Investment and Trading Capstone Project: Build a Stock Price Indicator

by

Xiaoke Fei

to

Udacity

in Partial Fulfillment of the Requirements

for the Nanodegree of

Machine Learning Engineer

January 2017

Chapter 1

Definition

1.1 Project Overview

This project consists of two main tasks. The first main task is to build a stock price predictor that takes daily Adjusted Close price data of certain stocks over a certain date range and a prediction interval as input, and outputs the prediction of future Adjusted Close price of the stocks for a given query date. The stock price predictor is built as a user-friendly interface such that users can specify stock(s) they are interested in, the training date range, and the prediction interval. The second main task is to use the stock price predictor we built to construct a trading strategy that suggests when to buy or sell a certain stock. Both of the stock price predictor and the trading strategy will be tested and evaluated in 3 different case studies.

The data we use in this project are queried directly from Yahoo! Finance. There are 3 case studies in this project. The first study uses daily trading data of Google. The Google's daily Adjusted Close prices ranging from 2011

to 2014 will be used to train a stock price predictor. Then a trading strategy that suggests when to buy or sell Google stock is constructed based on the predictor. The performance of both the predictor and the trading strategy will be tested using the daily Google stock data of the year 2015. In the second study, we use the daily trading data of Apple ranging from 2003 to 2007 as training data, and use the daily trading data of Apple in year 2008 as testing data to conduct a study in the same way as we do in the first study. Thirdly, the same study is conducted using the IBM daily stock data ranging from 2010 to 2013 as training data, and the IBM daily stock data in 2014 as testing data. In all the 3 studies, we will attempt 3 different prediction intervals: 5, 10 and 20 trading days respectively. The reason that we conduct 3 case studies is that we want to investigate if our stock price predictor and corresponding trading strategy can perform consistently well in different cases.

1.2 Problem Statement

The core problem of this project is to predict future Adjusted Close price of a certain stock at a given query date. The idea of solving this problem is as follows. First of all, we build several regression models that can predict the rate (in percentage) that the Adjusted Close price of the stock will change in the future using historical daily trading data, and select the best model among them as the final regression model. Then, at a given query date, we can predict the rate of change of the stock price using the final regression model. Finally, we can predict the future Adjusted Close price based on the

predicted rate of change together with the Adjusted Close price of the query date.

The regression models will be be built in the following several steps. First of all, we extract daily Adjusted Close price data of the stock we are interested in and the S&P 500 at a pre-defined date range (training date range) from the Yahoo! Finance. Then the extracted data will be pre-processed to generate features and target value for training regression models. The features in the regression models are the technical indicators (Bollinger Bands value, Relative Strength Index, etc.) generated based on the Adjusted Close price of both the stock and the S&P 500. The target value in the regression models is the rate of change of the stock Adjusted Close price in the future. Multiple regression models such as K-Nearest Neighbors (KNN), Bagging, Decision Tree and Random Forest are trained using the processed training data set. Different tuning parameters of each model will be attempted during the training, and the optimal one will be selected. Also, these models are evaluated in the training phase, and the best model is selected as the final regression model for future prediction.

1.3 Metrics

The metrics we will use for evaluating the performance of our regression models is mean absolute percentage error (MAPE). The definition of MAPE is as follows. Suppose the daily Adjusted Close prices in the testing data are $y_1, y_2, ..., y_n$, and the corresponding predictions of the Adjusted Close prices

are $\hat{y_1}, \hat{y_2}, ..., \hat{y_n}$, then the MAPE is defined as

$$\text{MAPE} = \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{n}.$$

MAPE measures the average difference between the actual value and the predicted value in terms of the percentage of the actual value. Smaller MAPE indicates better performance of the model. During the process of training, the optimal tuning parameter of each model will be selected based on MAPE metrics. The final regression model will also be selected using MAPE metrics.

To evaluate the performance of our trading strategy, cumulative return and Sharpe ratio on portfolio values will be used as metrics. The cumulative return measures the profit we make in percentage at the end of the trading date. On the other hand, Sharpe ratio not only consider the profit, but also consider the risk of the trading we make.

Chapter 2

Analysis

2.1 Data Exploration and Pre-processing

For a certain stock, the raw daily trading data we extract from Yahoo! Finance include Open, High, Low, Close, Volume and Adjusted Close. In this project, we only use Adjusted Close. So the first step is remove all other values except Adjust Close from the raw data. For each of Google, Apple and IBM, the daily Adjusted Close price data we use in the project is time series data shown below in Figure 2.1, Figure 2.2 and Figure 2.3 respectively.

In Figure 2.1, the Google daily adjusted close price data is ranging from 01/01/2011 to 12/31/2015. The data from 01/01/2011 to 12/31/2014 will be used to generate features and target values for training regression models. The data from 01/01/2015 to 12/31/2015 will be used for testing.

In Figure 2.2, the Apple daily adjusted close price data is ranging from 01/01/2004 to 12/31/2008. The data from 01/01/2004 to 12/31/2007 will be used to generate features and target values for training regression models.

