Abstract

Predicting the direction of stock prices is a widely studied subject in many fields including trading, finance, statistics and computer science. Investors in the stock market can maximize their profit by buying or selling their investment if they can determine when to enter and exit a position. Due to the non-linear, volatile and complex nature of the market, it is quite difficult to predict. The factors that influence stock prices are complex to model. Machine Learning algorithms have been widely used to predict financial markets with some degree of success. This project aims to study the application of these algorithms to predict one-day-ahead outcomes of the S&P500 Index with special emphasis on feature generation and analyzing the predictive ability of several algorithms. Three different outcomes based on Next Day Closing price, Next Day Opening Price and Next Day Returns were studied. The construction of the predictive models is based on historical information of the index extracted through Yahoo finance. In order to analyze the predictive quality of the algorithms, the final results were compared through the implementation of ROC curve.
# Index

1. **Introduction**  

2. **Background**  
   2.1 **Literature Review**  
      2.1.1 Random Walks and the Efficient Market Hypothesis  
      2.1.2 Arguments for the Random Walk Model  
      2.1.3 Arguments against the Random Walk Model  
      2.1.4 Technical vs Fundamental Analysis  
      2.1.5 Related Research  
      2.1.6 Application of Machine Learning algorithms  

2.2 **Scope**  

3. **Data Collection**  

4. **Feature Generation**  

5. **Model Construction**  
   5.1 Defining the Machine Learning Models  
      5.1.1 Support Vector Machine
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1.2. Random Forest and Adaptive Boosting (AdaBoost) Classifier</td>
<td>16</td>
</tr>
<tr>
<td>5.1.3. Logistic Regression</td>
<td>17</td>
</tr>
<tr>
<td>5.1.4. k Nearest Neighbour</td>
<td>17</td>
</tr>
<tr>
<td>5.1.5. Gaussian Naïve Bayes (GNB)</td>
<td>17</td>
</tr>
<tr>
<td>6  Current Progress</td>
<td>18</td>
</tr>
<tr>
<td>7  Conclusion and Future Work</td>
<td>20</td>
</tr>
</tbody>
</table>
1. Introduction

Forecasting is the process of predicting the future values based on historical data and analyzing the trend of current data. Processing powers of computers nowadays have become powerful enough to process large amount of data. Stock price time-series are often characterized by a chaotic and non-linear behavior which makes the forecast a challenging task. The factor that produces uncertainty in this field are complex and from different nature, from economic, political and investment decisions to unclear reasons that, somehow, produce effects and make hard to predict how the prices will evolve. The stock market attracts investments due to the ability of producing high revenues. However, owing to its risky nature, there is a need for an intelligent tool that minimizes risks and, hopefully, maximizes profits.

Predicting stock prices using historical data of the time-series to provide an estimate of future values is the most common approach among the literature. More recently, researchers have started to develop machine learning (ML) techniques that resemble biological and evolutionary process to solve complex and non-linear problems. This work contrasts the typical approach, where classical statistical methods are employed.

The application of ML algorithms can be helpful in various financial problems. It has already been applied successfully in financial forecasting, trading strategies optimization and financial modelling.

This project focus on forecasting stock prices time-series using a machine learning approach. Considering a short-term forecasting problem (one-day-ahead forecast), the objective is to predict the stock price in given \( day_{t+1} \) using a set of inputs variables that represents the past stock prices up to \( day_t \).
2. Background

2.1 Literature Review

Stock market prediction, which has the capacity to reap large profits if done wisely, has attracted much attention from academia and business. Due to the non-linear, volatile and complex nature of the stock market, it is quite difficult to predict.

For many years the following question has been a source of continuing controversy “To what extent can the past history of a common stock’s price be used to make meaningful predictions concerning the future price of the stock?” (Fama, 1965).

Answer to this question has been provided on the basis of two assumptions. The first being the theory of random walk says that successive price changes are independent, identically distributed random variables, which would make prediction impractical since future price will be no more predictable than the path of a series of cumulated random numbers. On the other hand, against the random walk model, the other theory suggests stocks past price holds patterns that which can be used to predict future behavior. The arguments for and against random walk model based on past research are stated below.

