Machine Learning Engineer Nanodegree

Capstone Project

Investment and Trading : Build a Stock Price Predictor[1]

Soyoung Park

March 31, 2017

I. Definition

Project Overview

Technical stock price analysis can be approached as a typical time series data analysis without understanding of health of a particular security while fundamental analysis requires deep domain knowledge about a security or domain of interest.

In this project, one particular security, Intel stock, was chosen for forecasting the future stock trend using various machine learning techniques without having to have domain knowledges. The input price data is from Yahoo! Finance. The baseline model was simple time series forecasting model (Autoregression Integrated Moving Average, ARIMA). Two different approaches were attempted for forecasting the trend.

First, the price or return of 7,14,28 days were predicted with various regression models and the performance of each model was ranked using root mean square of errors (RMSE)

Second, to understand how classification can be applied for this forecasting the continuous variable, the daily return was transformed into binary information as increase or decrease, and it was predicted with various classification models. Trading decision (Buy/Sell) was made for the days of query upon prediction. At the end, the total capital growth as a result [2] of trading upon decisions was backtested.

Problem Statement

Many day stock traders use their own various strategies to find patterns in one particular stock price chart and make their bets without having to understand the health of business or industry because they believe all information is already baked in the stock price. [3,4] This type of

[1] The given problem description of the project: https://docs.google.com/document/d/1ycGeb1QYKATG6jvz74SAMqxrlek9Ed4RYrzWNhWS-0Q/pub

[2] Complete code for this project is at:https://github.com/parksoy/Udacity_nanoDegree_MachineLearning/blob/master/capstone/CapstoneProject_ML4Trading_SoyoungPark.ipynb

[3] "How to Become a Day Trader on the Stock Market | Investopedia." 8 Mar. 2017, http://www.investopedia.com/trading/how-to-become-day-trader/.

[4] "8 Reasons Why You Should Never Become A Day Trader - Business ...." 8 Nov. 2010, http://www.businessinsider.com/reasons-not-to-be-a-day-trader-2010-11.

technical analysis process leading to investment is not only risky due to lack of fundamental analysis or statistical significance, but also, takes significant training efforts to build experience to make a consistently reasonable performance. If technical analysis is believed as valuable, machine learning can be very good alternative solution to guide the day traders to make their trading decisions on each signal. They can take a historical chart and apply high performance machine learning algorithm to understand the trend of price changes more systematically along with the accuracy of prediction for the future. [5] Ultimately, backtesting based on the trading decision can forecast how the asset grows over time, or verify if there is any risk.

Metrics

For ARIMA baseline study and regression approach with various models, RMSE between the prediction and the actual value was used as an evaluation metric. Precisely, RMSE between the predicted log return of 14 days and the actual log return of 14 day value of testset was recorded for ARIMA model, and each RMSE of price, return, log return of 7,14,28 days were recorded for various regression models.

For classification of increase/decrease of daily return, its f1 score was used. The INTC data was balanced as the number of bullish days and bearish days were equivalent, however, f1 score is more appropriate to use for the general classification task than just simple accuracy or hit rate for the possible other imbalanced dataset.

II. Analysis

Data Exploration

Input data is Intel (Ticker symbol, INTC) and S&P 500 (SPY) stock price data from January 1, 1997 to March 9, 2017 in csv file from Yahoo! Finance. Python and its various libraries were used to process the data for this project. User can query the following inputs

- symbol: of interested company
- start_date = datetime.datetime(1997,1,1)
- end_date = datetime.datetime(2017,3,9)

A function, get_data() will download two csv files, one for INTC and other for SPY. Two files are called and combined into a DataFrame and Figure 1 and Table 1 show the first two days of dataset as an example.

[5] "Forecasting stock market movement direction with support vector ...." https://pdfs.semanticscholar.org/ed03/6a6f69d192c98a750e8b937061eecf1aba50.pdf.

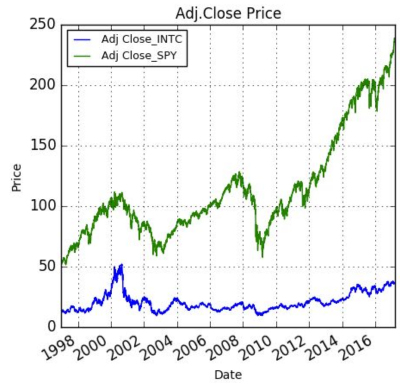


Figure 1.