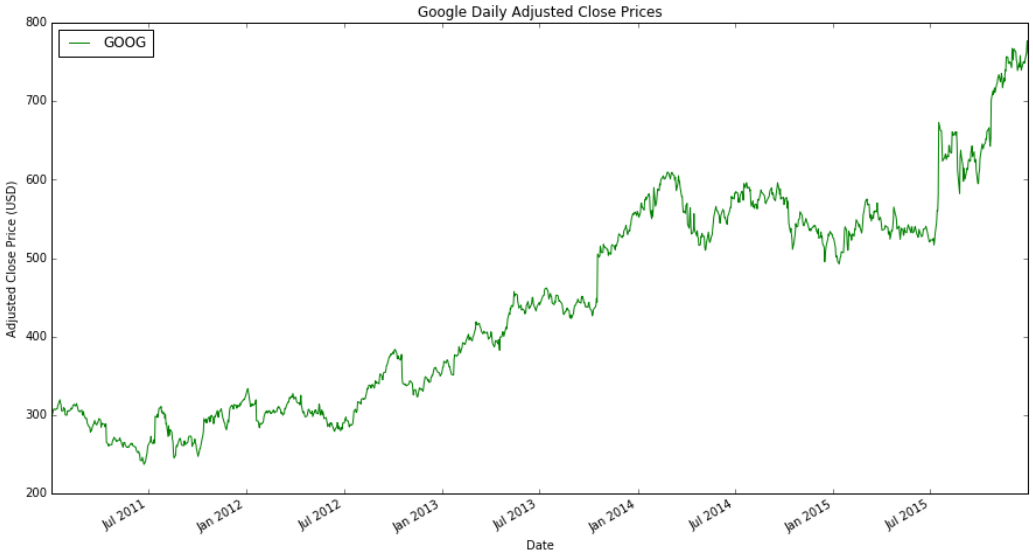

Figure 2.1: Time Series of Google Daily Adjusted Close Price

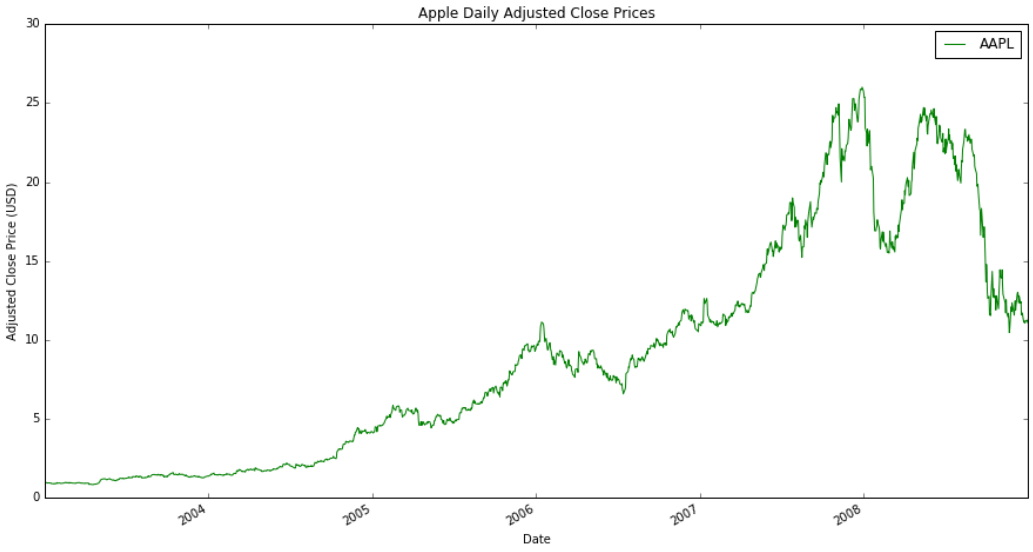

Figure 2.2: Time Series of Apple Daily Adjusted Close Price

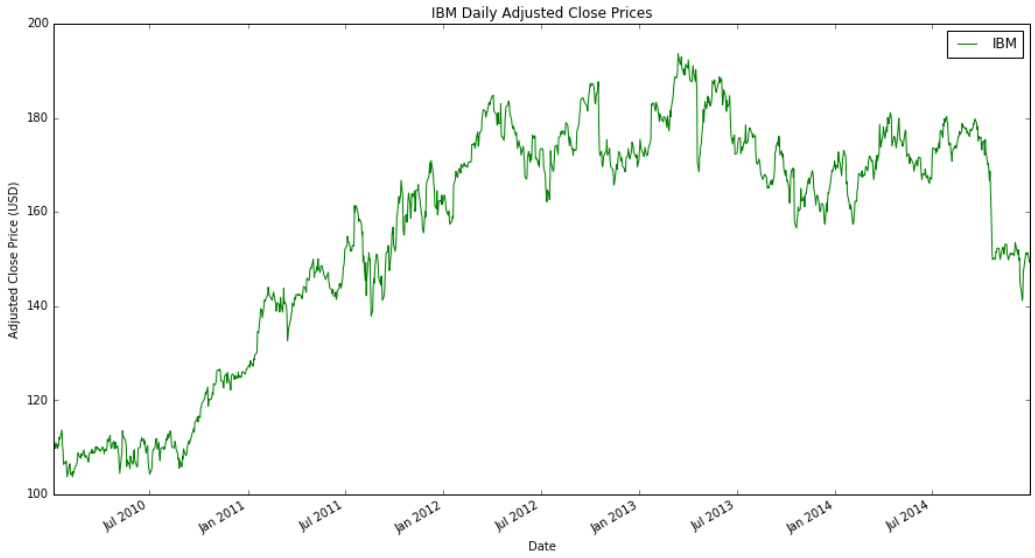

Figure 2.3: Time Series of IBM Daily Adjusted Close Price

The data from 01/01/2008 to 12/31/2008 will be used for testing.

In Figure 2.3, the IBM daily adjusted close price data is ranging from 01/01/2010 to 12/31/2014. The data from 01/01/2010 to 12/31/2013 will be used to generate features and target values for training regression models. The data from 01/01/2014 to 12/31/2014 will be used for testing.

For each stock, the technical indicators (features) will be computed based on the adjusted close price of the stock and S&P 500 at both training and testing range. Also, the rate of change in the future (target) will be computed based on the stock data at only training range.

The first technical indicator is Bollinger Bands Value. The Bollinger Bands Values for n trading days at the date t is computed by

$$BB(n)_t = \frac{y_t - sma(n)_t}{2 \times std(n)_t}$$

where y_t is the adjusted close price at the date t, $sma(n)$ is the simple moving average value of past n trading days at the date t, and $std(n)$ is the standard deviation of past n trading days at the date t.

The second technical indicator is Rate Of Change for past n trading days. It is computed by:

$$R(n)_t = \frac{y_t - y_{t-n}}{y_{t-n}}$$

where y_t denotes the adjusted close price at the date t, and y_{t-n} denotes the adjusted close price at the date that is n trading days before t.

The third technical indicator is Relative Strength Index (RSI). The RSI for n trading days at the date t is computed by

$$RSI(n)_t = 100 - \frac{100}{1 + \frac{up(n)_t}{down(n)_t}}$$

where $up(n)_t$ is the simple moving average of each day's positive price change for past n trading days at the date t, and $down(n)_t$ is the simple moving average of each day's negative price change for past n trading days at the date t.

We will compute the above mentioned technical indicators for 5, 10 and 20 trading days respectively based on both the stock price and the S&P 500 price. As a result, 18 technical indicators will be generated as feature for a given stock. 9 of them are generated based on the stock price using 3 different number of trading days, and the other 9 of them are generated based on the S&P 500 price. After all the features are generated, we need to normalize each feature such that the range of the feature values falls into $(-1, 1)$. The reason for doing this normalization is that some methods such as KNN and Bagging of KNN are sensitive to the scale of features because distances are

Table 2.1: Google Training Data

Date	BB5	BB10	BB20	...	Y
02/01/2011	0.118125	-0.183153	-0.244379	...	0.012012
02/02/2011	0.259904	-0.045392	-0.232173	...	0.007353
02/03/2011	0.282809	-0.072445	-0.336949	...	0.010309
02/04/2011	0.215050	-0.000001	-0.283076	...	0.022128
02/07/2011	0.815291	0.233651	-0.097964	...	0.022563
...	...	...	...	...	...

Prediction Interval: 5 trading days

computed during the process of fitting these models.

