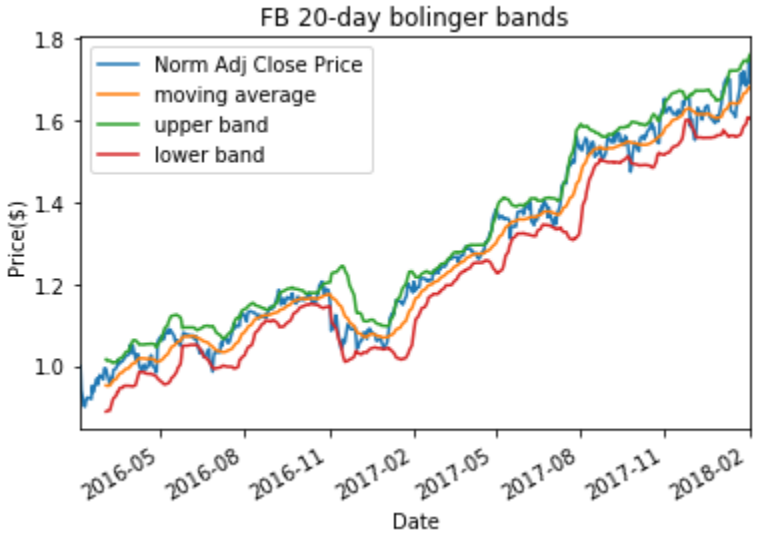


Figure 3 Deviation from 20-Day Rolling Mean

'price - ub' and 'price - lb' are created to detect signals that exceed the Bollinger band, beyond which the price is expected to come back close to the rolling mean. FB and NASDAQ Bollinger bands were shown in Figure 4. In most of the days, prices are within the bands; and in most of the cases when price reaches or exceeds the upper band, it decreases in the next day; and when price reaches or exceeds the lower band, it increases in the next day. Note that the first 19 days has no Bollinger band due to lack of data to calculate the 20-day rolling mean. Upper and lower bounds are calculated by 20-day rolling mean +- 2 * 20-day rolling standard deviation.



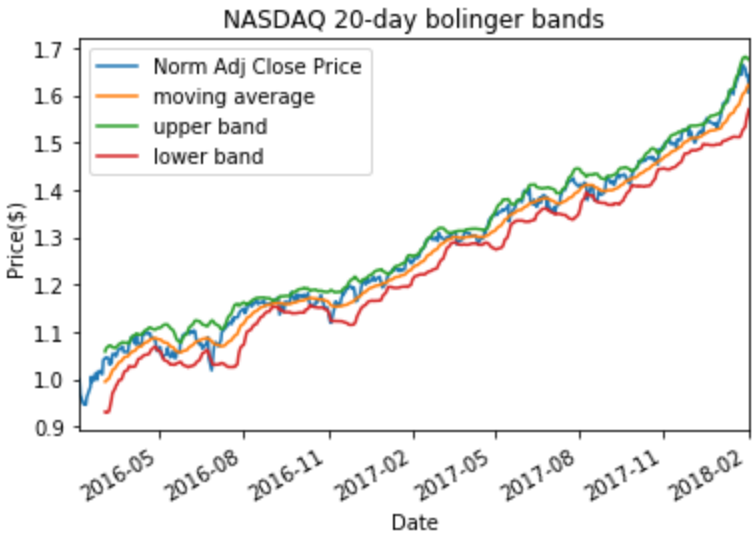


Figure 4 20-Day Bollinger Band (a) FB; (b) NASDAQ

'sharp ratio' measures the adjusted risk of return, therefore it may affect the price changing behavior. Daily trend is shown in Figure 5. Note that the first 59 days are empty due to lack of data.

