Machine Learning for Predicting Financial Market Movements

Jihye Hong

Submitted for the Degree of Master of Science in

Data Science and Analytics

Department of Computer Science
Royal Holloway University of London
Egham, Surrey TW20 0EX, UK

Sep 13, 2018

Declaration

This report has been prepared on the basis of my own work. Where other published and unpublished source materials have been used, these have been acknowledged.

Word Count: 14,125

Student Name: Jihye Hong

Date of Submission: 13 September 2018

Signature: Jihye Hong

Acknowledgement

I would like to express my very great appreciation to Professor Kalnishkan for his valuable suggestions with eye-opening resources and guidance during planning and development of this project. His willingness to give his time so generously and responsively has been very much appreciated.

Abstract

The objective of this project is to evaluate the predictive power of hidden features of financial market data with a set of machine learning classification algorithms. In the past, the prediction of financial market has been recognised as an impossible accomplishment with its characteristics such as uncertainty, noise and volatility. However, since the advent of adequate modern data sources and intelligent machine learning algorithms, we are enabled to reveal underlying patterns and insights related to financial markets. This project mainly used two financial data sources, cryptocurrency market and stock market data in order to show what degree of predictability exists in traditional or emerging financial markets. The machine learning algorithms for this project have been chosen to prove which classifier outperforms under a variety of settings. It is examined to compare the predictive performance of various predictors depending on the different financial dataset with a range of approaches. As a result, SVMs, Naïve Bayes and K-Nearest Neighbours (KNNs), are selected to demonstrate the predictability of financial market movement. Historical crypto-currency and stock market data are processed to label the price of financial assets as either bullish (1) or bearish (-1) in the next time unit to define the problem as a binary classification. The independent variables are the values of the last price movement of currencies or stocks.

Contents

1 Introduction

1.1 Summary

The predictability of financial market movements has been an active ongoing research topic for academics and practitioners with plenty of attention in industries. From traditional and statistical models to state-of-the-art artificial intelligence, predictive models have revealed more pieces of evidence about links between various market components and prices. It convinces people of the existence of predictability in financial markets, such as stock, foreign currency and bond markets. The definition of predictability in financial markets means that a future price of a security can be predicted based on information in the current time as the price reflects relevant information or trends acknowledged about the security.

This project aims to empirically find a robust model to predict price movements for a currency over time. There has been much research to look for useful features such as price-based or volume-based features to exploit them for algorithmic trading. In this research, only historical price data is used for features and no other technical indicators are considered. It is natural to believe historical data has some degree of patterns. Thus, this study will demonstrate the forecasting power based on pure price information.

Another goal of this project is to illustrate the contrary concepts between market efficiency and predictability. About the definition of market efficiency, in 1970, Eugene F. Fama [3] firstly introduced that a market to be "informationally efficient" if prices at each moment incorporate all available information about future values, which means the current price at the moment reflect all information about future values of securities. As we test machine learning models on cryptocurrency and stock market data, we will be able to see which market has more predictability. That concerns the more predictable market is less efficient than others. Those two markets have clear differences in many aspects. So, we can expect that the same experimental approaches applied to these two markets might show us different experimental results.

As previous academic research focused on proving, it is a rational belief that a financial market is not only driven by noises. Although financial market data are non-stationary, non-linear and noisy, there might be certain trends like any other time-series data. These underlying trends of training data allow machine learning models learn in order to predict labels based on the given data. With the characteristics of financial market data which are non-linear and noisy, one of the focus in the experiments is to test various type of ML algorithms and measure their performance. Therefore, this project selects machine learning algorithms with different attributes to enrich the selection of algorithms. The classifiers tested in this project are as follows:

- K-Nearest Neighbours
- Support Vector Machines
- Naïve Bayes

I believe this project will give me a great opportunity to learn various domains. Firstly, the utmost important achievement of this project is that this allowed me to have confidence about conducting a Data Science project from scratch. As carrying out experiments and researching relevant theories, I was able to get lots of exposure and deep understanding to the fundamental concepts of various machine learning algorithms in practice, which I have learnt from the MSc courses previously. Moreover, through this stage, I could get practical knowledge about not only the ML algorithms, but also the whole process of conducting a Data Science project such as data mining, processing, exploration, feature engineering and model comparison.

Secondly, I could get knowledge of the implementation of machine learning algorithms in financial forecasting in real-world trading. I started this project as a part of a proof of concept research in a project of my placement work, but it was not accompanied by the essential knowledge such as technical analysis or efficient market theory. With my supervisor's advice to look into related econometric studies, I was able to understand the history of financial forecasting and how the actual practitioners are using financial forecasting towards the implementation in markets such as algorithmic trading.

For the whole process, I have used Python as the main tool. Due to its popularity in industries as well as its functionality for both data science and software development, I decided to carry out this project in Python. I am proud of myself that I have learnt Python and Python data science stack by complete self-taught through this project. For my career as a data scientist, now I could achieve strong knowledge of Machine Learning in Financial Forecasting and adequate data science skill-sets. On the basis, what I learnt from this project will highly contribute to my next career to be as a data scientist with proper knowledge and experience in the financial technology field.

1.2 Stock and cryptocurrency markets

A financial market broadly means a marketplace for sellers and buyers who desire to trade financial securities such as stocks, bonds or more tangible assets like commodities and real estates based on their demand and supply. In other words, a financial market can function by sellers, buyers and pricing of an asset to trade. In a theoretical market, the only components drive pricing is the imbalance between sellers' supply and buyers' demands. However, in a real market, there are more things that can affect to the price, such as regulations or laws on the markets as well as costs and fees determining the prices of securities in exchanges. Moreover, prices among securities can be highly linked and affected the price of one another.

The financial market data for our experiments consist of two types of historical financial price data. One is the stock markets, and the other is the crypto-currency markets. As one of the major financial market, the stock markets have been the most popular destination for the research of statistical modelling and market simulations. There have been many academic studies targeting stock market prediction. On the other hands, cryptocurrency has only a little bit of studies going on. It is an emerging financial marketplace with massive volatility in prices. Its

market capitalisation is relatively small compared to stock markets. However, it is remarkably increasing regarding its volume and leverage. Cryptocurrency soared with full of attention from industries, and it peaked its price in 2017 with exponential growths. The extreme growth of the market capitalisation allured people to invest more capitals into cryptocurrency. Currently, the total market cap is over $300 billion. There is an increasing number of institutional investors due to its lowered entering barriers.

With this given information about cryptocurrency, now we are going to compare the similarities and differences between stock and cryptocurrency markets briefly to help us understand a better picture of how they would affect to the prediction performance. In terms of similarity between them, firstly, both function in similar ways. Prices are determined by buyers and sellers who willing to trade the asset. Secondly, it is not backed by a physical commodity as it values intangible idea behind them such as ownership for shares or a digital currency for technology. Thirdly, it has values in fiat currencies such as US dollars or British pounds.

About differences, with regard to trading hours, cryptocurrency is 24 hours available to trade continuously. Contrarily, stock markets are only available to trade for weekdays during the fixed trading hours. Furthermore, they have distinctions in fundamental factors. Although both securities value over intangible objects, buying a stock means you invest in an actual company. Therefore, you will get the share of the company. Unlike a stock, when you buy a cryptocurrency, you invest in the technology or the digital currency itself, which only returns you the future value of the coin not ownership or share of a company. This fact makes cryptocurrencies highly volatile compared to the stock market.

From these aspects on similarities and differences in these two various markets, I assumed those characteristics would bring some impacts on the performance of models for prediction of market movements. As a result, by application of the same experiments to those two markets data, we will be supposed to reveal which financial market is more predictable and efficiently acknowledging information for the future values.

2 Background research

In this part, we will see related concepts for prediction of financial market and studies conducted previously. It will show us how this subject field has been developed in various perspectives among economics, statistics and computer science.

2.1 Fundamental and Technical Analysis

Fundamental analysis is a trading approach that claims that economic factors influence supply and demand in financial markets. Thus, it focuses on the analysis of fundamental information such as the revenue of a company in the annual report, industry, competitors, and economics. Fundamental analysts use them as indicators to evaluate potential demands on an asset. Technical analysis, on the contrary, is an approach to forecast the direction of prices based upon historical price data. It is under an assumption that all necessary information is already reflected in an asset price. Therefore, many technical trading rules are based on patterns in historical prices [1]. As a result, technical analysts evaluate trends in historical price to predict future price movement to exploit past trend to make more profit. Technical analysis has proved its fundamental concept that prices economically and statistically have momentum over periods of times to a certain extent [2].

As technical analysts believe that there is all necessary information determining future prices of a stock in price data, they don't focus on fundamental information such as an annual report from a company. It develops quantitatively measurable indicators to forecast future price movements. There are a number of indicators mainly used such as ROC (Rate of Change), oscillators or moving averages in a given time window without regard to underlying economic [2]. They were adopted as tools widely applied to the evidence-based technical analysis.

2.2 Market Efficiency

In econometrics models, another notable concept against technical analysis is the Efficient Market Hypothesis (EMH). The notion of market efficiency was firstly introduced in 1970 by economist Eugene Fama, who later won the Nobel Prize for his effort. Fama claims that there is no relationship between past price or volume and future price movements because prices already "fully reflect" all available information. Under the Efficient Market Hypothesis, technical analysis is not valid and financial market prices are fundamentally unpredictable [3].

In this theory, there are three degrees of market efficiency types based upon the information subsets reflected in markets. Firstly, the weak-form of market efficiency assumes that past rates of return do not affect future rates because the market is efficient. Secondly, the semi-strong form of market efficiency supposes all public information is reflected in the current price of a security, and it will absorb new available public information to adjust prices quickly for market equilibrium. Finally, strong-form of market efficiency assumes that market prices reflect all public and even private information. Thus, even though you are an insider trader with monopolistic information, you can't earn excessive returns over an average investor.

Although Fama defined these three different degrees of efficiency, in reality, it is hard to measure quantitatively how efficient a market is. Given the assumption, a financial market is completely random thus prediction for financial market and exploiting them for trading strategy is invalid.

2.3 Machine learning in financial forecasting

Financial forecasting with machine learning has been mainly focusing on whether it will go bullish or bearish as a binary classification matter, although there were few regression approaches to predict markets. Bullish means the future price movement will be up, bearish means the opposite. Common class labelling is illustrated in (1). In machine learning, problems can be categorised into two groups, one is regression problems predicting numerical values, another is classification problems for prediction of qualitative labels. Binary classification is a classification problem having two classes. Thus, in a binary classification problem, it requires that each example in a dataset to be classified into two discrete classes. Although an output class should be labelled into qualitative classes, independent variables can have either continuous numerical values or discrete values as well. Our goal in this chapter is to look into popular binary classifiers utilised in related works.

$$f(X) = \begin{cases} "Up", & 1 \\ \text{Down}, & -1 \end{cases} \quad (1)$$

There are many binary classifiers which have been applied to predict financial market movements in different studies. It has been acknowledged that the Neural Network approach for long-term pattern recognition is quite powerful. As a result, the majority of academic studies has been developed on Neural Network (NN) before the advent of Support Vector Machine (SVM). Since then, SVM and NN oriented algorithms, such as ANN, have been most widely utilised for financial forecasting in preceding studies and demonstrated promising results due to their non-linearity and strength in time-series data. In [4], Kim examined SVM to predict stock price index. In the meantime, Kim examined the sensitivity of SVM towards its parameters, upper bound C and kernel parameters in terms of its forecasting power. Also, the prediction accuracy was compared with SVM, ANN and case-based reasoning (CBR) to evaluate the feasibility of applying SVM in financial forecasting. Input features in the approach consisted of technical indicators calculated from historical prices. It comprised total 14 features including %K, ROC (Rate of Change), Momentum, etc, to name a few. You can find the details of each indicator and how they are calculated in his study in Appendix A. According to the result of experiments, SVM achieved the best prediction performances with an accuracy of 57.8% outperforming the others. It demonstrated the model accuracy has a certain degree of sensitivity towards those parameters.

In another recent study, Yauheniya at al [5] experimented with three machine learning algorithms, SVM, ANN and K-NN to forecast price movements investigating the correlation between a forecast horizon and an input window length. Hence, respective data samples were collected depending on the input window length and forecast horizon for different approaches. In this research as

well, also technical indicators like Kim's are used as features. There are 10 features selected including SMA(Simple Moving Average), EMA(Exponential Market Average), ATR(Average True Range), and other price-based features. In the test result, among the three models, SVM resulted in the best performance with over 74%, ANN followed SVM with an accuracy of 71% and K-NN was far behind with the 57% accuracy. These results with high accuracy are contradictory with EMH as EMH claims excessive returns cannot be earned with predictive methods such as technical analysis or model-based training. With that respects, Ming-Wei at al [6] presented a comparison between the ML and EMH literature based on experiments across 34 financial markets. In this study, also ANN and SVM were selected for ML approaches, as well as SVM outperformed ANN. In conclusion, it proposed machine learning approaches for forecasting of financial markets have predictive power to a certain degree although the results contradict EMH.