The target value, rate of change in the future, is computed depending on the prediction interval we pre-defined. Suppose, the prediction interval we specified is n trading days, then the target value at the date t is computed by

$$ROC_t = \frac{y_{t+n} - y_t}{y_t}$$

where y_t is the adjusted close at the date t, and y_{t+n} is the the adjusted close at the date n trading days after t.

As an example, Table 2.1 shows the first 5 rows of the training data generated based on Google stock data where the pre-defined prediction interval is 5 trading days. We only shows 3 of the 18 features in the table because of the limited space. The column "Y" is the target value that denotes the rate the price will change after 5 trading days. For example, the "Y" in the first row is 0.012012, it means that the price of Google stock will arise 1.2012% 5 trading days after 02/01/2011. Note that our training data

starts from 02/01/2011 rather than 01/01/2011, this is because the Bollinger Bands Value for 20 trading days starts from 02/01/2011 when we used the price data frm 01/01/2011 to compute it. The training data will be used to fit regression models which will be illustrated in next section.

2.2 Algorithms and Techniques

The regression models we apply in this project include K-Nearest-Neighbor (KNN), Bagging, Decision Tree and Random Forest.

To predict a new instance, KNN model firstly find k nearest instances to the new instance in the training set, and then compute the mean target values of these instances as the prediction of the new instance. The k is a tuning parameter that will be adjusted during the training phase in this project.

In this project, the Bagging model uses KNN model as the base regression model. Each base KNN model is fitted using a bootstrapped data set of training data. The k in each KNN model we use will be the optimal k we obtained when building the KNN model. To make a prediction, Bagging averages the predictions of base KNN models. The ensemble size of Bagging is a tuning parameter that will be adjusted during the training phase.

The Decision Tree model uses Gini Index to split feature space recursively until the tree is large enough. To make a prediction, the new instance navigates down the tree from root based on the feature split conditions until it arrives a leaf node. Then the prediction is the average of target values of the instances in the leaf node. The performance of a Decision Tree model

often depends on the complexity of the tree. In this project, we will use the minimum number of instances required to split an internal node as the stopping criterion to control the tree complexity when building a tree model. Different choices of minimum number will be attempted during the training phase.

The Random Forest model uses Decision Tree model as base model. Each base model is built on a bootstrapped data set. The number of features to consider when looking for the best split we use in this project is the number of all features, which is the default value in sci-kit learn. The ensemble size of Random Forest model is a tuning parameter that will be adjusted during the training phase.

These models predict rate in percentage that the stock will change. This prediction can help us to predict the future stock price in the following way. Suppose $r\hat{o}c$ is the predicted rate of change at the date t, and y_t is the stock price at the date t. Also suppose n trading days is the prediction interval. Then the predicted price at the day that n trading days after the date t is computed by

$$\hat{y}_{t+n} = y_t \times (1 + r\hat{o}c)$$

Each model will select its optimal tuning parameter. Also, these models will be compared and the best model will be selected as the final model for future prediction. All the model evaluation and selection works are done in the training phase. They are illustrated in details in Section 3.1.

2.3 Benchmark

In this project, after a stock price predictor of a certain stock is built, we will construct a trading strategy based on predictions by the stock price predictor. Then we use the trading strategy to trade the stock at the testing date range, and compute the normalized portfolio values. The benchmark we use in this project to compare with the normalized portfolio values will be the normalized stock price at the testing range. In another word, the benchmark strategy is that we buy the stock using all our cash at the beginning of the testing date range and hold the stock all the way until the testing end date. To compare with the benchmark, we can tell whether our trading strategy is better than just buying the stock at the beginning and doing nothing else. The trading strategy will be illustrated in Section 3.1

In each of 3 case studies, the comparison between our trading strategy and benchmark strategy will be made in terms of cumulative return and Sharpe ratio. Also, graph will be made for visualizing the comparison.

Chapter 3

Methodology

3.1 Implementation and Refinement

In this section, we will illustrate the implementations on models and trading strategy separately. In the illustration of implementing models, how to refine models by adjusting tuning parameters are also included.

3.1.1 Implementation on Models

Suppose T is a training data set, the first step is to divide T into two parts: T_{small} and $T_{validation}$. T_{small} is the first $\frac{3}{4}$ of T and $T_{validation}$ is the other $\frac{1}{4}$ of T. T_{small} will be used to fit different models with different choices of tuning parameters using the 4 methods described in Section 2.2. These models then will be evaluated on $T_{validation}$ in terms of MAPE. Next, the best method and its corresponding optimal tuning parameter will be selected according to the performance of models on $T_{validation}$. Finally, the best method and its corresponding optimal tuning parameter will be used to fit the final model

on the training data set T.

The reason that we use the above mentioned way to do the models validation and selection instead of using cross-validation technique is as follows. In the field of trading, the data is time series data, i.e., the data set T is ordered by date. According to Professor Tucker Balch (instructor of "Machine Learning for Trading" class, Georgia Tech), the date range of validation set or testing set should always be after the date range of training set. If the date range we use to train models is after the date range we use to validate or test models, then the validating or testing results will be meaningless because our goal is to predict price of the future, not the past. Based on this idea, cross-validation technique is not appropriate for validating and testing models in this project because cross-validation will disorganize the dates order of T when randomly shuffling it. So, in our implementation described above, we divided T into T_{small} and $T_{validation}$ strictly by the date order.

When applying KNN method, $k = 3, 4, ..., 20$ will be attempted respectively to build 17 KNN models on T_{small}. Each KNN model is then evaluated by computing the MAPE on $T_{validation}$. The k that results in the minimum MAPE will be used as the k of base KNN model in Bagging. When applying Bagging method, we will attempt different ensemble size as 20, 30, ..., 100. 9 Bagging models will be built on T_{small} using different size respectively. Then MAPEs of these Bagging models are computed on $T_{validation}$. When Decision Tree method is applied, 6 different choices of the minimum number of instances required to split an internal node will be attempted to build 6 models on T_{small}. They are 5, 10, 15, 20, 25 and 30. For each of the 6 Decision Tree model, MAPE is computed on $T_{validation}$. When Random Forest

method is applied, the ensemble size of 20, 30, ..., 100 will be attempted respectively to build 9 forest model on T_{small}. Then each forest will be used to compute MAPE on $T_{validation}$.

After all the models are fitted, their MAPEs will be compared. The method and its corresponding tuning parameter resulting in minimum MAPE will be selected. Then the final regression model will be fitted on the training data T using the selected method and its corresponding tuning parameter. For example, if the minimum MAPE is obtained by a Random Forest model that has ensemble size of 70, then the final regression model is fitted on T using Random Forest method that sets the ensemble size to be 70. The final regression model will be used for future predictions. The whole process of models validation and selection described above will be coded in a complicated function (method) called **get_best_learner()** in the class of **BestLearner**. The details of the code can be found in the complementary ipython notebook file.