2.1.1 Random Walks and the Efficient Market Hypothesis

The idea of stock prices following a random walk is connected to that of the EMH. The premise is that investors react instantaneously to any informational advantages they have thereby eliminating profit opportunities. Thus, prices always fully reflect the information available and no profit can be made from information based trading (Lo and MacKinley, 1999). This leads to a random walk where the more efficient the market, the more random the sequence of price changes.
2.1.2 Arguments for the Random Walk Model

Shleifer (2000) identified three main arguments for EMH:

1. Investors are rational and hence value securities rationally.
2. Some investors are irrational but their trades are random and cancel each other out.
3. Some investors are irrational but rational arbitrageurs eliminate their influence on prices.

If all these exist, then both efficient markets and stock prices would be very unpredictable and thus would follow a random walk

2.1.3 Arguments against the Random Walk Model

Market Over- and Under-reaction:

Fama (1998) argues that investors initially over or under-react to the information and the serial correlation explained above is due to them fully reacting to the information over time. The phenomenon has also been attributed to the ‘bandwagon effect’.

Seasonal Trend:

Here, evidence is found of statistically significant differences in stock returns during particular months or days of the week. The ‘January effect’ is the most researched, but Bouman and Jacobsen (2002) also find evidence of lower market returns in the months between May and October compared with the rest of the year.

Size:

Fama and French (1993) found evidence of correlation between the size of a firm and its return. It appears that smaller, perhaps more liquid firms, garner a greater return than larger firms.

Dividend Yields:

Malkiel (2003) notes that generally a higher rate of return is seen when investors purchase a market basket of equities with a higher initial dividend yield.

Value vs. Growth Firms

It has been noted by Malkiel (2003) and Fama and French (1993) that in the long-term, value (low price to earnings (P/E) and price to book-value (P/BV) ratios) firms tend to generate larger returns than growth (high P/E and P/BV ratios) firms. In addition, Fama and French (1993) found there to be good explanatory power when the size and P/BV were used concurrently.
These arguments are powerful and could lead people to doubt the EMH and random walks. On the other hand, why are there investors with sophisticated tools if their efforts are futile?

2.1.4 Technical vs Fundamental Analysis

Investors and traders typically employ two classes of tools to decide what stocks to buy and sell; fundamental and technical analysis, both of which aim at analyzing and predicting shifts in supply and demand (Turner, 2007). Shifts in supply and demand are the basis of most economic and fundamental forecasting. If there are more sellers than buyers for a stock (i.e., increased supply), the theory states that the price should fall, and similarly, if there are more buyers than sellers (i.e., increased demand) the price should rise. Given the ability to foresee these shifts in supply and demand thus gives the trader the ability to establish profitable entry and exit positions, which is the ultimate goal of stock analysis.

While fundamental analysis involves the study of company fundamentals such as revenues and expenses, market position, annual growth rates, and so on, technical analysis is solely concerned with price and volume data, particularly price patterns and volume spikes (Turner, 2007). Price and volume data is readily available in real time, which makes technical analysis ideally suited for short-term swing trades. The underlying assumption in technical analysis is that stock prices evolve with certain regularity, forming reliable and predictable price and volume patterns that reveal market psychology which can be used to determine shifts in supply and demand (Turner, 2007).
In this section several systems and research papers that have investigated the profitability of computerized technical analysis are presented.

Lo, Mamaysky and Wang (2000) found that “through the use of sophisticated nonparametric statistical techniques... [analysts] may have some modest predictive power” (Malkiel, 2003). Many within the business community also highlight Warren Buffet’s ability to consistently beat the S&P index as an additional proof that the market can be predicted with an accuracy rate that exceeds the 50% threshold. Brock et al. (1992) describe an approach that employs two technical indicators (one of which is the moving average crossover rule discussed in detail in Section 3.2) to generate buy and sell signals. Profits generated by the signals are calculated by simulating transactions on the Dow Jones Industrial Average over 1897-1986. Their results shows that the system generate consistent returns.

