Stock Movement Predictor: A Machine Learning Approach

Ian Huang

Udacity Nanodegree Capstone Project

Table of Contents

Definition

Project Overview

When being asked about today's economic environment, most people will point to unemployment rate, Gross Domestic Product, and unarguably, stock indexes such as Dow Jones Industrial Average to measure the economic situation in modern times. Similar to the using stock index as an indicator, oftentimes we would look to how well the stock price is moving against others and compared to its historical performances when we are measuring the performance of overall market or a company. When companies operate beyond certain scale and require more funding for expansion, one of the most common ways is to "go public"—in which the company register itself on one of the major stock exchanges such as New York Stock Exchange and offer certain shares to the public. By offering public stock floatation, company would receive funding from investors, be it retail or institutional ones, and in return investor would have voting rights in the board. The stock price of each company is comprised of many aspects, but generally reflects its current operation performance, the business sector, and the macroeconomics. A stock receiving high valuation denotes investors are expecting the company to outperform in the future and thus are willing to buy the stock in the hope that the price would later increase to reflect its value. Vivid examples are that Tesla and Amazon are generating a lot waves in press not only because of their products and services, but also because investors hold high hope in the companies' business endeavors would further push up their stock prices. Other than these much-hyped businesses which people generally know a thing or two, investors often ask themselves, "Are there other ways to predict stock prices?"

Stock prices forecasting has always been a fascinating subject, and in recent years applying machine learning algorithms to predict the stock price trend, whether it is a classification issue in the format of prices going either up or down, or try to predict the actual price range. Various researches – including sentiment analysis using semantic frames to predict price movement[1] or predictions under high frequency trading scenarios[2] – have been put into light, and certain innovative approaches have made these tools more accessible to the public. Platforms such as Quantopian is using crowd sourcing method to find algorithms with the best performances, and research advisors for institutional investors such as Lucena Research have made testing trading strategy easier than before.

[1] Boyi Xie, Rebecca J. Passonneau, Leon Wu, Germ´an G. Creamer. 2013. Semantic Frames to Predict Stock Price Movement. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics.
[2] Michael David Rechenthin. 2014. Machine-learning classification techniques for the analysis and prediction of high-frequency stock direction. University of Iowa.

Problem Statement

With so many solutions available to the public, most investors simply care one thing—Will a stock go up or down? If the answer yes, if would certainly make the case easier for an investor to make investment decision. From my personal experiences, most people who label themselves as occasional investors usually speculated on certain information they received and thus invested in stocks relevant to the implication. Data suggesting that the average holding period for stocks listed at NYSE is eight months[3], therefore it is reasonable to assume most individual investors care more about short-term price movement as opposed to the long-term approach of identifying the intrinsic value.

The project would be building a straightforward tool for investors to inquire the probability of certain stock going up or down in the designated time frame. Therefore, the objective of the project is to predict whether the price of a stock would go up or down in a short time horizon after the time of the query being submitted. The machine learning task is framed as a binary classification task and the objective is to figure out the data input and the derived feature that are relevant in making these predictions.

For instance, if we are going to predict the price of Apple, Inc. stock (AAPL) in the next five days, should we simply use the historical data of its own or also look at other stocks that are in the same industry? In addition to that, what kind of information other than price, such as volume, should be featured in the dataset? These are all the problems that would be attempted to tackle in the project. For the clarity of the project design and the problem scope, the historical data of AAPL would be used as raw input, and features are generated based on commonly used technical indicators. The overall strategy is to test out several classifier algorithms and the one with best performance under evaluation metric would be selected to predict the movement of the stock.

Metrics

As mentioned earlier, the problem has been framed as a binary classification task, thus, the evaluation metric is the mean accuracy score, which is the number of correct predictions from all predictions made. Given the randomness characteristic of stock price moving direction, a threshold of 50% should be used to justify the accuracy score performance of a classifier.

[3] Massachusetts Financial Services Company. 2016. In White Paper Series. https://www.mfs.com/wps/FileServerServlet?servletCommand=serveUnprotectedFileAsset&fileAssetPath=/files/documents/news/mfse_time_wp.pdf

Analysis

Data Exploration

Using Quandl Python module to load Wiki EOD(link) data, a crowdfunded database that provides end-of-day stock prices, dividends and splits for 3,000 US companies. The raw data extracted from the database is consisted of opening price, intraday high price, intraday low price, closing price, trading volume, ex-dividend amount per share, stock split ratio, adjusted opening price, adjusted intraday high price, adjusted intraday low price, adjusted closing price, and adjusted volume. The benefit of using Quandl library also including that it is easier to get real-time and updated price without loading a pre-existing dataset.

It is very easy to get historical daily prices of the previous indices. Python provides easy libraries to handle the download. A table of stock information from the Wiki database using Quandl module is shown as below.

Date	Open	High	Low	Close	Volume	Ex-Divi-dend	Split Ratio	Adj. Open	Adj. High	Adj. Low	Adj. Close	Adj. Volume
2012-01-03	409.40	412.50	409.00	411.23	10793600.0	0.0	1.0	52.613606	53.011999	52.562200	52.848787	75555200.0
2012-01-04	410.00	414.68	409.28	413.44	9286500.0	0.0	1.0	52.690714	53.292160	52.598184	53.132802	65005500.0
2012-01-05	414.95	418.55	412.67	418.03	9688200.0	0.0	1.0	53.326858	53.789509	53.033847	53.722681	67817400.0
2012-01-06	419.77	422.75	419.22	422.40	11367600.0	0.0	1.0	53.946296	54.329267	53.875613	54.284287	79573200.0
2012-01-09	425.50	427.75	421.35	421.73	14072300.0	0.0	1.0	54.682693	54.971837	54.149348	54.198183	98506100.0

Table 1. *Stock Information of AAPL acquired from Wiki EOD Database*

Date: Only market open days are listed, closed and holidays are excluded. The exchange Apple registered at is NASDAQ and ticker being used is AAPL.

Open (in US dollars): Price of the stock at the opening of the trading day.

High (in US dollars): Highest price of the stock during the trading day.

Low (in US dollars): Lowest price of the stock during the trading day.

Close (in US dollars): Price of the stock at the closing of the trading.

Volume: Number of shares traded in a stock during the market hours. When buyers and sellers agree to make a transaction at a certain price, it is considered one transaction, and each transaction contributes to the count of total volume. If 500,000 transactions occur in a day, the volume for the day is 500,000.

Ex-Dividend (in US dollars): The amount each share receives, or dividend per share. For example, the most recent ex-dividend date of AAPL is August 10th, 2017 and the amount is 0.63, that means each shareholder will receive $0.63 per share. Investors need to own the stock one day before the ex-dividend date to receive the dividend. After the ex-dividend date has been declared, the stock will usually drop in price by the amount of the expected dividend, but often times dividend distribution has already been priced into the stock price.

Split Ratio: Split is defined as a company divides its existing shares into multiple shares. Split does not add any real value since the total dollar value of the shares remains the same compared to pre-split amounts. The number of shares outstanding increases by a specific multiple, for example, a 2 for 1 split means that the stockholder will have two shares for every share held earlier. A split ratio is 1 indicates that no split happens on the given date. Apple stock has four splits in its history, the most recent split took place on June 09, 2014 and it was a 7 for 1 split.

Adjusted Open/ High/ Low/ Close/ Volume (in US dollars): All of the above information may be affected by the underlying company action that changes the value of the stocks, typically dividends, stock splits, share buybacks, or material changes such as spinoffs, mergers, and acquisitions. The figure is applied retrospectively after an announcement is made, and might change overtime. For example, the adjusted close price of a company may change four times a year after the company announced quarterly dividend.

Data Sampling Range

Stock market raw data is considered as non-stationary and its behavior may be affected by various factors. The underlying stock of commodities companies might be affected by world demand, while consumer discretionary sector may show seasonal and cyclical patterns. Following the 2008 Financial Crisis turmoil, the market is demonstrating an upward trend since mid-2009. Thus, data points of the stock market are considered as non-stationary. Since the project would allow the user to make inquiries for major U.S. companies, transforming the data to a stationary one using a universal approach may cause biases as not all stocks are showing identical patterns or trends. Thus, the data would not be transformed to stationary in the project.

