Software Engineering II
Final Project
ECE 566, Spring 2014
Group # 9

Group members:

- Lakshmi Narasiman Veda Narayanan
- Shravan Sriram
- Priyanandhan Rajendhiran
- Prashanth Krishnakumar
- Sudarshan Kandi

Advised By:

Professor Shiyu Zhou

Table of Contents

Introduction

In this project we attempt to create a fully integrated application for a naïve user who aims to make a supplementary income by investing in the stock market. Stocks are a form of security that are not backed by any financial institution and are highly unpredictable if users take decisions without being fully informed of all the risks associated. Traditional banks that offer savings accounts, certificates of deposits etc. are usually FDIC insured and are able to guarantee a meager rate of return for the consumer; however, when it comes to securities, the returns can be quite substantial but they can end up being staggering losses as well. Precisely for this reason, our application aims to provide a stock trader some basic tools to help make some informed decisions about a particular security. Specifically, we allow the user to understand how the stock is doing on a short term as well as a long-term basis. Additionally, we provide some numerical metrics for the user so that he/she can make a decision to buy, sell, or hold a certain security either in the short term or long term. Numerical metrics that are considered in the application include, short-term prediction using machine learning & Bayesian estimation, RSI, MACD, and long-term shape prediction using support vector machine techniques (SVM). In the long-term prediction, the application detects for three specific cases where we believe the stock is at a pivotal point and a reaction from the user is warranted.

In this application, we use a MySQL database to store information, Python ™ to write scripts that pull and massage the stock data, and finally a PHP front end to make an interactive webpage for users to consume this information. In the following sections we will first define the functionality and features of the fully integrated application, followed by some discussion on the implementation, and finally we will show some results.

Functionality & Features

Prediction –

The prediction strategies used in this application are of two variants, short-term and long-term. In the short-term prediction, given the stock market can be extremely volatile, we employ multiple predictors and take a total vote to help the user make an informed decision. Namely, the Python ™ script predicts the future stock value using two different numerical approaches, followed by calculating the RSI (Relative Strength Index) and MACD (Moving Average Convergence Divergence) values respectively. No one metric has more value; the decision to buy/sell/hold is made based on a collective understanding of the security. The voting mechanism is further explored in the next section.

As for the long-term prediction, the only thing we afford the user is a simple, buy/sell/hold based on detecting long-term trends of securities. Specifically, we look for pivotal trends marking a huge turning point for the security. The exact specifics will be examined in the following sections.

With respect to the short-term predictions, the application provides a countdown timer that shows how long the prediction is valid for. By default, the application polls information every 60 seconds from the financial sources, therefore the prediction is only valid for 60 seconds, time taken to make the next value available. Given that 60 seconds is not a lot of time, without a specific visual cue to the user, there is no way for the application to communicate when the said prediction is valid till. To service this specific use case, there is a built-in countdown timer that shows the user how long a particular prediction is valid for

As for long-term predictions, their validity is on a stock-market day basis. In other words, the Python script pulls value at 8PM each night that shows the close of day status of the said security and adds it to the long-term table of the database.

User specific stocks & security –

The application affords the user to have a user specific login and be able to save the stocks he/she wishes to look at in different sessions. At start up, the user will have to first register with the application before gaining access to the application. The login is mandatory otherwise no portion of the website is accessible to the user.

Additionally, since this application is interfaced via PHP and on the back end interfaces with an SQL database, the team has worked painstakingly to ensure that proper countermeasures are present to prevent an SQL injection attack. An SQL injection attack usually corresponds to users trying to maliciously influence the back end DBMS by sending erroneous input values from the application front end. Since the user login information is stored in a database, using the login form allows malicious users to inject harmful queries to the database; by eliminating the use of special characters, we have mitigated this issue.

Ease of use –

The application makes no assumptions that users be aware of stock tickers; with close to 2400 tickers, knowing which company to look for makes the investment process even more daunting. Our application understands this issue and has implemented the auto complete feature where the user only needs to enter the first few letters of the ticker, which are usually the first few letters of the company name.

Even though the databases being used come pre-loaded with 15 companies, if the user specifies a ticker he/she wants to look at, the ticker is instantly added to the list of tickers being polled by the Python script. Although real time information starts flowing in minute-by-minute, we make the long-term data available to the user instantly. Using the real time information for prediction will not be available for at least 30 minutes as that's how long of a historical reference our application needs in order to make a useful prediction.

Hot stocks & other analysis –

Given only 15 ticker symbols that are part of our algorithm, we significantly restrict the user's field of vision on the entire stock market. Specifically to address this issue, our application makes a special attempt to look for the hot stocks by polling CNN Money, a popular financial analysis site. Based on the information provided through CNN, the user is able to lookup the companies that are in the market and are doing well (or not so well), and is able to lookup the ticker information through the autocomplete feature. Consequently, this ticker value is added to the user's profile such that short & long term information will be available for the user in the future.

In addition to looking up new stock tickers, the users also have access to an analytics section, where they can query different tables in the database. These queries include getting highest, average, lowest stock prices for the past year for a particular company, or comparing companies against one another, or listing the entire list of stocks contained in the database.

System Architecture

Figure 1 shows the basic system architecture for this application. As mentioned previously, Python has primarily been the scripting language of choice to retrieve and massage the data. Additionally, the data storage has been entirely done in MySQL so as to have the best storage and retrieval capabilities for large data sets. Finally, to have a good user experience, the front end has been crafted in PHP allowing for a variety of front-end additions (e.g. prediction countdown timer).

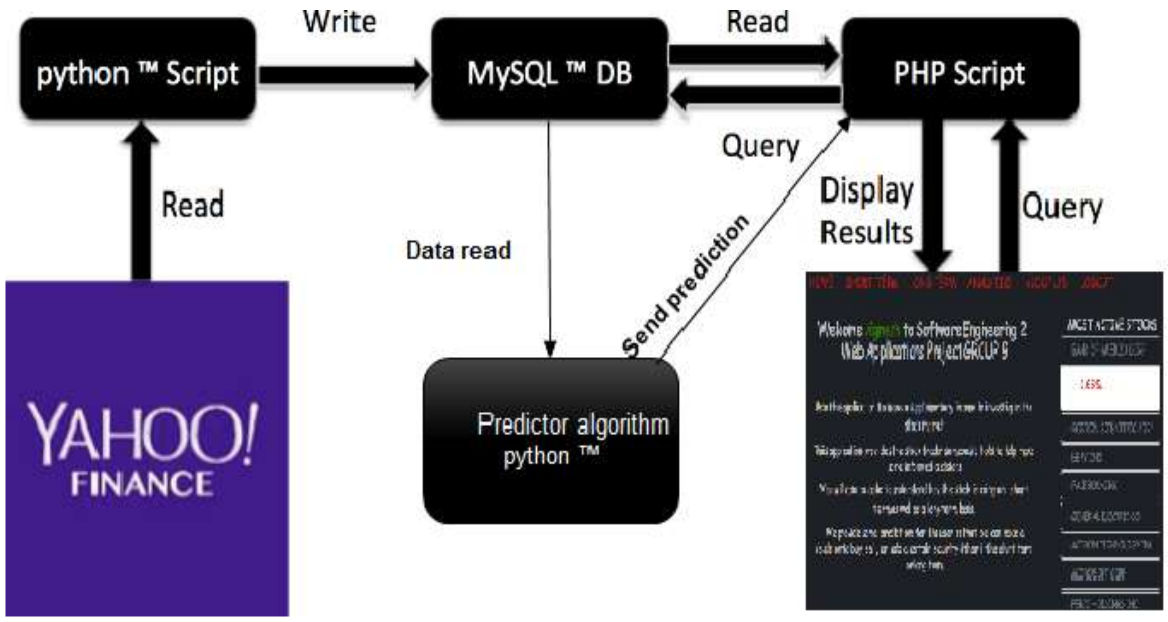


Figure 1: System architecture for the integrated application.

UML Diagram

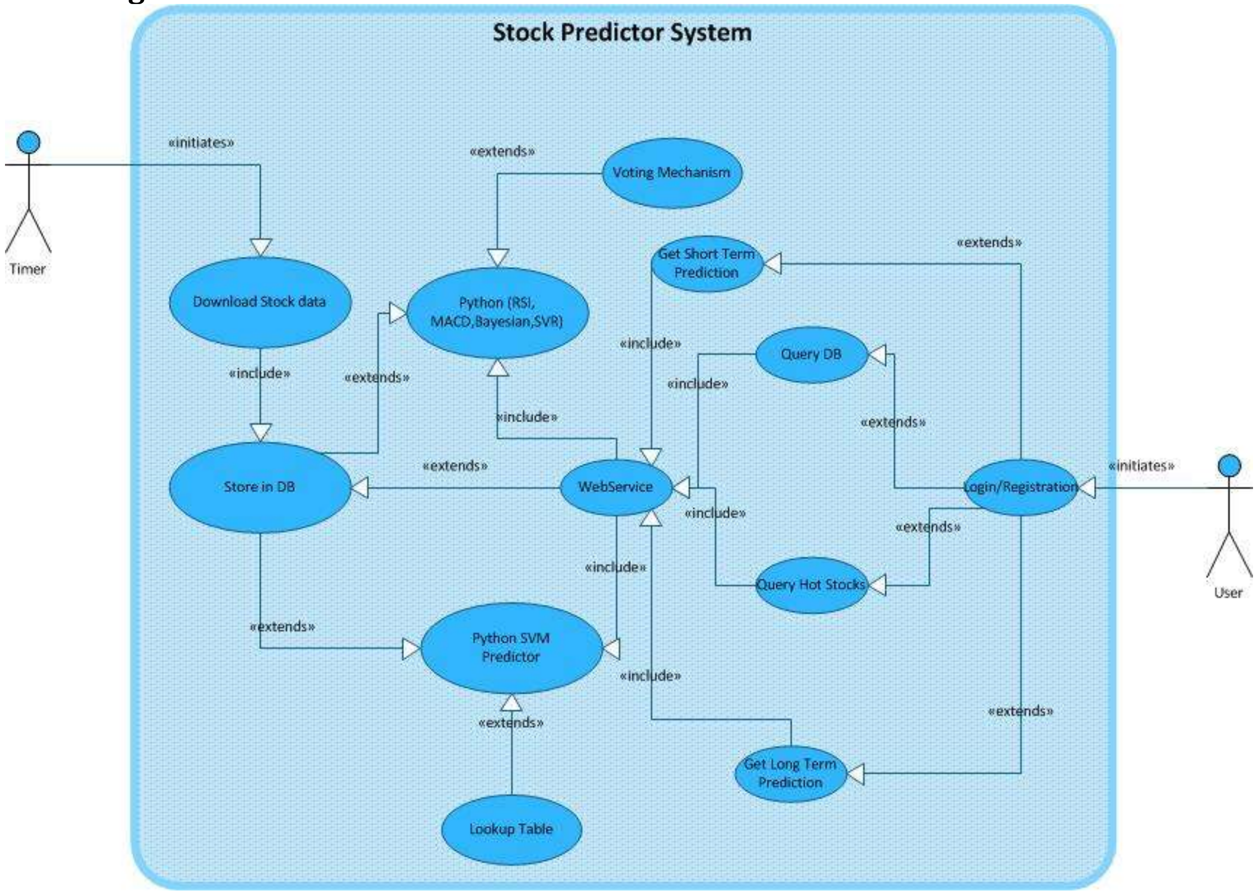


Figure 2: UML Diagram for the entire system.

Figure 2 shows the UML diagram for the application. The application has two actors, the user and a timer. The user initiates the login use case, which can be extended by four other use cases to assist with the goals of the application. The user is ultimately able to request predictions on a short-term as well as a long-term, query hot stocks obtained directly from CNN Money, and finally query the database for stored value and get specific stock information as mandated by the project requirements.

