Capstone Project: Stock Price Prediction

1. Definition

1.1 Project Overview

Investment firms, hedge funds and even individuals have been using financial models to better understand market behavior and make profitable investments and trades. A wealth of information is available in the form of historical stock prices and company performance data, suitable for machine learning algorithms to process.

The price variation of stock market is a very dynamic system. Predicting stock price and its movement has been considered as one of the most challenging applications of time series prediction. Three common analytical approaches are fundamental analysis, technical analysis, and quantitative analysis.

Fundamental analysis relies on the statistics of the macroeconomics data such as interest rates, money supply, inflationary rates, and foreign exchange rates as well as the basic financial status of the company. After taking all these factors into account, a decision of selling or buying a stock will be made.

Technical analysis is based on the historical financial time series data to explore different patterns and indicators, such as trends, abrupt changes, and volatility patterns. However, because the stock price data is nonstationary and even chaotic, technical indicators, generated by technical analysis of historical data, sometimes are not able to reveal the real variation of the stock market.

Quantitative analysis applies scientific methods, such as statistics and machine learning, while takes advantage of all kinds of data including datasets used by both fundamental and technical analysis, to make better and complex evaluation of future stock market, and help investor to make better decision.

In this project, we will build stock price predictors that takes daily trading data over a certain date range as input, and outputs projected estimates for given query dates with the help of different machine learning methods.

1.2 Problem Statement

In this project, we will use various machine learning methods to predict future stock price. More specifically, we will focus on the S&P 500 index, which is widely regarded as the best single gauge of large-cap U.S. equities. There is over USD 7.8 trillion benchmarked to the index, with index assets comprising approximately USD 2.2 trillion of this total. The index includes 500 leading companies and captures approximately 80% coverage of available market capitalization.

Because stock price is more like a random walk, models with original price information as independent variables will not give us much accurate prediction. Therefore, following previous studies, we use various technical indicators generated from historical stock price information as our model input. Because of stock splits and dividend, not all stock price changes are due to the market variation. Therefore, adjusted price (no influence from stock splits and dividends) is taken as model output.

The basic procedure is as follow:

1) Download daily historical stock data (Open, High, Low, Close, Adjusted Close, Volume)
2) Generate various technical indicators
3) Define prediction window (e.g., use previous 10 days' data to predict next day's price)
4) Split data into training and testing
5) Build different models and evaluate prediction performance based on defined metrics
6) Choose the best model based on defined metrics

Further in this report, I calculate 26 technical indicators, such as Relative Strength Index (RSI) and On Balance Volume (OBV), which are commonly used in technical analysis.

1.3 Metrics

The model prediction performance is defined as the mean absolute error (MAE) of testing dataset. That is $-\sum_{1^{5}=_{167}|_{1}-_{1}|}$

MAE measures the mean absolute difference between the prediction and real stock price, which is a typical choice for regression problem. It can directly tell the price difference between prediction and real data, and has been using for regression model performance for a long time. Therefore, MAE is taken as model performance metrics. The MAE metrics are also used for evaluating the training process of our models.

2. Analysis

2.1 Data Exploration

The S&P 500 daily historical price data is downloaded from Yahoo Finance using Zipline (https://github.com/quantopian/zipline). We focus on the period from 1/1/2000 to 4/30/2016. The data includes different stock information, such as open, high, low, close, price, as well as volume. The explanation of each variable is as follow:

- open: the stock price when the market starts in a day
- high: the highest price in a trading day
- low: the lowest price in a trading day
- close: the stock price when the market closes in a day
- price: adjusted close price considering stock splits and dividends
- volume: the number of shares traded in a trading day

Here, the "price" is the adjusted closing price, which is a stock's closing price on any given day of trading that has been amended to include any distributions and corporate actions that occurred at any time prior to the next day's open. It is often used when doing historical price analysis.

Because S&P 500 index doesn't have splits and dividends like stocks, the "price" and "close" are actually the same for the S&P 500 index. For stocks, they may be different. The basic statistics of the dataset is as follow:

Table 1. Basic Statistics of S&P500 Index (2001-2016)

	open	high	low	close	volume	price
count	4107	4107	4107	4107	4107	4107
mean	1344.02	1352.55	1334.78	1344.16	2.99E+09	1344.16
std	332.03	332.04	331.94	332.13	1.59E+09	332.13
min	679.28	695.27	666.79	676.53	3.56E+08	676.53
25%	1121.96	1127.79	1114.83	1121.74	1.50E+09	1121.74
50%	1280.85	1287.61	1272.66	1280.70	2.96E+09	1280.70
75%	1471.35	1480.14	1460.48	1471.53	3.99E+09	1471.53
max	2130.36	2134.72	2126.06	2130.82	1.15E+10	2130.82

2.2 Exploratory Visualization

To explore the dataset, all variables (open, high, low, close, price, and volume) are plotted as follow.