As highlighted before a period of serious disruption in the stock market happened between 2008 and mid-2009, and thus, the project would only look at data points after January 1, 2011.

As for sampling period, 5-year return is one of the commonly used measure to evaluate the long-term performance of a stock and it would be applied as the range of historical data. Overall, the project would allow user to predict using any data point samples between 2011 and the current date, with a maximum dataset length of 5 years. To test and trial, the examples generated throughout the report would be using the AAPL from 1/3/2012 to 12/30/2016, which would give us 1,258 data entries and 12 columns.

Statistical Description and Exploratory Visualization

	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj. Low	Adj. Close	Adj. Volume
count	1258	1258	1258	1258	1.26E+03	1258	1258	1258	1258	1258	1258	1.26E+03
mean	313.50	316.32	310.33	313.33	3.17E+07	0.02	1.00	88.17	88.96	87.32	88.14	7.73E+07
std	215.87	217.74	213.59	215.69	2.22E+07	0.24	0.17	20.96	21.11	20.81	20.97	5.06E+07
min	90.00	90.70	89.47	90.28	5.70E+06	0.00	1.00	50.59	52.11	50.22	50.93	1.15E+07
25%	109.74	110.78	108.66	109.61	1.39E+07	0.00	1.00	70.22	71.00	69.65	70.19	4.19E+07
50%	130.00	130.80	128.69	129.82	2.70E+07	0.00	1.00	89.16	89.79	88.22	89.18	6.32E+07
75%	527.90	531.95	522.36	527.41	4.38E+07	0.00	1.00	106.86	107.78	105.89	106.91	9.74E+07
max	702.41	705.07	699.57	702.10	1.90E+08	3.29	7.00	128.23	128.31	125.84	126.93	3.77E+08

Table 2. *Descriptive statistics of the AAPL dataset between 2012-2016*

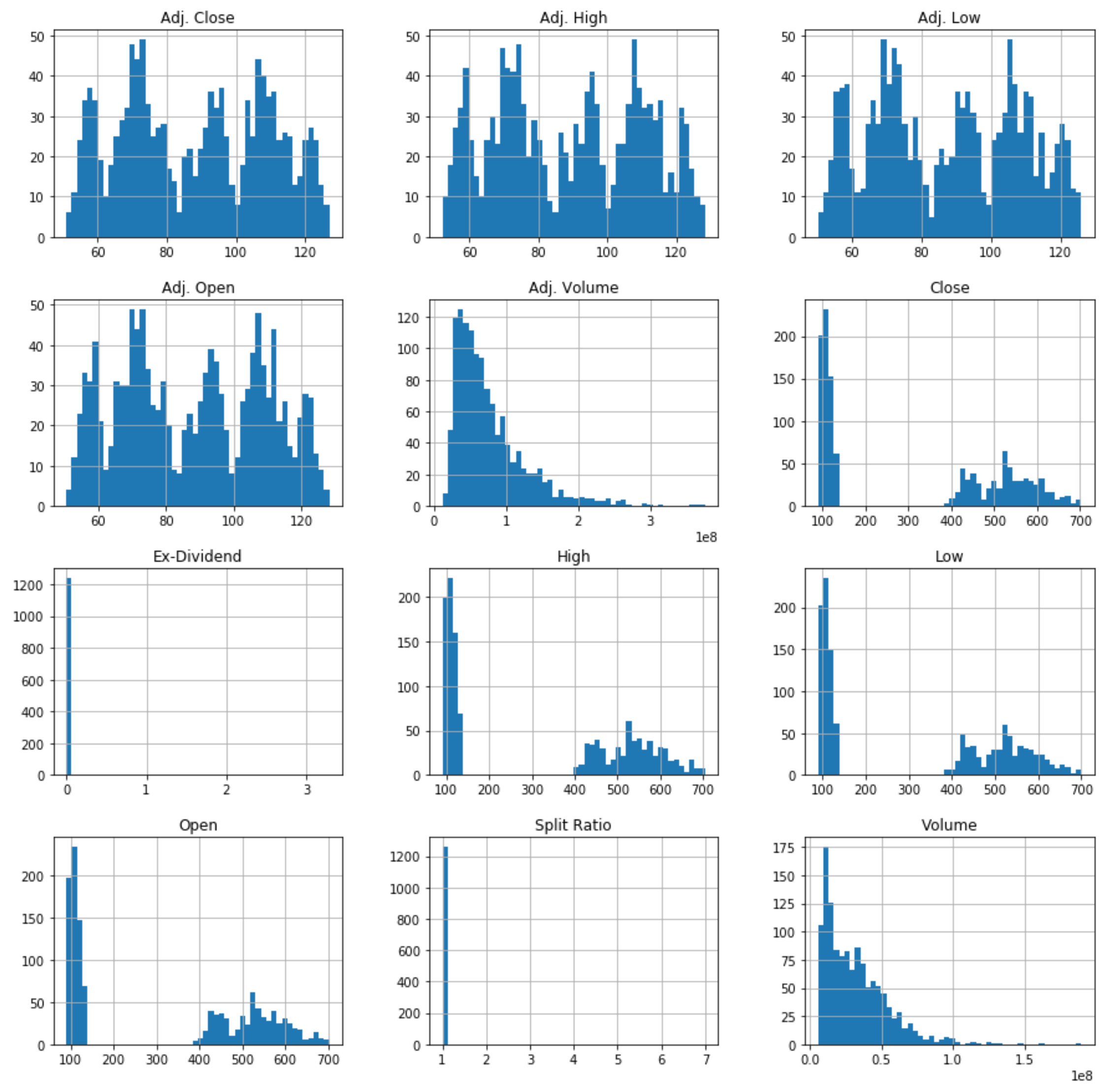

Figure 1. *Histogram of the AAPL dataset between 2012-2016*

There is no categorical variable and no missing value in raw dataset. One of the characteristics of AAPL stock that needs to be highlighted in this period is that the stock had a stock split in 2014, and as a result the stock price falls into two distinct groups, one is around \$100 a share and the other is in the range of \$400 to \$700. This demonstrates why this project will use adjusted price, instead of original price, to calculate and generate the features. Another important characteristic is that non-stationary pattern could be clearly seen in the column of adjusted prices. The trending factor is significant in AAPL stock in this period. As mentioned before, the project takes a general approach to predict all stocks listed in the database, therefore transformation would be applied despite stock data is generally considered as non-stationary. Lastly, volume is heavily skewed to the right as the market would sometime see extreme amounts of transactions, which contributes to the anomalies of massive volume.

Algorithms and Techniques

The binary classification objective and the multiple features used for prediction decides the algorithms used in the project. In general, algorithms and models that have good classification performance in high-dimensional space are considered. As the maximum query range of the project is set at 5-year, which is equivalent to approximately 1,260 sample sizes, the amount of training data is considered reasonable and thus computational efficiency should not be a huge factor. Apart from probability-based (Naïve Bayes) and distance-based (K-nearest neighbors) algorithms, boosting and bootstrapping algorithms are also tested in the project to see if they do improve the performance significantly.

As for data characteristics, time-series stock data is generally considered as non-stationary. However, as most of the chosen features and the generated features (explained in the later project section) have shown a pattern of Gaussian distribution, the likelihood of the features for classification should be assumed as Gaussian and therefore matches prerequisites of several algorithms identified below.

Logistic Regression: Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function, which is the cumulative logistic distribution.

The dependent variables of the project have been transformed to binary outcomes, where "1" represents the stock price of underlying stock is up from the previous day and "0" represents down. Thus, the regression model is essentially a binary logistic regression one. Binary logistic regression model predicts the odds of being a case based on the values of the independent variables(features), and the odds are defined as the probability that a particular outcome is a case divided by the probability that it is a non-case.

Gaussian Naïve Bayes: Naïve Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of independence between every pair of features. Naïve Bayes methods require a small amount of training data to estimate the necessary parameters.