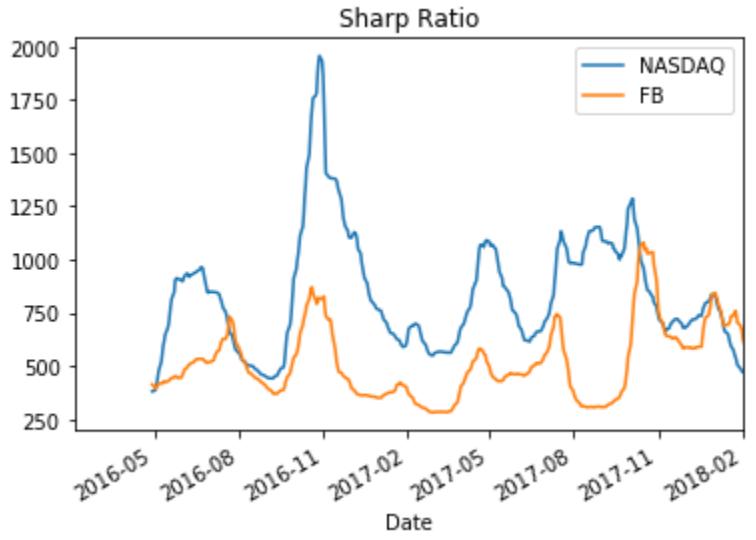


Figure 5 Sharp Ratio

'daily return' is created because the previous day's return may affect the next day's return to certain extent. The daily trend is shown in Figure 6. The high volatility of the daily return indicates a negative relationship between previous daily return and the next daily return.

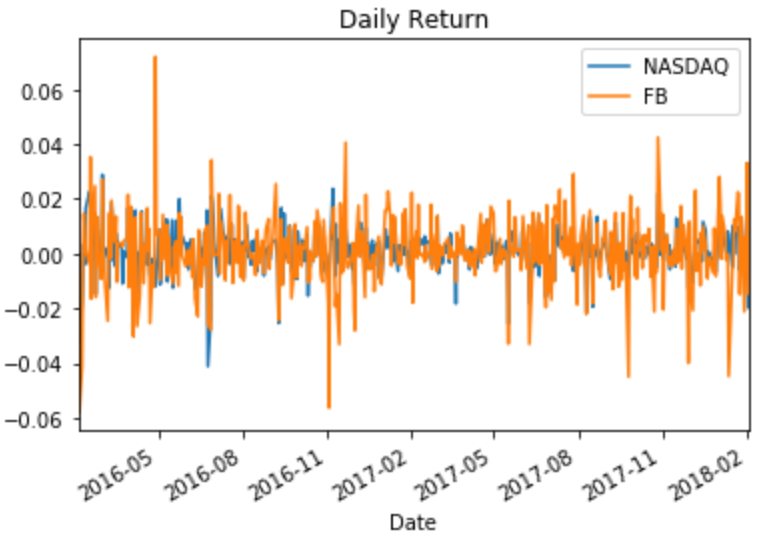


Figure 6 Daily Return

'rm momentum' measures the trend of the rolling mean, which may help identify the general trend of the price. The daily trend is shown in Figure 7.