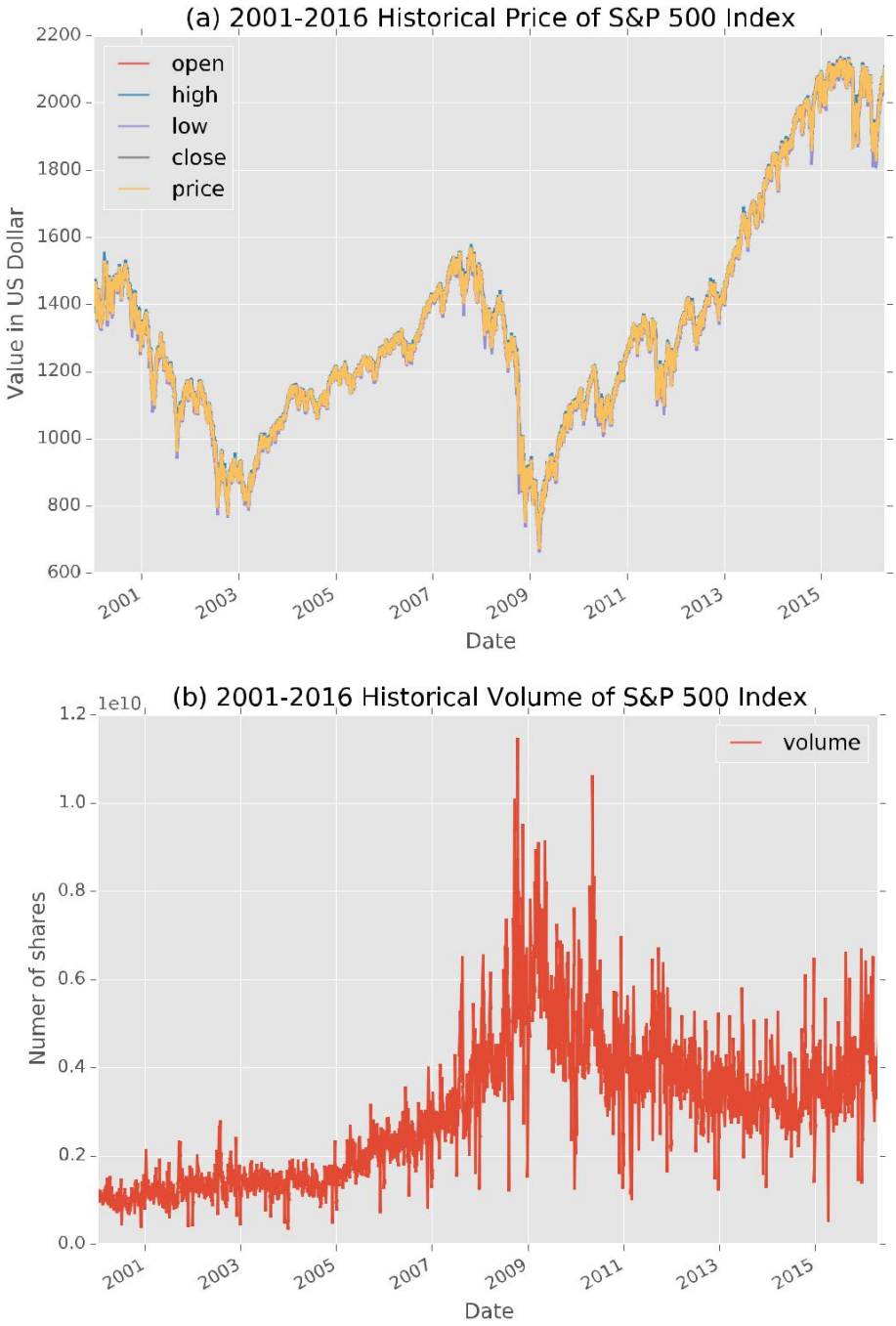

Figure 1. S&P500 Index historical price (a) and volume (b) from 2001 to 2016.

From the above figures, we can see the stock price has a lot of variation over the fifteen years (2001-2016). Generally, two low peaks exist, one in 2003 and other one in 2008, which are corresponding to the two financial crises. Because the "open", "high", "low", and "close" do not change a lot over a trading day, they tend to overlap with each other over such a long time period. Because we choose the S&P 500 index as our example, the adjusted close price ("price") is equal to the close price ("close"). Over the study period, the standard deviation of the daily "price" variation ($price_{n+1} - price_n$) is 15.18.

From the "volume" plot, we see more variability compared to that of the price over the last fifteen years. The largest variation over the study period occurred in 2008, which again corresponds to the 2008 financial crisis.

2.3 Algorithms and Techniques

Based on previous historical stock price, the future price will be predicted. Apparently, this is a regression problem. For the inputs, various technical indicators will be constructed based on the historical price information. This will be done in the Data Processing section. After the technical indicators generated, different models will be built with various technical indicators as inputs. Because this is a regression problem, the models chosen are as follows:

- **Linear Regression:**

This is the basic model for regression problem. The output ("price" variable) will be taken as a linear combination of all technical indicators. We will take the linear regression model results as the benchmark.

- **Lasso Regression:**

Lasso is the linear regression with L1 regularization. It is a shrinkage and selection method for linear regression. It minimizes the usual sum of squared errors, with a bound on the sum of the absolute values of the coefficients. Because the input variables themselves may be correlated or overlapped, and some input variables may not be related the output variable, Lasso can help us get rid of some variables.

- **Support Vector Machine (SVM):**

SVM is a set of supervised learning methods used for classification, regression and outliers detection. It is effective in high dimensional spaces. It uses a subset of training points in the

decision function, which is more efficient. For our cases, the various technical indicators can be better analyzed using the SVM in a high dimensional space.

- **Gradient Boosting Regression:**

It produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

- **Artificial Neural Network (ANN):**

ANN is a family of models inspired by biological neural networks which are used to estimate or approximate functions that can depend on a large number of inputs and are generally unknown. Artificial neural networks are generally presented as systems of interconnected "neurons" which exchange messages between each other. The connections have numeric weights that can be tuned based on experience, making neural nets adaptive to inputs and capable of learning.

Different models will be tested in the following sections. The performance of the models will be analyzed using the testing dataset.

2.4 Benchmark

Over the study period, the standard deviation of the S&P500 index daily price variation is 15.18. The long-term S&P500 Volatility Index (VIX) from Yahoo Finance shows that the volatility (standard deviation) of the daily S&P500 index ranges from 10 to 60. Therefore, we set the MAE benchmark of all the models to be 15, which is equal to the standard deviation of the S&P500 index daily price variation during the study period. That is, if the MAE of model prediction on the testing dataset is less than 15, we consider the model can beat the benchmark. Otherwise, the model cannot beat the benchmark. This is a reasonable choice for the benchmark given the historical variation of the VIX from Yahoo Finance.

Using the previous defined metrics (MAE), we can demonstrate the performance of different models and compare them with the benchmark in later sessions.