Quadratic Discriminant Analysis: Quadratic discriminant analysis (QDA) is closely related to linear discriminant analysis (LDA), where it uses Bayes' theorem and is assumed that the measurements from each class are normally distributed. Unlike LDA, however, in QDA there is no assumption that the covariance of each of the classes is identical. Each class uses its own estimate of covariance. QDA has a quadratic decision boundary, generated by fitting class conditional densities to the data.

K-Nearest Neighbors: K-Nearest Neighbors (KNN) classification is implemented based on voting of k-nearest neighbors of each query point. A new point is classified by a majority vote of its neighbors and is assigned to the data class which has the most representatives within the nearest neighbors of the point. For example, if k = 1, then the object is simply assigned to the class of that single nearest neighbor.

KNN is an instance-based algorithm, which means it does not attempt to construct a general internal model, but simply stores instances of the training data. This approach is well suited to solve the stock prediction problem as the training samples change over time, and a new query is classified based on the latest training samples or specified timeframe of training samples. The two major parameters that affect the performance of KNN are the number of neighbors and weights.

Support Vector Machines: Support Vector Machines (SVM) model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall. SVM algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary classifier.

SVM provides an effective mean to dissect high-dimensional data as a result of feature generation. The model uses a subset of training points (support vectors) in the decision function, therefore it is memory efficient and would provide an economical way if the data range would include more samples and extend beyond the five-year span in the project. In addition to performing linear classification, SVM can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces. In the case of the project during training phase, “linear” kernel simply would not fit or converge. “Poly” kernel seems to work better when more samples are contained in each fold (performs the best when around 370 samples per chunk), whereas “RBF” kernel seems to have better performance when the less samples per chunk, or splitting the training samples into more folds. “Sigmoid” kernel has the worst performance among all.

The goal of ensemble methods is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability and robustness over a single estimator. Ensemble methods are categorized into two families, averaging and boosting methods. Averaging method, such as Random Forest, the principle is to build several estimators independently and then to average the predictions. On average, the combined estimator is usually better than any of the single base estimator because its variance is reduced. On the other hand, in boosting methods (e.g. Adaptive Boosting, Gradient Tree Boosting), base estimators are built sequentially and one tries to reduce the bias of the combined estimator. The motivation is to combine several weak models to produce a powerful ensemble. By building instances on random subsets of the training set and then aggregate individual predictions to form a final prediction, ensemble learning methods provide a way to reduce overfitting. In general, bagging methods work best with strong and complex models (e.g., fully developed decision trees), in contrast with boosting methods which usually work best with weak models (e.g., shallow decision trees).

Random Forest: Random forests classifier is an ensemble learning method that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. Compared to decision tree classifier, random decision forests correct for decision trees' habit of overfitting to their training set. Random Forests is an improved evolution on the Bagging algorithm, which solves the high variance issue of a single tree but introduces an unavoidable high correlation among all the bootstrapped trees. The main improvement comes from that only a subset of the available features is selected each time a tree is built. As a result, the trees are generally uncorrelated one to another and the final result is much more reliable.

The sub-sample size is always the same as the original input sample size, and the samples are drawn with replacement by default. The parameters of number of estimators and max features are set to default, which is 10 and the square root of features(“auto”). In any case, the square root of the number of predictors is a good balance in terms of bias-variance trade-off, since setting max features equal to the total number of features would be theoretically applying Bagging.

Adaptive Boosting: Adaptive Boosting(AdaBoost) classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.

It can be used in conjunction with many other types of learning algorithms to improve their performance. The output of the other learning algorithms ('weak learners') is combined into a weighted sum that represents the final output of the boosted classifier. AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers. AdaBoost is sensitive to noisy data and outliers. In some problems, it can be less susceptible to the overfitting problem than other learning algorithms. The individual learners can be weak, but as long as the performance of each one is slightly better than random guessing (e.g., their error rate is smaller than 0.5 for binary classification), the final model can be proven to converge to a strong learner.

Every learning algorithm will tend to suit some problem types better than others, and will typically have many different parameters and configurations to be adjusted before achieving optimal performance on a dataset, AdaBoost (with decision trees as the weak learners) is often referred to as the best out-of-the-box classifier. When used with decision tree learning, information gathered at each stage of the AdaBoost algorithm about the relative 'hardness' of each training sample is fed into the tree growing algorithm such that later trees tend to focus on harder-to-classify examples.

Gradient Tree Boosting: Gradient Tree Boosting is a generalization of boosting to arbitrary differentiable loss functions for classification. The model builds an additive model in a forward stage-wise fashion, and in each stage regression trees (the parameter of n_classes_) are fit on the negative gradient of the binomial or multinomial deviance loss function. In the case of binary classification, only a single regression tree is induced.

Benchmark

The project referenced several different classification approaches to predict the stock trend. After reviewing several literatures available, the project decides to use one of the models that Mark Dunne[4] tests in his project. The model simply uses the stock prices in preceding 5 days to predict the stock performance in the 6th day. Essentially, the model uses single feature of time-series price to predict whether the stock price fall or rise immediately following the preceding five days.

For benchmarking purposes, the dataset is built using the dataset assembled in the Data Sampling Range section, and the same cross validation approach, time series split cross validation, is applied to get a fair comparison with other models. Under this cross-validation approach, the training samples are further being divided into training and validation sets. The data that the project will be using is the percentage change in closing price of the stock of the preceding 5 days. The dependent variable will be the trend of the 6th day. The table below is an extract of the first 5 rows in the dataset where the dependent variable, *6th Day Trend*, is encoded based on whether its return is

[4] Mark Dunne. 2016. Stock Market Prediction. University College Cork Computer Science Department. http://markdunne.github.io/public/mark-dunne-stock-market-prediction.pdf

positive or not. Columns labelled day 1 to day 5 are the percentage change in closing price of the preceding 5 days.

Date	1th Day	2th Day	3th Day	4th Day	5th Day	6th Day Return	6th Day Trend	UpDown
1/3/2012	0.011993	0.007934	-0.009569	0.006159	-0.000296	0.015383	Gain	1
1/4/2012	0.007934	-0.009569	0.006159	-0.000296	0.015383	0.005374	Gain	1
1/5/2012	-0.009569	0.006159	-0.000296	0.015383	0.005374	0.011102	Gain	1
1/6/2012	0.006159	-0.000296	0.015383	0.005374	0.011102	0.010454	Gain	1
1/9/2012	-0.000296	0.015383	0.005374	0.011102	0.010454	-0.001586	Loss	0

Table 3. *Data Extract of Benchmark Model*

Model Name	Hyper Parameters Tested	Average Accuracy
Logistic Regression	-	**0.5040**
KNeighbors Classifier	k: {5,**10**,25}, weights: {uniform, **distance**}	**0.5232**
GaussianNB	-	**0.5010**

Table 4. *Accuracy Scores of Benchmark Model*

Table 4 shows the accuracy scores of each model that we tried on the given data. The best result from the benchmark model is an accuracy of 0.5232 using K-nearest Neighbors classifier, with the parameters set at number of neighbors equals 5 and using distance method as its weighting approach. Using the best classifier to predict on test set resulted in an accuracy score of 0.4966.

Based on Efficient Market Hypothesis, the prices of stocks have already reflected all the information and the movement should be of absolute randomness, and therefore the expected accuracy should be of 50%. Thus, the mean accuracy score of the selected model and algorithm should achieve least 0.50 in order to validate its predictive efficacy. In this case, using the benchmark model approach does not really perform much better than a random coin toss, and its prediction accuracy on test dataset is even slightly less than 50%.

Methodology

Data Preprocessing

Common technique

Data range is extended beyond the query range of 5 years in preparation of generating the technical analysis features that would be used to predict the stock movement. After this feature generation process, samples that are outside of the query range would be dropped, and a function to check whether there is null data in the final dataset is executed in order to ensure data integrity.

Feature selection

As the project aims to predict short term price movement as opposed to forecasting a target price, thus, fundamental features of enterprise valuation such as split ratio, intraday trading prices, and ex-dividend figures that might be helpful for finding the intrinsic value, are dropped.

Technical analysis, on the other hand, is commonly used to identify stock price movement, momentum, and trend approach compared to fundamental analysis. Metrics such as Moving Averages and Moving Average Convergence Divergence are some of the popular technical analysis features.