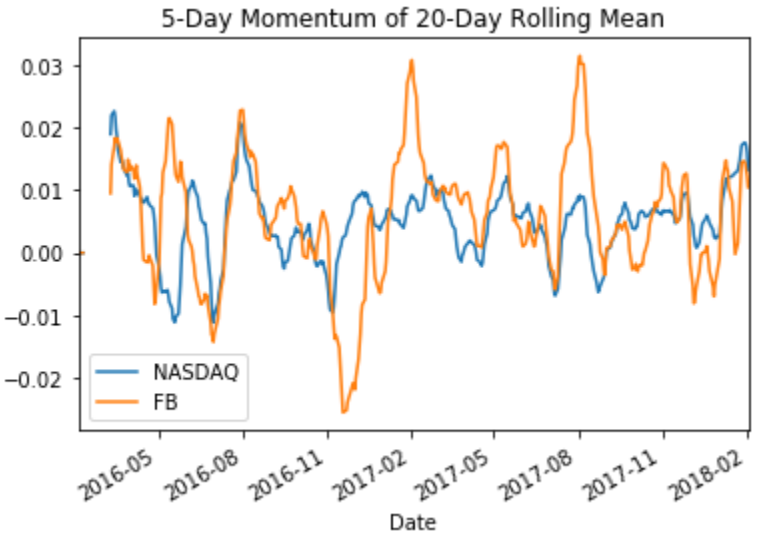


Figure 7 Momentum of 20-Day Rolling Mean

'market – market_rm' measures the benchmark deviation between daily price and 20-day rolling mean, which may identify the entire market trend. The daily trend is shown in Figure 3.

Because one of the variables uses previous 59-day rolling mean, the data used in the analysis ranges from April 29, 2016 to December 29, 2017, a total of 422 days. Train data ranges from April 29, 2016 to August 31, 2017 (339 days) and test data ranges from September 1, 2017 to December 29, 2017 (83 days).

The dependent variable 'y' is 1 if the end-of-day closing price is 1% or higher than the previous day closing price and 0 otherwise. Days that are positive are shown as blue bar in Figure 8. There are 87 days out of the 422 days with 1% or higher increase (21% positive observations).

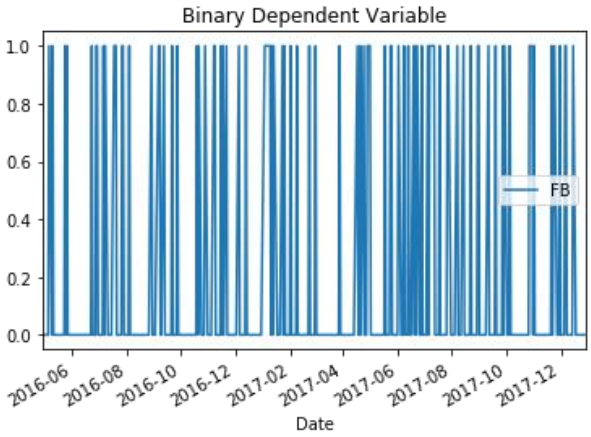


Figure 8 Binary Dependent Variable

The histograms of features and normalized price are shown in Figure 9. The histograms of 'price – ub' and 'price – lb' show that there are very few days the price is out of Bollinger bands. 'rm momentum' indicates an overall increasing trend. 'market – market_rm' also indicates an overall increasing trend. 'daily return' and 'price - rm' follow close-to-normal distributions, which indicates the number of increases and decreases are close to equal (not the amount of increases and decreases).

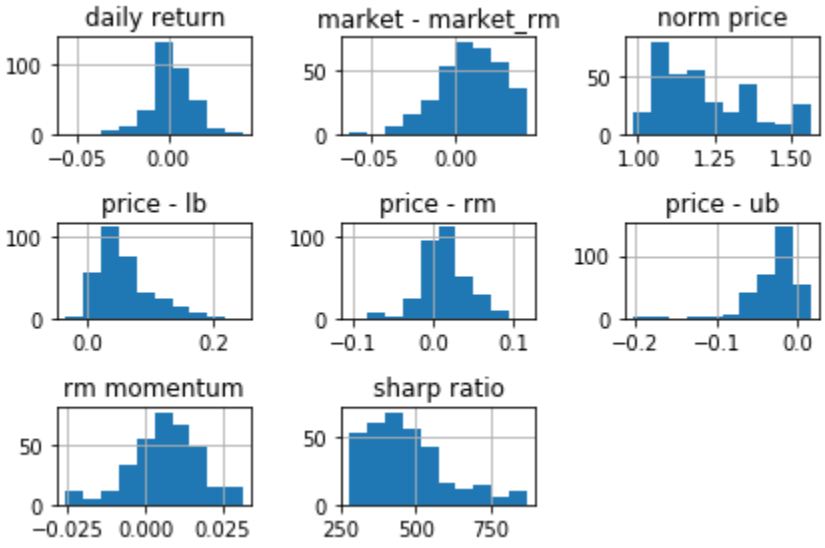


Figure 9 Histograms of Features

Figure 10 shows that there is little correlation between feature variables.