	Open_SPY	High_SPY	Low_SPY	Close_SPY	Volume_SPY	Adj Close_SPY	Open_INTC	High_INTC	Low_INTC	Close_INTC	Volume_INTC	Adj Close_INTC
1997-01-02	74.375	74.375	72.750000	74.031197	2031900.0	51.586402	131.75	132.0	127.625	130.375	97639200	11.206980
1997-01-03	74.375	75.125	74.078102	75.093697	2123200.0	52.326773	133.00	138.5	132.625	138.375	95648000	11.894656

Figure 1. Table 1. Input dataset that was used for the entire project

Additional features and targets were devised using adjusted close price, 'Adj Close_INTC':

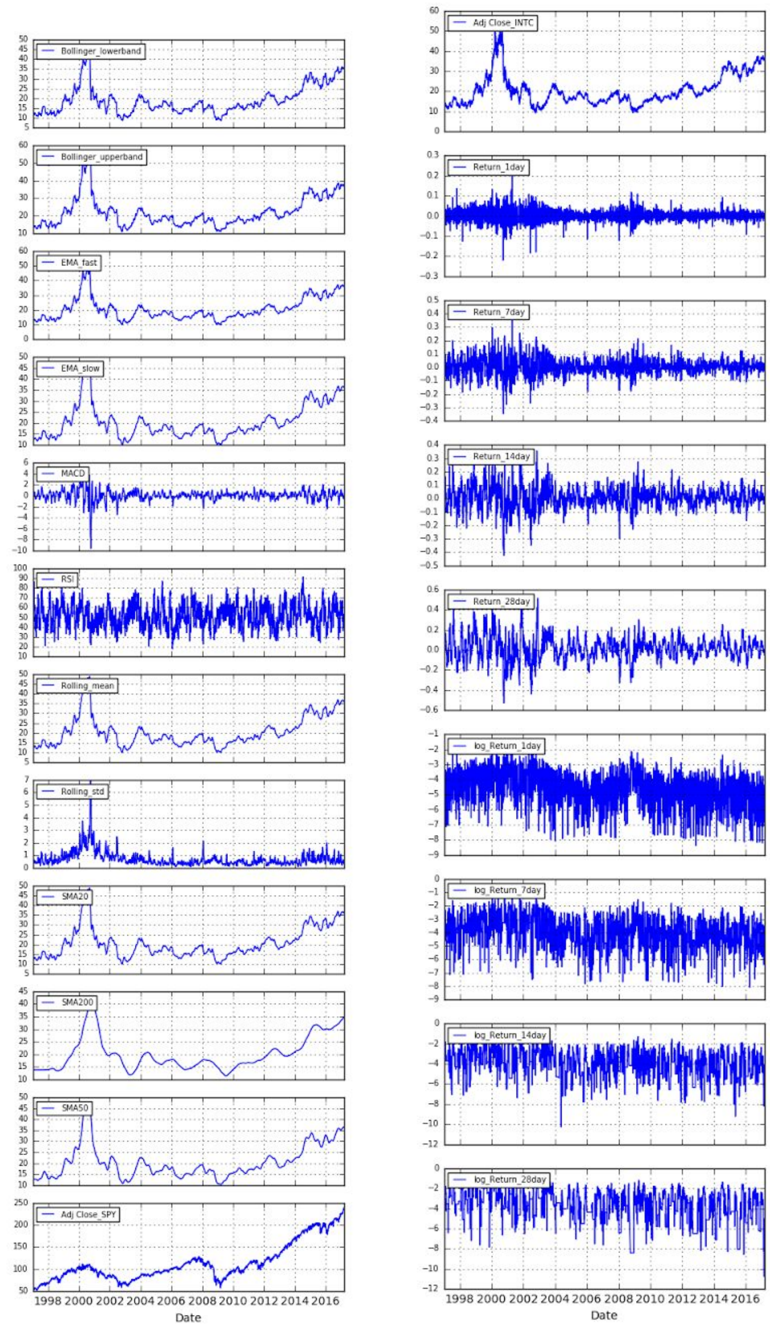
- 'Rolling_mean': rolling mean of adjusted close price for 20 days
- 'Rolling_std': rolling standard deviation of 20 days
- 'Bollinger_lowerband': rolling_mean-2*rolling_std
- 'Bollinger_upperband': rolling_mean+2*rolling_std
- 'EMA_fast': exponential moving average of 12 days
- 'EMA_slow':exponential moving average of 26 days
- 'MACD': Moving average convergence divergence. emafast - emaslow
- 'RSI': 14 days relative strength indicator 100-100/(1+rs) where rs is relative strength, number of price up divided by number of price down
- 'SMA20', 'SMA50', 'SMA200: simple moving average of 20,50,200 days
- 'Adj Close_SPY': Adjusted Close price of S&P 500 market.