Different from most of the stock prediction that will use price as one – if not the most important – of all features, the project will use return to achieve normalization of time-series data. This approach will measure all variables in a comparable metric, thus enabling evaluation of analytic relationships amongst two or more variables despite originating from price series of unequal values.[5] The same approach applies to using On Balance Volume, instead volume, as the indicator of the amount of transactions. A total of eight features are used in the final model.

Feature generation

The features generated in the process are described below. Explanation also includes the pre-processing steps required to make sure all columns in the data range have corresponding features.

Daily Return: In order to generate other technical indicators, daily return is calculated first. Daily return is calculated by subtracting the value of last (T-1) closing price from the present (T) closing price, and then dividing by the last closing price. It should be noted this is a trailing indicator, which means the result is lagged and not predictive of future trend. In other words, today's daily return is only reflective of the performance in the period between today and yesterday. To get the prediction, tomorrow's daily return will be shifted one day back to become today's prediction of pricing trend, and which would become the dataset's target value. Therefore, today's daily return will be one of the features, while tomorrow's daily return will be used as the target value.

[5] https://quantivity.wordpress.com/2011/02/21/why-log-returns/

Daily return will be calculated based on logarithm, instead of raw return. There are several reasons to use logarithm of returns, but the most significant one is log-normality. For instance, it is common to assume that stock returns are normally distributed. In this context, log returns are far superior to arithmetic returns since the sum of repeated samples from a normal distribution is normally distributed. However, the product of repeated samples from a normal distribution is not normally distributed.[6]

Rolling Return: Rolling return is the average return for a given period and is useful for examining the behavior of returns for holding periods. The period chosen in the project is 30 days, which could be interpreted as the stock performance of the past month.

Exponential Moving Average (EMA): To explain EMA, Simple Moving Average (SMA) should be briefed first. SMA is calculated by dividing the sum of a set of prices by the total number of prices found in the series, the result is also known as an arithmetic mean average.

SMA assigns equal weighting to all values, while EMA gives a higher weighting to recent prices than SMA does. In other words, EMA is more sensitive and reactive to the latest price changes in the data used in its calculation.

EMA is also different from SMA in that a given day's EMA calculation depends on the EMA calculations for all the days prior to that day. The steps to calculating EMA are as follow. First, a SMA is used as the previous period's EMA. Second, calculate the weighting multiplier. The weighting multiplier is defined as 2 divided by time periods plus 1, or (2 / (Time periods + 1)). Third, calculate the exponential moving average for each day between the initial EMA value and today, using the price, the multiplier, and the previous period's EMA value. The formula is as below.

EMA: {Close – EMA (previous day)} d days to achieve both accuracy and efficiency.

This project will take two different periods of EMA, 50 days and 200 days, respectively. It is generally considered that the longer moving average sets the tone for the bigger trend and the shorter moving average is used to generate the signals.[7] For example, as long as the 50-day EMA of a stock price remains above the 200-day EMA, the stock is generally thought to be trending upward (bullish). A crossover to the downside of the 200-day moving average is interpreted as be trending downward (bearish).

[6] http://www.dcfnerds.com/94/arithmetic-vs-logarithmic-rates-of-return/
[7] http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:moving_averages

Moving Average Convergence Divergence (MACD): MACD is largely due to its ability to help quickly spot increasing short-term momentum. MACD is designed to profit from the divergence by analyzing the difference between the two EMAs, which is calculated by subtracting long-term EMA from the short-term EMA. The default setting of long-term period is 26-day, while the short-term period is 12-day.

If a short-term EMA cross above a longer-term EMA, it is a signal of increasing upward momentum. This bullish crossover suggests that the price has recently been rising at a faster rate than it has in the past, so it is a common technical buy sign. Conversely, a short-term EMA crossing below a longer-term average illustrates that the stock's price has been moving downward at a faster rate, and that it indicates a technical sell sign.[8]

A positive MACD value, created when the short-term average is above the longer-term average, is used to signal increasing upward momentum.

On Balance Volume (OBV): OBV measures buying and selling pressure as a cumulative indicator that adds volume on up days and subtracts volume on down days. It is one of the indicators to measure positive and negative volume flow of a given stock. OBV rises when volume on up days outpaces volume on down days. OBV falls when volume on down days is stronger. A rising OBV reflects positive volume pressure that can lead to higher prices. Conversely, falling OBV reflects negative volume pressure that can foreshadow lower prices.

OBV is calculated using the below formula: If the closing price is above the prior close price then: Current OBV = Previous OBV + Current Volume. If the closing price is below the prior close price then: Current OBV = Previous OBV - Current Volume. If the closing prices equals the prior close price then: Current OBV = Previous OBV.[9]

Percentage Price Oscillator (PPO): PPO generates the same signals at MACD, but provides an added dimension as a percentage version of MACD. PPO levels in one security can be compared over extended periods of time, even if the price has doubled or tripled. While MACD measures the absolute difference between two moving averages, PPO makes this a relative value by dividing difference by the slower moving average (26-day EMA). PPO is simply the MACD value divided by the longer moving average. The result is multiplied by 100 to move the decimal place two spots.[10]

Relative Strength Index (RSI): RSI provides a relative evaluation of the strength of a security's recent price performance, thus making it a momentum indicator. RSI values

---

[8] http://www.investopedia.com/articles/technical/082701.asp

[9] http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:on_balance_volume_obv

[10] http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:price_oscillators_ppo

range from 0 to 100. The default time frame for comparing up periods to down periods is 14, as in 14 trading days.

Traditional interpretation and usage of the RSI is that RSI values of 70 or above indicate that a security is becoming overbought or overvalued, and therefore may be primed for a trend reversal or corrective pullback in price. On the other side of RSI values, a RSI reading of 30 or below is commonly interpreted as indicating an oversold or undervalued condition that may signal a trend change or corrective price reversal to the upside.[11]

$$\text{RSI} = 100 - \frac{100}{1 + \text{RS}}$$

* RS = Average Gain / Average Loss

* First Average Gain = Sum of Gains over the past 14 periods / 14.

* First Average Loss = Sum of Losses over the past 14 periods / 14

* Average Gain = [(previous Average Gain) x 13 + current Gain] / 14.

* Average Loss = [(previous Average Loss) x 13 + current Loss] / 14. [12]

[11] http://www.investopedia.com/terms/r/rsi.asp
[12] http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:relative_strength_index_rsi