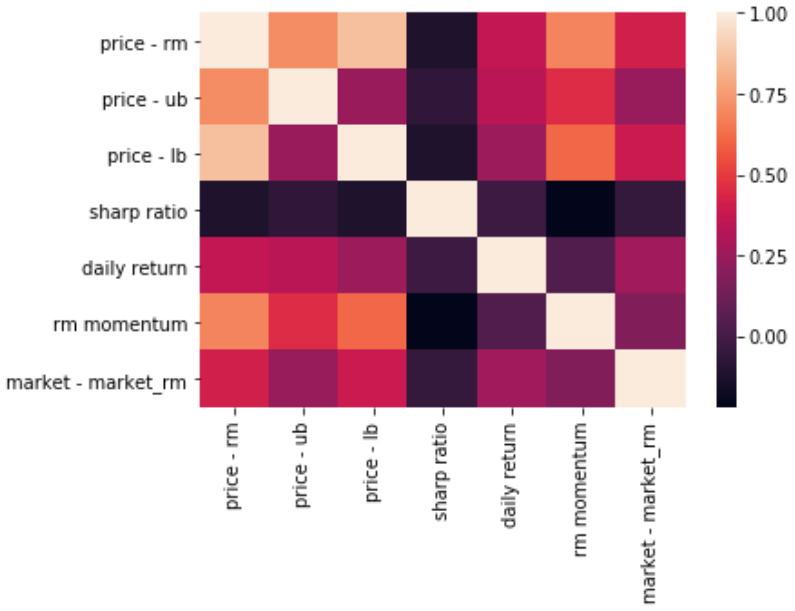


Figure 10 Feature Variables Correlation Heat Map

2.2 Algorithms and Techniques

Because this project tries to make daily trading decisions based on the prediction results, a binary variable dependent variable makes sense. Therefore, four classification algorithms will be used in this project: Logistic Regression, Support Vector Machine, Gradient Boosting Machine and Extreme Gradient Boosting (XGBoost).

Logistic Regression model is selected because 1) it has been used in both academia (Mironiuc & Robu, 2013) and industry (QuantDesk by Lucena Research); and 2) some of the features may have a close-to-linear probabilistic relationship with the outcome variable (e.g. 'price - ub' and 'price - lb'). Although, it may not capture more complicated non-linear relationship between features and the outcome.

Support Vector Machine is selected because 1) it is widely used in stock price prediction (Hu, Zhu, & Tse, 2014; Wen, Xiao, He, & Gong, 2014; Sheta, Ahmed, & Faris, 2015); 2) it can model various linear and non-linear decision boundaries with different kernels; and 3) it prevents overfitting well; 4) cross-validation can be used to further prevents overfitting and improve robustness. Although it is computationally expensive.

Gradient Boosting Machine and XGBoost are used because 1) they are one of the most widely used techniques in stock price prediction and other Kaggle prediction competitions (Khandelwal, 2017; Gorman, 2017; Paradkar, 2017); 2) these tree-based ensemble algorithms can capture complicated non-linear relationships between feature variables and the outcome variable; 3) cross-validation and grid search can be used to

further prevents overfitting and improve robustness; 4) computationally efficient, especially for XGBoost. According to the author of XGBoost (Chen & Guestrin, XGBoost: A Scalable Tree Boosting System, 2016), the key difference between XGBoost and GBM are: XGBoost used a more regularized model formulization to control overfitting and it is a computationally optimized GBM. Although XGBoost sounds superior to GBM theoretically and computationally, it is worth trying and comparing the two algorithms in this 1-day stock price exercise.

To maximize the performance of the above mentioned algorithms, cross validation will be used for SVM, GBM and XGBoost to tune parameters (Logistic Regression does not need parameter tuning). The cross validation sets are split chronologically instead of random split. Also, because it is not clear about the weights between precision and recall, three evaluation scorers were tested in SVM, GBM and XGBoost in the cross validation process using: accuracy score, fbeta scores with beta = 0.5 and beta = 2.0.