Including those preceding studies, both NN and SVM have been predominant approaches with technical indicators as features over other algorithms in financial forecasting. Consequently, many efforts to develop both approaches have been made with previous benchmarks. In the meantime, there have been only a few of past research presented feasibility of financial forecasting with novel features beyond technical indicators or other machine learning algorithms. Michel at al [7] conducted one of the interesting benchmarks introducing performance comparison of 7 different ML algorithms. In [7], it pointed out SVM and NN have been the majority of focuses in the field. Also, it claims ensemble models are able to outperform SVM and NN with its proven industrial practices in other industries such as face recognition, gene selection and credit scoring. In the result, it demonstrated Random Forest (RF) performed the best and proposed to consider RF for financial forecasting in future studies. Michel at al also presented a list of algorithms used for stock price prediction research between 1990 and 2015. The table is in Appendix B. Moreover, in most of the recent research, deep learning is also widely explored to affirm its predictive power in financial forecasting [8] [9].

However, to the best of my knowledge, even in [7] or other studies, the Naïve Bayes model has not been mainly to come into the spotlight in financial forecasting subjects. Due to the truth that there is no one perfect algorithm fits for all problems in machine learning, there are many factors that might affect to the performance. Thus, it is always the biggest task to choose the best performing algorithm to solve a specific problem according to several factors. From that perspective, after some novel feature engineering in this project, Naïve Bayes evidently will have outperformed over other algorithms in the experimental phases later on. The following section will briefly describe the fundamentals used in the experiment at high-level understanding and the evaluation standards for binary classification as well.

2.4 Discriminative vs Generative Classifiers

The type of output in a prediction problem can categorise one machine learning model if it is a regression or classification model (or both). Also, one algorithm might be either supervised or unsupervised learning model depending on its learning method. Another useful categorisation is if a model is either

discriminative or generative. The fundamental difference between discriminative and generative models is, 'Discriminative models learn the (hard or soft) boundary between classes' and 'Generative models model the distribution of individual classes' [13]. From the definition, Discriminative models are unlikely to make more assumptions on the underlying data as it learns based on the boundary between classes in the training dataset. On the contrary, generative models make more assumption on the fundamental of data.

To summarise, those two are mainly distinguished based on the degree of statistical modelling [14]. Discriminative models are rather non-statistical and model the conditional probability $P(Y|X = x)$; without assumption on the data. Even there are non-probabilistic models referred to "discriminative" [14]. On the other hand, generative models are often upon strongly statistical assumption above the conditional distribution. So, it is called a statistical model of the joint distribution.

In [15], Ng and Jordan presented the thorough comparison between logistic regression and Naïve Bayes as typified for discriminative and generative models. It was claimed that "there can often be two regimes of performances" with repeated experiments, although discriminative models are almost always preferred in wide applications as citing Vapnik [16] "one should solve the classification problem directly and never solve a more general problem as an intermediate step such as modelling $P(Y|X = x)$;." As a result of the study, it proposed "while discriminative learning has lower asymptotic error, a generative classifier may also approach its (higher) asymptotic error much faster" [15]. To name a few typical models from each approach, there are widely applied generative models are such as Gaussian Mixture model or Naïve Bayes, and discriminative models are Logistic regression, Support Vector Machine, Neural networks, etc. In the experiments section, various machine learning algorithms will have been explored for this project. Contrary to the better performance and preference of discriminative models commonly acknowledged, Naïve Bayes outperformed other discriminative algorithms in our study.

2.5 Support Vector Machines

Support Vector Machines (SVM) are supervised learning models that can be applied for both classification and regression analysis, but it is more widely used for classification problems. In a classification problem, SVMs try to find the best hyperplane to separate the labelled training examples into the number of discriminative classes geometrically with maximum margin. A hyperplane means a subspace of dimension p-1 in the p-dimensional space, $\mathbb{R}^p$. For example, in 2-dimensional spaces, a hyperplane is a line, as well as, in 3-dimensional spaces, a hyperplane is a plane. After building the model with training observations, SVM can assign a class label to a new instance based on where that instance lands to which side of the hyperplane in the p-dimensional spaces. When SVMs construct the hyperplane (decision boundary), it will choose the one that has the farthest distance apart from the closest data point in each category to provide a clear gap as well as find an optimal solution. That is why it is known as a maximum margin classifier as well. The closest data points from each side are called support vectors, which are a tiny subset of training samples.

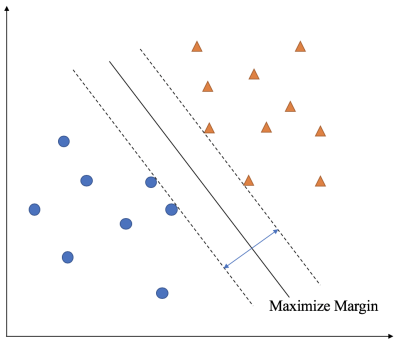


Figure 1: Separation by hyperplane of SVM

We assumed separation by hyperplanes for linearly separable samples so far, however, in reality, there are linearly non-separable data more often. To solve that problem, Vapnik at al (1992) suggested a way to create nonlinear classifiers by applying the kernel trick to maximum-margin hyperplanes [10]. The idea came to gain linear separation by mapping the data into a higher dimensional space. That means, even though the data samples don't look separable in the original dimension, in a higher dimension, it can be separable non-linearly. That suggests creating non-linear classifiers by applying the kernel trick. Although there are various non-linear kernels, these three are widely used in kernel tricks.

- Polynomial Kernel: $K(x, x') = (x \cdot x' + 1)^p$
- Radial Basis Kernel (Gaussian): $K(x, x') = \exp(-\gamma \parallel x - x' \parallel^2)$
- Sigmoid Kernel: $K(x, x') = \tanh(Kx \cdot x' - \delta)$

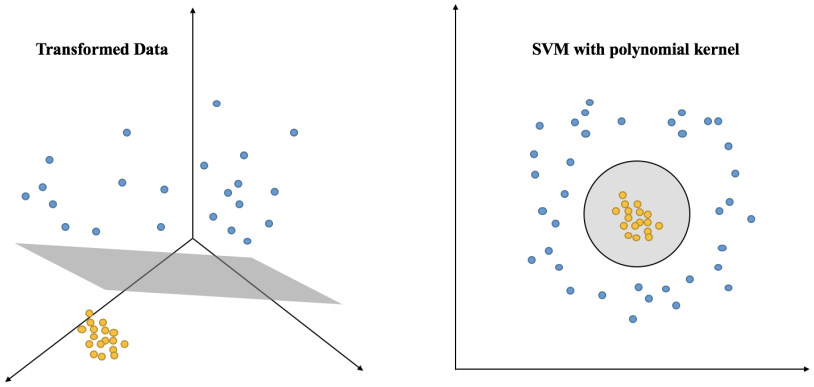


Figure 2: SVM with polynomial kernel

The following sequence of definitions is based on [11], we will get through the summary of the whole definitions due to the fact that we are only focusing on

the high-level understanding of various ML approaches. In order to utilise kernel tricks, all we need is the dot products. By that means, we can simply replace dot products with a kernel in the original linear classifier (2) to map it to a higher dimension.

$$f(x) = \beta_0 + \sum_{i=1}^{n} a_i \, (x \cdot x_i) \qquad (2)$$

To clarify, the linear classifier is called Support Vector Classifier and when the support vector classifier is combined with a kernel, it is called Support Vector Machine. With a kernel, SVM can produce very high variance to obtain best training performance. For example, in a case, every example in a training set can be a support vector if too much freedom is given to SVMs. To prevent that, a penalty parameter C is introduced to SVMs in order to reduce overfitting. As a regularisation parameter, C controls the trade-off between overfitting and reducing errors. If C is small, the classifier will try to find a larger margin, although it would be likely to get misclassified instances. When the value for C gets larger, the classifier will choose a smaller-margin hyperplane but achieve a better performance as classifying all the training examples perfectly. The regularization process for SVMs can be found in the book by Vapnik [12] with more details.

2.6 Naïve Bayes

Naïve Bayes is a typical generative model which make their predictions by so-called Bayes theorem (3). As a classification technique, Naïve Bayes strongly assume each feature is independent to other features based on Bayes rules. Thus, it is also called simple Bayes or independence Bayes. Naïve Bayes consider all of each property in the data independently contribute to the likelihood that the label of the example is a certain class, $P(c|x)$; here c is the label and x is the feature.

$$P(c_k|x_1, \dots, x_n)$$

$$P(c_k|X) = \frac{P(X|c_k)P(c_k)}{P(X)} \qquad (3)$$

$$X = (x_1, \dots, x_n) \text{ with n-features}$$

In practice, we can calculate the posterior probability $P(c_k|X)$ based on Bayes theorem (3). If you appraise the theorem, you can figure out there are three different probabilities consisting of the calculation of posterior probability. $P(X|c_k)$ is called the likelihood, which is the probability of predictor x given class k. $P(c_k)$ is the prior probability of class and $P(X)$ is the prior probability of predictor X. From the calculation, we can see the probability of an instance to have its label as class k. The class k with the highest probability will be predicted for the given example.

There are explicit pros and cons of Naïve Bayes, the biggest disadvantage it has is that it assumes the data features are independent although not always features are independent. So, if we apply Naïve Bayes towards a dataset with strongly

correlated features, it would not result in a good predictive performance. However, Naïve Bayes have many advantages at the same time. As we have seen in the theorem, it is relatively simple and easy to understand what happened inside of the box unlike black-boxed approaches such as Neural Network. Moreover, because it considers every feature is independent, it is not sensitive to irrelevant features.

In practice, Naïve Bayes has been popularly applied for medical science, anomaly detection as well as text classification but not mainly used for financial market prediction problems. However, I found out there was a research combining text classification towards financial market forecasting [17]. It exploited an approach of data mining and text classification to classify text news related to a stock market FTSE100, and the classifier was trained to predict the next price movement based on the given news. While it demonstrated a statistically significant performance, as the approach focused on the factors apart from historical price data or volume related features in the market it is rather on the basis of fundamental analysis for the forecast, unlike this project.

2.7 K-Nearest Neighbours

K-Nearest Neighbours (KNNs) are one of the most straight-forward machine learning algorithms for classification or regression. It is a non-parametric supervised learning machine. Non-parametric means it doesn't hold any assumption on the underlying data distribution. To that end, KNNs do not formulate any hypothesis, it simply results in a prediction on the test object. As we are focussing on a binary classification problem, here we will mainly go through K-NNs with the classification setting. The simplest form of K-NNs is the nearest neighbour algorithm. From the impression of the words "Nearest" in its name, it literally predicts the label of a test object to the same class of its nearest training sample after searching the nearest object in the training set. In the prior example of the most intuitive K-NNs is the Nearest Neighbour algorithm (1-NN), which K is 1. If the K is not 1, it follows the approach of majority voting classification. Majority voting defines the way to classify the test label among labels of K nearest objects'. As a result, the test label will be predicted as the majority of classes in K nearest objects to decide the best label for the new object. Due to tie can occur when we choose an even number as K, it is recommended to choose an odd number for the optimal vote. $K = 3$ or 5 is the most common choice.

There are a number of methods to calculate the distance from the test object to training objects. One of the most commonly used distance metrics for continuous variables is the Euclidean distance (4). In the meantime, it is important to choose the right metric to attain the best performance, although any kind of distance metrics can be used for K-NNs. In a p-dimensional space, p is the number of features in the dataset. If p is larger than 10, Euclidean distance is more or less unhelpful due to it can result in too many instances with same distances. In this context, it is called 'the curse of dimensionality'. As a result, for more sophisticated classification problems, other distance metrics can be preferred. For instance, the tangent distance can be used to resolve some problems of the Euclidean distance in handwritten image recognition.

K-NNs have a disadvantage due to its lazy learning method. It does not learn from the training set only use the training data itself for prediction. As it predicts test labels one by one without a common prediction rule, K-NN is known for its computational inefficiency in particular for high-dimensional data such as image classification. In the term of computational complexity using Big O notation, when p is the number of dimensions and n is the number of examples;

- O(p): to compute distance to one
- O(np): to find out the nearest neighbour
- O(knp): to find k nearest neighbours

As a result, O(knp) will be the eventual computational cost for K-NNs. For the large examples and dimensions, it can be extremely expensive in computation. In real-time setting such as for financial trading strategy, K-NNs with a number of features will be slow and might not be suitable. But still, K-NNs are one of the commonly used algorithms with its biggest advantage, simplicity. With a right setting for the right dataset, K-NN would have more predictive power than any complex machine learning models.