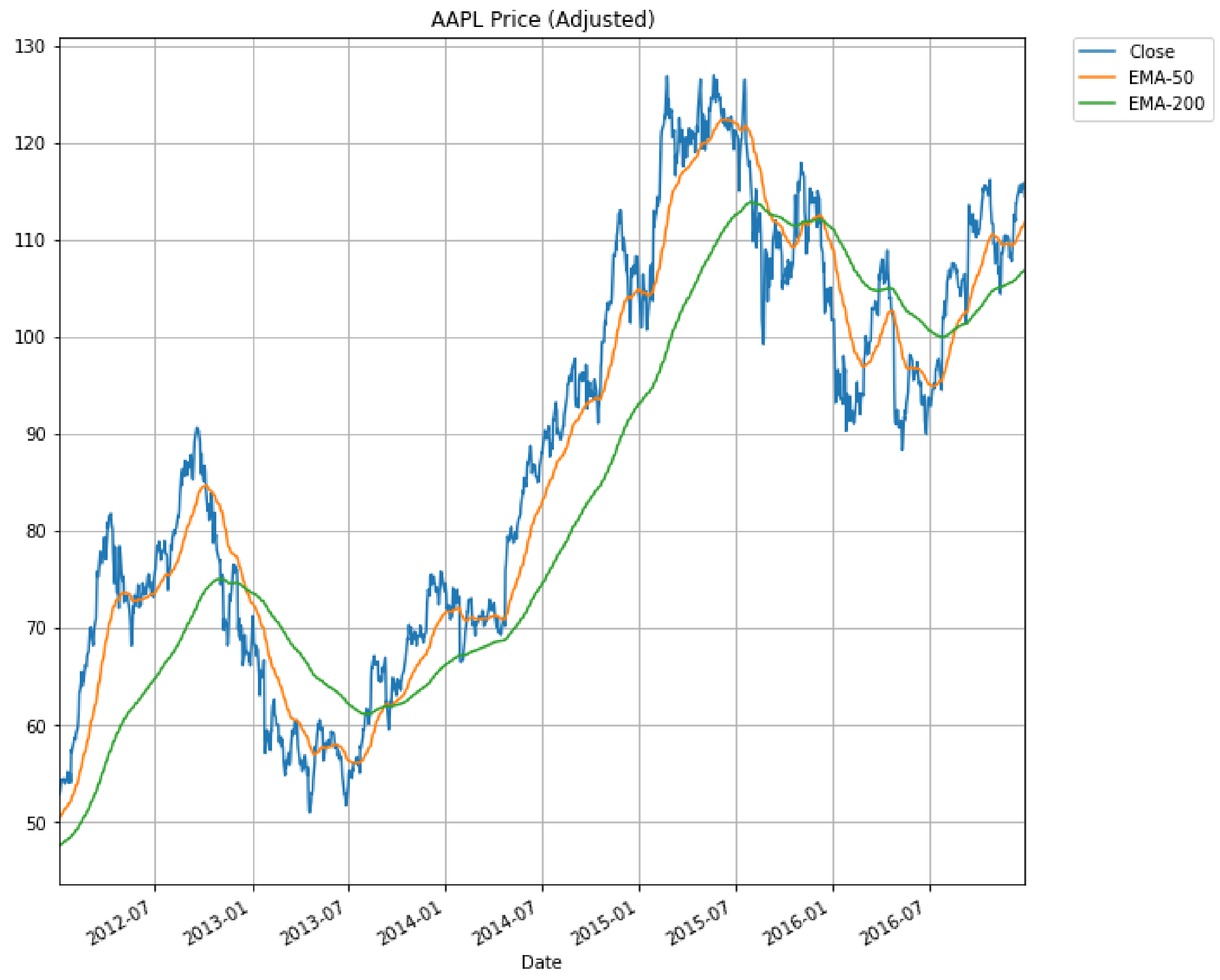

Figure 2. *Plot of AAPL price and rolling mean prices between 2012-2016*

	Close	Volume	Return	Return_T	RR-30	EMA-50	EMA-200	MACD	OBV	PPO	RSI
count	1258	1.26E+03	1258	1258	1258	1258	1258	1258	1.26E+03	1258	1258
mean	88.143	7.73E+07	0.000626	0.000616	0.000645	86.94	83.44	0.345	-6.95E+08	0.414	54.142
std	20.965	5.06E+07	0.016465	0.01646	0.003124	20.70	20.02	1.620	8.89E+08	1.939	13.449
min	50.929	1.15E+07	-0.131875	-0.131875	-0.007874	50.35	47.50	-4.131	-2.83E+09	-4.934	21.458
25%	70.195	4.19E+07	-0.007353	-0.007353	-0.00135	70.85	66.09	-0.627	-1.14E+09	-0.674	44.808
50%	89.183	6.32E+07	0.000473	0.000473	0.000772	84.51	75.00	0.359	-7.32E+08	0.408	53.270
75%	106.913	9.74E+07	0.009742	0.009716	0.002706	104.82	104.41	1.450	-1.67E+08	1.731	64.261
max	126.932	3.77E+08	0.085022	0.085022	0.008941	122.38	113.88	4.765	2.58E+09	5.382	88.056

Table 5. *Descriptive statistics of the AAPL dataset with generated features between 2012-2016*

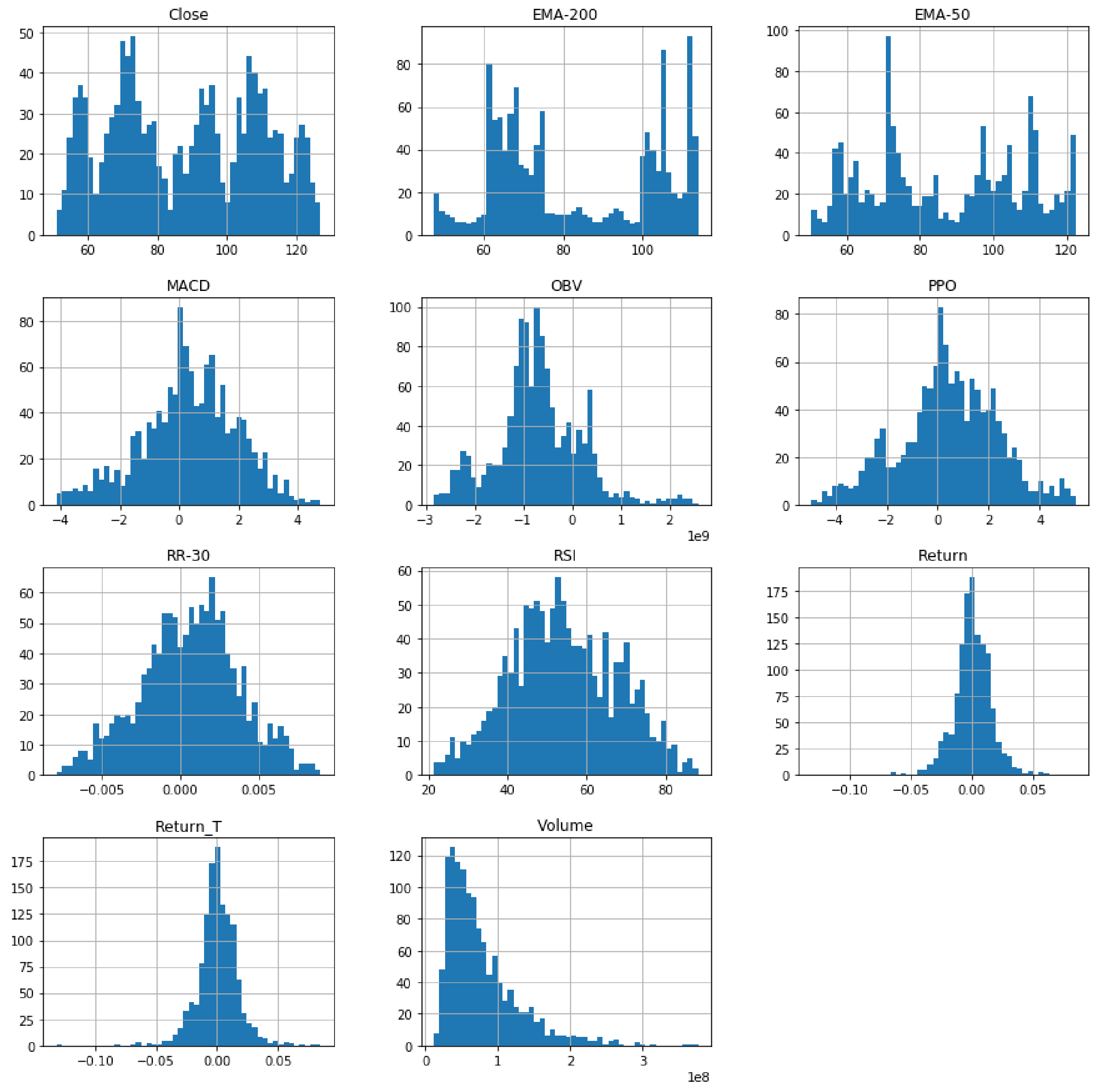

Figure 3. *Histogram of the AAPL dataset with generated features between 2012-2016*

Implementation

Classification Preparation: To transform stock movement into a binary result of price going up or down, daily return would be calculated to see if the result is positive or negative. As explained earlier in the feature generation section, the daily return feature needs to be shifted one day retrospectively to transform to target variable. A positive return, which is greater than or equal to zero, would be labeled as 1, whereas a negative return would be labeled as 0. Using label encoder in Sci-kit would get the job down, and a new column named "UpDown" would be created as a result.

Perform Classification: Models selected would be trained using default parameters defined in the Sci-kit. The function built in KNN, SVM, and AdaBoost models would be provided the capability to fine-tune the parameters later in the refinement step to see if a better performance could be achieved.