3. Methodology

3.1 Data Preprocessing

As we mentioned before, the direct use of historical price information will not give us accurate prediction due to the inherent property of stock price variation. Therefore, we need to transform the original historical price information into variables that we can use to do the prediction.

A natural choice is using the technical indicators. We calculate 26 technical indicators, such as Relative Strength Index (RSI) and On Balance Volume (OBV), which are commonly used in technical analysis. The RSI calculates a ratio of the recent upward price movements to the absolute price movement, and it ranges from 0 to 100. The RSI is interpreted as an overbought/oversold indicator when the value is over 70/below 30. The OBV is a cumulative total of the up and down volume. The OBV measures cumulative buying/selling pressure by adding the volume on up days and subtracting volume on losing days. More technical indicators and corresponding details can be found in appendix.

After generating technical indicators, all the indicators are transformed into certain range between -1 and 1, to ensure their influence to the output is generally in a similar level.

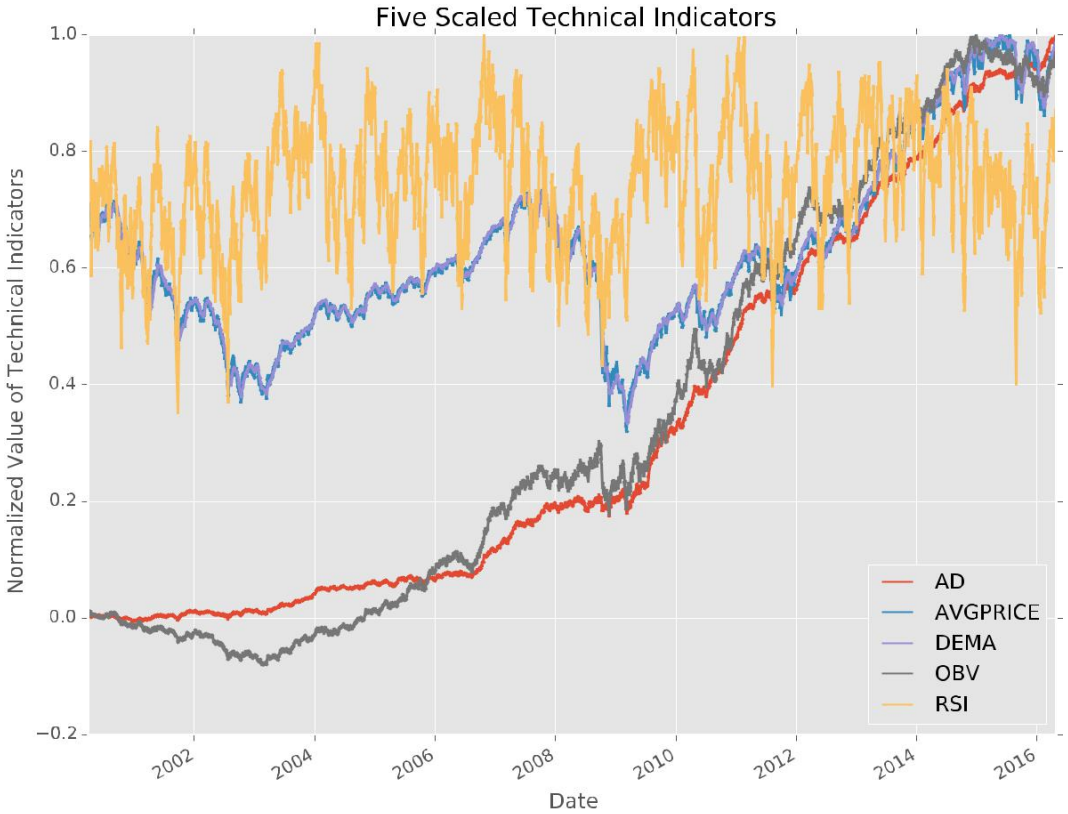

Figure 2. Scaled Technical Indicators (AD, AVGPRICE, DEMA, OBV, and RSI).

For better visualization, we only plot five indicators. As we can see from the above plot, all the five indicators give us pretty different variation. Therefore, they should contain different information, which could be useful for predicting future prices.

Preprocessing in this project is handled from sklearn which allows better maximum absolute scaler calculations to be operated without the use of additional code which would make the project less efficient.

```
“max_abs_scaler = preprocessing.MaxAbsScaler()
ti_nonan = ti.iloc[2*time_period:, :]
ti_scaled = ti_nonan.apply(max_abs_scaler.fit_transform)”
```

When plotting data on the graphs used above, the following code is used which sets forth 5 plot indicators in the project:

```
plt_data = ti_scaled[['AD', 'AVGPRICE', 'DEMA', 'OBV', 'RSI']]
ax = plt_data.plot(figsize=(10, 8), title='Five Scaled Technical Indicators'
        )
```

3.2 Implementation

In this part, we will implement the Linear Regression model. Because the way of calculating technical indicators, some non-number values are generated at the beginning of our data records. Therefore, we need to exclude those records before splitting the dataset into training and testing. The way we handle this problem is to find the first time record that all technical indicators have valid values and exclude the records before this one. After that, 70% of the remaining data is used for training the model, and the rest is used to test the performance of the model using the mean absolute error (MAE) as the metrics.

Here, we use a simple Linear Regression model = $\Sigma_{167\ 1\ 1}$ + to predict the “price” of S&P500 in the next day, where $_1$ is one of the technical indicators, is constant, and is the next day “price” of S&P500. The training data is used to fix the coefficients $_1$ and by trying to minimize the difference between model prediction and real data. After fixing the coefficients, the model is generated. The one issue in which became a problem upon implementation of the linear Regression model included some data interpretation issues when working with the coefficients highlighted in this report. These issues were easily fixed after further consideration was taken when implementing the algorithm. No problems were experienced other than the aforementioned issue in the implementation of the Linear Regression model– excluding outdated dependencies which caused conflicts

3.3 Refinement

In this session, we will implement more complicated models, such as Lasso Regression and SVM Regression. The purpose is to see if these complex models can give us better prediction.