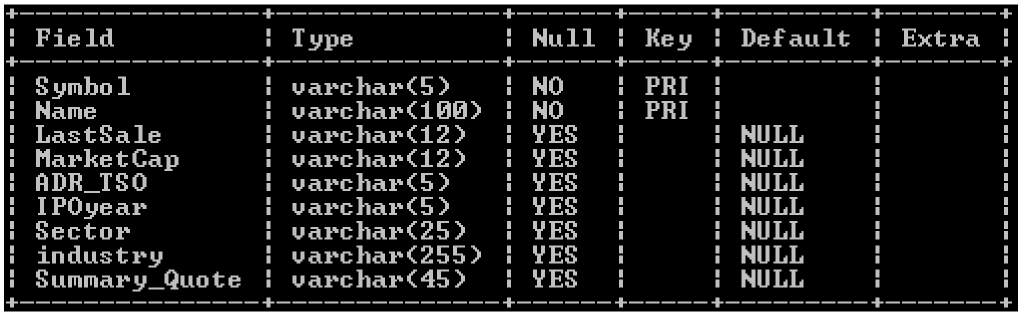
The predictor use cases (both short-term and long-term) make use of extensions that allow for them to produce an abstracted decision to the user; namely, the short-term predictor computes four separate variables and polls a decision-making entity to understand what the four variables are suggesting. Similarly, the long-term decision-making engine, polls a lookup table that correlates the shapes detected using SVM to proper recommendations sent to the user.

The timer actor initiates actions periodically compared to the user whose actions are quite transaction based. The timer actor forces the python ™ script to poll the relevant financial sources (Yahoo!) to get and store the stock information (short & long term) in the MySQL database.

MySQL Database –

This entire application consists of multiple tables that facilitate the variety of functionality specified above. The database used by this application has 6 tables in all:

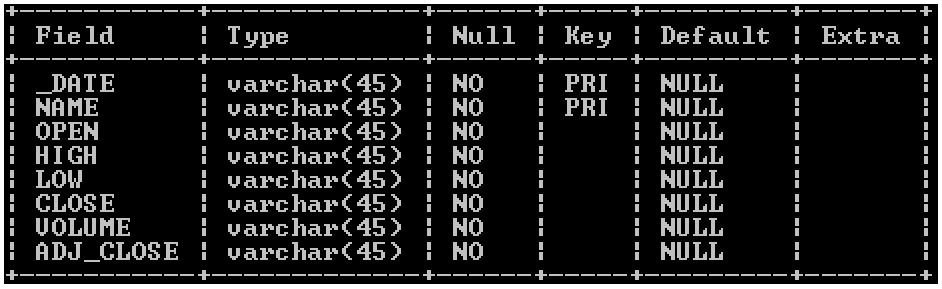
1. COMPANYLIST – Maintains a list of ticker symbols and company names. Table is specifically used to facilitate the autocomplete feature for the user. The table consists of approximately 2400 ticker symbols that allows the user to look up almost any company that is publicly traded. Figure 3 shows the schema for this table.



```
+---------------+---------------+------+-----+---------+-------+
| Field         | Type          | Null | Key | Default | Extra |
+---------------+---------------+------+-----+---------+-------+
| Symbol        | varchar(5)    | NO   | PRI |         |       |
| Name          | varchar(100)  | NO   | PRI |         |       |
| LastSale      | varchar(12)   | YES  |     | NULL    |       |
| MarketCap     | varchar(12)   | YES  |     | NULL    |       |
| ADR_TSO       | varchar(5)    | YES  |     | NULL    |       |
| IPOyear       | varchar(5)    | YES  |     | NULL    |       |
| Sector        | varchar(25)   | YES  |     | NULL    |       |
| industry      | varchar(255)  | YES  |     | NULL    |       |
| Summary_Quote | varchar(45)   | YES  |     | NULL    |       |
+---------------+---------------+------+-----+---------+-------+
```

Figure 3: Companylist table schema with all attributes and primary keys.

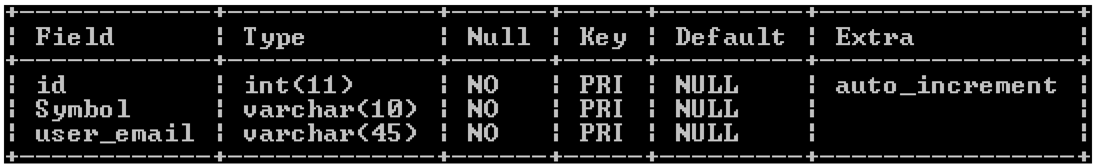
2. HIST_STOCK – This table consists primarily of historical stock information for all the tickers the user asks for and/or is present in the Python™ script by default. Attributes include Date, Name, Open value, High value, Low value, Close value, volume traded, and Adj_close values. In this table, the date and the name collectively are the primary keys as they uniquely identify any particular entry in the table. Figure 4 shows the table schema with all the relevant information.



```
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| _DATE     | varchar(45) | NO   | PRI | NULL    |       |
| NAME      | varchar(45) | NO   | PRI | NULL    |       |
| OPEN      | varchar(45) | NO   |     | NULL    |       |
| HIGH      | varchar(45) | NO   |     | NULL    |       |
| LOW       | varchar(45) | NO   |     | NULL    |       |
| CLOSE     | varchar(45) | NO   |     | NULL    |       |
| VOLUME    | varchar(45) | NO   |     | NULL    |       |
| ADJ_CLOSE | varchar(45) | NO   |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
```

Figure 4: HIST_STOCK table schema with all attributes and primary keys.

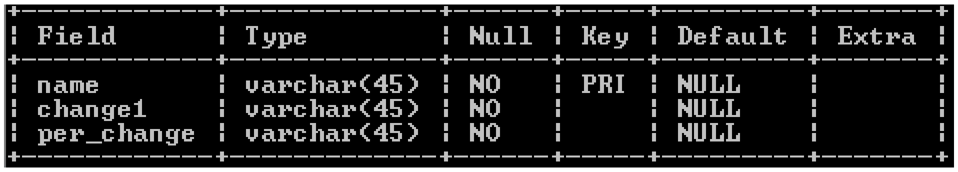
3. HISTORY – This table is used specifically to track stocks that are of interest to the users accessing the application. This table acts as a history for each user and upon their return, makes data from last seen stocks available to them assuming high interest on the user's part. The table has three attributes and all three of them together uniquely define an entry. The attributes are: row index, ticker symbol, and the user's e-mail. Figure 5 shows the table and all relevant information.



```
+------------+-------------+------+-----+---------+----------------+
| Field      | Type        | Null | Key | Default | Extra          |
+------------+-------------+------+-----+---------+----------------+
| id         | int(11)     | NO   | PRI | NULL    | auto_increment |
| Symbol     | varchar(10) | NO   | PRI | NULL    |                |
| user_email | varchar(45) | NO   | PRI | NULL    |                |
+------------+-------------+------+-----+---------+----------------+
```

Figure 5: HISTORY table schema with all attributes and primary keys.

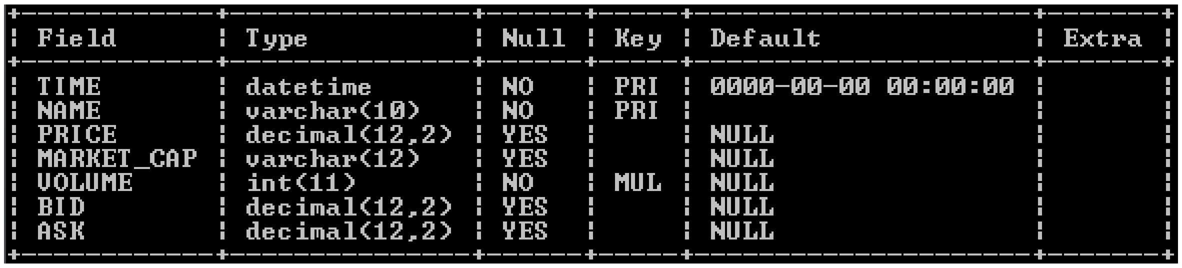
4. HOT_STOCK – This particular table allows the application to maintain a list of hot stocks as presented by CNN MONEY. The table contains three attributes: name, change, and percentage change where only the "name" is sufficient to identify any row uniquely. Figure 6 shows all the relevant information for this table.



```
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| name       | varchar(45) | NO   | PRI | NULL    |       |
| change1    | varchar(45) | NO   |     | NULL    |       |
| per_change | varchar(45) | NO   |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
```

Figure 6: HOT_STOCK table with all attributes and primary keys.

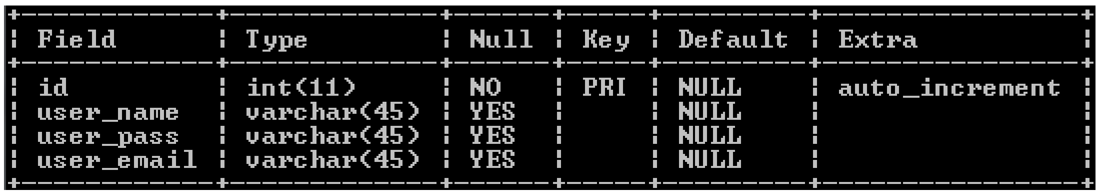
5. STOCKS – This table is used to maintain the real time data for each stock ticker. The table consists of 7 attributes including time, name, price, etc. where only time and name are used to uniquely identify any given row. This table is continuously written to every minute so that up-to-date stock information is present for all the tickers the user requested for in addition to all the predefined tickers in the Python™ script. Figure 7 shows all the relevant information for this table.



```
+------------+---------------+------+-----+---------------------+-------+
| Field      | Type          | Null | Key | Default             | Extra |
+------------+---------------+------+-----+---------------------+-------+
| TIME       | datetime      | NO   | PRI | 0000-00-00 00:00:00 |       |
| NAME       | varchar(10)   | NO   | PRI |                     |       |
| PRICE      | decimal(12,2) | YES  |     | NULL                |       |
| MARKET_CAP | varchar(12)   | YES  |     | NULL                |       |
| VOLUME     | int(11)       | NO   | MUL | NULL                |       |
| BID        | decimal(12,2) | YES  |     | NULL                |       |
| ASK        | decimal(12,2) | YES  |     | NULL                |       |
+------------+---------------+------+-----+---------------------+-------+
```

Figure 7: STOCKS table with all attributes and primary keys.

6. USERS – This table is specifically used to maintain a list of users and their login credentials. The table has 4 attributes: ID, username, password, and e-mail. Only the ID is used as a primary key. Figure 8 shows all the relevant information for this table.



```
+-------------+-------------+------+-----+---------+----------------+
| Field       | Type        | Null | Key | Default | Extra          |
+-------------+-------------+------+-----+---------+----------------+
| id          | int(11)     | NO   | PRI | NULL    | auto_increment |
| user_name   | varchar(45) | YES  |     | NULL    |                |
| user_pass   | varchar(45) | YES  |     | NULL    |                |
| user_email  | varchar(45) | YES  |     | NULL    |                |
+-------------+-------------+------+-----+---------+----------------+
```

Figure 8: USERS table with all attributes and primary keys.

Python Script –

The python script in addition to pulling information from financial sites also performs the prediction strategies. In the prediction strategies, we make use of a variety of mathematical prediction methods as well as derive technical analysis metrics as commonly used by investors. Table 1 summarizes all the prediction methods in play in this application.

Prediction Strategy	Term	Input	Output
Bayesian Prediction	Short Term	Stock training set values	Predicted stock value
Support Vector Regression	Short Term	Stock training set values	Predicted stock value
RSI	Short Term	Past stock values	RSI value
MACD	Short Term	Past stock values	MACD value
Support Vector Machine	Long Term	Historical data	Closest shape match

Table 1: Summary of prediction strategies.