Time Series Cross Validation: Stock market data is financial time-series data, which would easily cause look-ahead bias if we simply divide training and testing set into an 80-20 split ratio or other cross validation techniques such as KFold or Shuffle Split. To prevent this issue from happening, roll-forward cross validation should be applied as it validates the data based on time sequence and is still able to utilize the entire training dataset for validation purpose. Time Series Split Cross Validation in Sci-kit provides an effective and straightforward way to implement roll-forward cross validation. For example, for a 10-fold Time Series Split Cross Validation, the training set is divided into 10 chunks and being trained and validated based on time series sequence. The first testing subset used to validate the first training subset would be later incorporated into the second training subset, so the cross validation makes the best use of the training dataset. The performance of each model is valued based on the average accuracies of the number of folds being validated. The default number of folds tested in the project is 10.

Model Name	Hyper Parameters Tested	Cross Validation Average Accuracy
Logistic Regression	C: 1.0, solver: liblinear	**0.5222**
KNeighbors Classifier	k: 10, weights: distance	0.5000
GaussianNB	-	0.4939
Support Vector Machines	c: 1.0, kernel: rbf	0.4969
Random Forest		0.4989
Adaptive Boosting	number of estimators: 50, learning rate: l.0	0.4767
Gradient Tree Boosting		0.5010
Quadratic Discriminant Analysis		0.5121

Table 6. *Cross Validation Performance*

Refinement

After implementing time series cross validation, the model with the best result is Logistic Regression, which has an average accuracy score of 0.5222 across 10 folds. In this section, two approaches of refinement are applied. The first approach is by tuning various hyper parameters of Logistic Regression and other three models—KNN, SVM, and AdaBoost in order to achieve see a better performance. The second approach is to test if the number of folds in cross validation do impact model performance. Logistic Regression, along with the second and the third best performing models, Gradient Tree Boosting and Quadratic Discriminant Analysis respectively, are tested with a various number of folds, the training set size of each fold is also reported.

KNN: After comparing the tested hyper parameters, using “distance” as the weight calculation method is universally better than using “uniform” under different. This means the distance weight calculation would assign weights proportional to the inverse of the distance from the query point. In another word, the nearer neighbors contribute more to the average than the more distant ones, which could be interpreted in time-series data as a more recent(closer) data point has more impact on model performance than a point that is a while ago(further) from the prediction date. However, after testing different number of neighbors with the same weight calculation hyper parameter, it does not look like adjusting the number of neighbors would have positive impact on average prediction accuracy score.

SVM: Between “poly” and “sigmoid” kernels, “poly” always has a better score than “sigmoid”. The penalty score, C, also does not have an impact on performance. C is a regularization parameter that controls the trade-off between achieving a low error on the training data and minimizing the norm of the weights. As C increases, so does the complexity of the hypothesis class. From the result, it is shown that either increasing or decreasing C is not producing a different hyperplane.

AdaBoost: Under hyper parameter of 50 estimators and learning rate of 0.001, the AdaBoost model has the highest performance of average score of 0.5 in 10 folds. It is also concluded that increasing or decreasing the number of estimators does not increase model performance.

Logistic Regression: Besides the default “linlinear” solver, other solvers such as “sag” and “lbfgs” have exactly the same performance as “liblinear”, with the exception of “newton-cg”. The way that “newton-cg” calculates may fail to converge in some iterations, and thus is not robust enough for final model. As for C, the outcome is similar to that of Support Vector Machines, which has no impact on accuracy performance.

To summarize, the refinement approach by fine-tuning the hyper parameters of the three models does not seem to be effective in finding a better performing model.

Model Name	Hyper Parameters Tested	Cross Validation Average Accuracy
Logistic Regression	C: 1.0, solver: liblinear/ sag/ lbfgs	0.5222
	C: 1.0, solver: newton-cg	0.5030
KNeighbors Classifier	k: 10, weights: distance	0.5000
	k: 10, weights: uniform	0.4828
	k: 5, weights: distance	0.5000
	k: 5, weights: uniform	0.4929
	k: 25, weights: distance	0.4969
	k: 25, weights: uniform	0.4939
Support Vector Machines	C: 1.0, kernel: poly	**0.5070**
	C: 1.0, kernel: sigmoid	0.4969
Adaptive Boosting	number of estimators: 50, learning rate: l.0	0.4767
	number of estimators: 50, learning rate: 0.1	0.4747
	number of estimators: 50, learning rate: 0.001	**0.5000**
	number of estimators: 25, learning rate: 0.001	0.4838

Table 7. *Parameter refinement for KNN, SVM, and AdaBoost models*

Number of folds in cross validation: The project picks the three models with top average accuracy scores, which are Logistic Regression, Gradient Tree Boosting, and Quadratic Discriminant Analysis. A 5-fold cross validation of Logistic Regression model with 221 sample points in each chunk has the best performance of 0.5260. A cross validation fold of this size is training on approximately 11-month of stocking transaction data. In the other two models, neither significant impacts to the performance nor trends are observed by varying the number of folds. In addition, the way boosting algorithm works attributes to an inconsistent outcome of Gradient Tree Boosting when multiple iteration being implemented.

Despite using 10 folds is the rule of the thumb in implementing cross validation, the prediction performance in the project is mostly dependent on the sampling range of each fold. Since the project is applying time-series cross validation, sampling sequence is time sensitive and is probably deterministic based on the price trend within the range.

Model Name			**Logistic Regression**	**Gradient Tree Boosting**	**Quadratic Discriminant Analysis**
Number of Folds	Size of each fold	Project Model Average Accuracy			
10	110	0.5222	0.5222	**0.5010**	0.5121
8	138	0.5227	0.5227	0.4761	0.5031
5	221	0.5260	**0.5260**	0.4920	0.5158
3	369	0.4850	0.4850	0.4796	**0.5189**
15	73	0.4960	0.4960	0.4784	0.5019

Table 8. *Parameter refinement for KNN, SVM, and AdaBoost models*

Results

Model Evaluation and Validation

The performance of each algorithm is evaluated based on performance metrics, which is the average accuracy on training dataset using Time Series Split Cross Validation. After refinement, Logistic Regression with 5-fold Cross Validation has the best accuracy — 0.5260. To validate if the model is robust enough, a sensitivity analysis conducted by manipulating the input data. The input data range was shifted by moving the original start date one month ahead and behind. From Table.9 we can tell the accuracy scores of two different data ranges do not deviate too much from the performance of original input data, and thus conclude that the result of the chosen model, Logistic Regression, is reliable and trusted.

Perturbation	Number of Folds	Data Range	Cross Validation Accuracy Score
Shift one month ahead	5 folds of CV	2011/11/01-2016/11/30	**0.5192**
Shift one month behind	5 folds of CV	2012/02/01-2017/01/31	**0.5135**

Table 9. *Sensitivity Analysis for Logistic Regression*

Justification

After implementing, fine-tuning, and evaluating the model with the best performance, the final step is to compare the model's performance against benchmark model. In this stage, not only the accuracies of the three models in Cross Validation are compared, but also their accuracies in predicting the stock movement using the unseen test dataset. The prediction accuracy score could be seen as whether the model is an effective and significant solution to the problem.

In the benchmark model, KNN has the best accuracy score in validation stage, while Logistic Regression clearly has predicted much better in testing data set and is the only model that has better than 50% in accuracy score.

In the case of the final project model—Logistic Regression, its accuracy score in validation is not better than that of Logistic Regression and KNN in benchmark model, but the prediction score outperforms the ones in benchmark model and other models in the project. Logistic Regression with 5-fold Cross Validation has the best performance in both training/validation and testing data subset, and its prediction performance is better than that of benchmark model. Under 5-fold Cross Validation, the proportion between each set is: training set (70%), validation set (20%), and testing set (10%). Thus, it would be considered as a better approach compared to benchmark model and other tested solutions.

The predictions in the test period, however, both Logistic Regression models make the exact same prediction, which is consecutively upward for 149 days. Because of this identical prediction results, the comparison between benchmark and final model cannot be statistically distinguished.