Additional target indicators were created on top of simple adjusted close price, 'Adj Close_INTC':

- 'Return_1,7,14,28 day’
- 'log_Return_1,7,14,28 day'

All input data is continuous numeric variables, there was no abnormalities in terms of missing data or outlier data. However, when the additional features or targets were driven from the Adjusted Close_INTC data, missing data(NaN) were treated with backfill for the cases that when return or log return was calculated to avoid any unintentional information is interfered by filling with zero or mean. Dropping NaN was avoided to keep a good number of datapoints. Any issue with taking log of negative or zero return was resolved with backfill fill methodology.

Exploratory Visualization



(a) (b)

Figure 2. (a) 12 technical indicators (b) 9 different target indicators in time series before normalization.

Figure 2 shows (a) features and (b) targets that were used in this project. X axis is in time and y axis is annotated as in legend box. Price is in US dollar and any ratio, return, or log are unit less. In feature plots, time trend of various moving averages or adjusted close price are similar over the course of years. Ratio of two different moving average, such as MACD, or RSI metrics are, however, different trend from various moving average itself.

Algorithms and Techniques

A.ARIMA

As a baseline of the project, simple forecasting the price out of the sample with its own history was attempted with ARIMA model with guidance of the reference.[6] The price data itself, however, was non-stationary, therefore, transforming the price into log return was necessary. 14 day window were chosen to create log return metric to avoid any possible noise by too frequent sampling. After taking log return of 14 days of adjusted close price, the data became stationary, forecast is made one day ahead (one step) out of history as that point is added to history as progresses.

B.Regression

Various regressors were used to model price or return 7,14,28 days, log return of 7,14,28 days. The dataset consisted of 12 features, and 9 different target metrics. The dataset were splitted to trainset (first 2 out of 3 splits in time series) and testset (last one out of 3 splits. Using 'for loop' on 9 different targets, each model was fit on '12 features-one target' pair in train set and 12 features were input to predict the target on testset. RMSE was recorded on each model per each target and ranked with respect to baseline ARIMA model's performance. The following 4 regressors were attempted:

- KNeighborsRegressor(n_neighbors=3),
- RandomForestRegressor,
- DecisionTreeRegressor(max_depth=4),
- AdaBoostRegressor (DecisionTreeRegressor(max_depth=4),n_estimators=300),
- SVR

Fine hyperparameter tunings were not proceeded.

C.Classification

For target metric, daily return was converted to 1(bullish or no change) or 0(bearish) for each day, and 12 features were used to fit and predict those binary target metric. Grid search cross validation was used on 3 timely splitted datasets for the following classifiers with various hyperparameter configuration in the nested for loop:

---

[6] "How to Create an ARIMA Model for Time Series Forecasting with Python." 9 Jan. 2017, http://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/.

- AdaBoostClassifier
- GradientBoostingClassifier
- XGBClassifier
- LogisticRegression
- RandomForestClassifier
- KNeighborsClassifier
- SVC

F1 score was recorded for all cases, and the best model with the best hyperparameter set was selected. At the end, trade decision was made with the best prediction of everyday bullish, bearish information. Buy signal is made when two consecutive bullish days are predicted. Sell signal is recorded when two consecutive bearish days are predicted. Simple buy-sell-hold exercise based on prediction was backtested to understand how portfolio grows over the course of investment period.

Benchmark

One step forecasting out of sample using ARIMA model time was a benchmark/baseline for this project. The evaluation metric is RMSE of predicted value and the actual value of log return of 14 days to compare with other regression models.