Unlike the Linear Regression model, many hyperparameters in the complex models needs to be determined, such as the penalty parameter C in the SVM. To effectively choose the hyperparameters of these complex models, cross-validation method is implemented. This technique is widely used to perform feature and model selection once the data collection and cleaning phases have been carried out. The idea behind cross-validation is that in order to select the best predictors and algorithm it is mandatory to measure the accuracy of our process on a set of data which is different from the one used to train the model. The hyperparameters in which would be used to further refine this project include the use of Bayesian optimization and the use of Radial Basis Function (RBF) in the scikit-learn library in order to better refine and interpret the data being used in this project. Although further implementation of the hyperparamaters may be difficult, they would help optimize the project and lastly would be feasible.

The common used cross-validation method is picking samples out of the available data randomly to train, while leave the rest for validation. However, the stock price data is a time series dataset, which means we need to keep their sequential property and cannot pick them randomly.

Therefore, we need to choose a different cross-validation method (Pochetti, 2014). The basic procedure is as follow:

- Split data into train and test.
- Split train dataset into consecutive time folds (e.g. 5).
- Train on fold 1, and test on fold 2.
- Train on fold 1 and 2, and test on fold 3.
- Train on fold 1, 2, and 3, and test on fold 4.
- Train on fold 1, 2, 3, and 4, and test on fold 5.
- Compute the average of the accuracy of the above test folds as the model performance. Based on this cross-validation method, we can try different parameters for the complex models,

such as Lasso Regression and SVM Regression. The model setup procedure is similar as the Linear Regression model. The input variables are the scaled technical indicators, and the output variable is the next day's "price" of S&P500 index.

4. Results

4.1 Model Evaluation and Validation

Before applying the model to the testing dataset, we need to make sure the model is valid. To validate the robustness of the Linear Regression model's solution. We first plot the residuals of the training data as follows:

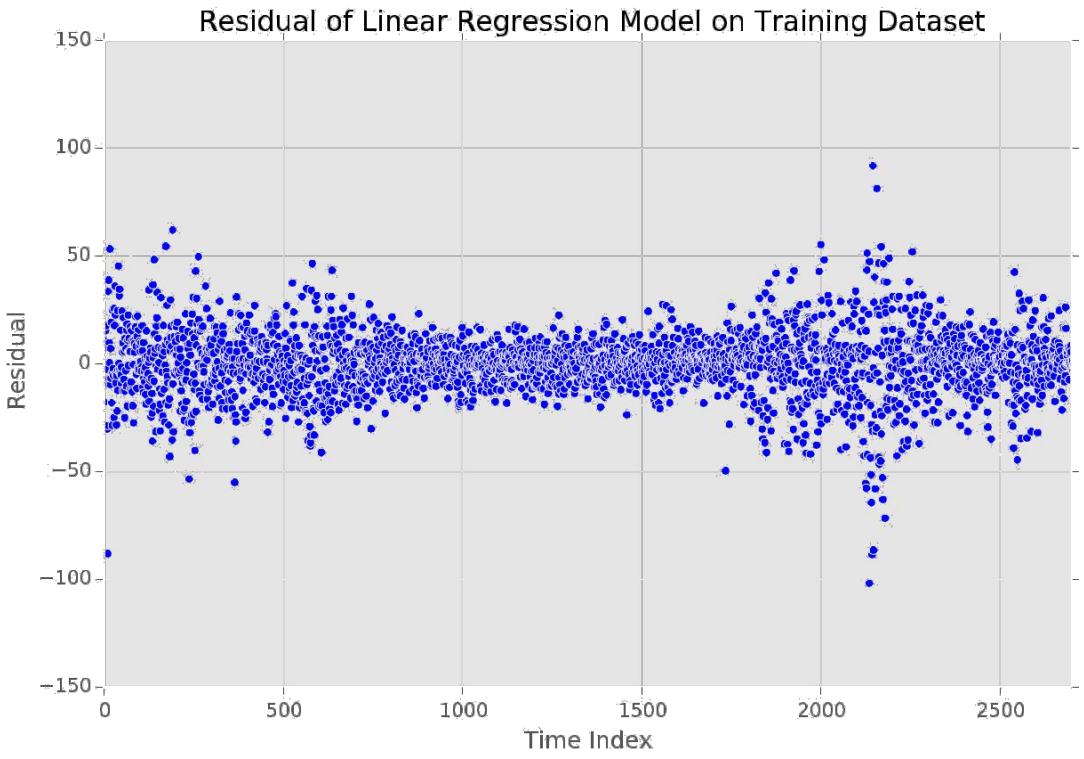

Figure 3. Residual of Linear Regression model for the training dataset.

The residuals generally are randomly distributed without certain symmetric pattern, indicating that the model is proper for this problem. We did a F-test for the overall significance of the regression model. Specifically, we test the null hypothesis that all of the regression coefficients are equal to zero. The p-value of the F-test is about 0.0001, which means we can reject the null hypothesis and implies our regression model does have some validity in fitting the data.

After validating our model, we can use it on the testing dataset, and generate the performance metrics-MAE. The MAE of the Linear Regression model on testing dataset is 11.64, which is reasonable and smaller than the benchmark value 15. Therefore, the Linear Regression model can beat the benchmark. In the following sections, we will use other more complicated models to see if they can beat the benchmark and produce better prediction than the Linear Regression.

For the Lasso Regression, we use the cross-validation method mentioned in the previous session to choose proper model settings. Here, the alpha parameter of Lass Regression is tested with different values (0.1, 1.0, 5.0, 10.0, 50.0, and 100.0) using the cross-validation method. Alpha is the constant that multiplies the L1 term, which is important for determining the coefficients of Lasso model. The training and validation MAE in the cross-validation dataset (70% of the whole

dataset) is calculated. The value of alpha (0.1) corresponding the least validation MAE is chosen as the best Lasso model. The MAE of the Lasso Regression model on testing dataset is 13.70, which is less than the benchmark, but greater than that of the Linear Regression model.

