Machine Intelligent Financial Trading - Capstone Project Final Report

Aditya Kant Sharma (SID: 470375615), Ignacio Colino (SID: 480201931), Pote Pongchaikul (SID: 311065775), Sergio Kulikovsky (SID: 480322960), Thomas Brown (SID: 470347346)

Abstract

People have always tried to forecast future asset prices to make gains. The rise of modern stock markets, with large data gathering capabilities, has opened a range of theories to help an investor forecast future prices. These range from fundamental analysis in which one values a security according to its underlying value to technical analysis through which the future price is predicted from its past time series data. The effect of many different price forming expectations is that stock markets are a very dynamic and crowded system. Time-series of past stock price data encodes a lot of information which has to be extracted through extensive use of feature engineering. Three main types of feature engineering are proposed here: (i) economic (fundamental) data, (ii) time series transformations and (iii) technical analysis feature extraction. Machine learning models can be successfully trained to learn from past data in order to predict future prices. Using machine learning output combined with expert knowledge and having clear risk minimisation strategies such as stop-loss and lock-gain strategies one can increase his portfolio return. Machine learning is a good tool to augment a trader's prediction accuracy. The system designed for this report can be easily extended to accept further economic and fundamental data and to accept further feature engineering. Different experiments can be easily trained and reports can be easily calculated for different machine learning techniques. This paper proposes an extensive range of supervised machine learning methods to process the engineered features into a solution for the stock price forecasting problem. The range of supervised learning methods include Gradient Boosting, Logistic Regression, State Space Modeling, Deep Neural Networks and LSTMs. The suggested implementation and specific architecture for these methods given the financial context is also discussed. Additionally insight will be given into the resources required to tackle the stock price forecasting problem and the allocated schedule for the project complete with expected outcomes. Solving this problem has motivation not just in the financial sense but also in the academic sphere where it has been debated since the emergence of the markets themselves.

Keywords

State Space — Kalman filter — Logistic Regression — Deep Learning — LSTM — Gradient Boosting — technical analysis — stock price prediction — Australian stock market — ASX — S&P500 — Financial Machine Learning

The University of Sydney, Australia

Contents

1. INTRODUCTION

1.1 General Background

Financial markets are a very dynamic system. Many different players are active in the market with different expectations on why a security's price may behave in the future, leading to a variety of price making expectations. While some investors may be looking for long term goals, other are looking for short term gains. Companies use the market to raise funds when interest rates are high and buy back when they don't have better investments for their cash or interest rates are low. Farmers, producers and companies use the financial markets to hedge their costs. The market as a whole can trend higher as a country's international rating improves. Employees can get equity in companies as a part of long term compensation incentives. Insider information can be used by the few who have it to try to make gains ahead of information becoming public. All of this leads to a very crowded market system which behaves erratically.

Different investor expectations create a very dynamic pricing composition which finds equilibrium at market prices. Since everyone's expectation is differently formed, markets tend to move very dynamically which leads to constant price fluctuations. The price of the market is always the right price and if an investor thinks it is wrong he can profit by l

The market is measured by indexes which are created to properly measure the market. Indexes can be built using many factors, but usually are based on market capitalisation or trading volume. They tend to reflect the market performance of its underlying stocks. In this study we used indexes such as S&P500 (Standard & Poor's) and ASX200 (Australian Securities Exchange), which reflect the largest companies in the U.S. and Australian markets respectively, but any other index such as Russell 2000, which reflects the performance of small caps stocks, could be used.

1.2 Motivation

The primary motivation for our study is to find out whether machine learning can be used to predict future asset prices. In a broad sense, investors are largely divided among fundamental and technical analysts. Fundamental investors take into account company underlying fundamentals to make price predictions, whereas technical investors use charts of past trading data to predict future price movements. Our objective is to study whether machine learning can augment an investors toolkit to help him beat the market.

The idea is that we can feed all the information that investors use to make their decisions, be they fundamental or technical analysts, into machine learning models. To do so we have to properly select which are the information that is used, how it is used, check for its availability for similar securities, and then apply those data to train a model that will hopefully give enhanced predictions.

We are not aiming at finding predictions with 100% accuracy, but predictions that are substantially higher than a base random guess rate. For example, if the random guess rate for whether a stock price would go up, down or stay flat is 33% in each category, getting a prediction accuracy of 60% is a substantial improvement.

1.3 Benefits

Most investors lose to the market annually. A recent survey of mutual funds [54] showed that 75% of listed equity mutual funds lose to the benchmark. This study was based on a wide range of benchmarks and there is no survivor bias analysis, meaning that the figure of 25% of funds that beat the market is probably overestimated.