III. Methodology

Data Preprocessing

As two (INTC, SPY) csv files were downloaded from Yahoo! Finance, they were read and merged into one DataFrame. Total 5080 data points in 14 different price metrics were starting points. There was no missing values, however, when adjusted close price was transformed into return of X days by shifting X rows (where X=1,7,14,28 days), the return before X days were missing(NaN) and they were backfilled with the first value of return. When log of return was taken, any zero or negative return became infinite. In this case, ‘infinite’ numbers were simply dropped in ARIMA modeling since there were enough number of datapoints (initially 2046 points in history + upto additional 681 points) to forecast one step as one data point is added to history set while training. However, for the case of regression task, the author prefered to keep the same number of datapoints for all targets, (price, return of 7,14,28 day, and log return of 7,14,28 day) to understand if any target is better metric than other or not by comparing RMSE in valid manner. So any ‘infinite’s before X days were backfilled to avoid any further projection of additional information or losing too much of datapoints.

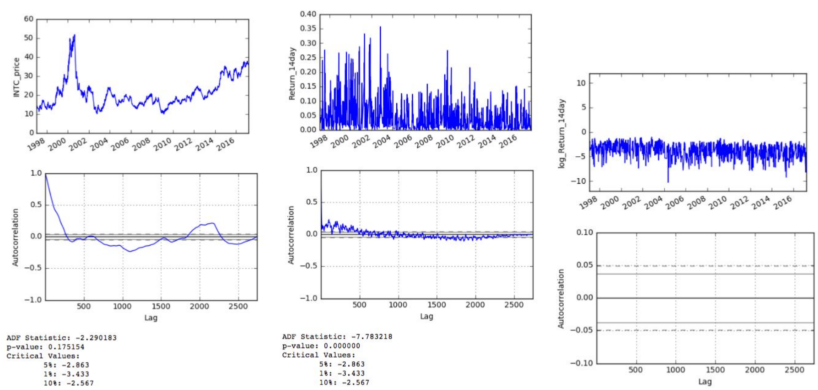
For classification task, the daily return was converted to binary format as 1 for increase/no change, 0 for decrease of price with respect to the previous day.

Implementation

A. ARIMA - Baseline

Based on this time series forecasting tutorial, [7,8] stationary test value on price data over the period of time was -2.290183, which is greater than all of the critical values at the 1%, 5%, and 10% confidence levels. This "non-stationary" data needed to be converted to stationary data to make a valid forecasting. As suggested in the tutorial, return of 14 days instead of price was calculated. Stationary test value on return became smaller than any of confidence level as shown in (b) compared to Figure 3(a). Furthermore, log was taken on return of 14 days to completely remove any autocorrelation. Log of return 14 days in Figure 3(c) suggests the data is stationary and good to use for valid forecasting as shown.

Taking negative or zero return led to infinite value, and they were dropped from the dataset. Final data points were time series splitted and first 2046 points were used as history in trainset, the last 681 points were used for one step forecasting in testset. As one step forecast is made, that datapoint is added to history. RMSE of forecast and actual value(log return of 14 days) in testset was 0.86 as shown in Figure 4(a). Figure 4(b) shows superimposed price of trainset and testset, especially out of sample log return of 14 days were transformed back to price after the first date in testset that the black arrow indicates.

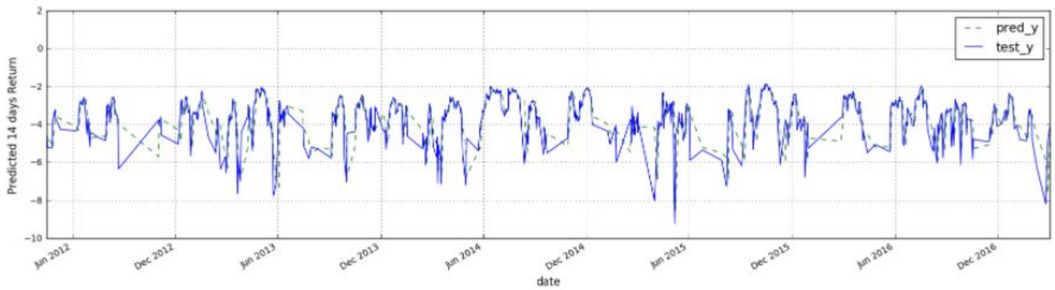


(a) (b) (c)