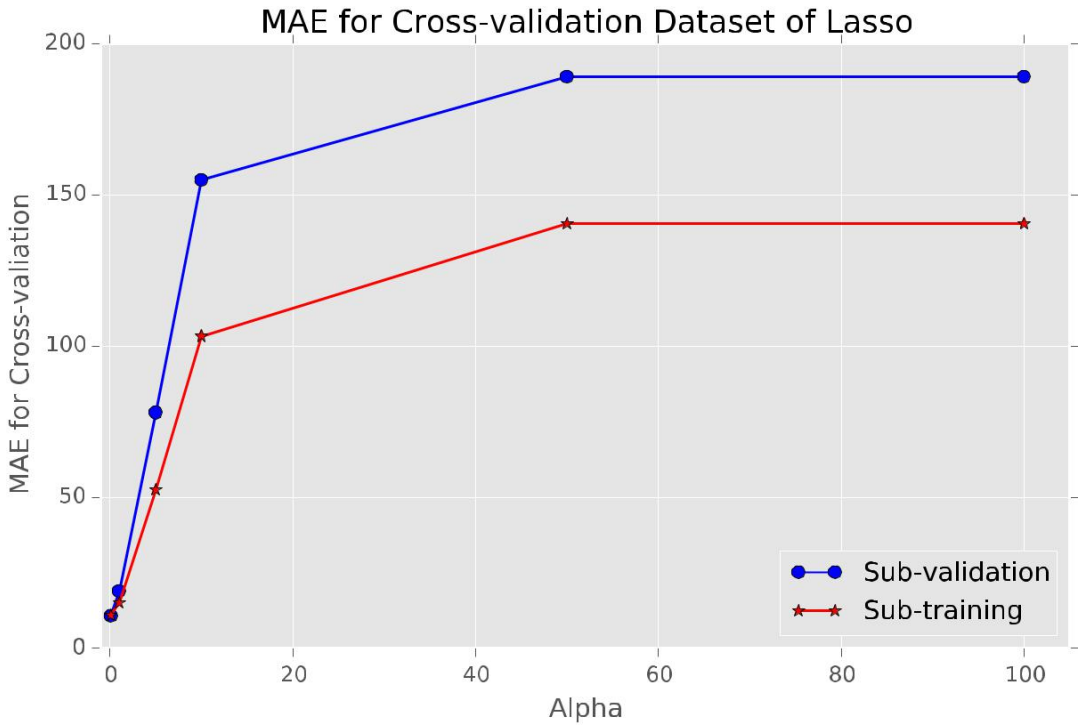

Figure 4. Mean Absolute Error (MAE) for the Lasso Regression on cross-validation dataset with different values of parameter alpha.

For the SVM Regression, similar procedure is implemented. The C parameter of SVM Regression is tested with different values (0.01, 0.1, 1.0, 5.0, 10.0, 50.0, and 100.0) using the cross-validation method. C is the penalty parameter for the error term in SVM. The value of C (100) corresponding the least validation MAE is chosen as the best SVM model. The MAE of the SVM Regression model on testing dataset is 51.41, which is greater than the benchmark. Therefore, the SVM model cannot beat the benchmark.

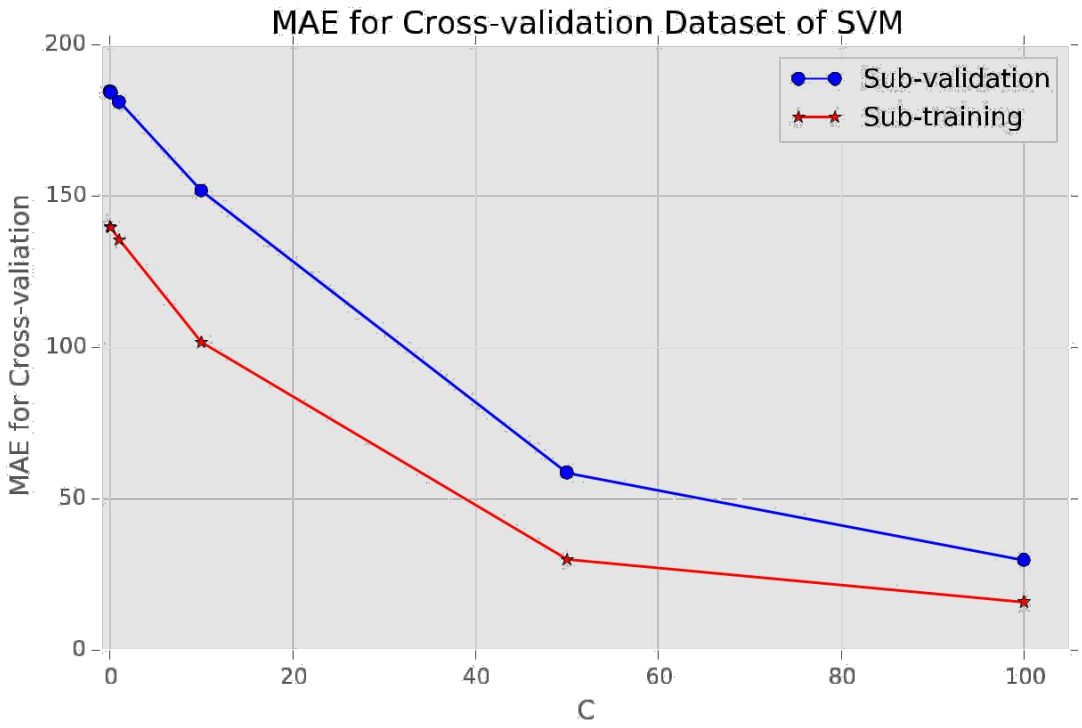

Figure 5. Mean Absolute Error (MAE) for the SVM Regression on cross-validation dataset with different values of parameter C.