Bayesian Prediction –

To implement Bayesian prediction, we make use of Bayesian curve fitting according to which, the training data **x** and **t**, along with a new test point x are provided. The end goal is used to predict the value of t, where x is the future time and t is the value taken on by the function at time x. The variables **x** and **t** signify two vectors constituting the training set for the prediction algorithm. Stated a different way, the problem becomes computing the distribution p(t|x, **x**, **t**). The expression is looking for the probability distribution of t (future value) given the future time, past times, and past values (Marsic, 2012).

Using the product rules of probability, the required distribution can be rewritten as follows:

$$p(t|x, \mathbf{x}, \mathbf{t}) = \int p(t|x, \boldsymbol{w}) p(w|\boldsymbol{x}, \boldsymbol{t})\, dw \qquad (1)$$

Here, **w** refers to a vector of consisting of different weights for the nth order polynomial that is being used to fit the data points.

Additionally, $p(t|x, \boldsymbol{w})$ is given by,

$$p(t|x, \boldsymbol{w}, \beta) = N(t|y(x, \mathbf{w}), \beta^{-1}) \tag{2}$$

where β is just the inverse of the variance for the normally distributed random variable.

The integration can also be performed analytically with the result that the predictive distribution is given by a Gaussian of the form

$$p(t|x, \mathbf{x}, \mathbf{t}) = N(t|m(x), s^2(x)) \tag{3}$$

Where the mean and variance are given by

$$m(x) = \beta\varphi(x)^T\mathbf{S}\textstyle\sum_{n=1}^{N} \emptyset(x_n)t_n \tag{4}$$

$$s^2(x) = \beta^{-1} + \varphi(x)^T\mathbf{S}\emptyset(x) \tag{5}$$

Here the matrix S is given by

$$\mathbf{S}^{-1} = \alpha I + \beta\textstyle\sum_{n=1}^{N} \emptyset(x_n)\emptyset^T \tag{6}$$

where I is the unit matrix, and we have defined the vector φ(x) with elements
$\emptyset_i(x) = x^i$ for i = 0, . . . , M.

In the Bayesian predictor implementation, the team has employed the use of a 6th order polynomial, which provides rather satisfactory predictions. The results section outlines the results further. Additionally, for this implementation, β was chosen to be 11.1 while α = 0.005 (Marsic, 2012).

SVR and SVM Prediction –

SVR and SVM stem from an area of machine learning known as supervised machine learning. The principle here is that data can be inferred based on past data points used as learning data sets. In other words, by giving the algorithm a sense of what to expect, it is able to come up with decision functions that make future classifications feasible without any human intervention. SVM or Support Vector Machine is one such machine-learning algorithm where the most basic construct is to come up with some sort of a decision that allows the algorithm to either classify the new inputs in certain categories or create a regression trend line around the input values.

In a two dimensional space, assuming there are only two types of data to be classified, classification can be rather simple if there is a clear distinction between the data sets. The decision boundary is what is known as a hyperplane (has one less dimension than the space). The concept of SVM is that the hyperplane should be chosen in such a way that it has the maximum feasible distance from the closest data point of either data set. Only with the maximum margin can the algorithm ensure minimum errors of classification.

Assume a scenario of two distinct data sets that can be clearly cut through by a maximum-marginal hyperplane, a vector **w** is defined as one being orthogonal to this hyperplane. Additionally, assume that an input vector **u** needs to be classified as either belonging to class 0 or class 1 (traditional classification problem).

By constraining the vector **u**_to be either on one side of the hyperplane or the other, it is easy to distinguish if **u** belongs to class 0 or class 1.
By taking the projection of the vector **u** on to **w** allows us to see how far from the hyperplane the new vector lies. However, given that the vector **w** itself is an unknown, it poses a significant ordeal to figure out the classification for **u** (Winston, 2013).

With this in mind, we constrain bounds on **w** as follows:

$$\begin{gathered} for\ class 1\colon \boldsymbol{w}.\boldsymbol{u_1} + b\ \geq 1 \\ for\ class\ 0\colon \boldsymbol{w}.\boldsymbol{u_0} + n\ \leq\ -1 \end{gathered} \tag{7}$$

where **u₁** and **u₀** refer to input vectors of the subscript class. Since the margin has to be maximal between the two data sets, we can make the conclusion that at least one of the data points of each case lie on the margin. Therefore the distance between the two margins is just the difference of the above equations:

$$\begin{gathered} \text{margin} = \mathbf{w.\ (u_1 - u_0)} = 2 \\ \text{margin } = \frac{2}{||\boldsymbol{w}||} \end{gathered} \tag{8}$$

In order to maximize this margin, we have to minimize the denominator, which is indeed the vector that is orthogonal to the hyperplane that was selected to segregate the two classes.

The alternate form of the margin equation can be restated as $\frac{1}{2}\left||\boldsymbol{w}|\right|^{\mathbf{2}}$ (9)

Using indicator variables above to designate a case 0 or a case 1 condition, we may also re-write the two equations from (1) as:

$$y_i\ (\boldsymbol{w}\ .\boldsymbol{u_i} + \boldsymbol{b}) \geq \mathbf{1} \tag{10}$$

$$here\ y_i\ is \pm 1\ depending\ on\ the\ case\ of\ the\ input\ vector.$$

At this point there are too many unknowns and not enough known parameters. In order to maximize/minimize **w**, mathematicians suggest using LaGrange multipliers leaving the above equations in the following form:

$$maximize\ \sum_i^l a_i - \frac{1}{2}\sum_{i,j=1}^l a_i a_j y_i y_j \boldsymbol{x_i}.\boldsymbol{x_j}\ \ where\ a_i the\ lagrange\ multipliers. \quad (11)$$

Consequently, this yields a very interesting notion that the support vector **w** relies solely on the dot product of the input data vectors and a few constants. By calculating the support vectors for a set of input vectors, we can certainly determine the constant b using a set of equations and consequently use the decision metrics defined above (Winston, 2013).

Clearly, the linear case is rather trivial to solve, but when the sample points are not trivial to demarcate the problem is not all that complicated either. Mathematicians suggest using kernels that take the dot product of the input vectors from the space they are non-linearly aligned in and transform them to a higher space. Instead of using a transformation matrix, they simply use a kernel that does the transformation.

In this algorithm, we employ SVM as the long-term predictor. Since stock values are highly erratic at times, there is almost never a clear demarcation in the Euclidean space. For this reason we employ the radial basis function (RBF) kernel, which uses an exponential function and maps, the input vectors to a higher dimension (Winston, 2013).

Similarly for SVR, we use the RBF kernel, which allows us to get past the non-linear demarcation issue. SVR, a regression technique relies solely on a subset of training data.

The equation to solve for an SV regression analysis is (Yang, Huang, King, & Lyu, 2008): (12)

$$f(x) = \boldsymbol{w^T x} + b$$

where **w** is again the same vector as before, **x** is a vector of training data, and b is a constant. Given this equation form, the task is to determine the vector **w** and the constant b while minimizing the regression risk function.

$$R_{Reg}(f) = ||\mathbf{w}|| + C\sum_{i-1}^{N} \daleth(f(x_i - y_i) \quad (13)$$

$$\text{where, } \daleth(f(x_i - y_i) = \begin{cases} 0\ if\ |y - f(x)| < \in \\ |y - f(x)| \quad otherwise \end{cases}$$

In this function, when the data points are in the range of $\pm\in$, they do not contribute to the empirical error (Basak, Pal, & Patranabis, 2007).

In this implementation, $\in$, the error threshold was set at 0.1. Further, in order to make a prediction, once we extrapolate the actual curve using the regression, we use the last two data points to make a linear interpolation for the next possible stock value. This precise value is being used as the SVR assisted prediction in the short-term realm.

RSI –

Relative Strength Index is a technical analysis tool that assists investors and financial analysts make decisions about a fund. The principle behind RSI is to observe the ratio of up closes of the security to the down closes. In other words, as the security is traded more, the RSI tends to swing to one of the extremes on the percentage scale signaling a bearish or bullish market (Investopedia).

It is important to preface the discussion that RSI by itself can be highly misleading given that it only looks at frequency of events. It is best if it is used in conjunction with another metric like MACD. RSI is calculated by the following equations:

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

$$RS = \frac{Average\ of\ x\ days'up\ closes}{Average\ of\ x\ days'down\ closes} \tag{14}$$

A higher RSI value indicates that the fund has been overbought driving the price up, while a lower RSI value indicates that it has been oversold. As a rule of thumb, a consistent RSI of 70 or higher is indicative of a fund being over bought. If the fund consistently shows a RSI value of 70 or higher, investors are likely to sell, while if the RSI is consistently under 30, investors are likely to buy the said security (Investopedia). Figure 9 shows the RSI for Google Inc. from Oct 2013 – Feb 2014. In the beginning of October, the RSI was well under 30, causing the investors to purchase this security. It is almost improbable to state that the RSI was the cause of increased share activity. Instead it could be the other way around. It is quite evident that the contour of the RSI is like that of the share price, however the causation isn't quite proportional. For instance, looking at the share price from Nov 1 – Nov 7, the stock price trended down with a slight negative slope. However, given that the stock was trending down (even slightly), the RSI plunged from 85 to 36. The volume of shares traded was not significant, however anyone looking at just the RSI would suggest purchasing after Nov 7. As prefaced previously, the RSI should be taken as a secondary technical metric and not a primary as it could be severely misleading without context.

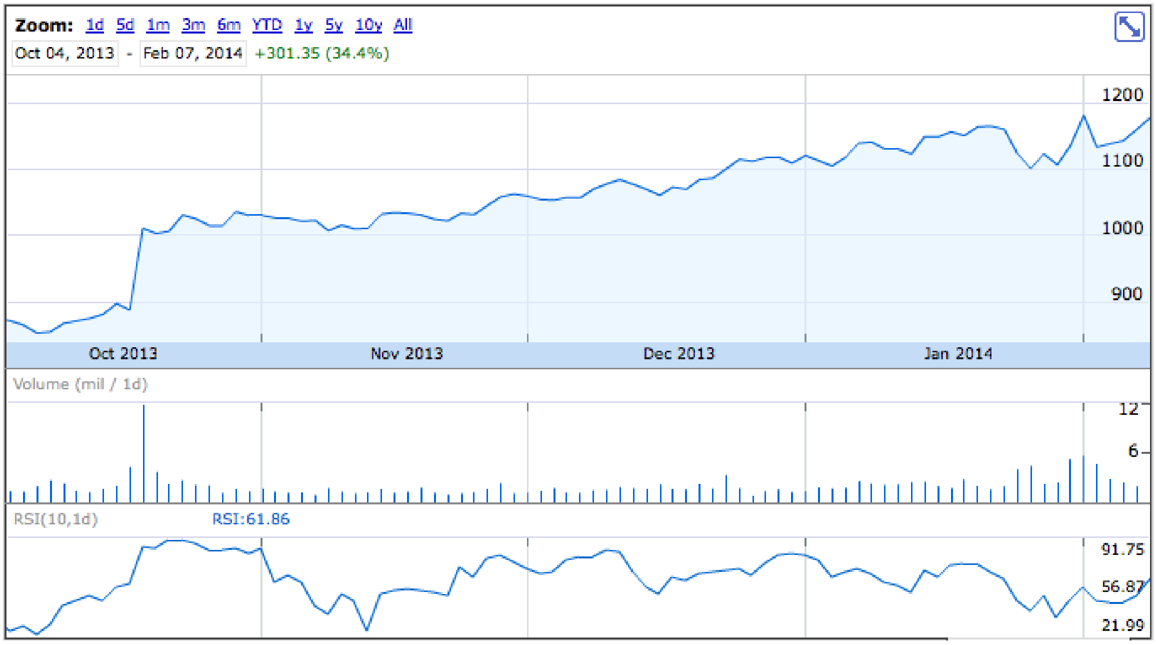


Figure 9: Stock for Google Inc (2014-2014) showing the RSI metric plotted along with the volume of shares traded. Clearly, volatility in the RSI value is disproportional to the volatility in the stock price (Google Finance).