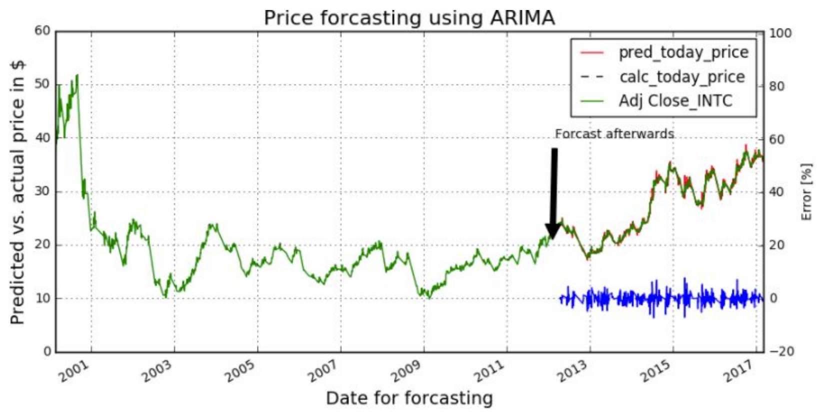
Figure 3. (a) non-stationary (b) stationary time series after taking return of 14 days

---

[7] "How to Check if Time Series Data is Stationary with Python - Machine ...." 30 Dec. 2016, http://machinelearningmastery.com/time-series-data-stationary-python/. Accessed 31 Mar. 2017.
[8] "How to Create an ARIMA Model for Time Series Forecasting with Python." 9 Jan. 2017, http://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/. Accessed 31 Mar. 2017.



(a)



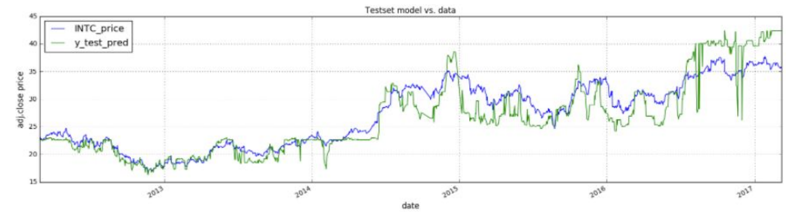
(b)

Figure 4. (a) ARIMA forecasting result on return of 14 days (b) Comparison of the price in and out of the sample: the price was transformed back from the forecasted return out of sample.

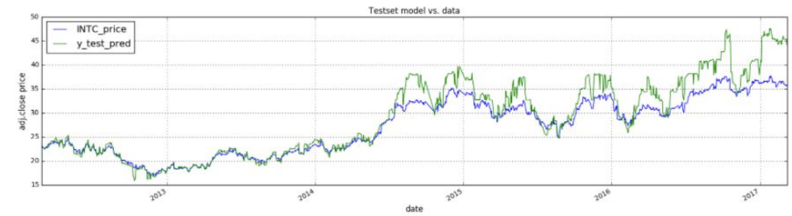
B. Regression

Various regressors were used to train and predict using the following 12 features and the superimpose price of predicted value and actual values for each regressor is shown in Figure 5(a)-(e).

- 'Bollinger_lowerband',
- 'Bollinger_upperband',
- 'EMA_fast',
- 'EMA_slow',
- 'MACD',
- 'RSI',
- 'Rolling_mean',
- 'Rolling_std',
- 'SMA20',
- 'SMA200',
- 'SMA50',
- 'SPY_price'