3.1.2 Implementation on Trading Strategy

We explain the Trading Strategy by illustrating one round of BUY and SELL or SELL and BUY operations starting at the date t. Suppose the prediction interval is n trading days. Also suppose at date t, y_t is the stock price and $r\hat{o}c$ is the predicted rate of change obtained by the final regression model. Then, the predicted price at the date that n trading days after the date t is obtained by

$$\hat{y}_{t+n} = y_t \times (1 + r\hat{o}c)$$

Then we compare $\hat{y}_{t+n}$ and y_t to make our trading operations:

1. If $\hat{y}_{t+n} > y_t$:

(1) BUY 100 shares of the stock at the date t and hold it until the date $t + n$.

(2) At $t+n$, if $y_{t+n} \geq \hat{y}_{t+n}$: SELL 100 shares to finish this round.
else: Continue holding until a date s such that $y_s \geq \hat{y}_{t+n}$, and SELL 100 shares at the date s to finish this round.

(3)Stop Loss: At any date d after t, if $y_d < 0.9 \times y_t$ and we haven't SELL our shares yet, then we SELL 100 shares to finish the round earlier.

2. If $\hat{y}_{t+n} < y_t$:

(1) Short (SELL) 100 shares of the stock at the date t and hold it until the date $t + n$.

(2) At $t+n$, if $y_{t+n} \leq \hat{y}_{t+n}$: BUY 100 shares to finish this round.
else: Continue holding until a date s such that $y_s \leq \hat{y}_{t+n}$, and BUY 100 shares at the date s to finish this round.

(3)Stop Loss: At any date d after t, if $y_d > 1.1 \times y_t$ and we haven't BUY our shares back yet, then we BUY 100 shares to finish the round earlier.

The above Trading Strategy algorithm can be summarized as follows. If the predicted price at $t + n$ is higher than the price at t, then we buy 100 shares, else we short 100 shares. Then we hold our position until $t + n$. If the actual price at $t + n$ is better than our prediction, we buy or sell our shares to finish this round of trading, else we continue holding our position until the price is better than our prediction and then buy or sell our shares to finish this round of trading. We also set stop loss thresholds

in our strategy. If we buy shares at t and the price drops more than 10% at some point after t, we sell our shares to stop loss and finish this round of trading earlier. On the other hand, if we short shares at t and the price increases more than 10% at some point after t, we buy our shares back to stop loss and finish this round of trading earlier. The trading strategy described above is coded as a complicated function (method) called **create_orders()** in the class **TradeStock**. The details of the code can also be found in the complementary ipython notebook file.

3.2 Experimental Design

In each study, we firstly build 3 stock price predictors that the pre-defined the prediction intervals are set to be 5, 10 and 20 trading days respectively using the implementation described in Section 3.1.1. Then we compare the MAPEs on $T_{validation}$ of the final regression models of these predictors to see which prediction interval results in the minimum MAPE. Suppose n is the prediction interval that has the minimum MAPE on $T_{validation}$, then the stock price predictor with the prediction interval of n will be used for the following study.

The predictor will be used to predict adjusted close prices on testing date range. Then the predicted prices will be compared with the actual prices of testing date range. The MAPE between them will be reported. The graph is also made to visualize their differences.

Then we trade the stock during the testing date range using the trading strategy described in Section 3.1.2, and compute the portfolio values. The

cumulative return and Sharpe ratio of our portfolio values are computed to evaluated our trading strategy. For comparison, we also compute the cumulative return and Sharpe ratio of our benchmark values (stock prices). We also plot the normalized portfolio value and normalized stock price together to visualize the difference between our trading strategy and the benchmark.

Chapter 4

Results

4.1 Model Evaluation and Validation

4.1.1 Google Case Study

In the study of Google trading data, the training date range is from 01/01/2011 to 12/31/2014, and the testing date range is from 01/01/2015 to 12/31/2015. Table 4.1 shows the result of regression models when the prediction interval is 5 trading days.

From Table 4.1, we can see that the Random Forest method of size 100 results in the minimum MAPE on $T_{validation}$ when the prediction interval is 5 trading days. So when the prediction interval is 5 trading days, the final regression model is fitted using Random Forest method of size 100 on the training data set T.

Table 4.2 shows the result of regression models when the prediction interval is 10 trading days. From Table 4.2, we can see that the KNN method

Table 4.1: Models Information, Prediction Interval: 5 trading days

Method	Optimal Tuning Parameter	MAPE on $T_{validation}$
KNN	$k = 19$	0.0255
Bagging	size=100	0.0257
Decision Tree	msp=30	0.0324
Random Forest	size=100	0.0248

Stock: Google

Table 4.2: Models Information, Prediction Interval: 10 trading days

Method	Optimal Tuning Parameter	MAPE on $T_{validation}$
KNN	$k = 19$	0.0359
Bagging	size=90	0.0364
Decision Tree	msp=25	0.0452
Random Forest	size=80	0.0363

Stock: Google

that sets k to be 19 results in the minimum MAPE on $T_{validation}$ when the prediction interval is 10 trading days. So when the prediction interval is 10 trading days, the final regression model is fitted on the training data set T using KNN method that sets k to be 19.

Table 4.3 shows the result of regression models when the prediction interval is 20 trading days. From Table 4.3, we can see that the KNN method that sets k to be 19 results in the minimum MAPE on $T_{validation}$ when the prediction interval is 20 trading days. So when the prediction interval is 20 trading days, the final regression model is fitted on the training data set T

Table 4.3: Models Information, Prediction Interval: 20 trading days

Method	Optimal Tuning Parameter	MAPE on $T_{validation}$
KNN	$k = 19$	0.0525
Bagging	size=90	0.0527
Decision Tree	msp=30	0.0723
Random Forest	size=100	0.0555

Stock: Google

Table 4.4: Final Models Comparison For Different Prediction Intervals

Prediction Interval	Final Method	Optimal Tuning Parameter	MAPE on $T_{validation}$
5	Random Forest	size=100	0.0248
10	KNN	$k = 19$	0.0359
20	KNN	$k = 19$	0.0525

Stock: Google

using KNN method that sets k to be 19.

Table 4.4 shows comparisons between the final models for different prediction intervals. As a result, the minimum MAPE on $T_{validation}$ is obtained when the prediction interval is 5 trading days. So, we will use 5 trading days as the prediction interval in the study of Google stock data. When the prediction interval is 5, the final regression model is the random forest model of size 100 built on the training data set T. The final model is then used to make predictions at testing date range. To quantify the difference between the predictions and the actual prices at the testing date range, the MAPE is computed. As a result, the MAPE between the predictions and the actual

Table 4.5: Models Information, Prediction Interval: 5 trading days

Method	Optimal Tuning Parameter	MAPE on $T_{validation}$
KNN	$k = 19$	0.0406
Bagging	size=50	0.0404
Decision Tree	msp=25	0.0496
Random Forest	size=80	0.0406

Stock: Apple

prices on testing set is 2.97%.