An interesting thing to note about RSI is that the first value computed requires some historical data of the security (~15 days), but every other value after that becomes part of a moving average and does not require as many data points as such. After the first value of RS is calculated, every other value is computed using the following equations:

- Average Gain for n > 1 = [(previous average gain x 13 + current gain)/14]
- Average loss for n > 1 = [(previous average loss x 13 + current loss)/ 14]

Even though RSI by itself is not a self-sufficient metric, we are using it in conjunction to 3 other metrics to make a short-term prediction.

MACD –

MACD is yet another technical analysis concept where a security is judged as a "buy" or a "sell" based on how the fund performs relative to its nine-day exponential moving average. Particularly, the difference between the 12-day EMA and the 26-day EMA is compared against the nine-day EMA. The nine-day EMA is also known as the signal while the MACD is the difference between the short term and long term moving averages. Ideally, if the short-term behavior is volatile compared to the long-term behavior (26 days), investors usually either buy/sell to compensate for this (Forexo).

For instance, if the MACD line goes above the signal line, it symbolizes a bull market and causes investors to purchase more securities. Conversely, if the MACD line goes below the signal, it symbolizes a bear market and causes investors to sell more securities. Finally, if the MACD line trends way too high and diverges too much from the signal, it signifies an over purchase of the security and the market reacts by selling to stabilize the price. Figure 10 shows the MACD plotted for Google Inc. from Mar 2008 – Dec 2009. The blue line in the bottom portion of the plot signifies the MACD while the red line is the signal. Focusing at the beginning of 2009, the MACD is higher than the signal, which correlates positively with the volume of shares traded and the value of the stock. As the investors purchased more stock, the demand went up causing the price of the stock to go up. Similarly. Looking at the chart prior to Jan 2009, the trend line is negative, the MACD is lower than the signal, the volume of shares traded are still high, but the stock price came down almost $300.00. In hindsight, the MACD certainly was a metric to purchase Google securities; however, it may not always be the case.

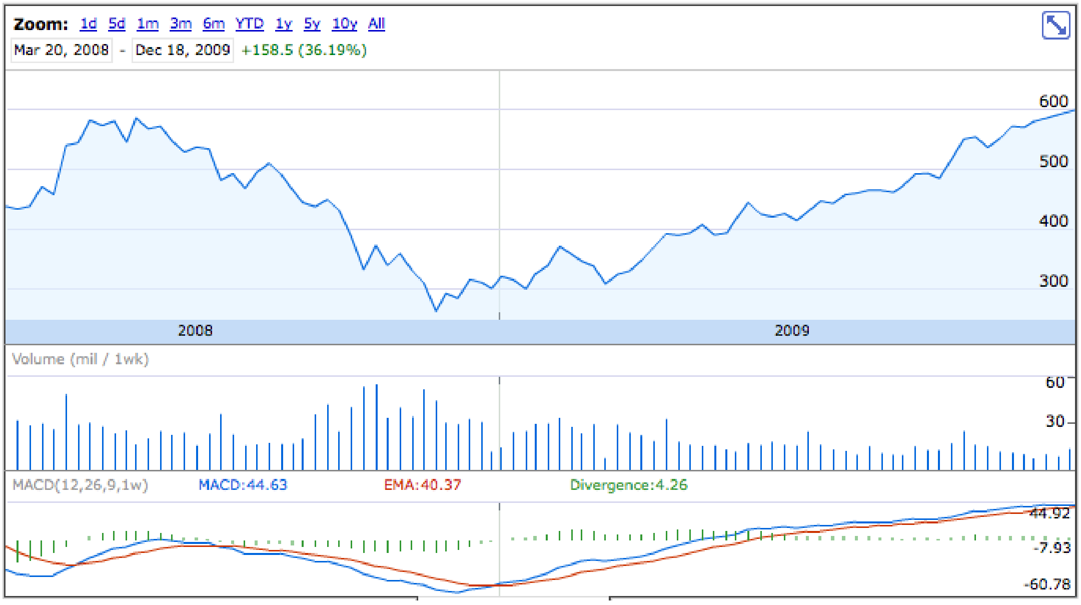


Figure 10: Stock price of Google Inc. showing the overlay of MACD and the volume of shares traded (Google Finance).

Short Term Prediction & Voting Mechanism –

In the prediction engine, we use two technical analysis trend metrics and two stock prediction methods to come up with four separate ways of making a recommendation to the user. Each one of these metrics takes in the stock value as the input, thus keeping a common basis. In the python™ code, we use the Urllib and re libraries to continuously poll the finance site for stock information. This information is written to the MySQL database every minute or at the end of the day for information gathering useful for long-term prediction.

The prediction engine has been designed such that it is only called upon user request. Once the user requests for a short-term prediction, the python™ code extracts data from the relevant tables of the database and starts crunching the numbers. It independently computes the RSI, MACD, Bayesian Prediction, and the SVR prediction and then calls a voting mechanism function.

The inputs to the voting mechanism function are as follows:

1. RSI value
2. MACD – Signal value
3. Bayesian Predicted Stock – Last known stock value
4. SVR Predicted Stock – Last known stock value

If the RSI was less than 30 then the application would consider it predicting a buy. If the (MACD – Signal Value) was positive then that would signify a buy as well. Consequently, the application considers a buy vote from the two stock predictors if the individual differences come out positive. If the differences come out positive, the indication is that the security will have a higher price at the expiration of the prediction time.

To recommend a "buy" to the user, the application requires at least 3 metrics to signal a buy. If only two signal a "buy" then the application recommends the user to "hold". Consequently, any less than 2 votes for "buy" results in a "sell" recommendation to the user. Since the confidence levels of any prediction vary in increments of 25%, the application does show this value to the user giving them an extra sense of assurance based on the four metrics defined.

Long-Term prediction –

In the long-term prediction arena, we do not employ any particular voting mechanism per say, rather we detect shapes. As described earlier, we use the SVM machine learning technique to look for specific patterns over the course of one year for any security. Specifically we are looking for pivotal points in the life of the security that allow us to make life-changing recommendations to the user. We detect, ascending triangles, descending triangles, and rounding bottom for the mere fact that these are three pivotal points in the life of any stock. They symbolize a turn for betterment or worse causing either significant profits or losses to the customers. Figure 11 provides a general overview of the process of performing the classification.

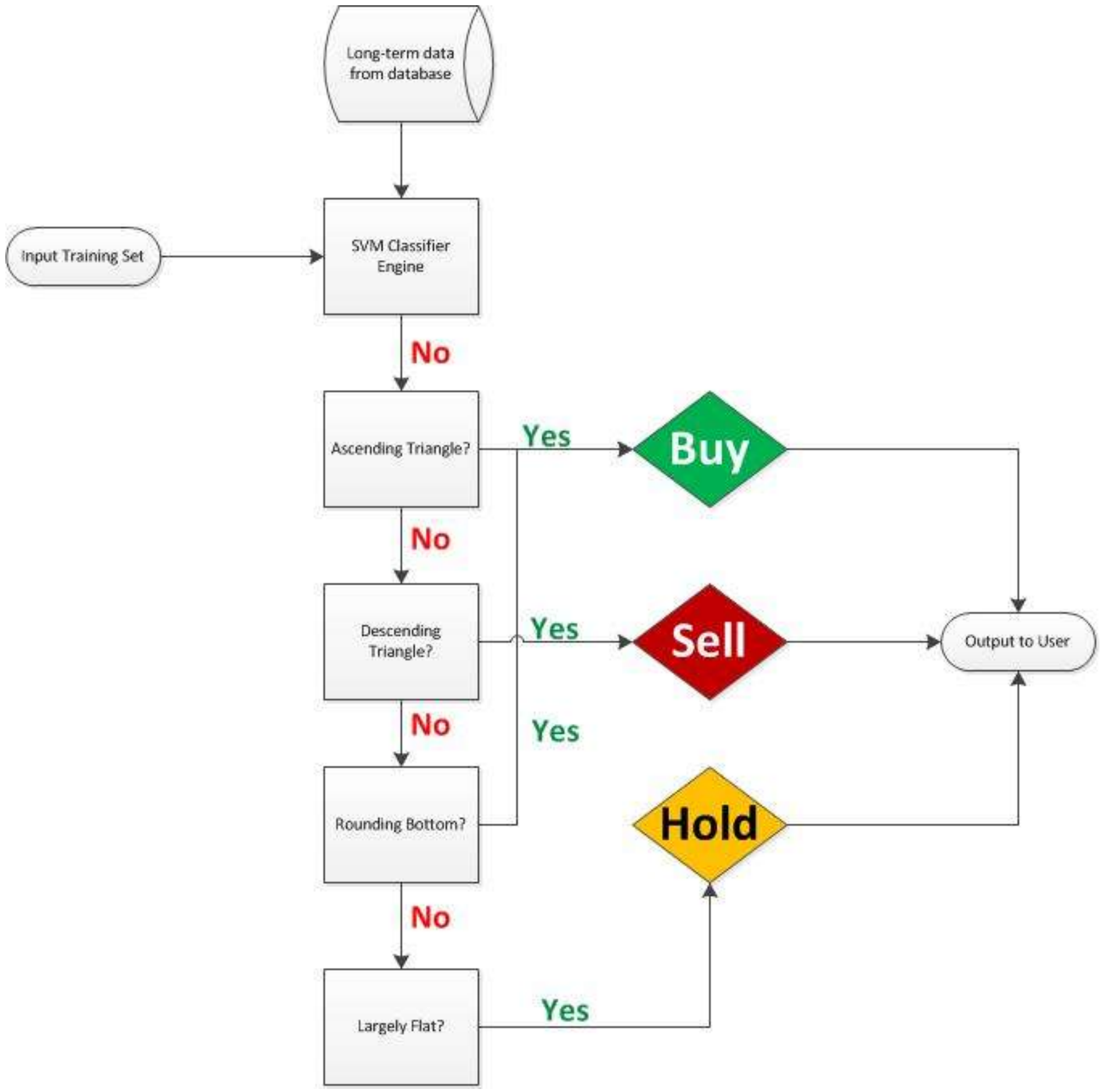


Figure 11: Provides the process flowchart of using the SVM algorithm for shape classification.

Ascending Triangle:

"Ascending triangle refers to a chart pattern in technical analysis that is easily recognizable by the distinct shape created by two trendlines. In an ascending triangle, one trendline is drawn horizontally at a level that has historically prevented the price from heading higher, while the second trendline connects a series of increasing troughs. Traders enter into long positions when the price of the asset breaks above the top resistance" (Investopedia). Figure 12 is an example of such an ascending triangle.

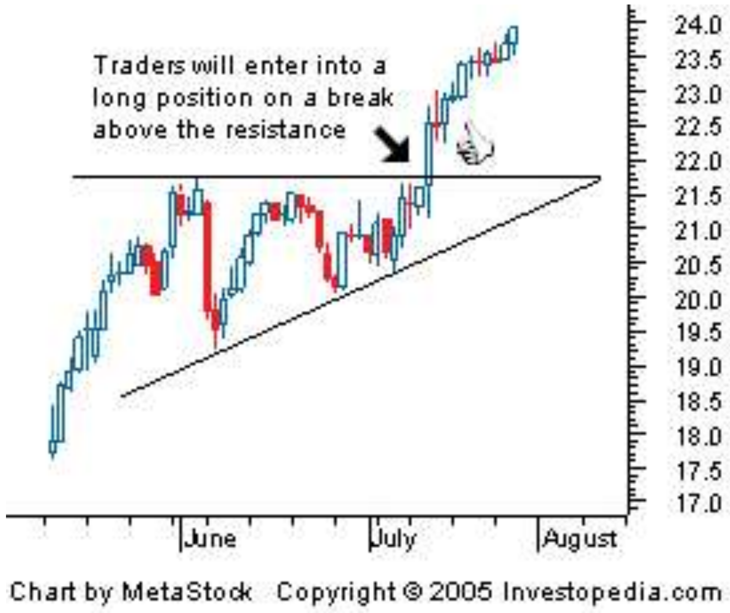


Figure 12: Chart showing ascending triangle (Investopedia).