2.3 Benchmark

Given that the goal of this project is to maximize the net return based on the predicted 1-day stock price increase (1%+) signal, two naïve predictions can be used as the benchmark: all positive and all negative. The expected metrics of these two naïve predictions on the test data is summarized in Table 1. Note that the net gain of the all positive prediction is simply the difference of normalized closing price between '2018-01-02' and '2017-09-01'. Definitions of the metrics are shown in Equations *(1) – (5)*.

Table 1 Benchmark Performance Metrics

Metrics	All Positive	All Negative
Accuracy	18%	79%
Precision	18%	NA
Recall	100%	0%
Net Gain	0.085	0

3 Methodology

3.1 Data Preprocessing

As shown in the descriptive analysis of the feature variables and the outcome variables in Section 2.1, the data is clean and complete without missing or abnormal observations, therefore, no additional data preprocessing steps are necessary.

3.2 Implementation and Refinement

All of the four algorithms were fitted on the training data and the fitted models were applied on the test data to collect the four performance metrics.

Logistic Regression is the easiest and fastest algorithm to implement, as expected. Because there are not too many feature variables, it is not necessary to perform variable selection.

To test different potential decision boundaries of the SVM, 'rbf' kernel was trained under three different penalty parameters C (1, 5 and 10) using three evaluation scorers (accuracy, fbeta score with beta = 0.5 and 2.0). The training time of SVM using 'linear' and 'poly' kernels are the longest among all four algorithms. At the beginning, 'linear' and 'sigmoid' kernels were tested against 'rbf' on all penalty parameters C using all three the evaluation scorers. 'rbf' was always chosen in the cross validation process. Kernel 'poly' was also trained at the beginning, but the training time was too long and it never ended. Therefore, the final implementation only used kernel 'rbf'. On the other hand, both C = 1 and C = 10 have been chosen in the original tests, thus the final implementation used C = 1, 5 and 10. Also, the original test shows that the performance under the three different evaluation scorers were different, the accuracy scorer led to the highest accuracy but the two fbeta scorers result in both higher precision and recall. Although, higher precision and recall do not necessarily lead to higher net gain due to the stochastic increase/decrease amount.

There are also some lessons learned in the coding process. At beginning, because there were not many scenarios to try and compare, the model was trained one at a time by manually changing the parameters and results were printed on the screen. However, as more parameters need to be tested and results to compare, it becomes inefficient. Therefore, cross validation training was looped across all combinations of parameters and results from all combinations were saved for better comparison.

Another lesson learned in coding is that the time series cross validation sets cannot be defined at once and used in each cross validation process, it has to be re-created at each iteration within the loop, although each iteration should generate the same cross validation sets.

To compare the performance between GBM and XGBoost, parameters were kept the same between the two algorithms. Because both of these two algorithms have relative short training time compared to SVM, multiple combinations of parameters were tested at the beginning, and the most common selected best parameters in the intermediate steps were selected in the final parameters with learning rate (0.1, 0.2, 0.3), number of trees (200, 300, 500) and max depth of trees (2, 3, 4). The models were also trained using the three different scorers: accuracy, fbeta score with beta = 0.5 and 2.0. Similarly as in

SVM, performance under the three different scorers are different. Precision and recall trained under fbeta score are usually both higher than them trained under the accuracy score.

When GBM and XGBoost were trained separately before the final implementation, the training time between the two algorithms were not noticeably different probably because of the relative small data size (the computation advantage of XGBoost was not noticeable in this project).

Another important difference between GBM and XGBoost is that XGBoost usually result in higher accuracy and lower precision, recall and net gain than GBM in all three evaluation scorers.

Similar coding lessons were learned in GBM and XGBoost as in SVM. As a result, grid search and cross validation was performed in a loop with all results saved in one execution in the final implementation.