Experienced analysts could apply some mathematical models that are proven based on the past data in order to evaluate company’s intrinsic value, such as Graham number. Graham number and Graham’s criteria is probably one of the most famous models (Graham, 1949). However, due to the increased volatility in the current market, it would be probably impossible to find a company that satisfies Graham’s principles on today’s stock exchanges. Because of these changes, the need for adjusted models rose. Also, stock market changes over time (Hendershott & Moulton, 2011). New investment strategies and new technology were introduced, which made some of the old models obsolete. Since financial literacy became higher, there are more market players than ever. However, for some of the old models cannot be easily adopted for the changes in stock market.

The introduction to algorithms in trading definitely changed the stock market. Algorithms made it easy to react fast to certain events on the stock market. Machine learning algorithms also enabled analysts to create models for predicting prices of stocks much easier. Introduction of machine learning caused that new models can be developed based on the past data. In this paper we will describe the method for predicting stock market prices using several machine learning algorithms.
2.1.6. Application of Machine Learning algorithms

Machine learning algorithms have been widely applied in many areas of finance. More specifically, ML techniques are common accepted to predict stock markets by means of a regression or classification problems. Usually, we have a quantitative output measurement (such as a stock price) or categorical (such as stock price goes up/down), that we wish to predict based on a set of variables, for example the stock prices of previous days or other indicators that could explain the final outcome. The use of ML algorithms allows us to build predictive models that can explain the relation between input and output variables on a set of training data. Considering a Supervised Learning approach, the agent is provided with known input-output values (labelled data) and tries to formulate a function to explain such relation.

Phua et al. performed a study predicting movement of major five stock indexes: DAX, DJIA, FTSE-100, HSI and NASDAQ. They used neural networks and they were able to predict the sign of price movement with accuracy higher than 60% by using component stocks as input for the prediction (Phua, et al., 2003).

Kim (Kim, 2003) trained SVM on daily time series from of Korean stock market. He reported a hit rate of around 56%.

Huang et al did a tried to use support vector machines in order to predict weekly movement of Japanese NIKKEI 225 index. Their approach achieved 73% hit rate with SVM and 75% with combined model. Also SVM outperformed in their approach backpropagation neural networks (Huang, et al., 2005).
2.2 Scope

The main objective of the project is to examine the important and potential factors/predictors that could impact the stock market by applying machine learning algorithms on historical past prices and implementation of various technical indicators as features.

Past research was based on various potential factors (e.g. economic, online news, social media) that could impact the stock market and acquired data from disparate data sources such as Yahoo Finance, Wikipedia, Google Trends.

To narrow our scope, we will only evaluate time-series data collected from Yahoo Finance and generate features such as technical indicators based on this data collected.

To predict stock movements historical prices will be collected from 2012-01-01 to 2017-12-17 of the S&P 500 index. The project will be implemented on Python 3.6 since it contains extensive libraries for Data Analysis and Data Visualization.
3. Data Collection

The S&P 500 Index data is first collected for the period of 2012-01-01 to 2017-12-17. To see how S&P is affected by other major world indexes the following world indexes are included: FTSE, NIKKEI, DJIA and NASDAQ.

Exchange rates may have an effect on the S&P Index as well.

The following exchange rates data are collected from FRED (Federal Reserve economic data) to be added as features:
USDJPY, USDCNY, USDGBP, and USDEUR.
4. Feature Generation

Technical Indicators

In addition to major indexes and commodities and exchange rate we introduce several technical indicators as features: Pandas.series API is used extensively for calculating the below well renowned technical indicators:

- **Relative Strength Index (RSI)**

  It compares the magnitude of recent gains to recent losses in an attempt to determine overbought and oversold conditions of an asset. RSI ranges from 0 to 100. In practice, investors sell if its value is \( \geq 80 \) and buy if it is \( \leq 20 \).