4.1.2 Apple Case Study

In the study of Apple trading data, the training date range is from 01/01/2003 to 12/31/2007, and the testing date range is from 01/01/2008 to 12/31/2008. Table 4.5 shows the result of regression models when the prediction interval is 5 trading days.

From Table 4.5, we can see that the Bagging method of size 50 results in the minimum MAPE on $T_{validation}$ when the prediction interval is 5 trading days. So when the prediction interval is 5 trading days, the final regression model is fitted using Bagging method of size 50 on the training data set T.

Table 4.6 shows the result of regression models when the prediction interval is 10 trading days. From Table 4.6, we can see that the Bagging method of ensemble size 30 results in the minimum MAPE on $T_{validation}$ when the prediction interval is 10 trading days. So when the prediction interval is 10 trading days, the final regression model is fitted using Bagging method of ensemble size 30 on the training data set T.

Table 4.6: Models Information, Prediction Interval: 10 trading days

Method	Optimal Tuning Parameter	MAPE on $T_{validation}$
KNN	$k = 19$	0.0565
Bagging	size=30	0.0564
Decision Tree	msp=25	0.0658
Random Forest	size=80	0.0567

Stock: Apple

Table 4.7: Models Information, Prediction Interval: 20 trading days

Method	Optimal Tuning Parameter	MAPE on $T_{validation}$
KNN	$k = 18$	0.0918
Bagging	size=80	0.0912
Decision Tree	msp=30	0.1031
Random Forest	size=50	0.0937

Stock: Apple

Table 4.8: Final Models Comparison For Different Prediction Intervals

Prediction Interval	Final Method	Optimal Tuning Parameter	MAPE on $T_{validation}$
5	Bagging	size=50	0.0404
10	Bagging	size=30	0.0564
20	Bagging	size=80	0.0912

Stock: Apple

Table 4.7 shows the result of regression models when the prediction interval is 20 trading days. From Table 4.7, we can see that the Bagging method of ensemble size 80 results in the minimum MAPE on $T_{validation}$ when the prediction interval is 20 trading days. So when the prediction interval is 20 trading days, the final regression model is fitted on the training data set T using Bagging method that sets the ensemble size to be 80.

Table 4.8 shows comparisons between the final models for different prediction intervals. As a result, the minimum MAPE on $T_{validation}$ is obtained when the prediction interval is 5 trading days. So, we will use 5 trading days as the prediction interval in the study of Apple stock data. When the prediction interval is 5, the final regression model is the Bagging model of size 50 built on the training data set T. The final model is then used to make predictions at testing date range. To quantify the difference between the predictions and the actual prices at the testing date range, the MAPE is computed. As a result, the MAPE between the predictions and the actual prices on testing set is 6.90% in this case study.

Table 4.9: Models Information, Prediction Interval: 5 trading days

Method	Optimal Tuning Parameter	MAPE on $T_{validation}$
KNN	$k = 19$	0.0215
Bagging	size=70	0.0214
Decision Tree	msp=30	0.0252
Random Forest	size=100	0.0218

Stock: IBM

Table 4.10: Models Information, Prediction Interval: 10 trading days

Method	Optimal Tuning Parameter	MAPE on $T_{validation}$
KNN	$k = 19$	0.0320
Bagging	size=90	0.0321
Decision Tree	msp=30	0.0353
Random Forest	size=50	0.0321

Stock: IBM

4.1.3 IBM Case Study

In the study of IBM trading data, the training date range is from 01/01/2010 to 12/31/2013, and the testing date range is from 01/01/2014 to 12/31/2014. Table 4.9 shows the result of regression models when the prediction interval is 5 trading days.

From Table 4.9, we can see that the Bagging method of size 70 results in the minimum MAPE on $T_{validation}$ when the prediction interval is 5 trading days. So when the prediction interval is 5 trading days, the final regression model is fitted using Bagging method of size 70 on the training data set T.

Table 4.11: Models Information, Prediction Interval: 20 trading days

Method	Optimal Tuning Parameter	MAPE on $T_{validation}$
KNN	$k = 18$	0.0436
Bagging	size=70	0.0435
Decision Tree	msp=25	0.0522
Random Forest	size=100	0.0442

Stock: IBM

Table 4.10 shows the result of regression models when the prediction interval is 10 trading days. From Table 4.10, we can see that the KNN method that sets k to be 19 results in the minimum MAPE on $T_{validation}$ when the prediction interval is 10 trading days. So when the prediction interval is 10 trading days, the final regression model is fitted on the training data T using the KNN method that sets k to be 19.

Table 4.11 shows the result of regression models when the prediction interval is 20 trading days. From Table 4.11, we can see that the Bagging method of ensemble size 70 results in the minimum MAPE on $T_{validation}$ when the prediction interval is 20 trading days. So when the prediction interval is 20 trading days, the final regression model is fitted on the training data set T using Bagging method that sets the ensemble size to be 70.

Table 4.12 shows comparisons between the final models for different prediction intervals. As a result, the minimum MAPE on $T_{validation}$ is obtained when the prediction interval is 5 trading days. So, we will use 5 trading days as the prediction interval in the study of IBM stock data. When the prediction interval is 5, the final regression model is the Bagging model of

Table 4.12: Final Models Comparison For Different Prediction Intervals

Prediction Interval	Final Method	Optimal Tuning Parameter	MAPE on $T_{validation}$
5	Bagging	size=70	0.0215
10	KNN	$k = 19$	0.0320
20	Bagging	size=70	0.0435

Stock: IBM

size 70 built on the training data set T. The final model is then used to make predictions at testing date range. To quantify the difference between the predictions and the actual prices at the testing date range, the MAPE is computed. As a result, the MAPE between the predictions and the actual prices on testing set is 2.00% in this case study.

4.2 Trading Strategy Justification

In this section, we use the trading strategy described in Section 3.1.2 to conduct 3 different case studies. In each study, we trade one certain stock during a testing date range. We set the starting cash value to be 100000 US dollars. Also, the leverage is not limited during the trading date range. The details of the studies are illustrated one by one as follows.

4.2.1 Case 1: Trade Google Stock

In this study, we trade Google Stock from 01/01/2015 to 12/31/2015 using the trading strategy in Section 3.1.2. The stock price predictor, as described in Section 4.1.1, uses 5 trading days as the prediction interval. We start

Table 4.13: Google: Portfolio Values vs Benchmark

	Cumulative Return	Sharpe Ratio
Portfolio Values	1.408	1.639
Stock Prices	0.446	1.395

our trading with 100000 US dollars. The daily portfolio values (sum of cash and stock values at each day) during the year 2015 will be computed. Then the cumulative return and the Sharpe ratio will be computed based on the portfolio values. Also, we compute the benchmark values: the cumulative return and the Sharpe ratio based on the Google stock prices from 01/01/2015 to 12/31/2015. The results are shown in Table 4.13.