4 Results

Logistic Regression predicted no positive outcome in the test data, which is probably because the key relationship between feature variables and the outcome variable is not linear. For example, the 'price - ub' and 'price - lb' may only be predictive when they change signs and the majority of these two variables are positive or negative; therefore, there is little linear marginal impact of these two variable on the outcome. As a result, the performance of the Logistic Regression is the same as the all negative benchmark prediction.

The performance metrics of the SVM algorithms on the test data and the best parameters under the three scorers are summarized in Table 2.

Table 2 SVM Performance Metrics

Metrics	Accuracy Score	F-beta Score (0.5)	F-beta Score (2.0)
Accuracy	0.71	0.63	0.63
Precision	0.09	0.10	0.10
Recall	0.07	0.13	0.13
Net Gain	-0.02	-0.08	-0.08
C	1	5	5

Results show that cross validation with accuracy score led to the highest accuracy (as expected) but less precision and recall. The best models trained under the two fbeta

scores have identify results, both with higher precision and recall, but the net gain was lower than the net gain trained under the accuracy score. This is possible if the positive predictions trained by the fbeta score models have smaller amount of increase than those trained by the accuracy score model.

The performance metrics of the GBM and XGBoost algorithms on the test data and the best parameters under the three scorers are summarized in Table 3 and Table 4.

Table 3 GBM Performance Metrics

Metrics	Accuracy Score	F-beta Score (0.5)	F-beta Score (2.0)
Accuracy	0.53	0.52	0.52
Precision	0.07	0.20	0.20
Recall	0.13	0.53	0.53
Net Gain	0.11	0.25	0.25
Learning Rate	0.1	0.3	0.3
# of Trees	200	500	500
Max Depth	4	2	2

Table 4 XGBoost Performance Metrics

Metrics	Accuracy Score	F-beta Score (0.5)	F-beta Score (2.0)
Accuracy	0.55	0.58	0.58
Precision	0.13	0.21	0.21
Recall	0.27	0.47	0.47
Net Gain	0.11	0.17	0.17
Learning Rate	0.2	0.3	0.3
# of Trees	200	500	500
Max Depth	4	2	2

Compared to GBM, XGBoost have better accuracy in all three scenarios probably due to the more regularized formulation that prevents overfitting, but XGBoost has lower precision and recall than GBM (and therefore lower net gain due to the overall trend of the price during the test data period is increasing) when trained with fbeta scores in the cross validation process.

By comparing the best models from the four algorithms and the two naïve benchmark predictions under the primary metric – net gain on the test data set, GBM trained with fbeta score (beta = 0.5 or 2.0) resulted in the highest net gain, followed by XGBoost, all positive naïve prediction, all negative naïve prediction and SVM. Logistic Regression produced the same results as the all negative naïve prediction.

For the other three key metrics, Logistic Regression and all negative naïve prediction have the highest accuracy due to the imbalanced distribution between positive and negative observations. However, they also have no positive predictions. SVM have higher accuracy than XGBoost, which is higher than GBM. Whereas, GBM has the highest precision and recall, followed by all positive naïve prediction, XGBoost, SVM, and Logistic Regression.

To test the robustness of these prediction models, another test data set was used between September 29, 2017 and February 1, 2018, a total of 86 days with 18% positive observations and 0.1757 net gain under all positive naïve prediction. GBM also has the highest net gain under fbeta score training (0.23), followed by all positive prediction, XGBoost, Logistic Regression and SVM. However, in this new test data, XGBoost has both higher precision and recall than GBM under the fbeta score trained models.

It is found that high accuracy is not always associated with high net gain. For example, SVM has higher accuracy than GBM and XGBoost (probably due to its advantage of preventing overfitting) but lower precision, recall and net gain. Also, higher precision and recall may not be associated with higher net gain due to the stock price change is stochastic (e.g. SVM and the new test data results between GBM and XGBoost).