2.8 Confusion Matrix

		True Class	
		1	**-1**
Predicted Class	**1**	**T**rue **P**ositive	**F**alse **P**ositive
	-1	**F**alse **N**egative	**T**rue **N**egative

Figure 3: Confusion Matrix

Recall (True Positive Rate): $\frac{TP}{TP+FN}$ False Positive Rate: $\frac{FP}{FP+TN}$

Precision: $\frac{TP}{TP+FP}$ Accuracy: $\frac{TP+TN}{TP+FP+TN+FN}$

Specificity (True Negative): $\frac{TN}{TN+FP}$ F-measure = $\frac{2}{\frac{1}{precision}+\frac{1}{recall}}$

The confusion matrix is a table that shows the number of actual class and predicted class in the test dataset from prediction results. It has been widely used in order to evaluate and compare classification algorithms. A performance of a

machine learning algorithm can be visualised in a confusion matrix with the number of correct or wrong predicted labels. A confusion matrix for binary classification has two rows and two columns that describe false positives, false negatives, true positives and true negatives. If we suppose there are two classes 'Positive (1)' and 'Negative (-1)', false positive means the model predicted it as 1 but the actual class was -1. You can easily assume what the other three mean in the same context based on Figure 3. False positive is called 'Type 1 error' and false negative is called 'Type 2 error'. If there are more than two classes, the table will grow bigger. From a confusion matrix, we can calculate a number of performance metrics such as accuracy, precision, recall(sensitivity), specificity and F1 score. Rather than being indicated by only the accuracy of a model, looking into different metrics can give us a better insight to investigate the classification result.

2.9 ROC curve and AUC

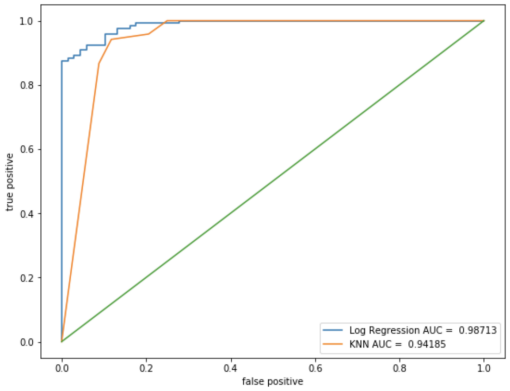


Figure 4: ROC curve

A receiver operating curve (ROC) graph is "a technique for visualising, organizing and selecting classifiers based on their performance" [18]. A ROC graph plots false positive on the x-axis, and true positive on the y-axis. With its simplicity in concept, ROC curve has been widely used in performance measure in diverse industries such as medicines, bioinformatics, psychology as well as of course in statistics, machine learning, data mining and etc. AUC is an acronym of "the area under the ROC curve" in the ROC graph. It is a score corresponding to a ROC curve which can be also used to measure the performance of classification algorithms in machine learning. The value of AUC can be usually between 0.5 and 1. 0.5 means the algorithm is like flipping a coin arbitrarily, there are 50:50 percent chance of false positive or true positive. If a classifier achieves its AUC score becomes 1, that means it is the possible best model. In figure 4, it shows two ROC curves. In the plot, the blue line is the ROC curve drawn from the prediction result of Log regression algorithm and the yellow one is the result from K-NNs. In this small experiment to visualise ROC curves, the breast cancer dataset in the Scikit-learn library in Python. The areas under those two ROC curves, we can assume the log regression model demonstrated the better average performance.

3 Experiments

3.1 Aim

This project aims to empirically analyse the hidden predictive power in historical prices of financial markets with unique features. This experiment has also focused on revealing the price movement patterns across different securities in the same financial exchange. For the goal, we will explore two distinct financial markets, which are stock market and cryptocurrency market. This experiment will address empirical approaches to attain the best prediction result with the diversity of parameter settings for the given datasets. After the best model is generated in the exploration stage, the experiment will be extended to conduct a hypothesis testing to confirm the model's forecasting power.

3.2 Data

3.2.1 Cryptocurrency Market Data

Cryptocurrency Data was extracted from a website called CryptoCompare via REST API requests collecting historical price of currencies. Cryptocompare provides free live pricing data, historical tick and volume on cryptocurrency from over 90 exchanges, 1,800 coins and 22,000 trading pairs listed. It provides free live pricing data, historical tick and volume data. Most of the APIs related to historical pricing is public REST APIs. Thus, I could easily manage to gather useful data for the object of this project. Three APIs, 'Histominute', 'Histohour' and 'Histoday' are mainly used for this project in order to collect historical price data in various time units, minute, hour and day.

Among those 22,000 available trading pairs on Crypto-Compare, our target data was selected based on certain conditions. A pair means a set of two coins or one coin and a fiat currency which are transferable, so they have an exchange rate from one unit to the other in exchanges. The target period of time is between 31 August 2017 and 31 August 2018 for a year, for example, the sample size using HistoDay API is 365. From various exchanges available on Crypto-Compare, Kraken was chosen to exploit its pricing data, because it is one of the top traded exchanges in volume since it launched in July 2011.

Base currency	**Coins**
US dollar (USD)	BTC, LTC, ETC, ETH, BCH, XMR, XRP, ZEC, DASH, REP, GNO, EOS, XLM
Bitcoin (BTC)	ETH, XLM, EOS, BCH, XRP, ETC, XMR, DASH, LTC, MLN, ZEC, REP, ICN, GNO

Table 1: Coins in available to trade in Bittrex

On Kraken, there are 52 markets currently available to trade (11 Aug 2018). From those markets, pairs having their base currency in either BTC or USD were chosen to look into. In order to see if there are any significant differences depending

on the base currency affecting our prediction results. As a result, there were 27 available pairs meeting the requirements. Full historical prices of 13 markets having base currency in US dollar and 14 markets with a base currency in Bitcoin, total 27 currencies were existing for over last 12 months. The available pairs are shown in the below table. As a result, each dataset for USD and BTC having same structures are prepared for the experiment.

We will examine how raw data looks like and see what it consists of in this chapter for the better understanding of the data first, while methods and approaches for data pre-processing and feature engineering will be discussed in the next section. The response of APIs has several variables and the return data is formatted in JSON. It is loaded into Python Pandas dataframe, which is a tabular data structure commonly used in Python for the purpose of data analysis or data science. It supports various operations on both row and column labels.

Features	Description
Close	Close price in each time
High	High price in each time
Low	Low price in each time
Open	Open price in each time
Time	Unix Timestamp of the data point
Volume from	Traded volume at the beginning of time
Volume to	Traded volume at the end of time

Table 2: Features in cryptocurrency price data

As the Table 2 shows above, it returns different 7 features. So, we can get open, high, low, close, volumefrom and volumeto with a Unix time stamp for the historical pricing data in each time unit. These features are price and volume-based features with a timestamp at each data point. This API has a limit for the number of pricing history; consequently, only maximum 2,000 recent data observations are available to under given parameters.

While exploring the data to look into if it has any outliers or null values, many abnormal data observations were detected. To see the price changes, the logarithmic scale was taken on the data. Plotting the data revealed that there was no price change for 4 coins in USD for more than the half of the time period. Therefore, the 4 coins, which are REP, GNO, EOS, XLM, were got rid of in the dataset. As a result, the historical price of 9 currencies in USD was exploited for this project. It will be demonstrated in detail in 3.2.3.

3.2.2 Stock Market Data

The data for historical stock markets were collected from Kaggle public datasets, which is so-called 'Home of Data Science'. The dataset contains prices of 7,195 NASDAQ stocks in USD obtained from Yahoo Finance. It has all available stocks on Nasdaq as of 10 November 2017, arbitrary stocks were chosen among them for this project. The features in the dataset are similar with the cryptocurrency dataset as it also has features about price as well as volume related. The dataset

describes Date, Open, High, Low, Close, Volume, OpenInt, therefore, we could get daily price data from the dataset for the entire history of stocks.

In stock data, there is a distinctive feature compared to cryptocurrency, that is the sector category. Hence, we can label which industry the stock is included in. In order to carry out different experimental approaches, two separate sets of data were processed. Those sets of different stock data were designed to experiment the impact of the number of independent variables and sector correlation of stocks based on their industries. The first set includes major companies in the Technology Industry, so the samples include price data of top 10 firms in order of market cap. The second sample consists of total 20 firms, 10 firms from the tech industry and another 10 firms from the utility industry also according to their market cap. These two industries were chosen by an analysis from Bloomberg about sector correlation in stock markets [20]. The research shows tech and utility industry have the smallest correlation across industries.

Group	Stocks
10 Top tech firms	GOOGL(Google), MSFT(Microsoft), FB(Facebook), T(AT&T), INTC(Intel), VZ(Verizon), TSM(Taiwan Semiconductor Manufacturing), CSCO(Cisco), ORCL(Oracle), CHL(China mobile)
10 Top utilities firms	NEE(NextEra), DUK(Duke Energy), SO(The Southern Energy), D(Dominion), EXC(Exelon), NGG(National Grid), AEP(American Electric Power), SRE(Sempra Energy), OKE(ONEOK), PEG(Public Service Enterprise Group)

Table 3: Stocks in stock data

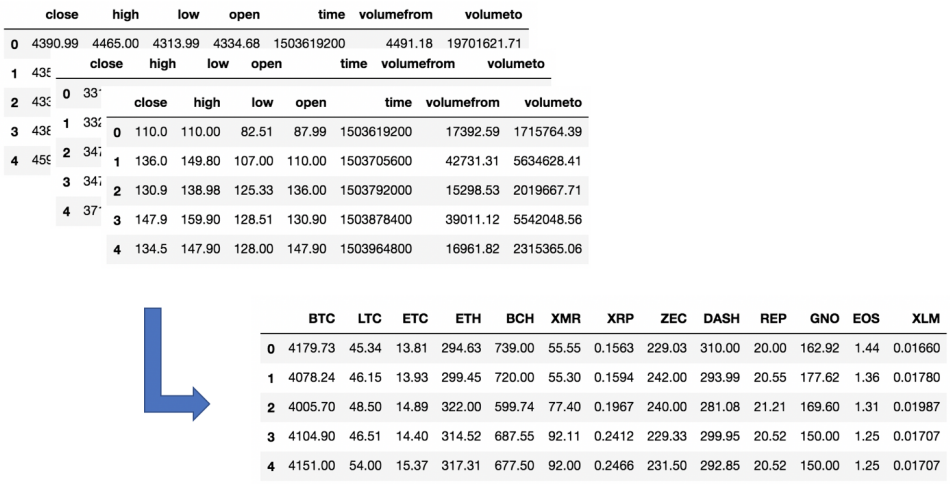
3.2.3 Data Analysis and Pre-Processing

	close	high	low	open	time	volumefrom	volumeto
0	4390.99	4465.00	4313.99	4334.68	1503619200	4491.18	19701621.71
1	4351.75	4392.10	4271.10	4390.99	1503705600	2657.67	11506698.74
2	4338.46	4410.00	4326.00	4351.75	1503792000	2235.46	9730291.34
3	4387.83	4399.00	4188.13	4338.46	1503878400	4683.99	20125265.04
4	4590.17	4665.45	4346.10	4387.83	1503964800	5533.78	25016504.28

Figure 5: Bitcoin Market Data in USD

Data collection for both cryptocurrency and stock markets is done and the initial datasets are ready to roll for actual experiments now. In this section, it will be demonstrated what sorts of data pre-processing have been done. After data extracted via REST APIs and collected from reliable public source, Kaggle and cross-checked with its original source Yahoo Finance, total 3 data sources are now prepared. All process to pre-process and mine data applied to three data sources equally. The same structure of dataset depicted in Figure 5 was prepared for each coin in USD price and Bitcoin price. Consequently, total 27 initial datasets for cryptocurrency were prepared. To attain integrated price information from those data, 'close' price was chosen for this purpose. Producing the integrated dataset was

done by development of a function. The process is depicted in Fig 6; All implemented codes in this experiment are also available to find on my Github repository. You can see the codes and visualisations without any pre-installation, how to use the code can be found out in chapter 7.