For the one-day ahead prediction, the MAE of different models are listed in Table 2. For training dataset, the three models (Linear, Lasso, and SVM Regressions) give similar MAE (~10) with slight difference. But for the testing dataset, the difference is significant. The Linear Regression generates the least MAE, while the SVM gives the largest, and the MAE of Lasso is larger than that of the Linear Regression but less than the benchmark (15).

Table 2. MAE (in US dollars) for different models

	MAE for training	**MAE for testing**
Linear Regression	10.20	11.64
Lasso Regression	10.64	13.70
SVM Regression	10.79	51.41

4.2 Justification

The above analysis shows that complex models such as SVM Regression may need further fine tuning or not suitable for this problem due to the large MAE on testing dataset. While both Linear and Lasso Regression beat the benchmark, Linear Regression model gives us the best and reasonable prediction according to the MAE of the testing dataset.

5. Conclusion

5.1 Free-Form Visualization

The predicted price with one-day sliding window as well as the true price and averaged prediction price are plotted as follow. Generally, the prediction of the three models agrees well with the real price. Therefore, the models we built have certain predictability.

Generally, from the testing dataset, we can see the Linear and Lasso Regression can give better performance than the SVM (Figure 6). To better see the difference between the Linear and Lasso Regression, subsampled comparison is plotted for a shorter time period (Figure 7), which shows that the Linear Regression model is generally better than that of the Lasso.

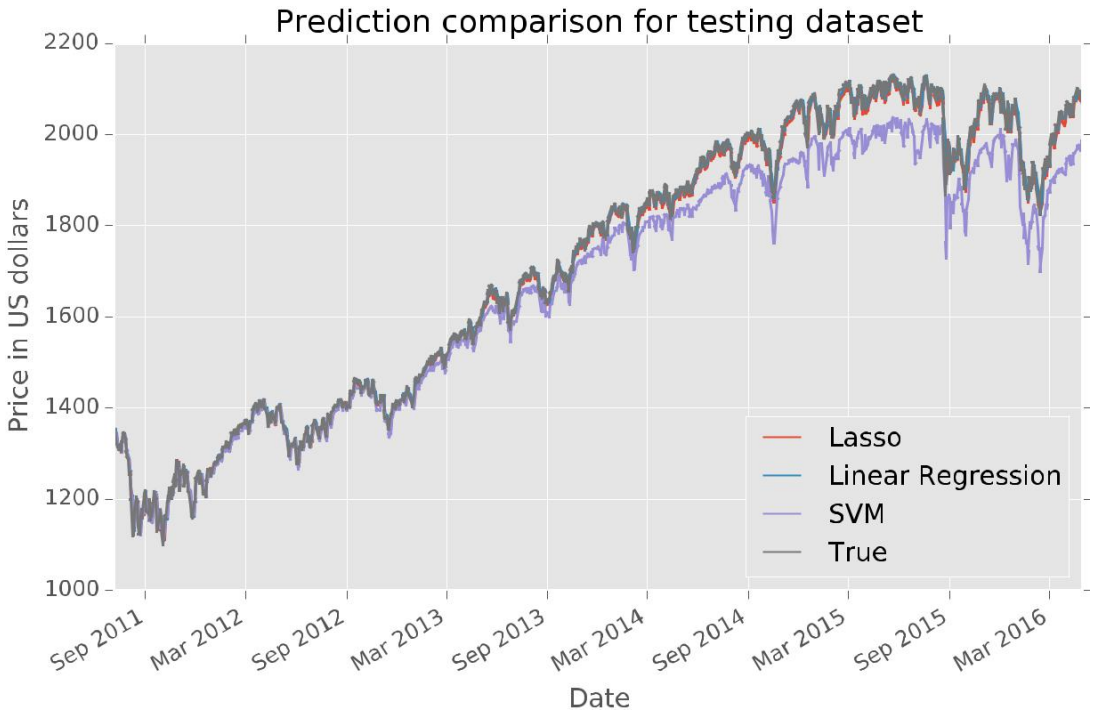

Figure 6. Comparison of prediction and true price for all three models.

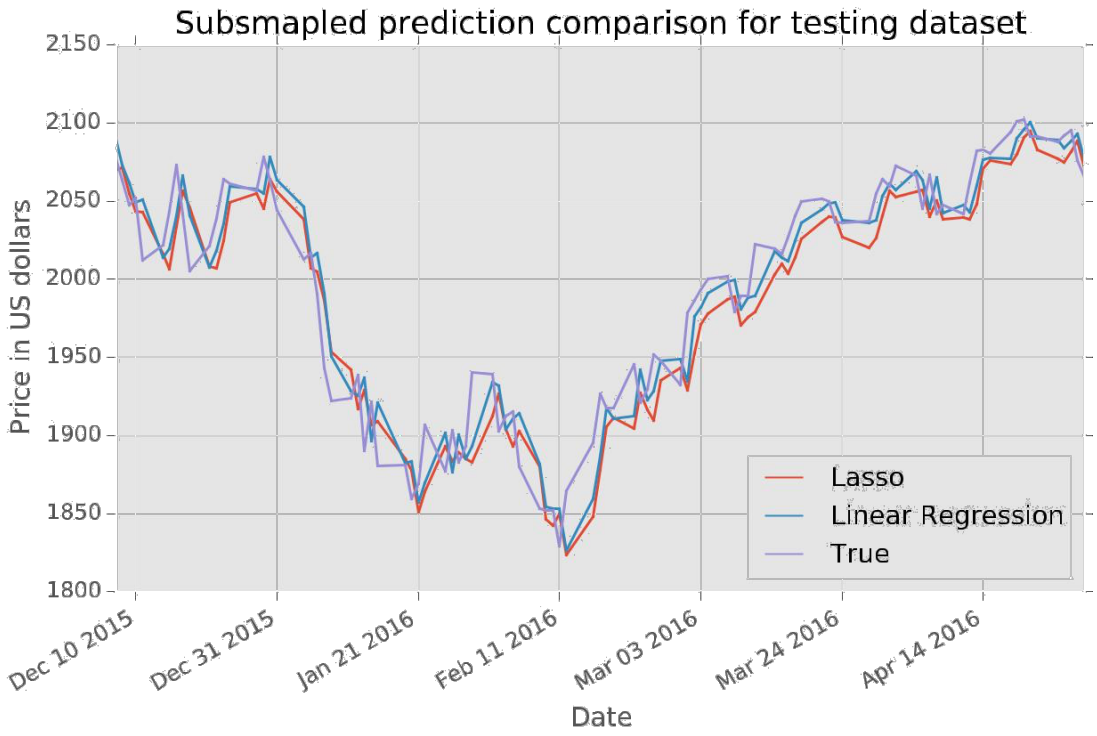

Figure 7. Comparison of prediction and true price for all three models (subsampled).