Model Name	Hyper Parameters	Benchmark Model		Final Model	
		Validation Average Accuracy	Prediction Accuracy Score	Validation Average Accuracy	Prediction Accuracy Score
Logistic Regression	-	0.4949 *(10 folds CV)*	**0.5436**	**0.5222** *(5 folds CV)*	**0.5436**
KNeighbors Classifier	k: **10**, weights: **distance**	**0.5232**	0.4966	0.5000	0.5167
GaussianNB	-	0.5010	0.4899	0.4939	0.5436

Table 10. *Performance comparison between the benchmark model and the project*

After choosing the algorithm with the best prediction, the project also offers a glimpse of the stock performance in the presumed scenario compared to actual stock by establishing back-testing portfolios. To simulate the back-testing results, various assumptions have to be made, including general trading assumptions and specific portfolio and strategy assumption. The details of various assumptions are explained in the appendix section. Three different trading strategies based on its timing on executing order (market close, market open, and intra-day, which is executed at both open and close) were tested and resulted in a rate of return around 18~20% in the six-month window between June 1st to December 31st, 2016. Compared to the actual return of simply holding the stock in the same period, the benchmark performance has a 18.87% (20.06% with dividends) of return. Since back-testing intrinsically has many assumptions, of which the greatest is the heavy transaction fees that come with frequent trading, the result could not be simply taken an apple-to-apple comparison. However, if we neglect these conditions and simply look at the final model being applied on the trading strategies, the project could conclude that the performance is on par with market performance despite the model's lacking in outstanding prediction accuracies.

Portfolio Strategy on Market Timing	Returns with Initial Capital of $100,000	Rate of Return
Close	$1,816.57	**18.16%**
Open	$1,843.81	**18.43%**
Intra-day	$2,012.11	**20.12%**

Table 11. *Back-testing Results of Portfolio Strategies*

Conclusion

After evaluating the model based on sensitivity analysis using slightly different sampling window and justifying the outcome based on back-testing, it is concluded that the model is able to predict the stock trend with some robustness. The final model performs slightly better than the benchmark model in the training and validation stage, but its accuracy is still in the range of 0.50 and 0.55. Since the project is framed as a binary classification task, this result is only marginally better than random guessing, or 50% of accuracy. Plausible explanations of model performance are provided in the following section.

Reflection

This section extends discussion from conclusion and provides some aspects of understanding why the model does not deliver significant improvement over benchmark model.

Model Complexity: Compared to the benchmark model, which uses the preceding 5-day stock prices as the features, the final model employs 8 features in total in order to address the model complexity. These 8 features are constructed based on the two main dimensions of stock information — historical price and transaction volume, whereas the benchmark model only takes price into consideration. Therefore, it is assumed that these added features should enhance the prediction performance significantly. It turns out that these features do not improve the performance much. It could be argued that despite these features are more complicated, they are still derivatives of price and volume. Thus, if price and volume alone can't predict the stock movement, these abstract features would not add much predictive robustness to the model either.

Amount of the Data: The model is built using five years of stock data preceding inquiry date. The time-series data contains approximately 1,260 raw entries per column, and would be trained on roughly 1,100 entries using a 9:1 train-test split ratio. The sample size is not really huge, thus make the case that Logistic Regression model achieves a better performance than other tested models. If the sample range is extended to a longer period, say 8 or 10 years, other models may be able to outperform Logistic Regression in terms of relative predictive accuracy, but the overall predictive performance across different models might not gain significant improvement due to the cyclical effect and the anomalies (e.g. Financial crisis that occurred in 2008) of the stock market. An alternative approach to predict stock movement is to use smaller window of sample points, e.g. 30 days or 90 days, since more recent data may have higher impact to forecasting accuracy, as suggested in using distance as weighting method in KNN.

Technical Analysis Approach: As aforementioned, the project takes the approach of technical analysis and the features are generated and selected accordingly. While fundamental analysis is mostly applied to finding intrinsic value of a stock, technical analysis is believed to be a more useful mean to analyze and summarize a longer trend, for example, price target or range in 30 days or 3 months; However, the project suggests that technical analysis does not provide much ground to predict the movement on a daily basis. This result is similar to what is proposed in Random Walk Theory that the past movement or trend of a stock price or market is not predictive of its future movement.

Improvement

As discussed above, changes and different approaches to model complexity and sampling data points may bring improve model performance. Algorithms that might be able to boost stock prediction performance might be deep learning approaches. Using neutral network architectures such as Convolutional Neutral Network to solve various problems has progressed significantly, especially in the fields of image classification and text recognition. Stock prediction might not be a particular area that has gained particular attention using these techniques, but experimenting different amounts of multiple layers and perceptron is certainly a potential path to achieve a higher prediction performance.

Free-Form Visualization
The section provides some visualization tools during model selection and validation. After using Principle Component Analysis (PCA) to determine which feature accounts for most of the variabilities, it is unsurprisingly that OBV has explained all of the variances due the fact that PCA is sensitive to the relative scaling of the original variables. In fact, the project has implemented an alternative feature selection by removing OBV from the variables in order to see if a better prediction could be achieved; however, the performance does not improve and has deteriorated marginally. Potential reasons might be that OBV itself is a hybrid indicator based on stock movement and transaction volume, and therefore retaining it in the model has more benefits than simply removing it due to the result of PCA.

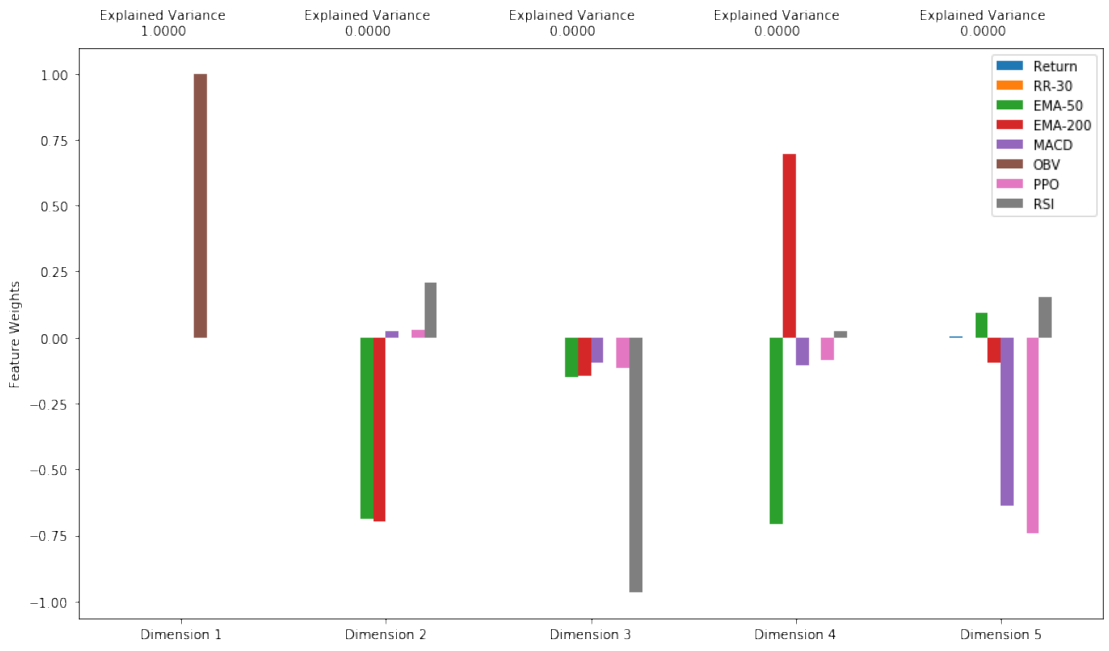

Figure 4. *Principle Component Analysis*

Another visualization is provided here as the importance scores for each feature is retrieved after the trees are constructed in Gradient Boosting. The score indicates the value of each feature in constructing the boosted decision tree. The more an attribute is used to make key decisions with decision trees, the higher its relative importance. In the case here, RSI has the highest relative importance, but is not significantly higher than the others. If the Gradient Boosting model is the best performing model in the project, an elaboration on feature importance should be helpful in refinement stage to construct a model with higher accuracy.