Descending Triangle:

A descending triangle is a technical analysis tool "created by drawing one trendline that connects a series of lower highs and a second trendline that has historically proven to be a strong level of support. Traders watch for a move below support, as it suggests that downward momentum is building. Once the breakdown occurs, traders enter into a short positions and aggressively push the price of the asset lower" (Investopedia). Figure 13 shows an example of such phenomenon.

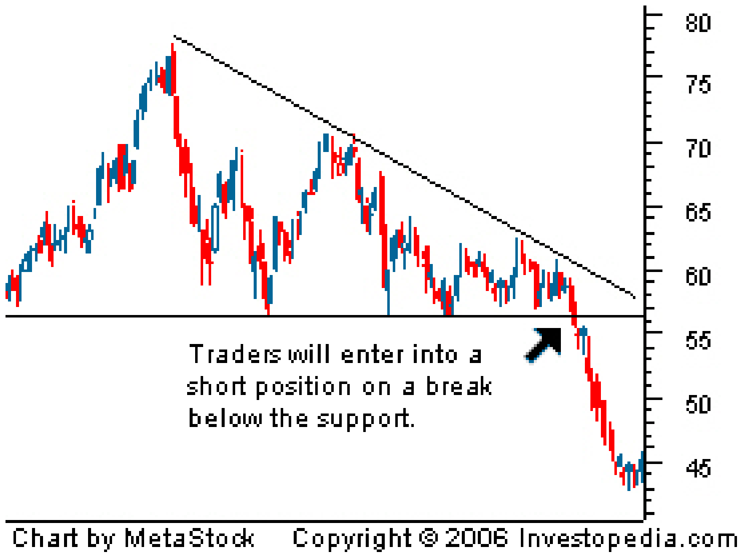


Figure 13: Chart showing descending triangle (Investopedia).

Rounding Bottom:

"A chart pattern used in technical analysis, which is identified by a series of price movements that, when graphed, form the shape of a "U". Rounding bottoms are found at the end of extended downward trends and signify a reversal in long-term price movements. This pattern's time frame can vary from several weeks to several months" (Investopedia). Figure 14 shows an example of rounding bottom technical analysis trend.

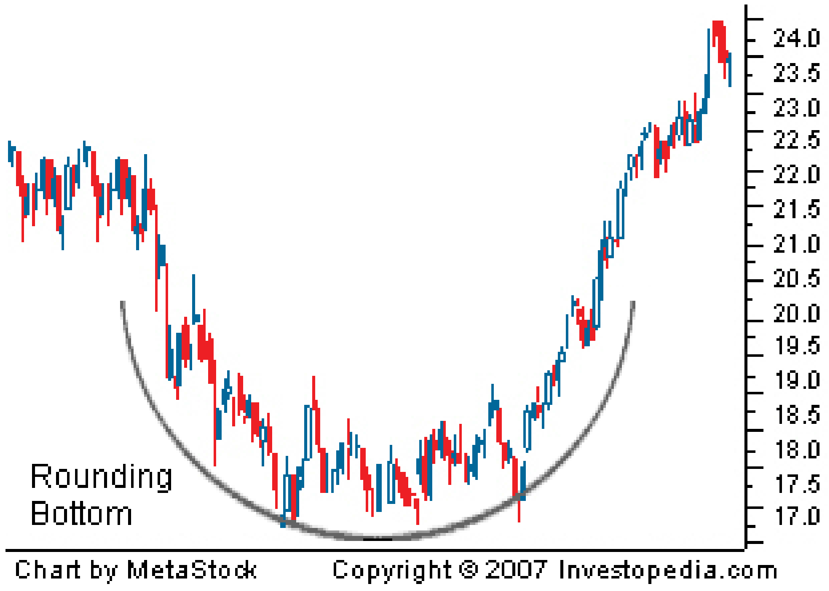
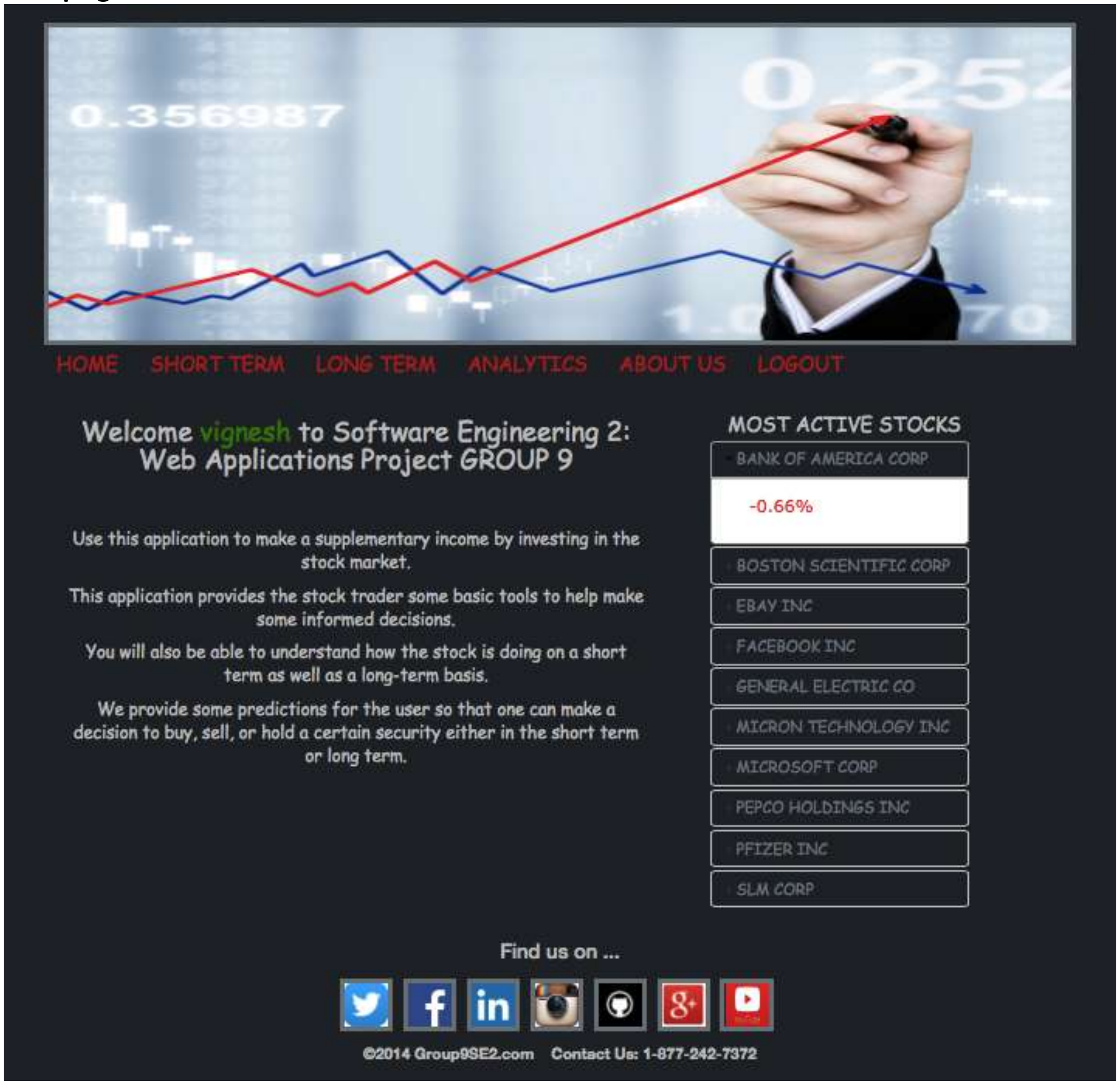


Figure 14: Chart showing rounding bottom.

PHP Front End –

The front end of the application, though written mainly in PHP, also has good mix of CSS, HTML, and JQuery. The first page is the registration page where the user enters his name, e-mail, and password. Once the user is registered and logs in, the use can access the application. The user can access the short-term predictions by selecting his/her stocks of choice. The user can also access the long-term predictions by entering his/her choice of stocks. The website also contains a special section where the user can do specialized queries to the MySQL database. The following figures show some examples of the front end.

Homepage



Welcome vignesh to Software Engineering 2: Web Applications Project GROUP 9

Use this application to make a supplementary income by investing in the stock market.

This application provides the stock trader some basic tools to help make some informed decisions.

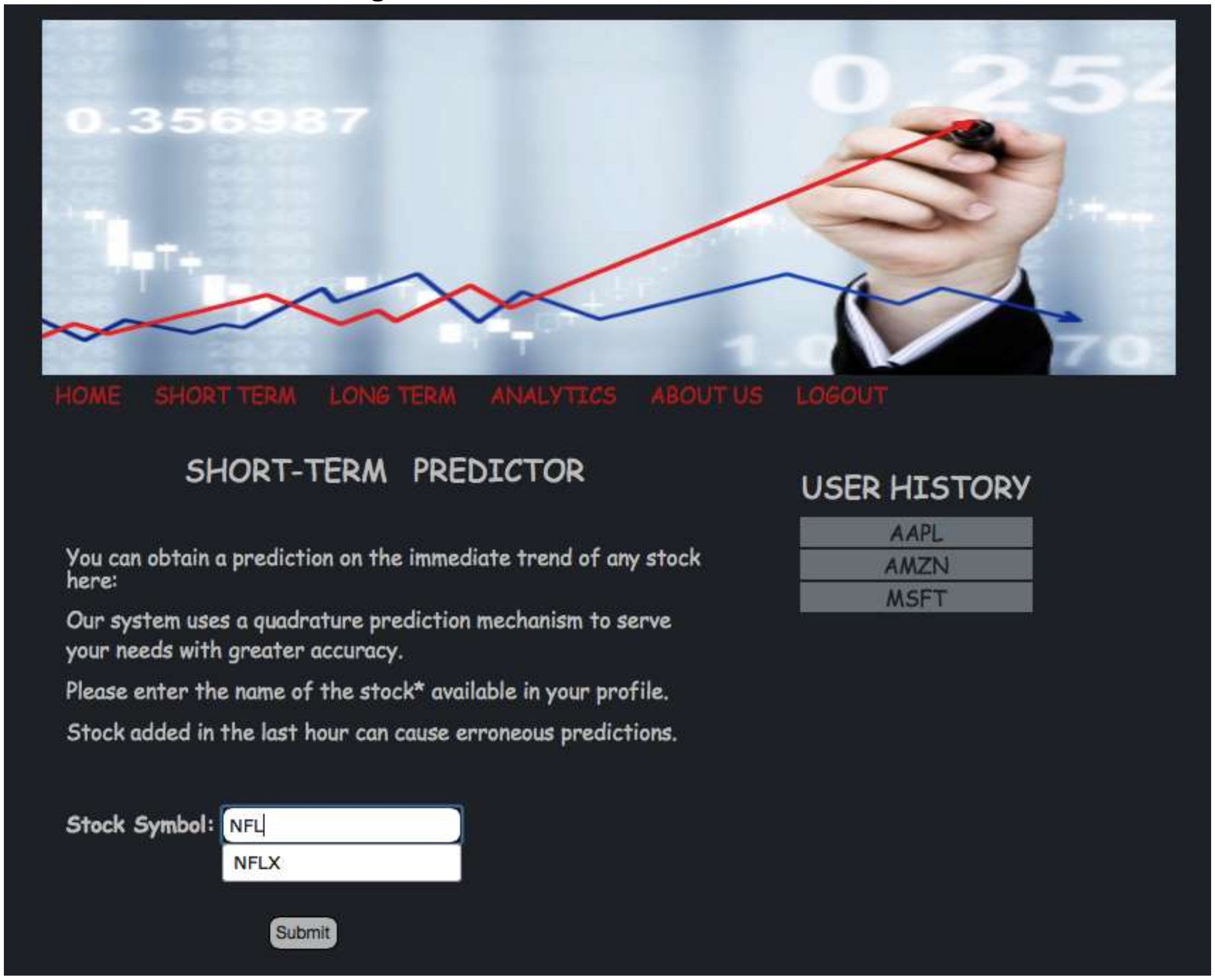
You will also be able to understand how the stock is doing on a short term as well as a long-term basis.