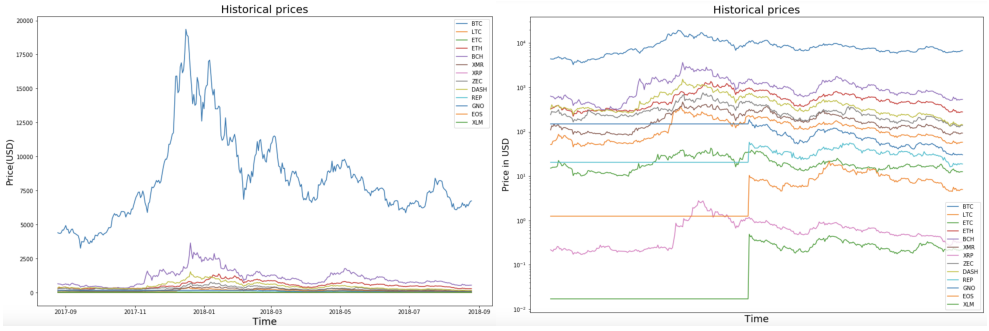
	close	high	low	open	time	volumefrom	volumeto
0	4390.99	4465.00	4313.99	4334.68	1503619200	4491.18	19701621.71

	close	high	low	open	time	volumefrom	volumeto
0	110.0	110.00	82.51	87.99	1503619200	17392.59	1715764.39
1	136.0	149.80	107.00	110.00	1503705600	42731.31	5634628.41
2	130.9	138.98	125.33	136.00	1503792000	15298.53	2019667.71
3	147.9	159.90	128.51	130.90	1503878400	39011.12	5542048.56
4	134.5	147.90	128.00	147.90	1503964800	16961.82	2315365.06

	BTC	LTC	ETC	ETH	BCH	XMR	XRP	ZEC	DASH	REP	GNO	EOS	XLM
0	4179.73	45.34	13.81	294.63	739.00	55.55	0.1563	229.03	310.00	20.00	162.92	1.44	0.01660
1	4078.24	46.15	13.93	299.45	720.00	55.30	0.1594	242.00	293.99	20.55	177.62	1.36	0.01780
2	4005.70	48.50	14.89	322.00	599.74	77.40	0.1967	240.00	281.08	21.21	169.60	1.31	0.01987
3	4104.90	46.51	14.40	314.52	687.55	92.11	0.2412	229.33	299.95	20.52	150.00	1.25	0.01707
4	4151.00	54.00	15.37	317.31	677.50	92.00	0.2466	231.50	292.85	20.52	150.00	1.25	0.01707

Figure 6: Integration phase of price data

After this initial pre-processing phase, we can apparently explore the price data at a look easily. In unclean public datasets, there are likely to have a few outliers or missing values. To investigate if there are any unusual data point such as some outliers or missing values included, I visualised time-series plots as well as box plots for each coin. In terms of the historical price of cryptocurrencies, in a linear metric, it is hardly able to detect anything specific apart from the dominance of Bitcoin. So, a log-log plot was depicted to visualise the vivid price change of each cryptocurrency from time to time. In this stage, 4 odd lines were detected in Fig 7. Although it was confirmed all coins had historical price data for the given period, the log plot showed during almost a half of time prices of 4 coins (GNO, REP, EOS, and XLM) in USD have not changed at all. After closely looking into data again, it was discovered those 4 coins have not been traded in USD once it was introduced to the exchange.



(a) Linear Prices in USD (b) Log Prices in USD

Figure 7. Historical price of crypto-coins in USD from August 2017 to August 2018

More plots for data visualisation of historical crypto-price data can be found in the execution of the **'DataProcessing_crypto.py'** code.

The next phase in data pre-processing was to turn continuous price data into categorical value either "1"(up) or "-1"(down). Each example was compared to the next row's price data as copying and shifting the next row's prices to the same row. Then based on the comparison, it got a new column addressing the price movement from the current time point to the next. To make the 'price movement' column have two categorical values, if the current price and the next price is same 1(up) was assigned to the instance as the instances having 0 was quite tiny proportion except for the case of 4 coins which were not traded for a long term as mentioned previously. Figure 8 shows the counts of observations in each price movement case. Fortunately, the data shows evenly distributed categorical values without much imbalance. Also, in the ends, the 4 coins were removed from the dataset for data cleansing purpose. The resulting dataset looks like below Figure 9. Other datasets for cryptocurrency price movement in BTC and stock price movement also processed in the same approaches.

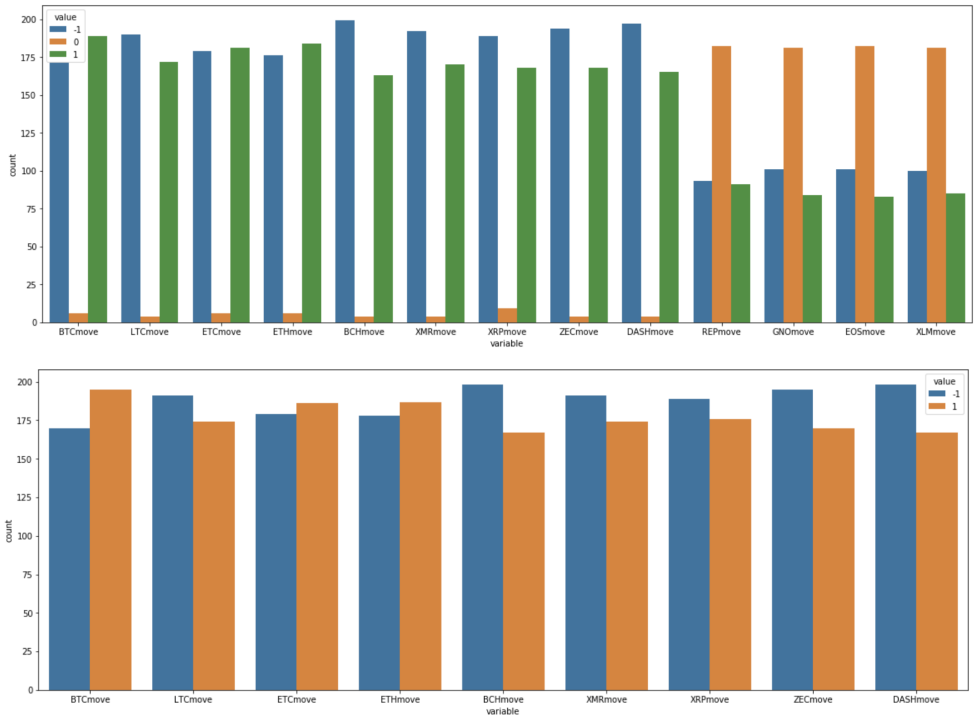


Figure 8. Histogram of price movement values for each coin

	BTCmove	LTCmove	ETCmove	ETHmove	BCHmove	XMRmove	XRPmove	ZECmove	DASHmove
0	-1	-1	1	1	-1	1	-1	1	1
1	-1	1	1	1	-1	-1	-1	-1	-1
2	1	1	-1	-1	-1	1	1	1	-1
3	1	1	1	1	-1	-1	-1	-1	1
4	-1	1	-1	1	1	-1	1	1	1

Figure 9. Resulting dataset after pre-processing

3.3 Analysis and Results

This experiment firstly focuses on the prediction of the directions of Bitcoin (BTC) price change in the cryptocurrency market. The prediction label is categorized as "1" or "-1" in the samples as described in the data pre-processing section. "-1" means that the next time unit's BTC price is lower than the current price, and "1" means the next price is higher than now. In order to build the best machine learning model, diverse factors which can potentially affect to the prediction performance have been empirically explored. While pure price directions across markets were considered as attributes in the dataset, traditional technical indicators were not exploited. Defining the direction of crypto-coins' price as features was a straight-forward but a novel approach.

There are two major hypothesises in this study; First is to validate the correlation across different crypto-coins simultaneously and uncover their forecasting power. Second is to address the predictability of price movement with its historical price of a currency with different samples and parametric setting. Furthermore, this experiment will be extended to see how the model built performs in the stock market. Finally, it will be demonstrated how statistically significant the model is. In all experiments in chapter 3, a validation set approach was adapted for clear visualisation in this report. Training and test set split ratio is 70:30 or a reasonable ratio close to 70:30.

3.3.1 Approach 1: Patterns in other cryptocurrencies

[Phase 1.1]

- Label: Price movement of BTC at the same time step
- Features: Price movements of other cryptocurrencies
- Time-unit: Day, Hour, Minute

For a starter, a simple approach to show if there is a clear correlation among different cryptocurrencies and if they have predictive power. K-NNs was randomly chosen to conduct this initial test. Concerning dataset size, the maximum available historical price data for each time unit was collected from Crypto-compare, which are 2,000 for the hour and minute API, 365 for day API. The number of rows from day API also has a maximum limit of 2,000 though, as cryptocurrency markets do not have enough daily price data for 2,000 days due to its short history, the period for daily API was chosen for 365 which is a year. Due to three available choices of time units in cryptocurrency data from Crypto-compare API (day, hour, minute), a decision for which time-unit to examine was required to be made first. The daily price dataset has 365 observations, and the validation set size is 110. The minute and second datasets have 2,000 total samples and 600 observations in the validation set. After testing all of the datasets collected from the three APIs, below results were demonstrated.

It seems the price movement in BTC reflects other cryptocurrencies movements happening in parallel especially in days and hours but not in minutes. It was assumed each minute might be relatively too short to reflect any price movement patterns across different currencies as the movements are rather noisy. In terms of accuracy for each case, each confusion matrix with the prediction

accuracy on different time-units, days, hours and minutes are shown in Figure 10. As we can confirm there are certain relations in currencies from this result, I decided to investigate the correlation across crypto-coins for the next time unit.

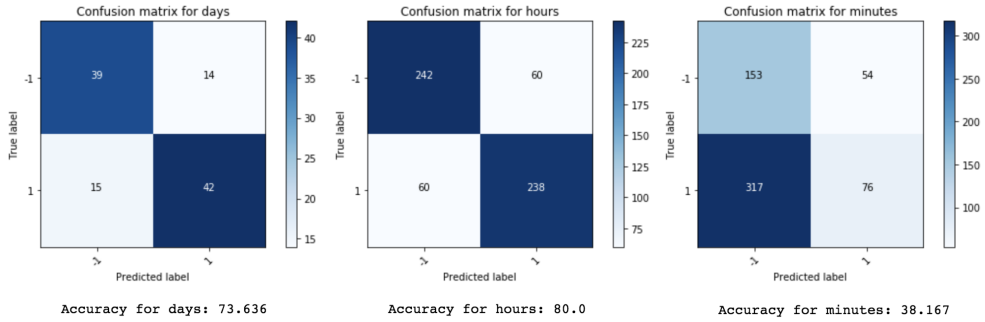


Figure 10. Confusion matrix of prediction in the current time step

[Phase 1.2]

- Label: Next price movement of BTC
- Features: Price movements of other cryptocurrencies
- Time-unit: Day, Hour, Minute

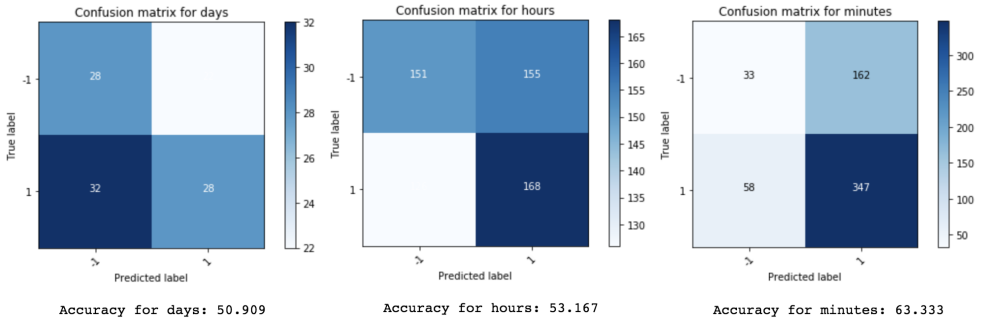


Figure 11. Confusion matrix of prediction in the next time step

From the result shown in Figure 11, it is contrary to the outcome of previous phase. Although there have been found quite strong price movement patterns in the parallel time horizon, there was no strong predictive power across diverse currencies in short-term forecasting for the next time unit. Moreover, I extended this experiment to show if there are co-movements of crypto-coins in aggregate price movement from different input time horizon. Three different input time horizons, 5, 10, 15 were tested but it was not effective to improve the prediction accuracy. As a result, it was hardly able to find any evidence that price movements of different coins have patterns over times. Nevertheless, there is certain correlation in movements of crypto-coins in parallel.

3.3.2 Approach 2: Aggregate past movements

This approach is designed to examine if there are patterns in past price movements of a currency. Unlike approach 1 that took an attempt to predict the next price movement of BTC based on price movements across different coins, in approach 2 it was investigated to show the predictability within one coin's historical price data. In this approach as well, BTC price was the label to predict. If you look into the dataset in Figure 12, it is processed as to aggregate the last 'n' time-units price movements. For example, in the table, column 'BTC5' means the sum of the last 5 price movements at the given time. Like the preceding, the other columns are aggregated in the same manner. As the price information was turned into discrete integer value '1' or '- 1', this approach was available to calculate the rolling sum of past price movements in the given time window.