5 Conclusion

In summary, this project selected Facebook stock price data over the past two years to build a next-day stock price prediction model and applied a simply daily trading strategy based on the prediction results. Four modeling techniques were trained, optimized through cross validation and compared with two naïve benchmark predictions (all positive and all negative). GBM generated the highest net gain among all models (including the benchmark model) on two test data sets. XGBoost outperformed GBM on test accuracy but not always on precision and recall. The final comparison across the four models and the naïve benchmark model is summarized and shown in Figure 11.

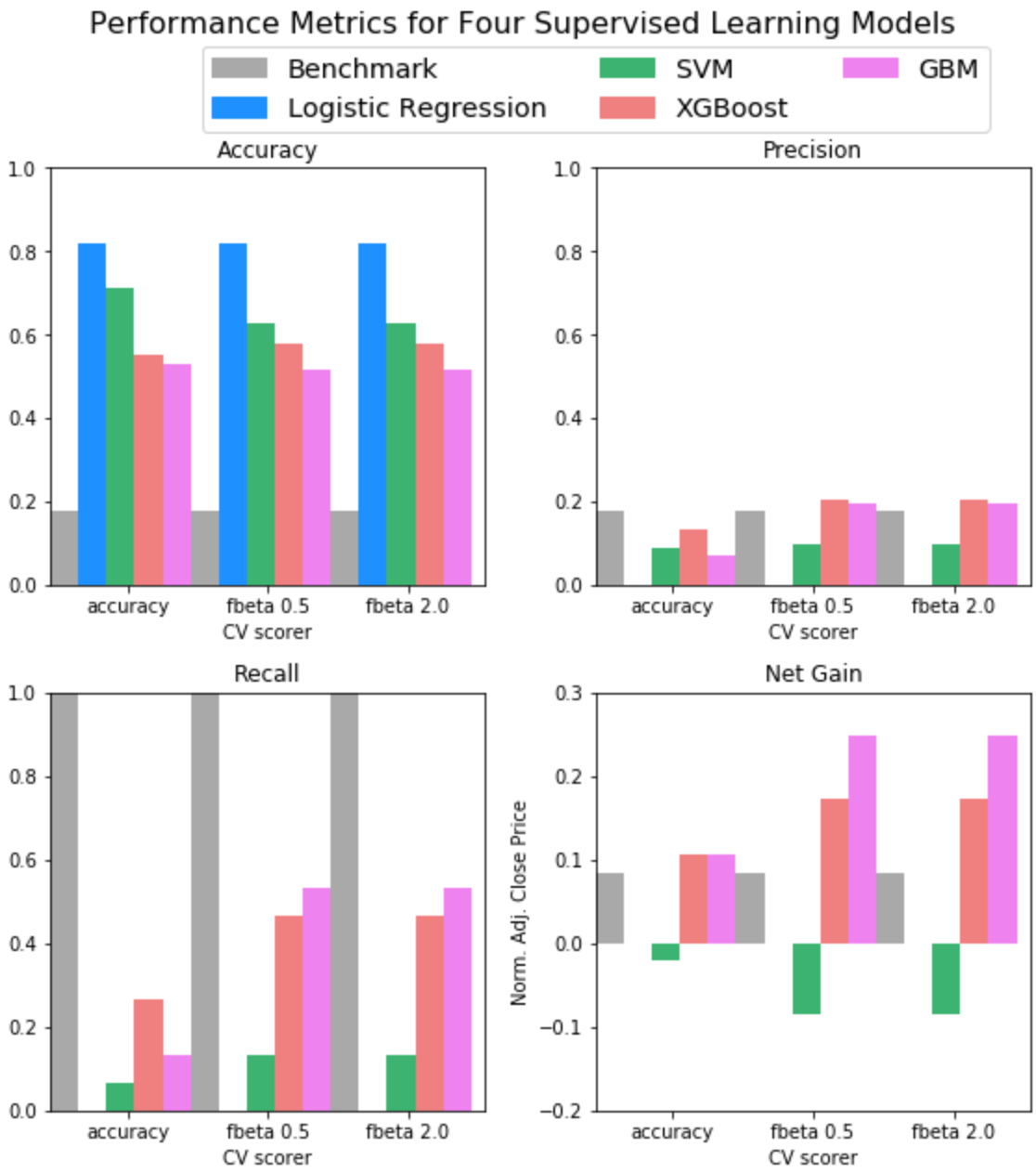


Figure 11 Performance Metrics for Four Supervised Learning Models and the Naïve Benchmark Model