We provide some predictions for the user so that one can make a decision to buy, sell, or hold a certain security either in the short term or long term.

©2014 Group9SE2.com Contact Us: 1-877-242-7372

Figure 15: Homepage of the application designed in PHP. The page shows a navigation bar up on the top and a list of host-stocks to the right of the page.

Short Term Prediction Page



SHORT-TERM PREDICTOR

You can obtain a prediction on the immediate trend of any stock here:

Our system uses a quadrature prediction mechanism to serve your needs with greater accuracy.

Please enter the name of the stock* available in your profile.

Stock added in the last hour can cause erroneous predictions.

Stock Symbol: NFL

NFLX

Submit

USER HISTORY

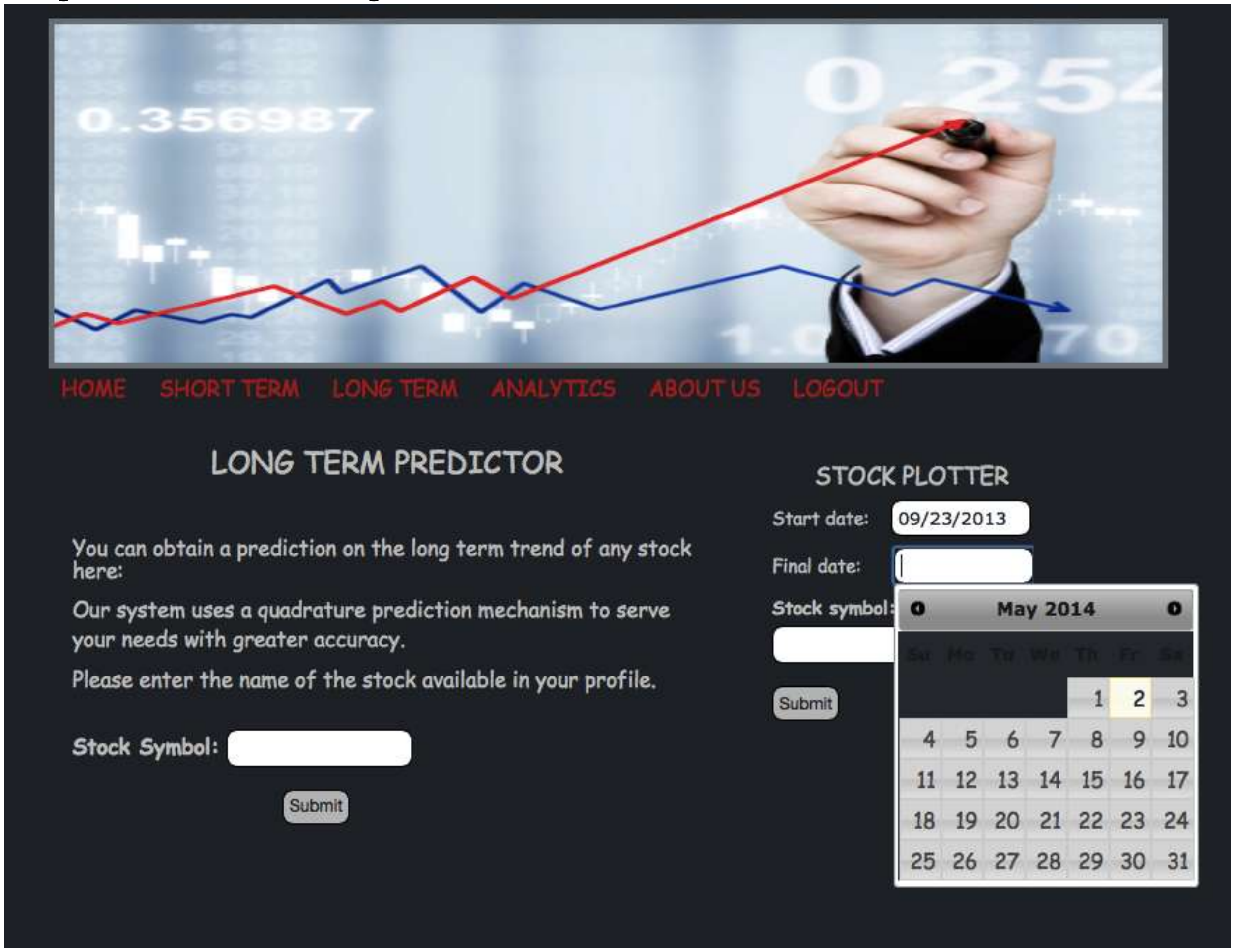
AAPL

AMZN

MSFT

Figure 16: Short-term prediction page; the page allows the user to enter a ticker symbol to get a short-term prediction. Additionally, the user is shown the last few stocks searched to assist their future searching needs.

Long Term Prediction Page



LONG TERM PREDICTOR

You can obtain a prediction on the long term trend of any stock here:

Our system uses a quadrature prediction mechanism to serve your needs with greater accuracy.

Please enter the name of the stock available in your profile.

Stock Symbol:

Submit

STOCK PLOTTER

Start date: 09/23/2013

Final date:

Stock symbol:

Submit

Figure 17: Webpage that allows the user to get a long-term prediction on any ticker of his/her choice. Additionally, the user is given the option to plot a stock where he/she can specify the start & stop dates.

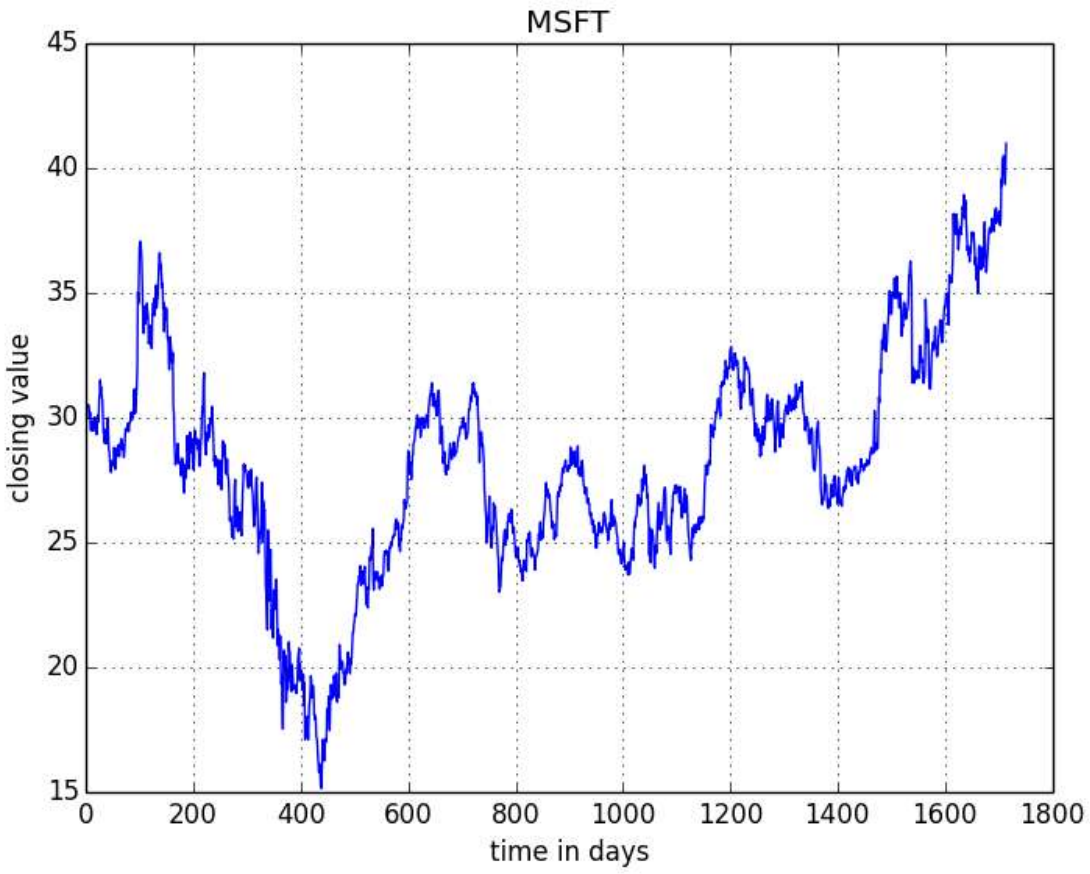


Figure 18: A pop-up window showing the long-term performance of the stock based on user input time values.