5.2 Reflection

In this project, we try to take various technical indicators derived from historical daily S&P500 index price (2000-2016) over a certain date range as input, and generate projected estimates for the index price for the next trading day with the help of different machine learning models. The historical S&P500 index data first is downloaded from Yahoo Finance, and then used to generate common technical indicators of stock market analysis. With these historical technical indicators as input, we try to predict the next day's S&P500 index price with different models. Three models (Linear, Lasso, and Support Vector Machine Regressions) are implemented and compared using the mean absolute error (MAE) as performance metrics. We set the benchmark of MAE to be 15 according to the variation of the daily S&P500 index and corresponding historical range of volatility index (VIX). Through model testing, we find that the Linear Regression model gives us the best performance with MAE of 11.64 on testing dataset (20% of whole dataset), while Lasso Regression generates MAE of 13.70, and SVM Regression produces 51.41 one testing dataset.

Stock price prediction is very difficult problems due to its complexity. From the performance of different models, I find that complex models do not always beat simple ones. One reason is that

complex models typically have more parameters to tune. Sometimes, it is hard to find proper parameters suitable for certain problem. Complex models may perform well on training dataset, but they are easy to overfit the data, which means their performance may be worse on testing dataset. Therefore, overfitting needs more attention when using complex models. In a word, start with the simplest model when doing a project, and then gradually increase complexity of the model and compare the performance with the previous one to check if the model complexity can benefit the performance.

5.3 Improvement

Because we use S&P500 as our example, luckily, we do not need to fill the historical dataset. For other stock or index, we may need to fill the nans with the forward fill first and backward fill if necessary to make sure we don't spy the future data information.

From the analysis and performance, Lasso and Linear Regression give us more reasonable results, while Linear Regression generates the least MAE. For the SVM, more parameters can be fine tuned, which may give us better performance. One reason could be that the kernel function of the SVM needs further adjustment.

We only considered one-day ahead prediction in this project. Longer prediction window can be applied in the future. For longer prediction window, the complex models may produce better results than that of the simple models.

In this project, only the historical price information (open, high, low, close, adjusted close, and volume) is used as input. In reality, other fundamental information, such as interest rate, inflationary rates, and foreign exchange rates as well as the basic financial status of the company, can also be included in the model. Because this information can also affect the stock market and price, models with this information may give us better prediction accuracy.

Appendices

Setup

- NumPy (http://www.numpy.org)
- Pandas (http://pandas.pydata.org)
- Zipline (https://github.com/quantopian/zipline)

- TA-Lib (http://ta-lib.org)
- Python wrapper for TA-Lib (http://mrjbq7.github.io/ta-lib/)

Technical Indicators

- AD - Chaikin A/D Line: The Accumulation/Distribution Line is similar to the On Balance Volume (OBV), which sums the volume times +1/-1 based on whether the close is higher than the previous close. The Accumulation/Distribution indicator, however multiplies the volume by the close location value (CLV). The CLV is based on the movement of the issue within a single bar and can be +1, -1 or zero. Details can be found at http://www.fmlabs.com/reference/default.htm?url=AccumDist.htm.
- ADX - Average Directional Movement Index: The ADX is a Welles Wilder style moving average of the Directional Movement Index (DX). The values range from 0 to 100, but rarely get above 60. To interpret the ADX, consider a high number to be a strong trend, and a low number, a weak trend.
- AROONOSC - Aroon Oscillator: The Aroon Oscillator is calculated by subtracting the Aroon Down from the Aroon Up. The resultant number will oscillate between 100 and -100. The Aroon Oscillator will be high when the Aroon Up is high and the Aroon Down is low, indicating a strong upward trend. The Aroon Oscillator will be low when the Aroon Down is high and the Aroon Up is low, indicating a strong downward trend. When the Up and Down are approximately equal, the Aroon Oscillator will hover around zero, indicating a weak trend or consolidation. See the Aroon indicator for more information.
- ATR - Average True Range: The ATR is a Welles Wilder style moving average of the True Range. The ATR is a measure of volatility. High ATR values indicate high volatility, and low values indicate low volatility, often seen when the price is flat (http://www.fmlabs.com/reference/default.htm?url=ATR.htm).
- AVGPRICE - Average Price: The Average Price is the average of the open + high + low + close of a bar. It can be used to smooth an indicator that normally takes just the closing price as input.
- CMO - Chande Momentum Oscillator: The Chande Momentum Oscillator is a modified RSI. Where the RSI divides the upward movement by the net movement (up / (up + down)), the CMO divides the total movement by the net movement ((up - down) / (up + down)). Please see http://www.fmlabs.com/reference/default.htm?url=CMO.htm.