From Table 4.13, we can see that the cumulative return of our portfolio values is much higher than that of the stock prices. It can be interpreted as follows. If we use our trading strategy to trade Google stock in year 2015, then at the end of year 2015, our total value will increase 140.8%. On the other hand, if we only buy the Google stock at the beginning of year 2015 using all our cash and hold it, our total value will only increase 44.6% at the end of year 2015. So, our trading strategy can help us make a lot more profits than the benchmark strategy in this study. Besides, the Sharpe ratio of our portfolio values is also higher than that of the stock price. It means that our trading strategy is less risk than the benchmark strategy in terms of stability of daily returns.

Figure 4.1 shows the comparison between the normalized daily portfolio value and the normalized stock price. We can find that our normalized portfolio value is higher than the normalized stock price at any day during

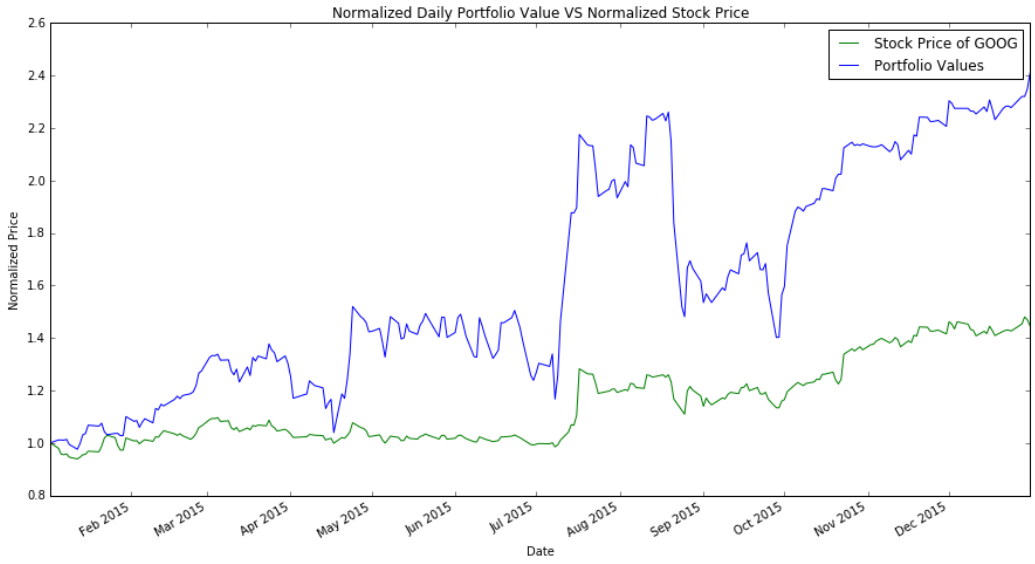

Figure 4.1: Google: Normalized Portfolio Value vs Normalized Stock Price

the year of 2015. It means that the total value by our trading strategy is always higher than that by the benchmark during the testing date range. In another word, our trading strategy not only beats the Google stock market at the end of the year 2015, but also beats the market through the year 2015.

4.2.2 Case 2: Trade Apple Stock

In the study 2, we trade Apple Stock from 01/01/2008 to 12/31/2008 using the trading strategy in Section 3.1.2. The stock price predictor, as described in Section 4.1.1, uses 5 trading days as the prediction interval. We start our trading with 100000 US dollars. The daily portfolio values (sum of cash and stock values at each day) during the year 2008 will be computed. Then the cumulative return and the Sharpe ratio will be computed based on the daily portfolio values. Also, we compute the benchmark values: the cumulative

Table 4.14: Apple: Portfolio Values vs Benchmark

	Cumulative Return	Sharpe Ratio
Portfolio Values	-0.155	-1.693
Stock Prices	-0.562	-1.120

return and the Sharpe ratio based on the Google stock prices from 01/01/2008 to 12/31/2008. The results are shown in Table 4.14.

From Table 4.14, we can see that the cumulative return of our portfolio values is much higher than that of the stock prices. If we use our trading strategy to trade Apple stock in year of 2008, then at the end of year 2008, our total value will decrease 15.5%. On the other hand, if we use the benchmark strategy buying the Apple stock at the beginning of year 2008 using all our cash and holding it, our total value will decrease 56.2% at the end of year 2008. So in this study, we can see that even though the Apple stock market is bad during 2008, our trading strategy still can help us lose much less money than the benchmark strategy. Besides, the Sharpe ratios of both portfolio values and stock price are negative. It means that both of our trading strategy and benchmark strategy are worse than just doing nothing. But still, our trading strategy is better than the benchmark strategy in terms of cumulative return.

Figure 4.2 shows the comparison between the normalized daily portfolio value and the normalized stock price. We can find that our normalized portfolio value is higher than the normalized stock price at any day during the year of 2008. It means that the total value by our trading strategy is always higher than that by the benchmark during the testing date range. In

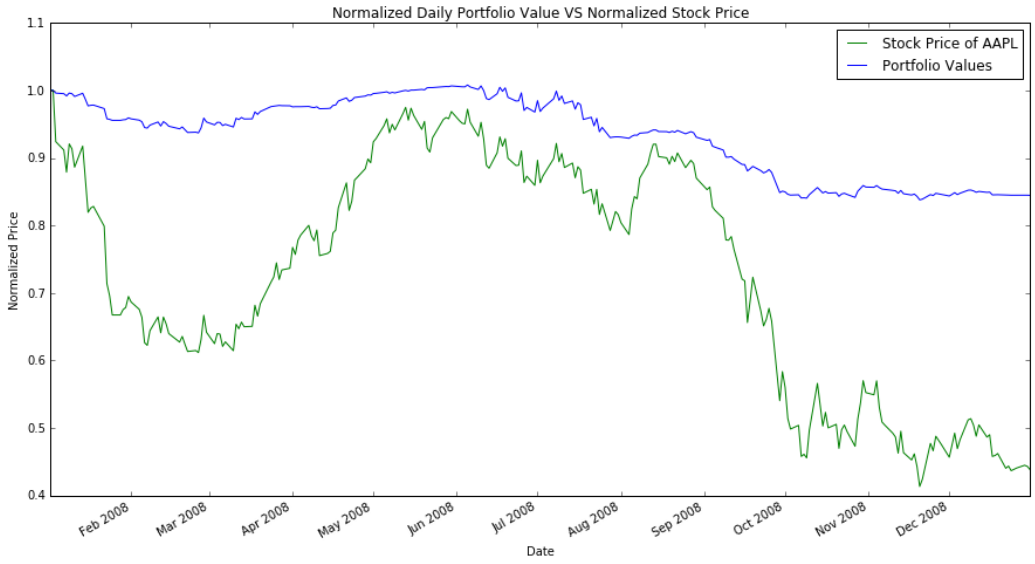

Figure 4.2: Google: Normalized Portfolio Value vs Normalized Stock Price

another word, our trading strategy not only beats the Apple stock market at the end of the year 2008, but also beats the market through the year 2008.