Analytics Page

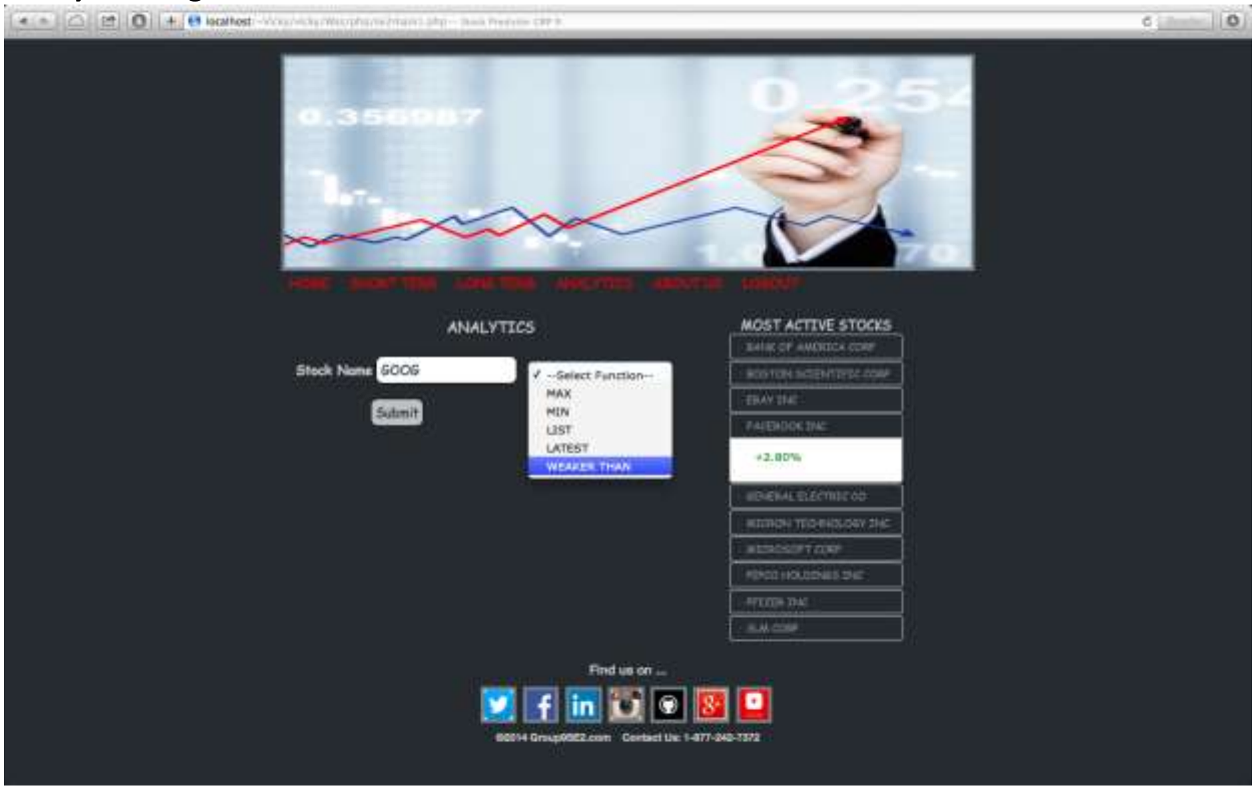
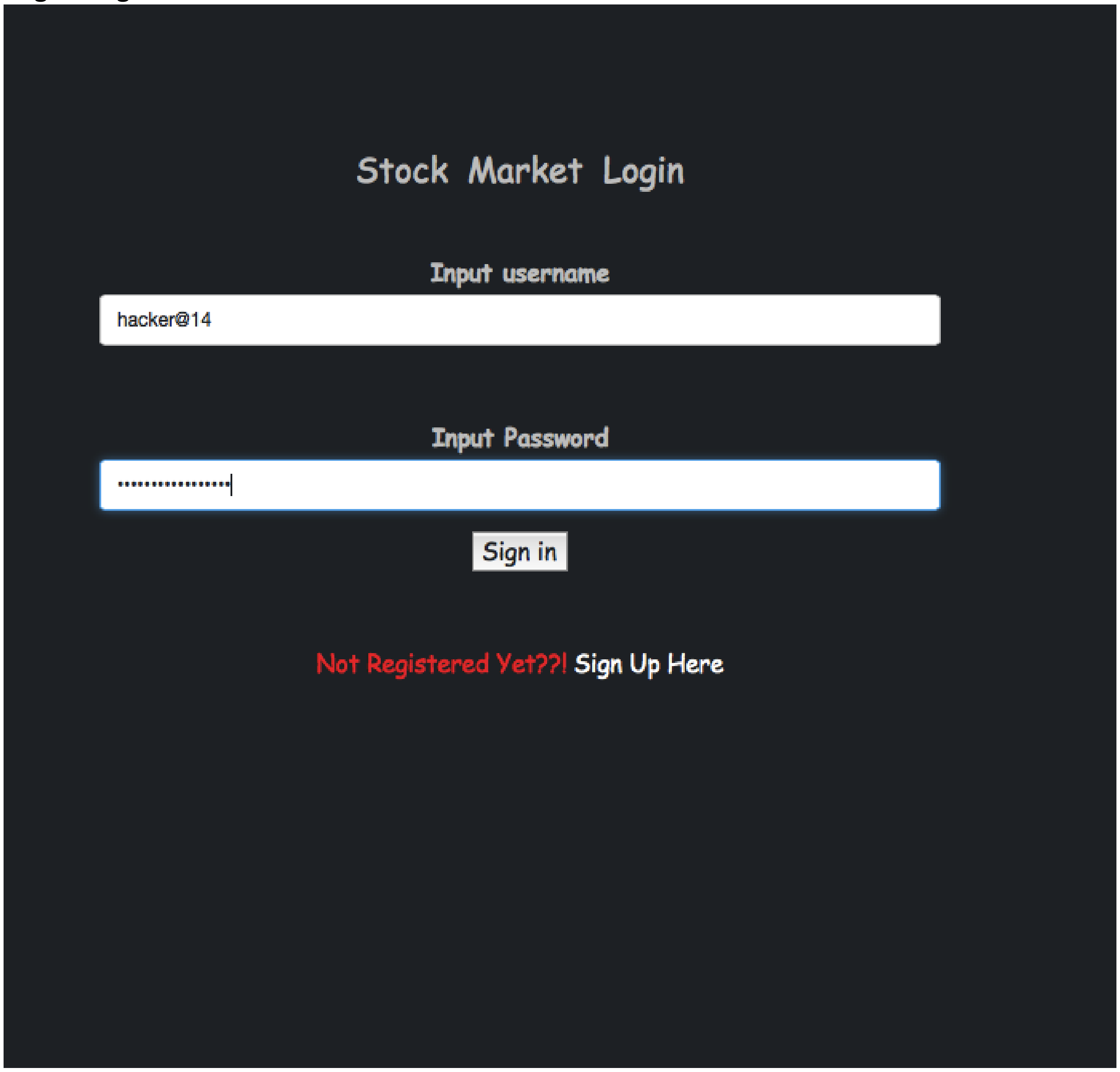


Figure 19: Shows the analytics tab in the application. Users can query the MySQL database directly from this front-end location.

Login Page


Stock Market Login

Input username

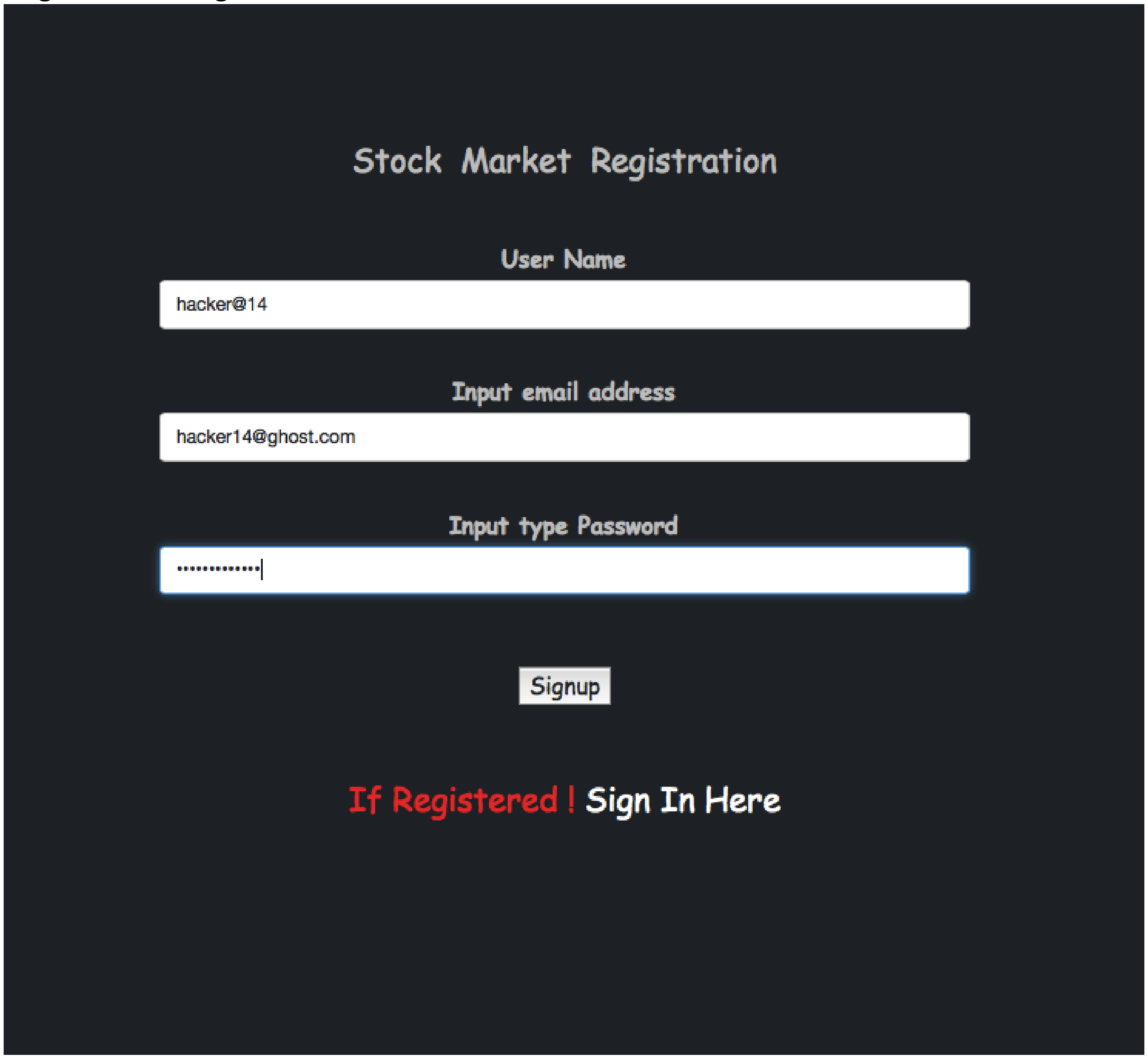
hacker@14

Input Password

••••••••••••••••

Figure 20: User login page; the page requires the user to login before accessing the website. Optionally, if the user is not registered, he/she has the option to do so from the website.

Registration Page


Stock Market Registration

User Name

hacker@14

Input email address

hacker14@ghost.com

Input type Password

•••••••••••••

Figure 21: User registration page; website allows new users to create an account with the application and gain access to its features. If the user is already registered, he/she is given the option to go back to the login page.

Results –

Figure 22 shows the actual output from the SVR regression test. The red dot at the end of the dataset is the actual prediction based on the regression performed. It is quite evident that we are able to predict the next value without much of a hassle and the regression line is rather noteworthy given the erratic behavior of the input data.

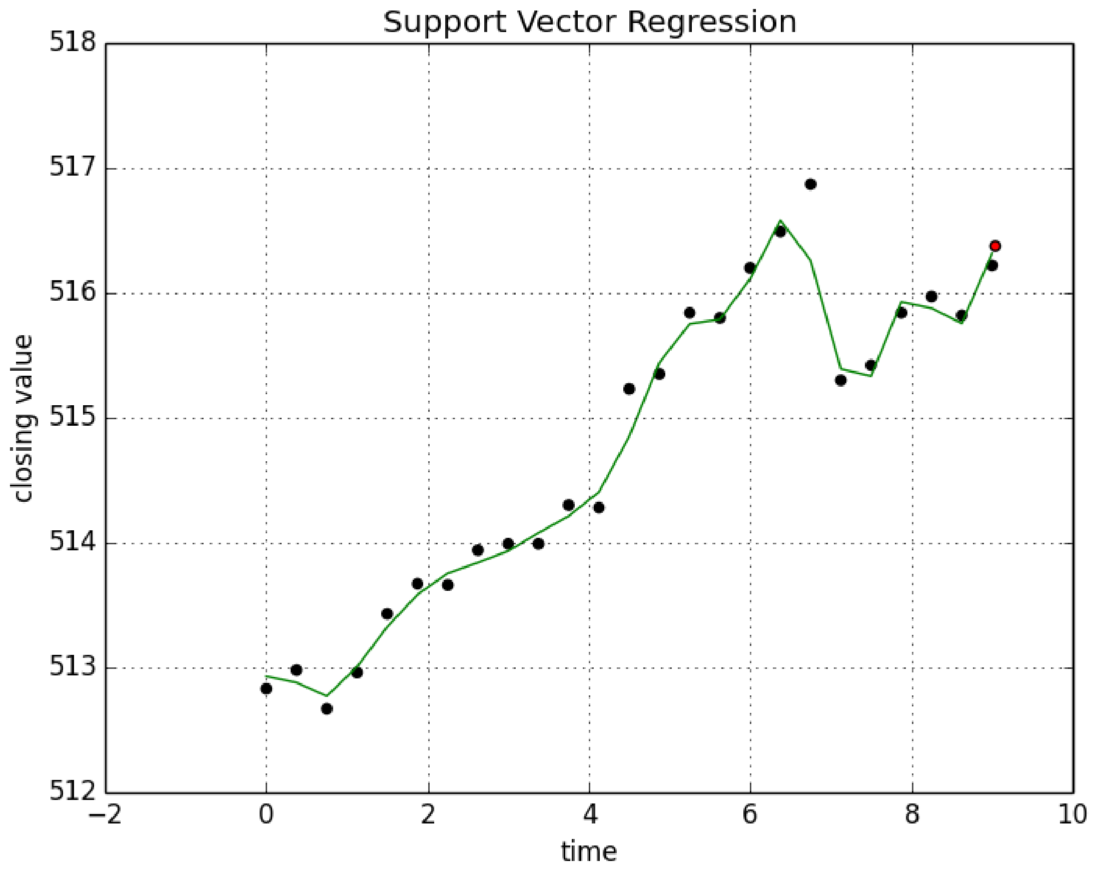


Figure 22: Output plot form SVR regression; note the red dot at the end signifies the predicted value.