- DEMA - Double Exponential Moving Average: The DEMA is a smoothing indicator with less lag than a straight exponential moving average.
- DX - Directional Movement Index: The DX is usually smoothed with a moving average (i.e. the ADX). The values range from 0 to 100, but rarely get above 60. To interpret the DX, consider a high number to be a strong trend, and a low number, a weak trend.
- EMA - Exponential Moving Average: The Exponential Moving Average is a staple of technical analysis and is used in countless technical indicators. In a Simple Moving Average, each value in the time period carries equal weight, and values outside of the time period are not included in the average. However, the Exponential Moving Average is a cumulative calculation, including all data. Past values have a diminishing contribution to the average, while more recent values have a greater contribution. This method allows the moving average to be more responsive to changes in the data.
- MACDEXT - MACD with controllable MA type: The Moving Average Convergence Divergence (MACD) is the difference between two Exponential Moving Averages. The MACD signals trend changes and indicates the start of new trend direction. High values indicate overbought conditions; low values indicate oversold conditions. Divergence with the price indicates an end to the current trend, especially if the MACD is at extreme high or low values. When the MACD line crosses above the signal line a buy signal is generated. When the MACD crosses below the signal line a sell signal is generated. To confirm the signal, the MACD should be above zero for a buy, and below zero for a sell. See http://www.fmlabs.com/reference/default.htm?url=MACD.htm.
- MEDPRICE - Median Price: The Median Price is the average of the high + low of a bar. It can be used to smooth an indicator that normally takes just the closing price as input.
- MFI - Money Flow Index: The Money Flow Index calculates the ratio of money flowing into and out of a security. To interpret the Money Flow Index, look for divergence with price to signal reversals. Money Flow Index values range from 0 to 100. Values above 80/below 20 indicate market tops/bottoms. Details can be found http://www.fmlabs.com/reference/default.htm?url=MoneyFlowIndex.htm.
- MINUS_DI - Minus Directional Indicator: The -DI is the percentage of the true range that is down. A buy signal is generated when the +DI crosses up over the -DI. A sell signal is generated when the -DI crosses up over the +DI. You should wait to enter a trade until the

extreme point is reached. That is, you should wait to enter a long trade until the price reaches the high of the bar on which the +DI crossed over the -DI, and wait to enter a short trade until the price reaches the low of the bar on which the -DI crossed over the +DI. See http://www.fmlabs.com/reference/default.htm?url=DI.htm.

- MOM - Momentum: The Momentum is a measurement of the acceleration and deceleration of prices. It indicates if prices are increasing at an increasing rate or decreasing at a decreasing rate. The Momentum function can be applied to the price, or to any other data series.
- OBV - On Balance Volume: The On Balance Volume (OBV) is a cumulative total of the up and down volume. When the close is higher than the previous close, the volume is added to the running total, and when the close is lower than the previous close, the volume is subtracted from the running total. See http://www.fmlabs.com/reference/default.htm?url=OBV.htm.
- PLUS_DI - Plus Directional Indicator: The +DI is the percentage of the true range that is up.
- ROCP - Rate of change Percentage: The Rate of Change function measures rate of change relative to previous periods.
- ROCR - Rate of change ratio: price/prevPrice
- RSI - Relative Strength Index: The Relative Strength Index (RSI) calculates a ratio of the recent upward price movements to the absolute price movement. The RSI ranges from 0 to 100. The RSI is interpreted as an overbought/oversold indicator when the value is over 70/below 30. You can also look for divergence with price. If the price is making new highs/lows, and the RSI is not, it indicates a reversal. Details can be found at http://www.fmlabs.com/reference/default.htm?url=RSI.htm.
- SMA - Simple Moving Average: Moving Averages are used to smooth the data in an array to help eliminate noise and identify trends. The Simple Moving Average is literally the simplest form of a moving average. Each output value is the average of the previous n values. In a Simple Moving Average, each value in the time period carries equal weight, and values outside of the time period are not included in the average. This makes it less responsive to recent changes in the data, which can be useful for filtering out those changes.

- STOCHRSI - Stochastic Relative Strength Index: Stochastic RSI (StochRSI) is an indicator of an indicator. It calculates the RSI relative to its range in order to increase the sensitivity of the standard RSI. The values of the StochRSI are from zero to one. See http://www.fmlabs.com/reference/default.htm?url=StochRSI.htm.
- TRANGE - True Range: The True Range function is used in the calculation of many indicators, most notably, the Welles Wilder DX. It is a base calculation that is used to determine the normal trading range of a stock or commodity. See http://www.fmlabs.com/reference/default.htm?url=TR.htm.
- TYPPRICE - Typical Price: The Typical Price is the average of the high + low + close of a bar. It is used in the calculation of several indicators. It can be used to smooth an indicator that normally takes just the closing price as input.
- WCLPRICE - Weighted Close Price: The Weighted Close is the average of the high, low and close of a bar, but the close is weighted, actually counted twice. It is used in the calculation of several indicators. It can be used to smooth an indicator that normally takes just the closing price as input.
- WILLR - Williams' %R: The Williams %R is similar to an unsmoothed Stochastic %K. The values range from zero to 100, and are charted on an inverted scale, that is, with zero at the top and 100 at the bottom. Values below 20 indicate an overbought condition and a sell signal is generated when it crosses the 20 line. Values over 80 indicate an oversold condition and a buy signal is generated when it crosses the 80 line. Details can be found at http://www.fmlabs.com/reference/default.htm?url=WilliamsR.htm.
- WMA - Weighted Moving Average: The Weighted Moving Average calculates a weight for each value in the series. The more recent values are assigned greater weights. The Weighted Moving Average is similar to a Simple Moving average in that it is not cumulative, that is, it only includes values in the time period (unlike an Exponential Moving Average). The Weighted Moving Average is similar to an Exponential Moving Average in that more recent data has a greater contribution to the average.

References

- Zhang and Wu (2009), Stock market prediction of S&P 500 via combination of improved BCO approach and BP neural network. Expert Systems with Applications 36.

- Chang and Liu (2008), A TSK type fuzzy rule based system for stock price prediction. Expert Systems with Applications 34.
- Chang et al. (2009), A neural network with a case based dynamic window for stock trading prediction. Expert Systems with Applications 36.
- Kara et al. (2011), Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange. Expert Systems with Applications 38.
- http://www.nlreg.com/results.htm