4.2.3 Case 3: Trade IBM Stock

In this study, we trade IBM Stock from 01/01/2014 to 12/31/2014 using the trading strategy in Section 3.1.2. The stock price predictor, as described in Section 4.1.1, uses 5 trading days as the prediction interval. We start our trading with 100000 US dollars. The daily portfolio values (sum of cash and stock values at each day) during the year 2014 will be computed. Then the cumulative return and the Sharpe ratio will be computed based on the portfolio values. Also, we compute the benchmark values: the cumulative return and the Sharpe ratio based on the IBM stock prices from 01/01/2014 to 12/31/2014. The results are shown in Table 4.15.

Table 4.15: IBM: Portfolio Values vs Benchmark

	Cumulative Return	Sharpe Ratio
Portfolio Values	0.102	0.462
Stock Prices	-0.114	-0.623

From Table 4.15, we can see that the cumulative return of our portfolio values is much higher than that of the stock prices. It can be interpreted as follows. If we use our trading strategy to trade IBM stock in year 2014, then at the end of year 2014, our total value will increase 10.2%. On the other hand, if we only buy the IBM stock at the beginning of year 2014 using all our cash and hold it, our total value will decrease 11.4% at the end of year 2014. So, our trading strategy can help us make more than 10% profit while the benchmark strategy can let us lose more than 10% total values in this study. Besides, the Sharpe ratio of our portfolio values is positive and higher than that of the stock price.

Figure 4.3 shows the comparison between the normalized daily portfolio value and the normalized stock price. We can see that our normalized portfolio value is higher than the normalized stock price at most of the days in the year of 2014. It means that in most of the time of year 2014, the total value by our trading strategy is higher than that by the benchmark strategy. In another word, our trading strategy not only beats the IBM stock market at the end of the year 2014, but also beats the market at most of the days in the year 2014

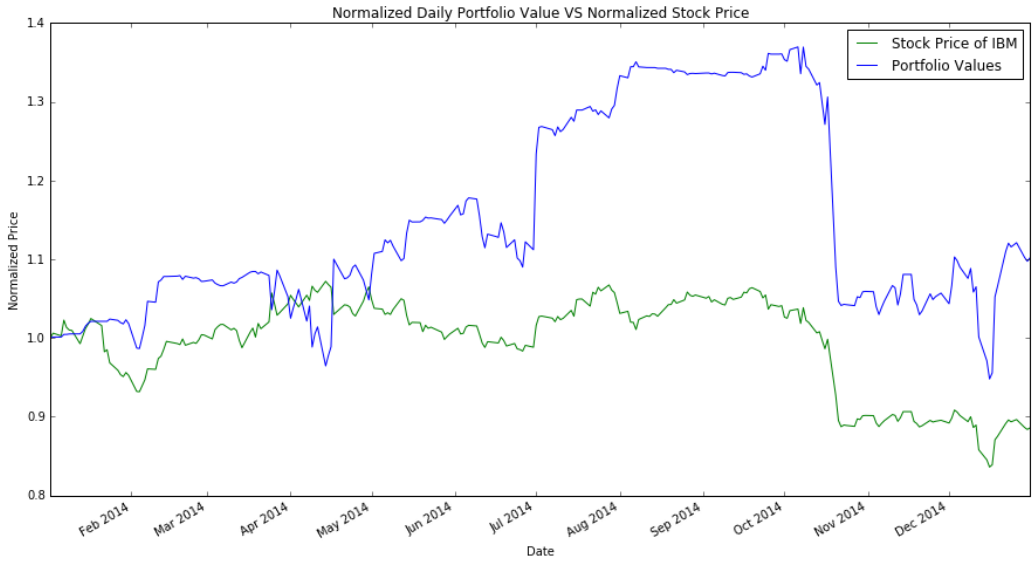

Figure 4.3: IBM: Normalized Portfolio Value vs Normalized Stock Price

Chapter 5

Conclusion

In this project, we applied various regression models to construct a stock price predictor. Then, a trading strategy is constructed based on the predictions given by the stock price predictor. Both the predictor and the trading strategy are tested by 3 case studies which are Google, Apple and IBM trading data respectively.

In each study, we built 3 different predictors using the prediction interval as 5, 10 and 20 trading days respectively. So, all the 3 studies produce 9 stock price predictors based on 9 different data sets. Each predictor selects the best model from KNN, Bagging, Decision Tree and Random Forest models. According to the result, 5 of the 9 predictors use Bagging, 3 of them use KNN and 1 of them uses Random Forest as their final regression models respectively. In all the 3 studies, the predictor of prediction interval of 5 trading days performs the best on the validation set among different choices of prediction interval. When the prediction interval is 5 trading days, the stock price predictor performs consistently well on the testing set. In the

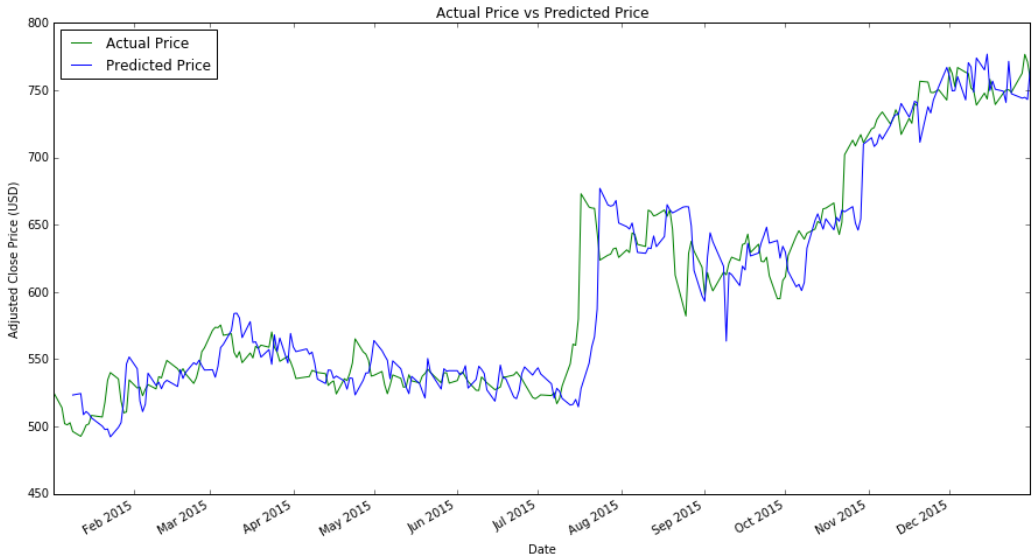

Figure 5.1: Google: Actual Price vs Predicted Price

studies of Google and IBM stocks, both MAPEs on testing data are below 3%. In the study of Apple Stock, the MAPE on testing data is slightly above 6%. Figure 5.1, Figure 5.2 and Figure 5.3 show the comparison between the predictions and the actual prices during the testing date range in the studies of Google, Apple and IBM respectively. From the figures, we can see that the predictions given by our stock price predictors captured the same trend of the actual prices in all the 3 case studies. In summary, we tested our predictor on 3 different stocks at different date ranges, and the result shows that the predictor we built is reliable for constructing a trading strategy.