	BTC1	BTC2	BTC3	BTC4	BTC5	BTC6	BTC7	BTC8	BTC9	BTC10	...	BTC21	BTC22	BTC23	BTC24	BTC25	BTC26	BTC27	BTC28	BTC29	BTC30
0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	...	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1	1.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	...	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
2	-1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	...	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
3	1.0	0.0	1.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	...	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
4	1.0	2.0	1.0	2.0	3.0	3.0	3.0	3.0	3.0	3.0	...	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0
5	-1.0	0.0	1.0	0.0	1.0	2.0	2.0	2.0	2.0	2.0	...	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
6	1.0	0.0	1.0	2.0	1.0	2.0	3.0	3.0	3.0	3.0	...	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0
7	-1.0	0.0	-1.0	0.0	1.0	0.0	1.0	2.0	2.0	2.0	...	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
8	1.0	0.0	1.0	0.0	1.0	2.0	1.0	2.0	3.0	3.0	...	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0
9	1.0	2.0	1.0	2.0	1.0	2.0	3.0	2.0	3.0	4.0	...	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0

Figure 12. Pre-processed data set for approach 2

[Phase 2.1]

- Label: Next price movement of Bitcoin in USD ('BTC1' in Fig 12)
- Features: Sum of price movements in last 'n' time-units
- Time-unit: Day
- Total number of days (d): 365
- Training sample size (t): 265
- Validation sample size (m): 100
- Number of features (p): 1, 2, 3, …, 30

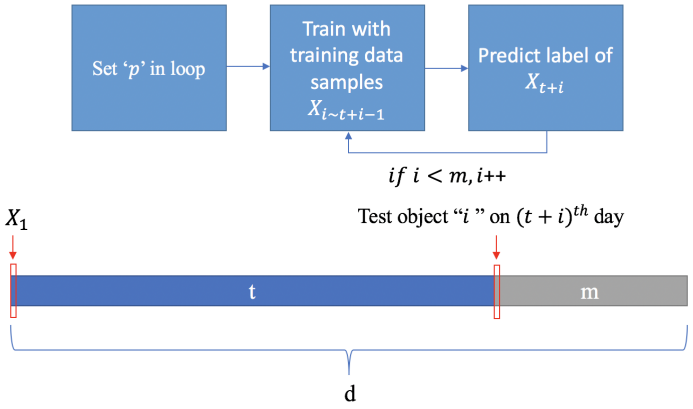


Figure 13. Architecture of training and validation set

In this experiment, the model test is implemented in a different way with the approach 1. Not only splitting the dataset into a training and validation set, this experiment was executed with a hypothetical on-line learning method. The hold-out data was used to run the trading simulation for "m" days hypothetically. As a result, there are total "m" models built and each model trains with data between X_i and X_{t+i-1}, while X_i is a training example. The test data consists of only one test example X_{t+i}. Thus, each model predicts predict the Bitcoin price direction on $(t+i)^{th}$ day. The prediction performance term P is defined by the following equation; PO_i is the predicted outcome from the model for the i^{th} trading day, and TO_i is the actual outcome. m is the number of test examples.

$$P = \frac{1}{m}\sum_{i=1}^{m} R_i \; (i = 1, 2, \dots, m)$$

$$Prediction\left(X_{i,t+i-1}\right) = \; PO_i = \begin{cases} 1 \\ -1 \end{cases}$$

$$R_i = \begin{cases} 1, & if \; PO_i = TO_i \\ 0, & otherwise \end{cases}$$

As the architecture of this experiment depicted in Figure 14, this one-day-ahead forecasting was implemented to show the impact of varying rolling window size 'p' (It differs with performance term P) of the historical price. To investigate how far back of the past aggregate movements predicts the highest prediction accuracy, from 1 to 30 days input window length were tested in a loop as shown in Figure 14. Therefore, the number of features incremented like below. As a result, total 30 models were built by the time window and each model tested for 100 test cases moving its cursor from i to m, which is the number of hold-out examples.

```
['BTC1']
['BTC1', 'BTC2']
['BTC1', 'BTC2', 'BTC3']
['BTC1', 'BTC2', 'BTC3', 'BTC4']
['BTC1', 'BTC2', 'BTC3', 'BTC4', 'BTC5']
['BTC1', 'BTC2', 'BTC3', 'BTC4', 'BTC5', 'BTC6']
['BTC1', 'BTC2', 'BTC3', 'BTC4', 'BTC5', 'BTC6', 'BTC7']
['BTC1', 'BTC2', 'BTC3', 'BTC4', 'BTC5', 'BTC6', 'BTC7', 'BTC8']
['BTC1', 'BTC2', 'BTC3', 'BTC4', 'BTC5', 'BTC6', 'BTC7', 'BTC8', 'BTC9']
['BTC1', 'BTC2', 'BTC3', 'BTC4', 'BTC5', 'BTC6', 'BTC7', 'BTC8', 'BTC9', 'BTC10']

   ...

['BTC1', 'BTC2', 'BTC3', 'BTC4', 'BTC5', 'BTC6', 'BTC7', 'BTC8', 'BTC9', 'BTC10', 'BTC11', 'BTC12', 'BTC13', 'BTC14',
'BTC15', 'BTC16', 'BTC17', 'BTC18', 'BTC19', 'BTC20', 'BTC21', 'BTC22', 'BTC23', 'BTC24', 'BTC25', 'BTC26', 'BTC27',
'BTC28', 'BTC29', 'BTC30']
```

Three machine learning algorithm, K-NNs(k=3), SVM with Gaussian kernel and Naïve Bayes are implemented towards the dataset. From the result, it was able to see that after a few features added first, the prediction accuracy was decreased by adding more further past data. By that means, adding too far back data is likely to decrease forecasting performance to some extent. The plot in Figure 14 provides useful insight into the correlation between the accuracy and the number of features, it clearly shows the decreasing trend depending on the number of features in time-series. The result let us interpret that the market price is influenced by the patterns

of short-term recent price movements only. Surprisingly, Naïve Bayes outperformed KNN and SVM with a noticeable difference. The highest prediction accuracy 63% was achieved from the Naïve Bayes model with 5 and 6 features which are recent aggregate price movements from 1 to 5(or 6) previous days'. In other words, the model learnt the most useful patterns for a day-ahead-forecast from the last 5 days. You can find the classification rate data of Figure 14, 15 in Table 4.

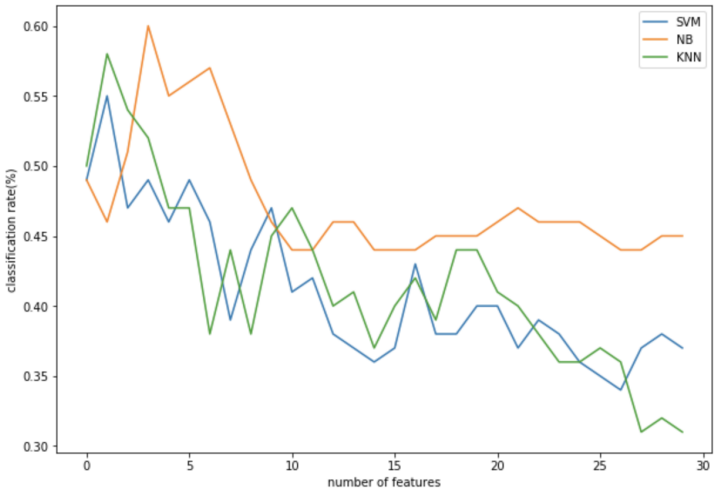


Figure 14. BTC price movement prediction results

As some degree of predictability was found in the experiments, I decided to extend this approach to a different market. I applied the same method to predict Bitcoin price with different time-units, hours and minutes, but the results were rather too noisy to find a pattern and it seemed the model didn't learn much from the input data.

To look into the feasibility of forecasting power of this model in cryptocurrency market with further experiments, the method has been extended to implement on another cryptocurrency market. The target cryptocurrency to predict is Ethereum (ETH) and the base coin is in Bitcoin. Unlike the preceding test cases were targeting the market with the base in USD, which is a fiat money, in this approach the base currency in the pair to predict is Bitcoin. It drew attention to address which market has more degree of predictability in those two different cryptocurrency markets.

[Phase 2.2]

- Label: Next price movement of Ethereum based in BTC
- Features: Sum of price movements in last 'n' time-units
- Time-unit: Day
- Total number of days (d): 365
- Training sample size (t): 265
- Validation sample size (m): 100
- Number of features (p): 1, 2, 3, …, 30

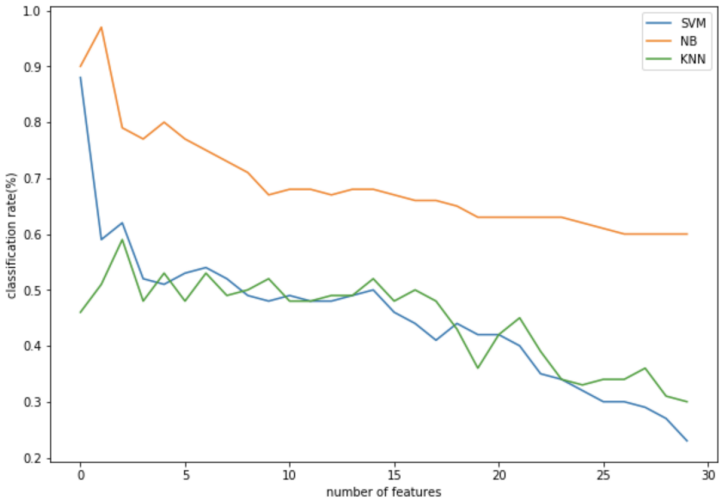


Figure 15. ETH price movement prediction results

In Figure 15, the highest prediction accuracy of 89% was observed with SVM with 1 feature, the latest price movement only. The result also described similar outcome trends which are decreasing of classification rate according to the increasing number of features. Although the best accuracy was observed from SVM, in terms of the overall prediction performance, Naïve Bayes also outperformed as previous. Moreover, Naïve Bayes demonstrated its own best result with the past 6 days price movement patterns for one-day-ahead forecasting which is the same number of features as in the experiment of phase 2.1. The improvement of classification rate observed in forecasting for the ETH_BTC pair is quite remarkable as compared to the forecasting result of the BTC_USD market.

3.3.3 Approach 3: Application of approach 1 to stock markets

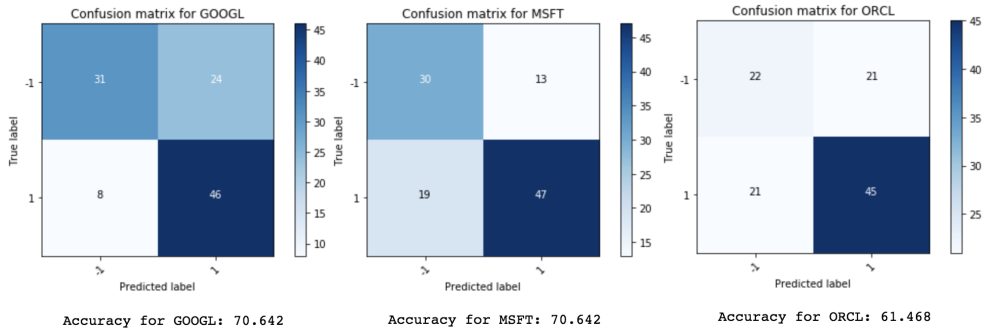


Figure 16. Confusion matrix of classification result in the current time step

In this section, we will apply the previous two approaches into stock markets to examine how the models perform differently within the stock market. One of the most distinguishable differences between stock market and cryptocurrency market is that cryptocurrency market is continuous for 24 hours. On the contrary, stock market opens and closes at a certain time and don't open on

weekend. Consequently, this discontinuity existing in the open and close prices in stock markets over time would impact its forecasting result.

Three prediction tests for three different stocks have been implemented. Due to the dataset was collected from Yahoo Finance's historical daily price, the time unit is 'day'.

$$C_i = \begin{cases} 1, & if\ CP_i \geq OP_i \\ -1, & otherwise \end{cases}$$

The label in each test is the price movement of 'GOOGLE', 'MSFT' and 'ORCL' respectively from the left panel. Features are the rest of 9 stocks described in Table 3 except the label stock. The movement was classified as shown in the equation above. C_i is the direction of price movement on day "i". CP_i is the close price on the i^{th} day. The available days of the stock market data sample are much more than cryptocurrency markets as the stock market has more historical data over time. But to give the same size of data as cryptocurrency forecasting, here also the total trading days in the dataset is 365, which is not actually one year unlike cryptocurrency because stock markets don't open on weekends. The total sample was split into 70:30 ratio of training and test set.

Concerning of the experimental results, it seems obviously similar with the preceding result of the approach 1. The classification rate of prediction using other stocks movements at the same time step is approximately 10-20% higher than the prediction for the next time step. To that ends, from the obtained performance, we can confirm the directional price movement of each stock price in markets has a certain level of relevance in parallel time. But the comprehensive outcome shows it was not very much feasible to obtain a high classification rate for the next time step based on other stocks movement observations at the previous time step.