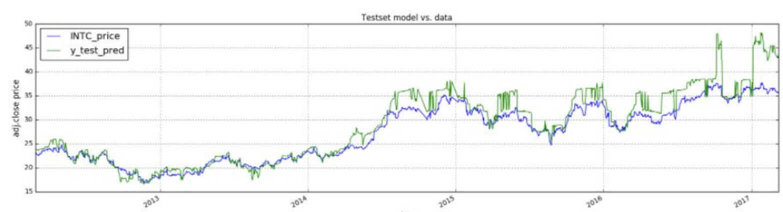


(a) knn

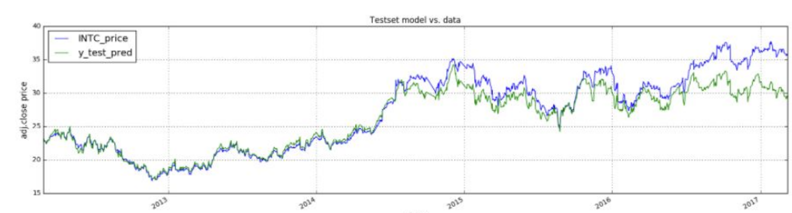


(b) RandomForest

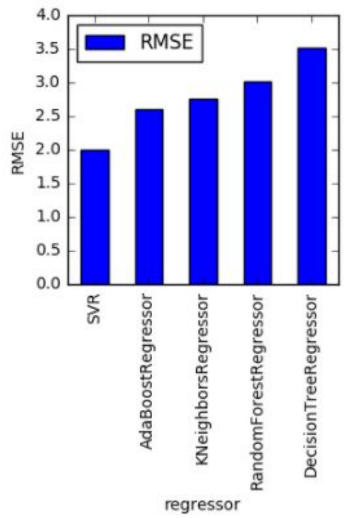
(c) Decision tree



(d) Adaboost with decision tree



(e) SVR regressor



(f) Summary of RMSE in prediction of price in order for various regressors with default args

Figure 5. Prediction result of various regressors (a) Knn (b) RandomForest (c) Decision tree (d) Adaboost with decision tree (e) SVR regressor (f) Summary of RMSE in prediction of price in order for various regressors with default args

Figure 5(f) is summary of RMSE in order between the predicted value and actual value in testset for various regressors. There was no missing data for target value(price), but for the features, backfill was used to fill any missing data due to shifting X days to make moving average or return.

C. Classification

Given with the significant RMSE of all regressors, and a concern of time dependence of price data itself or the correlation of moving average characteristics with the target price that may leads to false interpretation, the daily return was taken and converted to binary information. 1(bullish or no change) or 0(bearish) for each day were predicted for each day, and initially 12 features were used to fit and predict those binary target metric. When various hyperparameters were attempted to be tuned, it was very computationally heavy process, therefore, feature importance was taken and sorted in descending order using random forest classifier as a start as following:

1. feature RSI (0.187172)
2. feature MACD (0.105709)
3. feature Rolling_std (0.089269)
4. feature SPY_price (0.083011)
5. feature EMA_fast (0.072642)
6. feature Bollinger_lowerband (0.070900)
7. feature Bollinger_upperband (0.070351)
8. feature SMA50 (0.069771)
9. feature SMA200 (0.066733)
10. feature EMA_slow (0.066463)

11. feature Rolling_mean (0.061869)
12. feature SMA20 (0.056111)

Therefore, only top 3 meaningful features, 'RSI','MACD', and 'SPY_price' are selected to predict increase(or no change) or decrease of price. The data was balanced between increase/no change(1.0: 2564 points) and decrease (0.0: 2365).

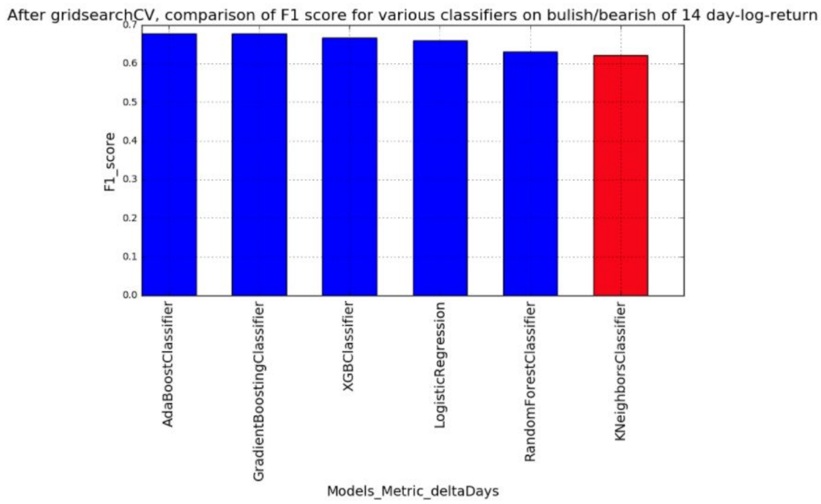


Figure 6. Summary of f1 score in prediction of price increase/decrease in order for various classifiers with many

7 different classifiers were used along with grid search cross validation with different sets of hyper parameters. Cross validation was done on time series splitted 3 chunks of the entire dataset. Figure 6 shows f1 score ranks in descending order for 6 classifiers. SVC was dropped due to significant computational time. The best classifier was AdaBoostClassifier with the highest f1 score, 0.68. Among the sets of hyperparameters, [{'n_estimators': [5, 50], 'learning_rate': [0.05, 0.5, 1.0]}], {'n_estimators': 50, 'learning_rate': 0.5} made the highest f1 score.