Because the overall trend of the stock price is increasing during the test data time period, models with higher precision and recall usually (though not always, e.g. all positive naïve prediction) lead to better net gain. SVM, GBM and XGBoost tend to result in higher precision and recall in the test data when trained by fbeta score during the cross validation process.

To conclude, there is no single model and parameters that works best for every stock in all time periods. To maximize the expected net gain, it is highly recommended to execute the entire modeling process for each new stock.

5.1 Future Improvement

There are several improvements that could be made to this project:

1) The positive and negative observations are not balanced (21% vs. 79%), this imbalance may lead certain algorithms to favor negative prediction when they are optimized over the accuracy (e.g. Logistic Regression). It may worth the effort to try to balance the positive and negative observation in the training data set, and there are multiple techniques to balance the outcome variable as described in (Upasana, 217)
2) There are several other algorithms and techniques that can be used for stock price prediction such as neural network, deep learning, ARIMA, etc. These algorithms may discover other relationship between feature variables and outcome variable.
3) It may be worth to predict the next 7-day price or long-term prediction and develop different trading strategies based on prediction results, different prediction interval may favors different modeling techniques and parameters.
4) It might also be worth to create more feature variables or to try different thresholds of the existing feature variables (e.g. different window of moving average)
5) It is also worth to test model performance on different stocks and in different time periods, the results may be different too. There might not be a single technique that works best in all stock price predictions, it may be better to collect the most recent stock price data for the desired stock and build and test models and techniques on the particular stock on the desired trading frequency.

6 References

Chen, T. (n.d.). *Introduction to Boosted Trees*. Retrieved from XGBoost: http://xgboost.readthedocs.io/en/latest/model.html

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *KDD '16 Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (pp. 785 - 794). San Francisco.

Gorman, B. (2017). *A Kaggle Master Explains Gradient Boosting*. Retrieved from No Free Hunch: http://blog.kaggle.com/2017/01/23/a-kaggle-master-explains-gradient-boosting/

Hu, Z., Zhu, J., & Tse, K. (2014). Stocks market prediction using Support Vector Machine. *6th International Conference on Information Management, Innovation Management and Industrial Engineering (ICIII).* Xi'an: IEEE.

Khandelwal, P. (2017). *Which algorithm takes the crown: Light GBM vs XGBOOST?* Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-takes-the-crown-light-gbm-vs-xgboost/

Mironiuc, M., & Robu, M.-A. (2013). Obtaining a Practical Model for Estimating Stock Performance on an Emerging Market Using Logistic Regression Analysis. *Procedia - Social and Behavioral Sciences*, 422 - 427.

Paradkar, M. (2017). *Forecasting Markets using eXtreme Gradient Boosting (XGBoost)*. Retrieved from R-bloggers: https://www.r-bloggers.com/forecasting-markets-using-extreme-gradient-boosting-xgboost/

Sheta, A., Ahmed, S. E., & Faris, H. (2015). A Comparison between Regression, Artificial Neural Networks and Support Vector Machines for Predicting Stock Market Index. *International Journal of Advanced Research in Artificial Intelligence*, 55 - 63.

Upasana. (217). *How to handle Imbalanced Classification Problems in machine learning?* Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2017/03/imbalanced-classification-problem/

Wen, F., Xiao, J., He, Z., & Gong, X. (2014). Stock Price Prediction based on SSA and SVM. *Procedia Computer Science*, 625-631.