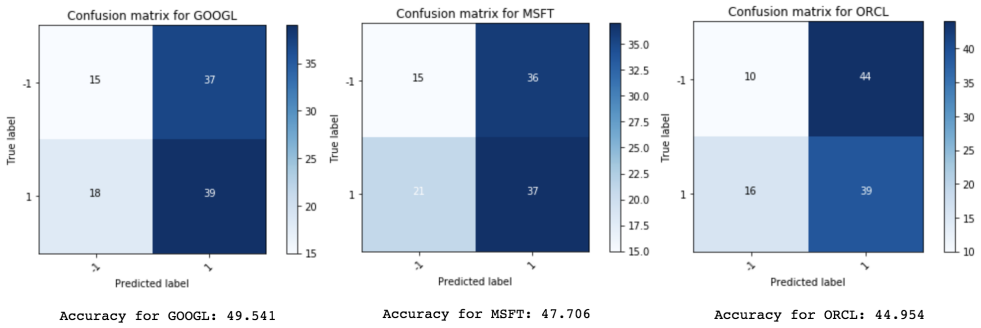


Figure 17. Confusion matrix of classification result in the next time step

The experiment has been carried out using price data in top 10 tech firms. Meanwhile, to exploit the industry information which is available from the stock market data, for next I tried to forecast the 'GOOGL' directional movement in the current and next time unit based on 10 other stocks directions in utility industry with the given method. As it was described in the data chapter previously, tech and utility industry are the least correlated industry. Therefore, the forecast result is supposed to have a lower prediction performance. In Figure 18, the assumption is affirmed as

true. The left panel is for the classification at the same time step and the right panel is the outcome for the forecasting the next time step's movement. Interestingly the accuracies resulted the same rate in both cases.

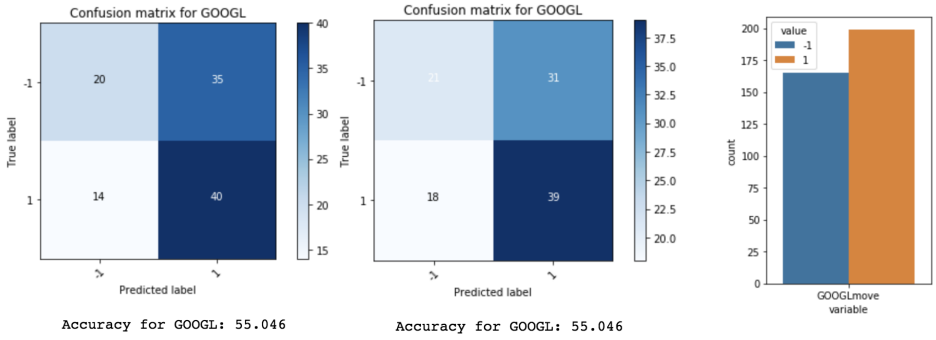


Figure 18. Confusion matrix of classification result using different industry's price data (Left) & the number of labels in GOOGL (Right)

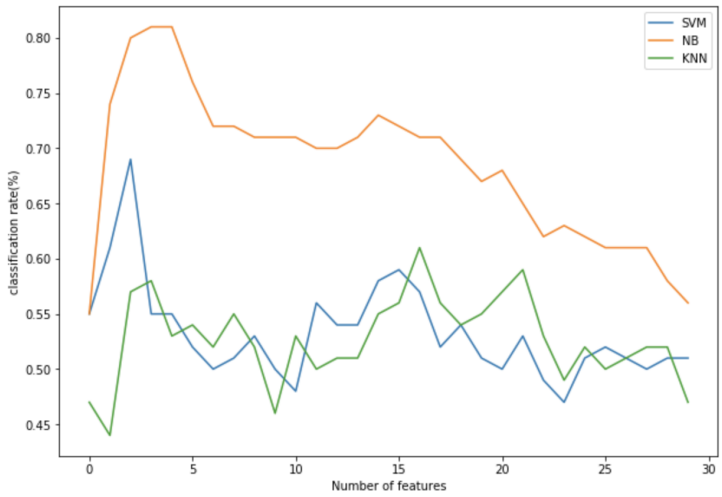


Figure 19. 'GOOGL' stock price movement prediction results

3.3.4 Approach 4: Application of approach 2 to stock markets

This study compares the prediction performance in stock markets with respect to the same three machine learning algorithms, SVMs, Naïve Bayes and K-NNs depending on the number of input features, which are previous directions of price movements as tested in approach 2 for cryptocurrency markets. The outcome from the approach 2 addressed more promising forecasting power than approach 1. Hence, the application of approach 2 to stock markets had a more optimistic expectation compared to approach 1. The same method as described in approach 2 has been applied to the daily stock market price data. Overall results of the experiment are quite convincing as shown in Figure 19. The method affirmed Naïve Bayes has absolute outperforming results with its highest accuracy of 82%. Table 4 presents the prediction performance of algorithms in different datasets.

Data	BTC_USD crypto			ETH_BTC crypto			GOOGLE stock		
Counts	+1s: 51		-1s: 49	+1s: 40		-1s: 60	+1s: 56		-1s: 44
(*)	45%			46%			49%		
p	SVM	NB	KNN	SVM	NB	KNN	SVM	NB	KNN
1	0.49	0.49	0.5	**0.88**	0.9	0.46	0.55	0.55	0.47
2	**0.55**	0.46	**0.58**	0.59	**0.97**	0.51	0.61	0.74	0.44
3	0.47	0.51	0.54	0.62	0.79	**0.59**	**0.69**	0.8	0.57
4	0.49	**0.6**	0.52	0.52	0.77	0.48	0.55	**0.81**	**0.58**
5	0.46	0.55	0.47	0.51	0.8	0.53	0.55	**0.81**	0.53
6	0.49	0.56	0.47	0.53	0.77	0.48	0.52	0.76	0.54
7	0.46	0.57	0.38	0.54	0.75	0.53	0.5	0.72	0.52
8	0.39	0.53	0.44	0.52	0.73	0.49	0.51	0.72	0.55
9	0.44	0.49	0.38	0.49	0.71	0.5	0.53	0.71	0.52
10	0.47	0.46	0.45	0.48	0.67	0.52	0.5	0.71	0.46
11	0.41	0.44	0.47	0.49	0.68	0.48	0.48	0.71	0.53
12	0.42	0.44	0.44	0.48	0.68	0.48	0.56	0.7	0.5
13	0.38	0.46	0.4	0.48	0.67	0.49	0.54	0.7	0.51
14	0.37	0.46	0.41	0.49	0.68	0.49	0.54	0.71	0.51
15	0.36	0.44	0.37	0.5	0.68	0.52	0.58	0.73	0.55
16	0.37	0.44	0.4	0.46	0.67	0.48	0.59	0.72	0.56
17	0.43	0.44	0.42	0.44	0.66	0.5	0.57	0.71	0.61
18	0.38	0.45	0.39	0.41	0.66	0.48	0.52	0.71	0.56
19	0.38	0.45	0.44	0.44	0.65	0.43	0.54	0.69	0.54
20	0.4	0.45	0.44	0.42	0.63	0.36	0.51	0.67	0.55
21	0.4	0.46	0.41	0.42	0.63	0.42	0.5	0.68	0.57
22	0.37	0.47	0.4	0.4	0.63	0.45	0.53	0.65	0.59
23	0.39	0.46	0.38	0.35	0.63	0.39	0.49	0.62	0.53
24	0.38	0.46	0.36	0.34	0.63	0.34	0.47	0.63	0.49
25	0.36	0.46	0.36	0.32	0.62	0.33	0.51	0.62	0.52
26	0.35	0.45	0.37	0.3	0.61	0.34	0.52	0.61	0.5
27	0.34	0.44	0.36	0.3	0.6	0.34	0.51	0.61	0.51
28	0.37	0.44	0.31	0.29	0.6	0.36	0.5	0.61	0.52
29	0.38	0.45	0.32	0.27	0.6	0.31	0.51	0.58	0.52
30	0.37	0.45	0.31	0.23	0.6	0.3	0.51	0.56	0.47

(*) Classification performance of the naive model in Approach 5 for datasets

Table 4. One-day-ahead prediction results for cryptocurrency and stock markets

Apart from accuracy, another performance metric, the ROC-AUC curve demonstrated the performance of models with $p = 3$. In this stage, there was an odd performance observed from the SVM model. The SVM model showed a very low AUC value of 0.279 although the corresponding accuracy is 0.69. As the output class in the test set is well balanced, the code was also speculated to find. However, as the ROC curves of NB and KNN depict reasonable results, it was hard to have a doubt on the code. From further research to find a reason under this, there were a number of claims by researchers including Lobo at al [21], that AUC can be a misleading

measure by its characteristic that it ignores the predicted probability values and the goodness-of-fit of the model [21];

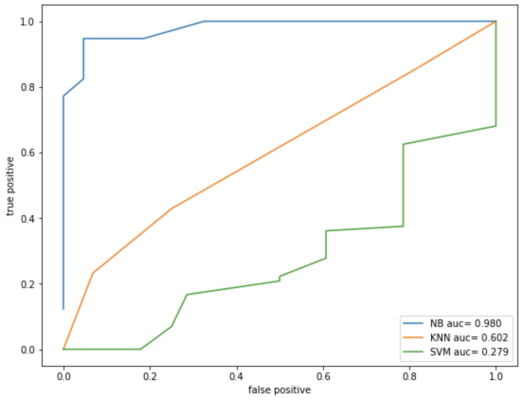


Figure 20. the ROC-AUC curve of 'GOOGL' stock price movement prediction with p=3

3.3.5 Approach 5: Statistical significance test with a naive model

According to Fama's theory, financial markets are unpredictable, and it cannot be reasonably predicted with any approaches. On the contrary, we affirmed a certain degree of predictability exists in both the cryptocurrency and stock markets. To validate the theory, a naive model will be compared to our prediction model. The naive model is often called the naive forecast, or a persistence model. With the given assumption that no excess return is achievable with any prediction model as the market is unpredictable, the best naive classifier I can make would be to use the observation at the previous time as the next price movement. Consequently, this naive model predicts what will happen in the next time step based on the latest observed price movement over time. The result of this naive approach for each dataset is shown in Table 4 (*). I will demonstrate if the prediction model is significant enough to reject the null hypothesis "The prediction performances from the best prediction rule and the naive approach are actually the same in the stock market". In this comparison to calculate p-values, the dataset of 'GOOGL' prediction given in approach 4 is used. Figure 21 depicted the results. Algorithm 1 is the Naïve Base model with 4 input features which performed the best for 'GOOGL' stock prediction. Algorithm 0 is the naive approach for the same forecasting problem.

		Algorithm 1	
		Correct	Wrong
Algorithm 0	Correct	49 (A)	0 (B)
	Wrong	32 (C)	19 (D)

Figure 21. 'GOOGL' stock price movement prediction results

Total number of observations: 49 + 0 + 32 + 19 = 100
Algorithm 0's error rate: (32+19)/100 = 51%
Algorithm 1's error rate: (0+19)/100 = 19%

$$P(T \leq t_0),\ t_0 = 0 \qquad (4)$$

binom_test(0, 32, ½, 'less') = 0.00000000023283064365386962890 6 (5)

I tested the null hypothesis that the prediction result is random and there is 50:50 chance that our model predicts better than the naive approach. Under that assumption, we can interpret the difference in error rates from two models could be just a 'statistical fluke' [19]. In other words, the probability that the algorithm 1 predicts the right label and algorithm 0 predicts the wrong label is 50%, vice versa. So, I focused on the cell B and C in Figure 21. To compare these two models, it was addressed that how rare the event of algorithm 0 got wrong 32 and algorithm got wrong 0 total out of 32 observations. With the given 50% chance under the null hypothesis, it can be restated as "Can we get 0 heads in 32 tosses of a fair coin?" [19]. It seems an extremely rare event. To find out the statistics under the question, I calculated the p-value of it as (4). As a result, the value in (5) is the p-value of $P(T \leq 0)$. With a chosen significance level of 1%, we can easily reject the null hypothesis, as the p-value corresponding to our observation of 0 head out of 32 coin-tosses is highly statistically significant. The result (5) was calculated in Python using "binom_test" function in scipy.stats library.

4 Conclusions

In this project, the main goal was to prove there are predictive powers with hidden features in the historical price of financial markets. The empirical experiments present that there is a certain predictive power in historical price data. There are two major underlying hypothesises focused on the experimental phases.

The first investigation in this study was focused on the dependency of the price movements across different securities (stocks or cryptocurrencies). To analyse the correlation of price movements patterns across various stocks or cryptocurrencies, the experiment in approach 1 was designed to predict the next price movement of BTC using other cryptocurrencies price movements as features. The following classification result is observed: when we predict for the label in parallel time steps, the highest performance was achieved in the time unit of hours. However, the presence of the price movement patterns across different cryptocurrencies wasn't applied to forecasting in the next time step. The patterns were also investigated in stock markets in approach 3 to validate the findings. It resulted in the similar outcome as cryptocurrency markets, as it proved links of stocks' price movements in the same time step to some degree but not in the next time step forecasting. Furthermore, experiments have been also conducted to demonstrate if the pattern exists in stocks across different industries. Consequently, the result shows that the link of price movement patterns only appears in the same industry.