After prediction is made on the testset with the best model, Adaboost('n_estimators': 50, 'learning_rate': 0.5), the total capital was backtested based on the following rules: when price increase(1's) occurs in two consecutive days, trade decision is 'buy', and 'sell' for two consecutive 0's. Initial capital is $100,000. Each day, 500 shares were purchased in the morning (at open price) when trade decision is 'buy', and sold at 'sell' signal. Figure 7(c) shows portfolio of initial cap + gain from trades over the course of investment period (testset). Total 54 times of 'sell' in red and 54 times of 'buy' in blue are made over the course of 5 years from February 27, 2012 to March 9, 2017 and it made ~$45,000 profit. Trading cost for each transaction (e-trade $4.95 per trade[9]) was not included in this plot.

[9] "E*TRADE Fees and Rates | Pricing for Investing & Trading | E*TRADE." https://us.etrade.com/what-we-offer/pricing-and-rates.

Refinement

For the regression test, only price was used as target initially as discussed and shown in Figure 5(f). In order to remove a concern on time dependency of the price, return of 7,14,28 days, and further log of them were used as target for regression tasks. Figure 8 shows a significant RMSE reduction in prediction of return as opposed to use price only or ARIMA model. RandomForestRegressor predicts the best with the lowest RMSE of 0.027769 on return of 14 days.

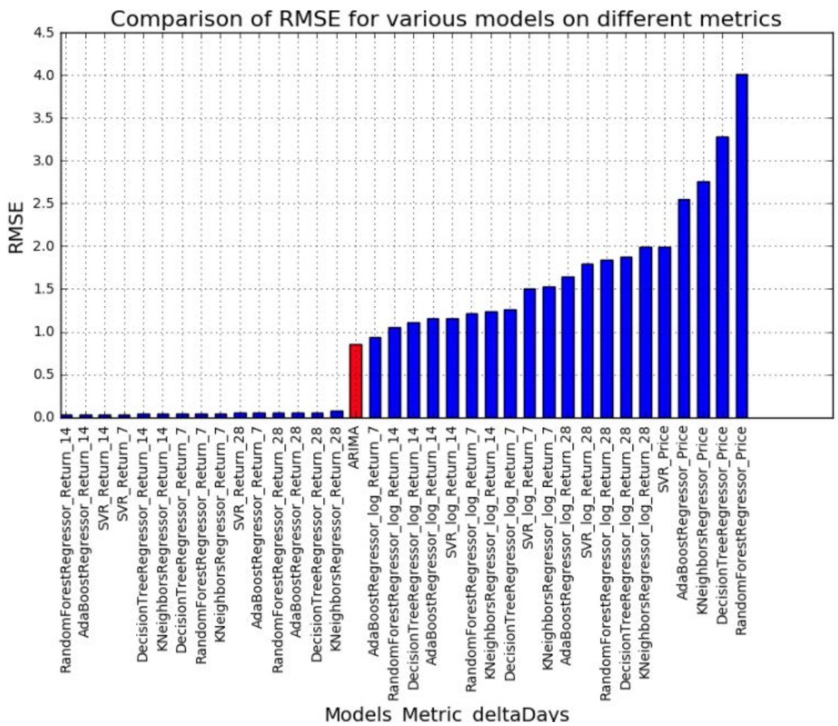


Figure 8. (f) Summary of RMSE in prediction of price, return of 7,14,28 day, log return of 7,14,28 days in order for various regressors with default args

IV. Results

Model Evaluation and Validation

For regression, the final model is RandomForestRegressor on Return of 14 with the smallest RMSE, 0.027769. For classification, the final model is Adaboost classifier with the highest f1 score, 0.68.

Justification/Free-Form Visualization

To comment about net profit of this trading scheme with respect to just holding INTC or SPY, holding INTC stock (3846 shares which is $100,000 worth on the one time purchase) for 5 years without any transaction will gain $38k.

On the other hand, this machine learning based day trading scheme will make $45k minus $535 trading cost for ‘Buy’ and ‘Sell’ 108 times over 5 years, which is slightly more than what holding INTC stock only ($~6400) with the same amount of money. Most important justification of this scheme is that even if there is down trend in INTC stock, the portfolio in Figure 7(b) shows only uptrend. That is meaningful enough to convince that machine learning based trading scheme is worth to investigate more about strategy.