There are two aspects that I found interesting about the predictors we built. Fistly, as described above, out of 9 predictors we built, 3 of them use KNN model as the final model, and 5 of them use Bagging model as the final model. Since our Bagging model uses KNN as the base model,

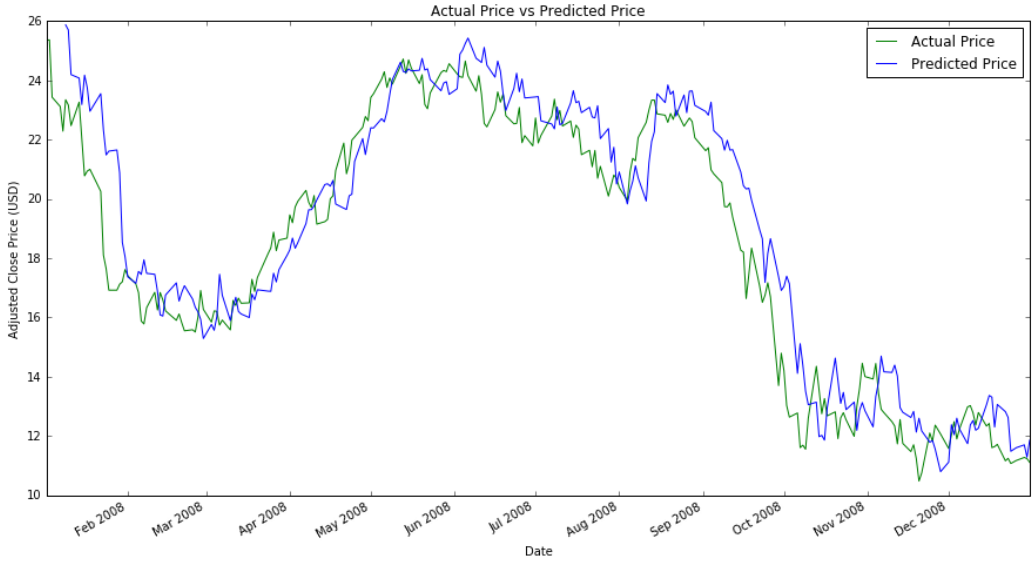

Figure 5.2: Apple: Actual Price vs Predicted Price

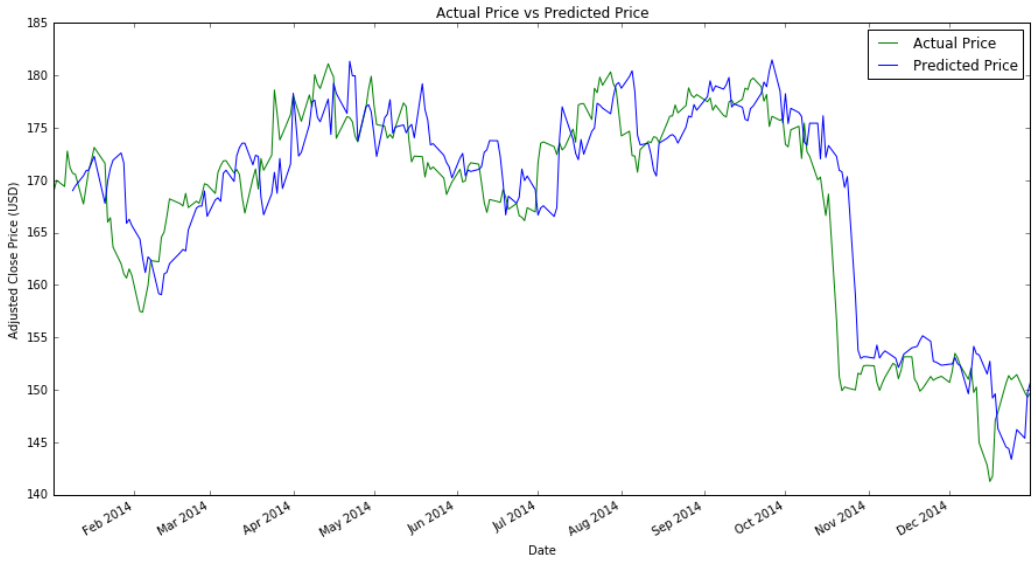

Figure 5.3: IBM: Actual Price vs Predicted Price

we can see that 8 of 9 predictors we built use KNN-related models as their final regression model. So, it is reasonable to say that KNN model is a good choice for applying in the field of Trading Stocks and Investment. The second aspect I found interesting is that in all the 3 studies, the predictor that uses 5 trading days as the prediction interval performs better than the predictor that uses either 10 or 20 trading days as the prediction interval. This fact might indicate that the stock price predictor built based on regression methods performs better in predicting shorter period's price.

In all the 3 studies, our trading strategy is better than the benchmark strategy in terms of cumulative return. In the studies of Google and IBM stocks, the profits our strategy made are much higher compared to the benchmark strategy. Also, our trading strategy generated higher positive Sharpe ratio in both studies indicating less trading risk. In the study of Apple, although the total value decreased by our strategy, but the total valued decreased more when using the benchmark strategy. Generally speaking, our trading strategy beats the stock market in all the 3 case studies.

Since we have tested our stock price predictor and corresponding trading strategy on 3 different stocks at different testing date ranges, and their performance are consistently good, it is reasonable to say that this project successfully applied machine learning methods to the filed of stock trading. However, there are still some points needed to be discussed. First of all, we found that all the 9 predictors we built in the project never used Decision Tree as the final model. So, it might mean that Decision Tree is not a very suitable method to be applied in the trading field in the way we did in this project.Secondly, for each regression method we applied in this project, we

didn't attempt too many choices for the tuning parameters because of the expensiveness of computation. Because of limited attempts, the optimal tuning parameter for each method might not be found. So, more attempts of tuning parameters might improve the results. Thirdly, there are many more regression models that can be applied in the project. Other models such as Generalized Least Square model and SVMs etc. can be attempted to see if there is improvement. Finally, the trading strategy might be adjusted by setting a different stop loss threshold. This might also improve the results.