The second experimental design is forecasting the one-day-ahead price movement based on a number of aggregate movements of historical price. The experiments discovered a strong correlation between the direction of price change based on the recent price movement patterns of a security. Each different rolling window size of historical price movements yields a feature to forecast the price movement of target label. The parameter, maximum window size, was investigated to find the best performing model. From the result of approach 2 and 4 summarised in Table 4, it seems certain that there is forecasting power in recent patterns of price movements. Efficient market theory claimed past price has no relationship to future price movement, thus it is not available to achieve excess return with predictive systems. Contrary to the claim, our forecasting models using historical price patterns clearly outperform in different financial markets over the simple naive approach. In a sense, we can see the market has a certain short-term predictability.

Moreover, we compared three machine learning classifiers under the hypothesis. While Naïve Bayes has shown to be the top performer, it was followed by SVM and K-NNs in order. Naïve Bayes has performed the top in many domains but in financial market prediction, it has not been focused a lot. I recommend Naïve Bayes as a good predictor in financial markets. Due to the attribute of Naïve Bayes that it only learns by a simple hypothesis function, it makes a high bias and low variance model. In other words, it adopts only representative observations in training data and ignores unrelated information. As a result, the Naïve Bayes classifier performs surprisingly well with small, insufficient training data sets than other classifiers. By that means, the given design of feature engineering in the experiments showed Naïve Bayes doesn't do overfitting compared to K-NNs and SVM when we add more irrelevant, stale input features from old price history. In

the meantime, KNN showed the lowest accuracy in every dataset with similar performance.

The last focus of this empirical research is about the different degree of predictability in various markets. The approach 2 models forecast three different financial markets in stocks and cryptocurrencies. The forecasting targets are BTC/USD, ETH/BTC price on a crypto exchange 'Kraken' and GOOGL/USD price on NASDAQ. The difference between BTC/USD and ETH/BTC is whether the pair is crypto/fiat or crypto/crypto. The underlying idea is to demonstrate either cryptocurrency or stock market has more forecasting power. Crypto-currency markets are emerging and rather unstable with a base of its anonymity. So, the markets are often hugely affected by false information on SNS. The initial assumption was the degree of predictability would be distinctly different in those two markets. However, the actual result was rather unexpected. While the classification models (NB, SVM) for ETH/BTC has outperformed BTC/USD, performance in the stock market was about 15-20% lower than ETH/BTC forecast. As a result, it is hard to clearly interpret if stock or crypto market has more predictability. Also, we can say both markets correct their inefficient prices quite quickly because we found only short-term predictability over time in historical price.

For an extension of this study, it will be interesting to present the classification performance difference from various crypto-fiat and crypto-crypto as well. What attributes of those markets are making the difference will be an exciting topic to build an improved forecasting model. Furthermore, the poor performance in the ROC-AUC of the SVM model contrary to its fairly good accuracy in approach 4 would have been investigated, but it was not within the scope of this experiment.

5 Self-Assessment

This chapter will demonstrate how the process of this individual project has gone and my progress via this study.

5.1 Contribution of placement

The idea of this project first came across me when I was doing experimental research for cryptocurrency market in my placement job as a quantitative analyst at Lindgren Laboratories. I was developing algorithmic trading bots while I quickly cycle a number of hypothesises through discovery phases of exploratory analysis. As the placement project was focusing on the traditional market making at that moment, there was no chance to develop further this specific idea. In the ends, the initial finding captivated me to extend that idea into this thesis. Apart from getting initial insights, the placement work contributed me to finish this project in Python, which I am self-taught through the placement year. As a software engineer, I have had my background mainly in Java. R and MATLAB that I have learned from my MSc taught modules are incredibly useful analytical programming languages though, Python is one of the most popular data science languages in this era. By that means I have started to develop every work in my placement job in Python from a few months ago, although I was inexperienced in Python. The placement work became a stepping stone that I could gain analysis and development skills in Python particularly with statistical machine learning and plotting libraries. After carrying out this project I could firmly achieve my skill-sets in practice.

5.2 Progress of project

It was my first time to dedicate myself to an academic project on my own with this much of a substantial amount. I have learnt a lot about planning and executing a project from this experience. First, I established the embryonic structure of contents in this thesis. Microsoft OneNote is my favourite work management tool. It was used to manage the contents of this thesis making respective notes. I collected and documented relevant resources I found out in each online note. It helped me a lot to follow my project plan efficiently.

The start of this research was just a rough idea via exploratory analysis without any hypothesis design. As I didn't study financial computing or econometrics previously, I was with lack of knowledge in underlying theories. Once I submitted my first abstract to propose this project, Dr. Kalnishkan suggested me to read resources such as Eugene Fama's efficient market hypothesis and research papers about technical analysis. I was happy to learn through the rightful resources. Moreover, those resources helped me a lot to think of how to organise the architecture of this experimental analysis and how to approach to this given idea.

After the phase of the literature review, I started to work on data validation and pre-processing. In the meantime, I realised my initial finding from the placement work about forecasting power in the crypto market was under a mistake. The reason was that I had been using the validation set to train models unintentionally. As a result, possible ideas from the finding have had dead ends. I

needed to figure out other ideas to build models and test after I have already spent a few weeks to develop the initial models. Eventually, I managed to design new approaches with a number of trial and error. I was able to work on next steps but the whole progress was a little bit behind schedule I planned. In the ends, the new approaches show more promising results than the initially designed approach. It was such a rewarding experience to design experiments, test hypothesises and validate findings iteratively in order to understand new concepts and come up with an original solution to a given problem.

6 Professional Issues

In this section, I would like to propose professional issues and concerns I have encountered during this project.

6.1 Data Reliability and Limitation

As I exploited public data sources such as open source APIs and Kaggle datasets, one of the concerns was how reliable the data are. Public data such as these might have flaws or deliberate corrections. In particular, as the given topic was forecasting problem based on time series data, it was important to be certain if there are any wrong data or huge time difference at observations in the datasets. Thus, it was necessary to cross-check if the values are identical between the actual market price in the exchange 'Kraken' and returned price data from the 'Crypto-Compare' public open APIs, as well as Yahoo finance data and Kaggle data sets. It can be also a practical issue of this approach based on predictive models as suggested in this study. Although the prediction accuracy is impressive, in practice, this unreliability of data might highly affect to the ability to achieve excess returns in trading strategies.

Another issue was the limitation of data. There are few public financial data sources, but there are limitations to the available amounts of data. For example, Crypto-Compare APIs only maximum 2,000 observations. In the meantime, Yahoo Finance data was only available to historical daily price data. That fact brought me some limitations in the experiments phases. As a result, the models were applied on stock markets for daily forecasting only. On the other hands, for the cryptocurrency, data for each day, hour and minute are available but under 2,000 rows. Even, there are several companies selling complete historical data commercially. This limitation of public data would delay the development of research in the area.

6.2 Safety of cryptocurrency trading

Important financial data can be often monopolized by exchanges and it can be used by internal traders. It might be one of the major obstacles to execute forecasting models in financial markets as well as hard to make trading strategies with prediction models profitable. Data monopoly does not only exist in financial markets but also in many industries. Nowadays, we are living in the era of data. By that means, having valuable data exclusively can be a great benefit in many ways. For example, a cryptocurrency exchange usually has internal market-makers. They have exclusive offers for commission rate per trade under a contract. Also, they get important information such as launching of new coin trading or new ICOs (Initial Coin Offering) first. The preoccupancy of information can hugely drive profit or loss of other high-frequency traders.

Cryptocurrency markets are still quite controversial and there is no official regulatory body in the cryptocurrency market explicitly. Someone says it is the future of the monetary unit and the others argue that it is just a scam. I called cryptocurrency as a security in this thesis, but it is also a quite controversial topic. As there are no clear regulations of it yet, cryptocurrency exchanges like Kraken,

which I tested its data in this study, are vulnerable to hacks or scams due to its anonymity and open source technology. Major hacks targeting crypto exchanges have happened from time to time.

In terms of scams on cryptocurrency, there are many concerns. As the volume of cryptocurrency market increases, governments in a number of countries are concerned about its attributes of high volatility, anonymity and open source concepts. Although cryptocurrency is decentralised digital currency, exchanges are still centralised. Unless you trade peer-to-peer directly to the seller or buyer with high risks, you need to trade on exchanges to exploit its supply and demands. So, one of the most common scams is shady exchanges. These exchanges can steal customers' precious coin deposit easily. Therefore, I highly recommend trading on reliable, legit exchanges carefully. Several events of cryptocurrency price manipulation done by exchanges with malicious intent have resulted in people to emphasize the needs of transparent public disclosure of market prices in cryptocurrency markets. To be a viable investment market, cryptocurrency will need to solve this aspect of problems.

Finally, sometimes the exchanges' server might be unstable and struggle from the on-and-off server down. I have experienced many times of connection failure to exchange servers while I am working on projects for cryptocurrency trading. Even when the market is un-approachable by traders, the markets on the exchange are still alive. It causes a big risk that you can't sell or buy when you want to.

6.3 Risk of Real Application

The actual application of financial forecasting models will accompany some risks. To prove the underlying hypothesises of research, it requires actual investments in reality. From my placement work, I have tried paper trading with predictive models several times but always it demonstrated the defference between theoretical and practical results. In particular, by the characteristic of cryptocurrencies which they only exist in the blockchain networks without underlying values, investments on cryptocurrency markets would have higher risks. Nevertheless, any kinds of time-series analysis or predictive modelling would have the same problem to validate findings.

6.4 Difficulty of Citation

One of the most challenging parts was the management of references and citations. I kept on track of my work using Microsoft OneNote as I mentioned, but it wasn't easy to organise contents I referenced every time. In particular, in the literature review chapter, I had a little concerned about how much and in what convention of citation is allowable. Later on, I found out there are several reference management tools such as Mendeley, it would have been so much easier to manage references if I had known the existence of the tools from the first. However, through this experience, I could learn how to manage references more efficiently.

7 How to use my code

7.1 Installation and run

All code and relevant data to this project are in the **MyProject/** folder. The code was tested on macOS 10.12.6 and Linux openSUSE 13.1 with Python version 3.6. Follow the instruction in this chapter to execute and explore the code from Terminal.

```
# Install libraries
$ pip3 install seaborn
$ pip3 install pandas
$ pip3 install matplotlib
$ pip3 install numpy
$ pip3 install requests
$ pip3 install json
$ pip3 install sklearn
$ pip3 install itertools
```

Above shows how to install the necessary libraries used in the code. If there is any library not installed in your machine, you can install using the command **pip3 install 'packagename'**. Before installing everything, check if your machine has some of the libraries already installed as import libraries like below. If there is a package not installed, it will return a '**ModuleNotFound'** exception. In that case, the reader can try to install the particular package only using the **pip3** command from Terminal.

```
# Run Python3
$ python3

# Import libraries
>>> import seaborn
>>> import pandas
>>> import matplotlib
>>> import numpy
>>> import requests
>>> import json
>>> import sklearn
>>> import itertools
>>> quit()
```

```
# Run code
$ python3 filename.py
```

To run a code, open Terminal **in the directory where the file exists**, and type in **python3 filename.py** from Terminal. In the script, it uses a relative path to call data files, therefore each code should be executed in the folder.

7.2 Code List

There are four folders in the **MyProject/** directory, Exploration, CsvTest, RealTimeTest and data. It is recommended to run in order from Exploration -> CsvTest -> RealTimeTest. Below shows the hierarchy of the directories as well as simple description of the resource folders.