However, in simply investment point of view, holding S&P 500 stocks (724 shares) will gain $74k over 5 years of period of time. Even if the tax rate is assumed to be the same for the simplicity of calculation, it may not be worth to pursue buy/sell trading INTC stocks over 5 years compared to just holding S&P500 stock. More volatile stock with the greater range of swinging rather than stable INTC stock might be of next interest to try the model.

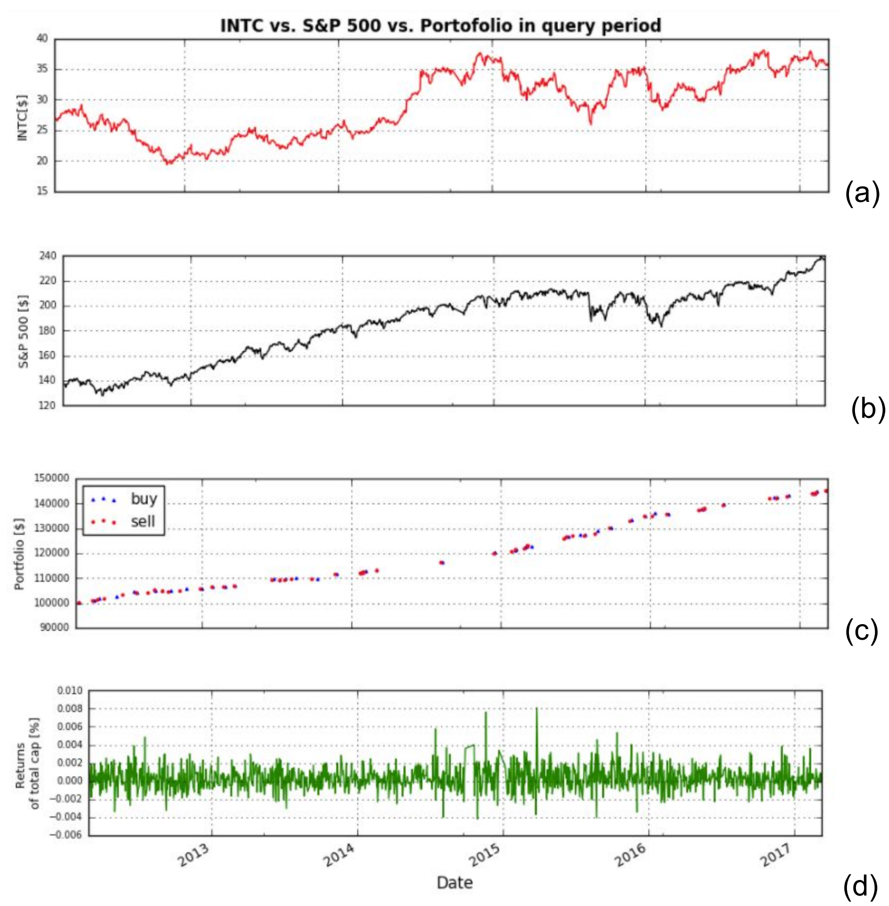


Figure 7. (a) INTC price per share in testset (b) SPY price per share (c) portfolio growth over the investment period. Initial capital is $100,000, trading is made based on the prediction. Total value in y axis

is initial capital+ cumulated profit based on 54 ‘buy’ and 54 ‘sell’ actions. (d) daily return of portfolio. Daily return itself doesn’t grow over time.

V. Conclusion

Reflection

One particular security, Intel stock, was chosen for forecasting the future stock trend using various machine learning techniques. Regression and classification approaches were attempted for forecasting the trend. Instead of forecasting log return using ARIMA or price itself by regression, return of 14 is better target metric to predict than any other metrics or return of different number of days. For classification, the daily return was transformed into binary information and predicted with various classification models. Adaboost was the best performer. Trades (Buy/Sell) were simulated for the days of query upon prediction and portfolio backtest suggests that this machine learning based trading scheme on this INTC stock is proven to be slightly more profitable compared to just holding the equivalent amount of its stocks.

Improvement

Only INTC stock was tested for various regression and classification tasks. However, it would be interesting to know if this scheme is applicable for any trend or volatility of stock. Also once its robustness is proven, it can be expanded to create a portfolio of multiple stocks to make even greater profits with lesser risk.