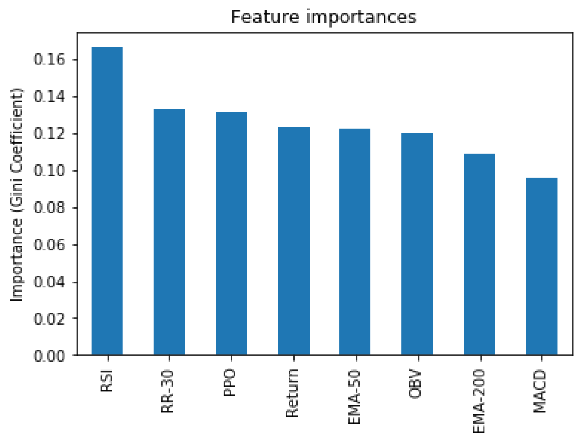

Figure 5. *Feature Importance - Gradient Boosting*

Appendix

The reference of general trading assumptions and various portfolio building strategies is from Quant Start[13].

General Trading Assumption:

1. Assuming no transaction fees, no interests charged through short-selling.

2. If the signal is 1, then buy 100 shares of underlying stock. If the signal is -1, then sell 100 shares short.

Market on Close Portfolio Strategy:

1. Use prediction for tomorrow to establish positions, the order is executed after the market closed, therefore it is a Close-to-Close price difference.

2. First day of test period is nullified as there is no close price difference from the previous period. No position left at the beginning, so the trading execution starts on the second day.

3. Every transaction is completed at the day's end. Bought shares would be sold at the closing price of the day, while the short-selling positons would be filled at the closing price.

Date	Open	Close	Signal	Positions	Close-to-Close Price Difference	Profit	Total	Returns
2016-06-01	96.771017	96.223736	1	NaN	0.000000	0.000000	100000.000000	NaN
2016-06-02	95.383269	95.500543	1	0.0	-0.723193	-72.319282	99927.680718	-0.000723
2016-06-03	95.568953	95.696001	1	0.0	0.195458	19.545752	99947.226470	0.000196

[13] https://www.quantstart.com/articles/Research-Backtesting-Environments-in-Python-with-pandas

Date	Open	Close	Signal	Positions	Close-to-Close Price Difference	Profit	Total	Returns
2016-06-06	95.764411	96.389875	1	0.0	0.693874	69.387419	100016.613889	0.000694
2016-06-07	96.995793	96.780790	1	0.0	0.390915	39.091504	100055.705393	0.000391

Table A-1. *Market on Close Portfolio Snapshot*

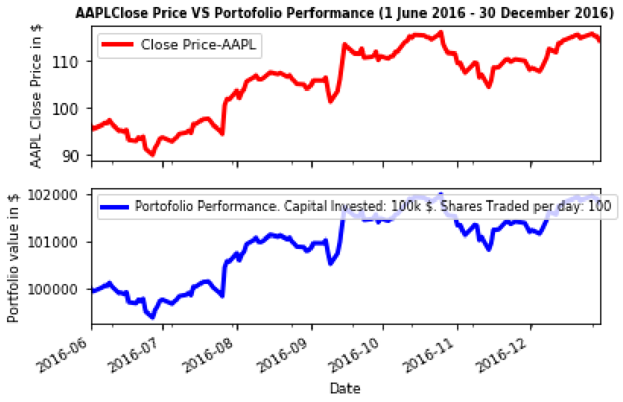

Figure A-1. *Market on Close Portfolio compared with Actual Performance*

Market on Open Portfolio Strategy:

1. Use prediction for tomorrow to establish positions, the order is executed at the market open, therefore it is an Open-to-Open price difference.

2. First day of test period is nullified as there is no open price difference from the previous period. No position left at the beginning, so the trading execution starts on the second day.

3. Every transaction is completed at the day's beginning. Bought shares would be sold at the opening price of the day, while the short-selling positons would be filled at the opening price.

Date	Open	Close	Signal	Positions	Open-to-Open Price Difference	Profit	Total	Returns
2016-06-01	96.771017	96.223736	1	NaN	0.000000	0.000000	100000.000000	NaN
2016-06-02	95.383269	95.500543	1	0.0	-1.387748	-138.774837	99861.225163	-0.001388
2016-06-03	95.568953	95.696001	1	0.0	0.185685	18.568464	99879.793627	0.000186

Date	Open	Close	Signal	Positions	Open-to-Open Price Difference	Profit	Total	Returns
2016-06-06	95.764411	96.389875	1	0.0	0.195458	19.545752	99899.339378	0.000196
2016-06-07	96.995793	96.780790	1	0.0	1.231382	123.138236	100022.477615	0.001233

Table A-2. *Market on Open Portfolio Snapshot*

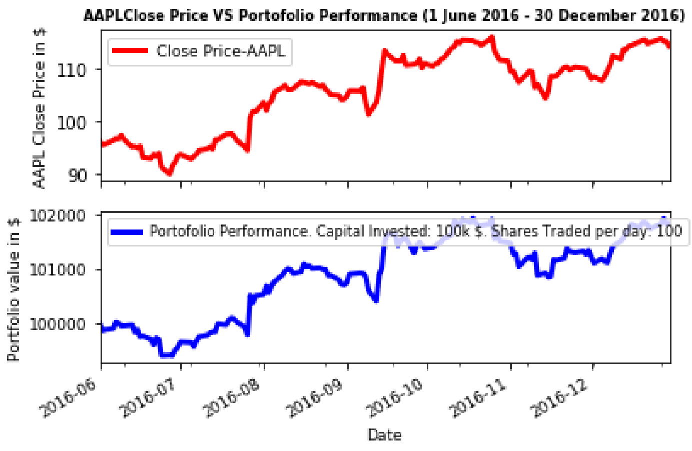

Figure A-2. *Market on Open Portfolio compared with Actual Performance*

Market Intra-Day Portfolio Strategy:

1. Use prediction for tomorrow to establish positions, the order is executed at the market open and after the market closed, therefore it is a Close-to-Open price difference.

2. Every transaction is completed at the day's end. Bought shares would be sold at the opening price of the day, while the short-selling positons would be filled at the closing price.

Date	Open	Close	Signal	Positions	Close-to-Open Price Difference	Profit	Total	Returns
2016-06-01	96.771017	96.223736	1	NaN	0.000000	0.000000	100000.000000	NaN
2016-06-02	95.383269	95.500543	1	0.0	0.117275	11.727451	100011.727451	0.000117
2016-06-03	95.568953	95.696001	1	0.0	0.127047	12.704739	100024.432190	0.000127
2016-06-06	95.764411	96.389875	1	0.0	0.625464	62.546406	100086.978595	0.000625
2016-06-07	96.995793	96.780790	1	0.0	-0.215003	-21.500327	100065.478268	-0.000215

Table A-3. *Market Intra-Day Portfolio Snapshot*

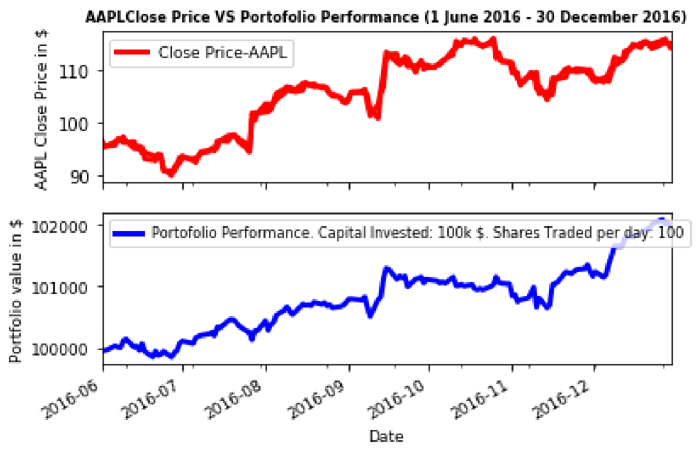

Figure A-3. *Market Intra-Day Portfolio compared with Actual Performance*

Reference

Pochetti, Francesco (2014). Stock Market Prediction in Python Intro. (http://francescopochetti.com/stock-market-prediction-part-introduction/)