  The relative strength index is calculated using the following formula: 
  (https://www.investopedia.com/terms/r/rsi.asp)

  \[
  RSI = \frac{100 - 100}{1 + RS}
  \]

  Where \( RS = \) Average gain of up periods during the specified time frame / Average loss of down periods during the specified time frame


- **Commodity Channel Index (CCI)**

  The Commodity Channel Index (CCI) is a momentum based technical trading tool used most often to help determine when an investment vehicle is reaching a condition of being overbought or oversold. (https://www.investopedia.com/terms/c/commoditychannelindex.asp)

  The Commodity Channel Index is computed with the following formula:

  \[
  CCI = \frac{(Typical \ Price - 20-period \ Moving \ Average \ of \ TP)}{(.015 \times Mean \ Deviation)}
  \]

  Typical Price (TP) = (High + Low + Close) / 3
Accumulation/distribution is a momentum indicator that attempts to gauge supply and demand by determining whether investors are generally "accumulating," or buying, or "distributing," or selling, a certain stock by identifying divergences between stock price and volume flow. The accumulation/distribution is calculated by first calculating the money flow multiplier, and then multiplying the money flow multiplier by the period's volume. 

\[ \text{Money Flow Multiplier} = \frac{(\text{Close} - \text{Low}) - (\text{High} - \text{Close})}{(\text{High} - \text{Low})} \]

\[ \text{Money Flow Volume} = \text{Money Flow Multiplier} \times \text{Volume for the Period} \]

\[ \text{ACCDIST} = \text{Previous ACCDIST} + \text{Current Period's Money Flow Volume} \]

In python: (https://github.com/Crypto-toolbox/pandas-technical-indicators/blob/master/technical_indicators.py)

The Momentum (MOM) indicator compares the current price with the previous price from a selected number of periods ago. This indicator is similar to the “Rate of Change” indicator, but the MOM does not normalize the price, so different instruments can have different indicator values based on their point values.

\[ \text{MOM} = \text{Close} - \text{Close of n days ago} \]

In Python: (https://github.com/Crypto-toolbox/pandas-technical-indicators/blob/master/technical_indicators.py)

The Rate-of-Change (ROC) indicator is a pure momentum oscillator that measures the percent change in price from one period to the next.
\[ ROC = \frac{(\text{Close} - \text{Close n days ago})}{(\text{Close n days ago})} \times 100 \]


- **Simple Moving Average (MA)**

  A simple moving average is formed by computing the average price of the index over a specific number of periods.

  In Python: ([http://pandas.pydata.org/pandas-docs/version/0.17.0/generated/pandas.rolling_mean.html](http://pandas.pydata.org/pandas-docs/version/0.17.0/generated/pandas.rolling_mean.html))

- **Exponential Moving Average (EMA)**

  Exponential moving averages (EMAs) reduce the lag by applying more weight to recent prices. The weighting applied to the most recent price depends on the number of periods in the moving average.

  Calculation:

  Initial Simple Moving Average: \( \frac{\text{10-period sum}}{10} \)

  Multiplier: \( \frac{2}{(\text{Time periods} + 1)} = \frac{2}{(10 + 1)} = 0.1818 \) (18.18%)

  EMA: \( \{\text{Close} - \text{EMA(previous day)}\} \times \text{multiplier} + \text{EMA(previous day)} \).


  In Python: ([http://pandas.pydata.org/pandas-docs/version/0.17.0/generated/pandas.ewma.html](http://pandas.pydata.org/pandas-docs/version/0.17.0/generated/pandas.ewma.html))

- **Standard Deviation (STDDEV)**

  Standard deviation is an indicator that measures the size of recent price moves of index, to predict how volatile the price may be in future.

5. Model Construction

5.1 Defining the Machine Learning Models

5.1.1 Support Vector Machine

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection. SVM finds a “large margin” so that the decision boundary stays as far away from the closest samples as possible.

5.1.2 Random Forest and Adaptive Boosting (AdaBoost) Classifier

Random Forest and AdaBoost are a set of ensemble learning method where the goal is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability over a single estimator.

In Random forest, each tree in the ensemble is built from a sample drawn with replacement from the training set. When splitting a node during the construction of the tree, the split that is chosen is no longer the best split among all features. Instead, the split that is picked is the best split among a random subset of the features. As a result of this randomness, the bias of the forest usually slightly increases but, due to averaging, its variance also decreases, usually more than compensating for the increase in bias, hence yielding an overall better model.