Figure 23 shows the output graph a Bayesian prediction scheme; the red asterisk at the end of the stream represents the predicted value based on the training set provided. The training set in each case consists of 30 values from the short-term table of the MySQL database. Finally, Figure 24 shows how a sample prediction would look like to the user. Here we only show a long-term prediction, but needless to say, short-term prediction looks exactly the same.

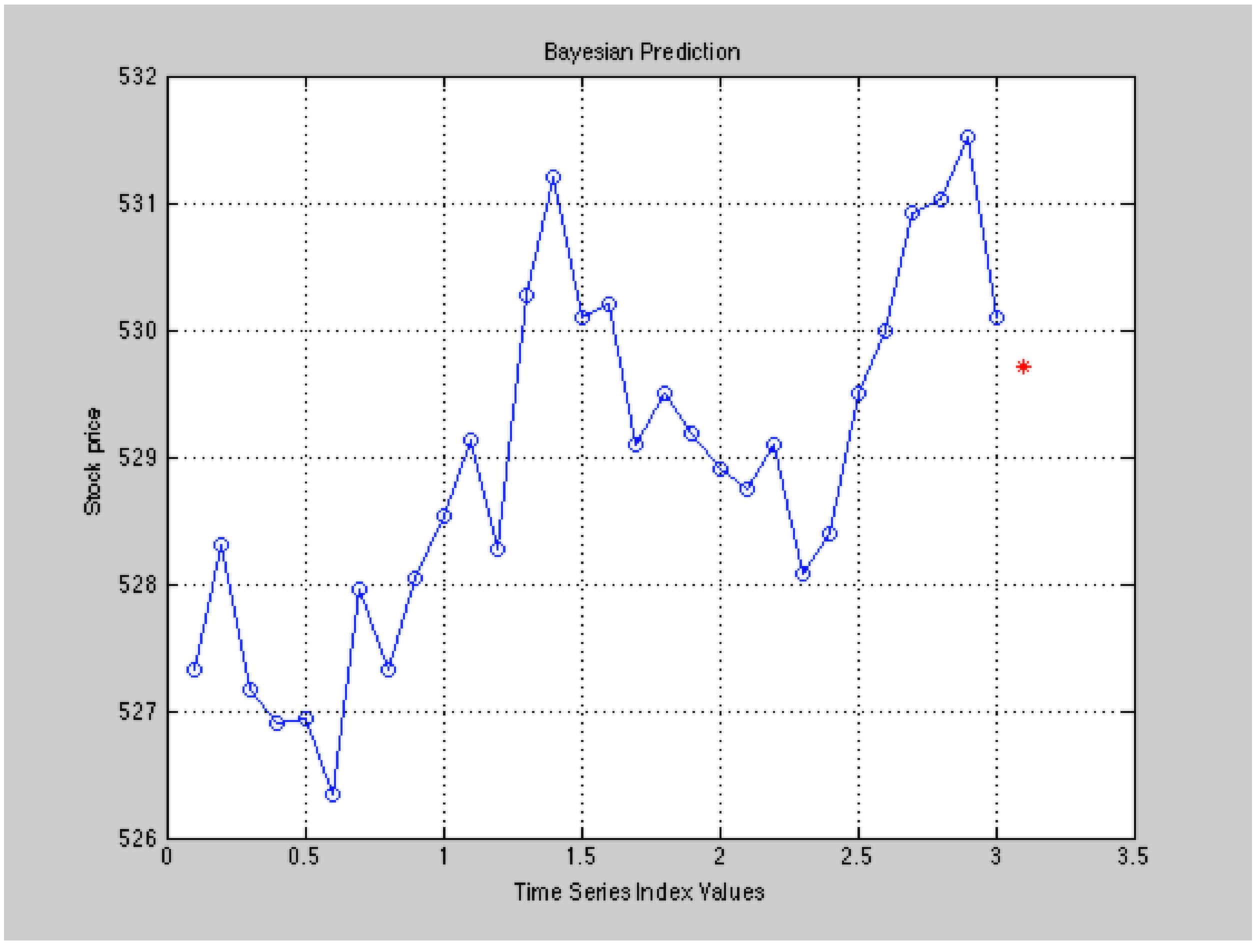
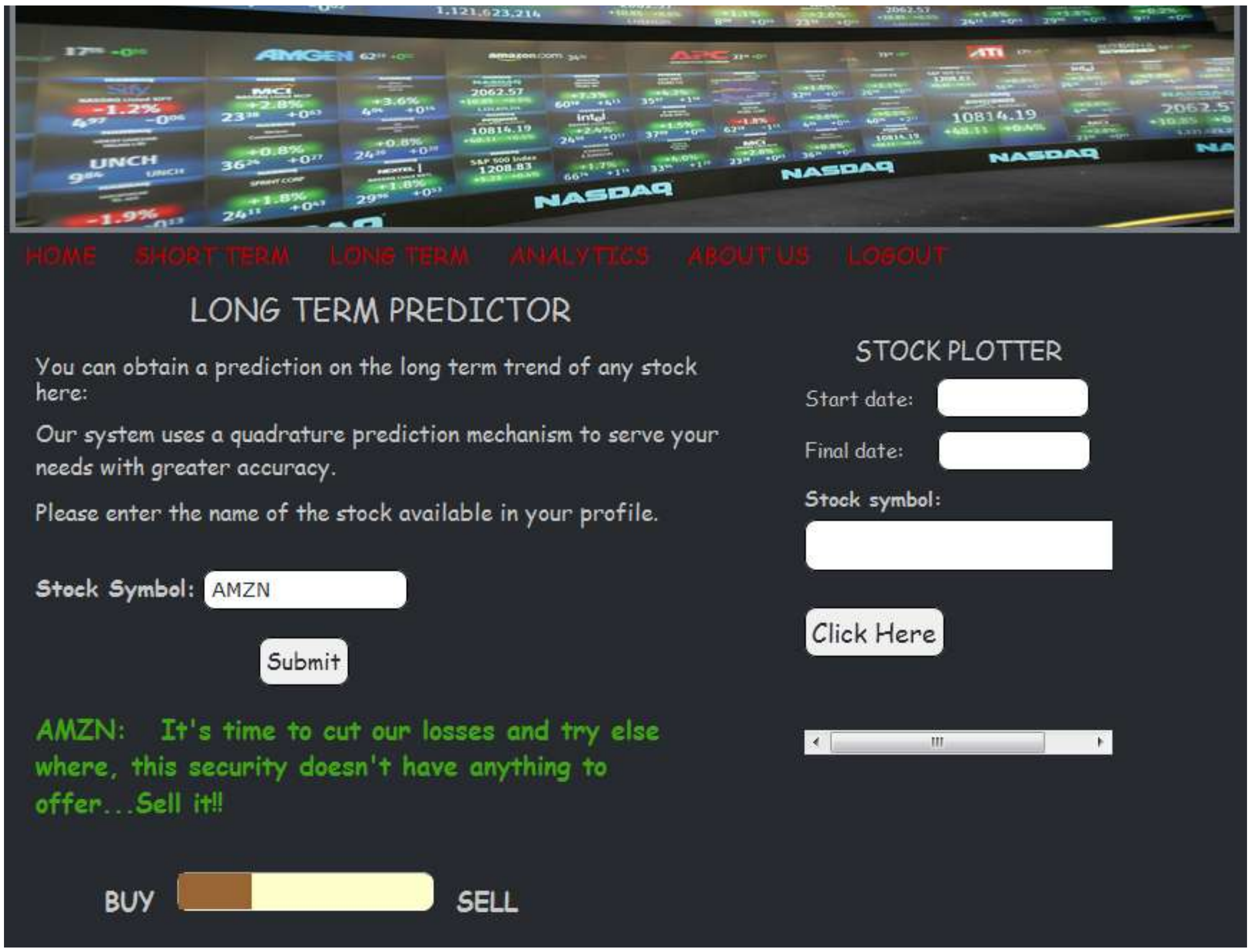


Figure 23: Shows the Bayesian prediction with respect to the training data.



LONG TERM PREDICTOR

You can obtain a prediction on the long term trend of any stock here:

Our system uses a quadrature prediction mechanism to serve your needs with greater accuracy.

Please enter the name of the stock available in your profile.

Stock Symbol: AMZN

Submit

AMZN: It's time to cut our losses and try else where, this security doesn't have anything to offer...Sell it!!

BUY SELL

STOCK PLOTTER

Start date:

Final date:

Stock symbol:

Click Here

Figure 24: Output of a long-term prediction; short-term prediction using voting also outputs similar text to the screen.

<u>Conclusion</u> –

In this project we have been able to successfully integrate three distinct programs to create an end-to-end service. We use python™ to pull data from Yahoo! Finance © and then store it into a MySQL database. Further, we are able to pull data from this database into the python environment again to create predictions and service user specific stock requirements. We are also able to inject all of the back end analysis on to a PHP based front end environment that not only gives a rather appealing look to the end user, but also allows the user to have an abstracted access to the python scripting environment and the MySQL database.

We have been able to successfully implement two versions of the same machine-learning algorithm to detect not only shapes but also create regression lines. Additionally, we have made use of common technical analysis metrics like RSI and MACD and combined them with a machine learning prediction scheme to give a rather robust recommendation to the user.

As part of further study and implementation, we would like to refine the machine learning algorithms and be able to detect more long-term trends. Furthermore, we would like to include additional technical analysis trends and make a much more informed decision. Clearly, by adding more variables to the recommendation process, will allow us to give the user a variety of output responses instead of just a ‘buy’, ‘sell’, ‘hold’ response. However, given the parameters of this exercise and the time constraints, this project was executed to our satisfaction with decent results.

Contributions –

Each member contributed equally to the project.

Lakshmi	-	20%
Shravan	-	20%
Prashanth	-	20%
Priyan	-	20%
Sudarshan	-	20%

Works Cited -

[1] Basak, D., Pal, S., & Patranabis, D. C. (2007). Support Vector Regression. *Neural Information Processing , 11* (10), 203-223.

[2] Forexo. (n.d.). *MACD - Moving Average Convergence Divergence*. Retrieved February 10, 2015, from Forexo - Forex online trading guide: www.forexo.co.uk/macd-moving-average-onvergence-divergence

[3] Google Finance. (n.d.). *Google Inc.* Retrieved February 9, 2014, from Summary.: http://www.google.com/finance?chdnp=1&chdd=1&chds=1&chdv=1&chvs=maximized&chdeh=0&chfdeh=0&chdet=1388523600000&chddm=38709&chls=IntervalBasedLine&q=NASDAQ:CSCO&ntsp=0&ei=gz74UoCOJabC6AHebA

[4] Investopedia. (n.d.). *Ascending Triangle*. Retrieved May 1, 2014, from Investopedia: http://www.investopedia.com/terms/a/ascendingtriangle.asp

[5] Investopedia. (n.d.). *Descending Triangle*. Retrieved May 1, 2014, from Investopedia: http://www.investopedia.com/terms/d/descendingtriangle.asp

[6] Investopedia. (n.d.). *Investopedia*. Retrieved May 1, 2014, from Rounding Bottom: http://www.investopedia.com/terms/r/roundingbottom.asp

[7] Investopedia. (n.d.). *Relative Strength Index - RSI*. Retrieved Febrary 9, 2014, from Investopedia.com: www.investopedia.com/terms/r/rsi.asp

[8] Marsic, I. (2012). *Software Engineering.* New Brunswick, NJ.

[9] Winston, P. H. (2013, May 28). *Reference material and playlist*. Retrieved May 2, 2014, from 6.034 Wiki: http://courses.csail.mit.edu/6.034f/ai3/SVM.pdf