- **Exploration:** has scripts to explore the data with data visualisation plots.
- **CsvTest:** scripts for experiments with csv files stored in **data/** directory**.** It will show the forecasting results as described in this thesis. It tests three machine learning approaches (SVM, NB, KNN) only.
- **RealTimeTest:** scripts for experiments with real-time data. It shows the experimental result according to the real-time data from CryptoCompare APIs. Therefore, it would return different results with the thesis. The data will be collected at the moment of when the code executed.
- **data:** has pre-stored csv data files to demonstrate the more or less same result as this thesis.

```
MyProject/
|-- Exploration/
| |-- DataProcessing_crypto.py
| |-- DataProcessing_stock.py
|
|-- CsvTest/
| |-- Approach1.py
| |-- Approach2BTC_USD.py
| |-- Approach2ETH_BTC.py
| |-- Approach3.py
| |-- Approach4.py
| |-- Approach4ROC.py
| |-- Approach5.py
|
|-- RealTimeTest/
| |-- Approach1.py
| |-- Approach2.py
|
|-- data/
| |-- data_days_BTC.csv
| |-- data_days_USD.csv
| |-- data_hour_USD.csv
| |-- data_days_GOOGL.csv
| |-- stock/
```

It does not show .csv in the **data/stock/** and **RealTimeTest/AppendixC.py**

Note: **RealTimeTest/Approach2.py** provides more range of choices for various machine learning models then **CsvTest/Approach2.py**. The reader can test the function, as open the **RealTimeTest/Approach2.py** file in an editor program. There are the lines of code as below at the end of the file. It will show the prediction result of ML models as calling the **predictMove** function in the last line with different parameters. For example, the last line of code forecasts ETH_BTC pair with SVM when $p = 5$, p is the maximum input feature size given to the model.

```
# Models: nb, knn, svm, clf, logreg, boost, forest, nn
nb = GaussianNB() # Naive Bayes
knn = KNeighborsClassifier(n_neighbors = 3) # 3 KNNs
svm = SVC(kernel='rbf') # SVM with Gaussian kernel
clf = tree.DecisionTreeClassifier(class_weight="balanced") # Decision Tree
logreg = LogisticRegression(class_weight="balanced") # Logistic Regression
boost = GradientBoostingClassifier() # Gradiant Boost
forest = RandomForestClassifier(n_estimators = 18) # Random Forest
nn = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(5, 2), random_state=1) # Neural Network

# predictMove(target_coin, base_coin, MLmodel, max_feature_size)
predictMove('ETH','BTC',svm,5)
```

References

[1] Friesen, G. C., Weller, P. A. and Dunham, L. M. (2008) 'Price trends and patterns in technical analysis: A theoretical and empirical examination', *Journal of Banking & Finance*. doi: 10.1016/j.jbankfin.2008.12.010.

[2] Taylor, M. P. and Allen, H. (1992) 'The use of technical analysis in the foreign exchange market', *Journal of International Money and Finance*.

[3] Fama, E. F. (1970) 'Efficient Capital Markets: A Review of Theory and Empirical Work', *Journal of Finance*.

[4] Kim, K.-J. (2003) 'Financial time series forecasting using support vector machines', *Neurocomputing*, 55, pp. 307–319. doi: 10.1016/S0925-2312(03)00372-2.

[5] Shynkevich, Y. *et al.* (2017) 'Forecasting price movements using technical indicators: Investigating the impact of varying input window length', *Neurocomputing*, 264, pp. 71–88. doi: 10.1016/j.neucom.2016.11.095.

[6] Hsu, M.-W. *et al.* (2016) 'Bridging the divide in financial market forecasting: machine learners vs. financial economists', *Expert Systems With Applications*, 61, pp. 215–234. doi: 10.1016/j.eswa.2016.05.033.

[7] Ballings, M. *et al.* (2015) 'Evaluating multiple classifiers for stock price direction prediction', *Journal of Banking & Finance*. doi: 10.1016/j.eswa.2015.05.013.

[8] Singh, R. and Srivastava, S. (2017) 'Stock prediction using deep learning', 76, pp. 18569–18584. doi: 10.1007/s11042-016-4159-7.

[9] Chong, E., Han, C. and Park, F. C. (2017) 'Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies', *Expert Systems With Applications*, 83, pp. 187–205. doi: 10.1016/j.eswa.2017.04.030.

[10] Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). 'A training algorithm for optimal margin classifiers', *Proceedings of the fifth annual workshop on Computational learning theory - COLT*, 92, pp. 144–152. New York, New York, USA: ACM Press. doi: 10.1145/130385.130401.

[11] Luo, Z., Vovk, V. (2016) 'Support vector machines' [Lecture] CS5920 Machine Learning. Royal Holloway. 25th November 2016.

[12] Cortes, C; Vapnik, V. N. (1995). 'Support-vector networks', *Machine Learning*. 20 (3): 273–297. doi:10.1007/BF00994018.

[13] Ben, H. (2011) [Online] *machine learning - Generative vs. discriminative - Cross Validated*. Available at: https://stats.stackexchange.com/questions/12421/generative-vs-discriminative (Accessed: 10 September 2018).

[14] Jebara, T. (2004) *Machine learning : discriminative and generative*. Kluwer Academic Publishers.

[15] Ng, A.Y., Jordan, M.I. (2002) 'A Comparison of Logistic Regression and Naive Bayes', *Advances in Neural Information Processing Systems*. Cambridge, Mass., MIT Press.

[16] Vapnik, V. N. (1998) 'Statistical Learning Theory', John Wiley & Sons.

[17] Shihavuddin, A. S. M. *et al.* (2010) 'Prediction of stock price analyzing the online financial news using Naive Bayes classifier and local economic trends', *ICACTE 2010 - 2010 3rd International Conference on Advanced Computer Theory and Engineering, Proceedings*, 4, pp. 22–26. doi: 10.1109/ICACTE.2010.5579624.

[18] Fawcett, T. (2005) 'An introduction to ROC analysis', *Pattern Recognition Letters 27*, pp. 861–874. doi: 10.1016/j.patrec.2005.10.010.

[19] Vovk, V. (2016) 'Evaluating and comparing prediction algorithms', [Lecture] CS5100 Data Analysis. Royal Holloway. 27th October 2016.

[20] Jon Maier (2017) *CIO Insights: Sector Investing and Correlations – Global X Funds*. Available at: https://www.globalxfunds.com/cio-insights-sector-investing-and-correlations/ (Accessed: 10 September 2018).

[21] Lobo, J. M., Jiménez-valverde, A., & Real, R. (2008) 'ECOLOGICAL SOUNDING AUC: a misleading measure of the performance of predictive distribution models', *Global Ecology and Biogeography*, 17, pp. 145–151. doi: 10.1111/j.1466-8238.2007.00358.x.

Appendix A: Technical indicator features, Refs. [4]

Table 1
Initially selected features and their formulas

Feature name	Description	Formula	Refs.
%K	Stochastic %K. It compares where a security's price closed relative to its price range over a given time period.	$\frac{C_t - LL_{t-n}}{HH_{t-n} - LL_{t-n}} \times 100$, where LL_t and HH_t mean lowest low and highest high in the last t days, respectively.	[1]
%D	Stochastic %D. Moving average of %K.	$\frac{\sum_{i=0}^{n-1} \%K_{t-i}}{n}$	[1]
Slow %D	Stochastic slow %D. Moving average of %D.	$\frac{\sum_{i=0}^{n-1} \%D_{t-i}}{n}$	[9]
Momentum	It measures the amount that a security's price has changed over a given time span.	$C_t - C_{t-4}$	[3]
ROC	Price rate-of-change. It displays the difference between the current price and the price n days ago.	$\frac{C_t}{C_{t-n}} \times 100$	[16]
Williams' %R	Larry William's %R. It is a momentum indicator that measures overbought/oversold levels.	$\frac{H_n - C_t}{H_n - L_n} \times 100$	[1]
A/D Oscillator	Accumulation/distribution oscillator. It is a momentum indicator that associates changes in price.	$\frac{H_t - C_{t-1}}{H_t - L_t}$	[3]
Disparity5	5-day disparity. It means the distance of current price and the moving average of 5 days.	$\frac{C_t}{MA_5} \times 100$	[5]
Disparity10	10-day disparity.	$\frac{C_t}{MA_{10}} \times 100$	[5]
OSCP	Price oscillator. It displays the difference between two moving averages of a security's price.	$\frac{MA_5 - MA_{10}}{MA_5}$	[1]
CCI	Commodity channel index. It measures the variation of a security's price from its statistical mean.	$\frac{(M_t - SM_t)}{(0.015\, D_t)}$ where $M_t = (H_t + L_t + C_t)/3$, $SM_t = \frac{\sum_{i=1}^{n} M_{t-i+1}}{n}$, and $D_t = \frac{\sum_{i=1}^{n} \lvert M_{t-i+1} - SM_t \rvert}{n}$.	[1,3]
RSI	Relative strength index. It is a price following an oscillator that ranges from 0 to 100.	$100 - \frac{100}{1 + (\sum_{i=0}^{n-1} Up_{t-i}/n)/(\sum_{i=0}^{n-1} Dw_{t-i}/n)}$ where Up_t means upward-price-change and Dw_t means downward-price-change at time t.	[1]

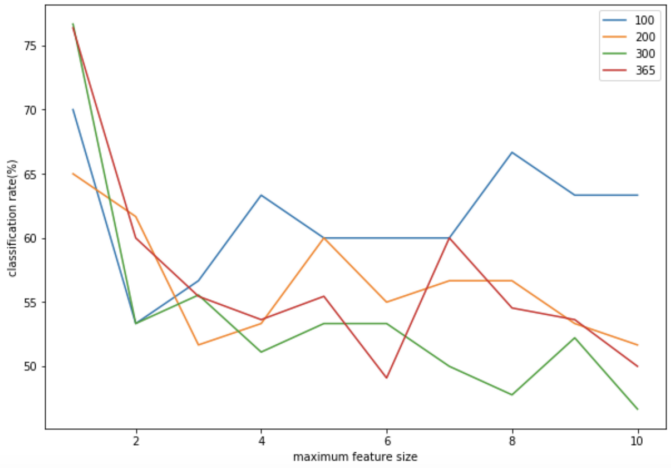
C_t is the closing price at time t, L_t the low price at time t, H_t the high price at time t and, MA_t the moving average of t days.

Appendix B: ML algorithms used in literature, Refs. [7]

Algorithms for stock price direction prediction used in literature.

	Prediction method						
	LR	NN	KN	SVM	AB	RF	KF
Schöneburg (1990)		x					
Bessembinder and Chan (1995)	x						
Brownstone (1996)	x	x					
Saad, Prokhorov, and Wunsch (1996)		x					
Kim and Chun (1998)		x					
Saad, Prokhorov, and Wunsch (1998)		x					
Kim and Han (2000)		x					
Kuo et al. (2001)		x					
Kim (2003)		x		x			
Kim and Lee (2004)		x					
Rodriguez and Rodriguez (2004)	x	x			x	x	
Huang et al. (2005)		x		x			
Kumar and Thenmozhi (2006)	x	x		x		x	
Lunga and Marwala (2006)					x		
Wu et al. (2006)							
Wang and Chan (2007)	x						
Huang et al. (2008)	x	x	x	x			
Senol and Ozturan (2008)	x	x					
Lai, Fan, and Chang (2009)							
Lee (2009)		x		x			
Ou and Wang (2009)	x	x	x	x			
Kara, Boyaciogly and Baykan (2011)		x		x			
Wei and Cheng (2011)		x					
Subha and Nambi (2012)	x		x				
Lin, Guo, and Hu (2013)				x			
De Oliveira, Nobre, and Zárate (2013)		x					
Chen, Chen, Fan, and Huang (2013)		x					
Rechenthin et al. (2013)			x	x	x		
Ji, Che, and Zong (2014)		x					
Bisoi and Dash (2014)		x					
Zikowski (2015)				x			
Hafezi, Shahrabi, and Hadavandi (2015)		x					
Patel et al. (2015)		x		x		x	
This study	x	x	x	x	x	x	x

Appendix C: Approach 1 with various ML algorithms



	nrows	feeds	scores
30	365	1	81.818182
31	365	2	60.000000
32	365	3	56.363636
33	365	4	55.454545
34	365	5	54.545455
35	365	6	55.454545
36	365	7	58.181818
37	365	8	55.454545
38	365	9	54.545455
39	365	10	53.636364

Figure 22. Result of approach 1.1 with KNN (k=3)

The plot and table demonstrate the result of Approach 1 with KNN(k=3) depending on the sample size (100, 200, 300, 365) and the maximum input feature size p. It was an experiment to address if there is a relationship between the given sample size to the model and the prediction performance. However, as a result, it did not show any significant result.

	Model	Score
2	Naive Bayes	80.000
1	KNN	78.182
3	Logistic Regression	77.273
6	svm	77.273
0	Random Forest	72.727
5	Gradient Boosting	72.727
7	nn	69.091
4	Decision Tree	67.273

	Model	Score
0	Random Forest	56.364
5	Gradient Boosting	56.364
2	Naive Bayes	54.545
1	KNN	52.727
3	Logistic Regression	51.818
6	svm	51.818
7	nn	50.909
4	Decision Tree	47.273

Table 5. Experimental results with various model
Approach 1.1 (Left) & 1.2 (Right)

These two tables show the result of experiments from the model selection for Approach 1.1 (Left) and 1.2 (Right). This phase of initial experiments resulted in choosing SVM, KNN and Naïve Bayes to study in detail in this project.

NOTE: The code for Appendix C can be found and tested with the real-time cryptocurrency market data in **RealTimeTest/AppendixC.py.** Run $ **python3 AppendixC.py** from Terminal in the **RealTimeTest/** directory.