The core principle of AdaBoost is to fit a sequence of weak learners (i.e., models that are only slightly better than random guessing, such as small decision trees) on repeatedly modified versions of the data. The predictions from all of them are then combined through a weighted majority vote (or sum) to produce the final prediction.
5.1.3. Logistic Regression

Logistic regression is a linear model for classification rather than regression. The target value is expected to be a linear combination of the input variables. Like in support vector machines, smaller value of parameter $C$ specifies stronger regularization.

5.1.4. k Nearest Neighbour

Nearest neighbour (NN) classifiers assign an input example the class (label) of similar labelled examples. The algorithm requires a training dataset made up of examples that have been classified into several categories.

The value of $k$ determines how well the model will generalize to future data. Choosing a large $k$ reduces the impact or variance caused by noisy data, but can bias the learner so that it runs the risk of ignoring small, but important patterns.

5.1.5. Gaussian Naïve Bayes (GNB)

k-NN uses neighboring points to classify/label an input point. Sometimes it is not that clear cut that a data point falls entirely into one particular category. GNB assumes that all features in a dataset are independent and equally important.
6. Current Progress

Data is collected for S&P Index and four other major world indexes. For S&P, the following features are implemented at the current scope of the project.

**Relative Strength Index**

def RSI(close, period):
    delta = close.diff().dropna()
    up = delta * 0
    down = up.copy()
    up[delta > 0] = delta[delta > 0]
    down[delta < 0] = -delta[delta < 0]
    up[up.index[period-1]] = np.mean(up[:period]) # first value is sum of avg gains
    up = up.drop(up.index[(period-1)])
    down[down.index[period-1]] = np.mean(down[:period]) # first value is sum of avg losses
    down = down.drop(down.index[(period-1)])
    rs = pd.stats.moments.ewma(up, com=period-1, adjust=False) / \n         pd.stats.moments.ewma(down, com=period-1, adjust=False)
    return 100 - 100 / (1 + rs)

**Commodity Channel Index**

def CCI(close, high, low, n, constant):
    TP = (high + low + close) / 3
    CCI = pd.Series((TP - pd.rolling_mean(TP, n)) / (constant * pd.rolling_std(TP, n)), name = 'CCI_' + str(n))
    return CCI

**Accumulation/Distribution (ACCDIST)**

def ACCDIST(Close, High, Low, Volume, n):
    ad = (2 * Close - High - Low) / (High - Low) * Volume
    M = ad.diff(n - 1)
    N = ad.shift(n - 1)
    ROC = M / N
    AD = pd.Series(ROC, name = 'Acc/Dist_ROC_' + str(n))
    return AD
Momentum

```python
def MOM(Close, n):
    M = pd.Series(Close.diff(n), name = 'Momentum_' + str(n))
```

Rate Of Change

```python
def ROC(close, n):
    M = close.diff(n - 1)
    N = close.shift(n - 1)
    ROC = pd.Series(((M / N) * 100), name = 'ROC_' + str(n))
    return ROC
```

Moving Average

```python
def MA(Close,n):
    MA = pd.Series(pd.rolling_mean(Close,n),name='MA_'+str(n))
    return MA
```

Exponential Moving Average

```python
def EMA(close, n):
    EMA = pd.Series(pd.ewma(close, span = n, min_periods = n - 1), name = 'EMA_' + str(n))
    return EMA
```

7. Future Work

The S&P Index data together with the Technical Indicators will be used as our feature for the implementation of our Machine Learning Algorithms.

To evaluate the performance, a ROC curve will be plotted for each algorithms.

The ROC curve plots True positive rate vs. False positive rate at various threshold settings. Therefore, True Positive Rate (TPR) is the hit rate or sensitivity and False Positive Rate (FPR) is the fall-out or (1-specificity). The TPR defines how many correct positive results occur among all positive samples available during the test. FPR, on the other hand, defines how many incorrect positive results occur among all negative samples available during the test.

![Figure 1: A sample ROC curve](image)

From figure 1, a ROC curve demonstrates several things:

1. The closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate the test.

2. The closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate the test.

3. The area under the curve is a measure of test accuracy. An area of 1 represents a perfect test; an area of 0.5 represents a test similar to a coin-flip.
The Area Under Curve (AUC) is equal to the probability that a classifier will rank a randomly chosen positive chance as positive, in other words, higher than a randomly chosen negative chance.
References