The fact that the index is known beforehand should make it easy for a portfolio manager to closely match it and optimise stock allocation to beat it, yet most fail to. One simple reason are fees charged and trading costs. That was one of the reasons for the huge growth in indexed funds. Given that only 25% of

the funds can actually beat the benchmark, that explains why investors are migrating towards passive (indexed) investment strategies. Our goal is to uncover ”hidden” information in price time series as well as other factors in order to be able to try to outperform market.
The key objectives of our study are thus:

- Research machine learning and financial trading: explore what was done in the past, the current state of deployments and understand some of its limitations
- Explore different techniques and models: by extracting features from data as well as adding external features which may affect a stock's price, explore different machine learning techniques
- Implement a platform for experiments: to create a generalisable framework for machine learning in the financial markets.

An eventual secondary objective of our research is to explore possible productisation of the tools here developed. As a group, we'd like to continue studying financial machine learning and intending to eventually create a financial trading system which could be used for live trading.

1.4 Problem Description

Markets are always at a dynamic equilibrium. The pressures of buying and selling are met at the market price, which is the last price at which a transaction has occurred. Given a certain market equilibrium, different players create expectations as to where the price of a security should be at. They arrive to those conclusions using a combination of different analysis techniques:

- Fundamental analysis: a security's price is created by the underlying value of its assets, its future profit potential, the market it operates in, the regulation of its industry
- Technical analysis: the price of a security is mostly influenced by the pressures of buy and sell in the market, and studying past trading data one can extract future information as to where a price is moving to. These analysts use mostly charts known as candlestick stick charts which show, for each trading period (bar) whether the price is trending up or down, the open, close and minimum and maximum price a security has reached in that period. It's a great visual aid for the trader and from those charts a trader can derive substantial trend information.
- Macroeconomic analysis: by studying the macroeconomic factors in the economy one can invest in specific sectors which would benefit from economic trends. As an example, a currency devaluation usually benefits export oriented industry while simultaneously being bad for companies which depend on imports into its supply chain.

The market tends to become very crowded with lots of different players investing with different expectations, but always at an equilibrium. Automating the analysis by using factors available to fundamental, technical and macroeconomic analysts, is our objective in this report.

1.5 Propose Solution

The approach we used to construct solutions to this complex project proposition is to derive factors which can encode information from stock market securities. The factors that were derived comprised some of the common analytical techniques employed by traders.
The first step in our project was to define the features that can be extracted for all the securities being studied. We thus extracted mostly price based features, such as momentum, volume strength and statistical smoothing averages. Chapter 5 describes those features in further detail. Additionally, we also extracted macroeconomic data which became features of our models.
An interesting set of features were the technical analysis candlestick features. We identified some classic candlestick patterns used by traders, such as hammer, shooting and morning star, three line strike and bullish abandoned baby. For each of those patterns we created features which would identify in the data set whether those patterns were occurring.
We created a large data set of over 80 features and those features were then fed through different machine learning models. We then created 3 different sets of experiments and run each of the models under identical parameters. We could then compare model accuracy as well study which kind of data sets were better suited for predictions.
The way we set up our experiments is very flexible with the objective of being able to augment our features in the future. Having more features is a way to introduce complexity and possible overfitting to a model, so proper cross validation has to be done. That is not the scope of our project and we intend to continue working on that in the future on this subject. It is also critical to define proper techniques of cross validation, as a time series should not learn on data which it will predict. Thus a proper time series cross validation has to be employed, in which the model learns up to a point in time and predicts the near future. That future is then added to the learning series and the predictions are moved one step further.
The framework that was developed was also critical to the success of this project as it's very flexible in order to be able to change the type of experiments we can run. All code was developed in Python and the availability of extensive Python libraries for machine learning makes it easy to keep expanding the breadth and depth of this challenging project.

2. RELATED LITERATURE

Finance professionals have been trying to model financial markets using statistical techniques since the middle of the 20th century. Harry Markowitz[55] was awarded the Nobel prize in 1990 for his work on efficient portfolios and the

efficient risk frontier. It was the start of the theory of portfolio optimisation according to risk.
William Sharpe[70], of Sharpe ratio fame, also shared the 1990 Nobel prize. He used linear regression when developing the Capital Asset Pricing Model (CAPM) in order to optimise portfolios according to risk. Most importantly, he wanted to properly attribute returns to the asset rather than the risk of the asset. The CAPM model states that a given security's return should be proportional to the market's return (or risk), called beta. If the expected return is above that which is predicted by the model then the company is underpriced (positive alpha). The work of a portfolio manager is thus to optimise alpha in the portfolio, given that beta is just the assumption of risk.
CAPM thus concludes that a stock's return is dependent on the systemic risk of the market. The CAPM function states that the expected return of a security is the risk free rate plus a fraction (beta) of the return of the market above the risk free rate:

$$E(R_s) = R_f + \beta(E(R_m) - R_f)$$

Fama and French[33] expanded the model into a 3 factor return portfolio by recognising that there are other factors which affect the return of a stock. In 1992 at the time of publishing their paper they identified two additional factors: (i) company size and (ii) value stocks. They identified those two factors by analysing the historical overperformance of (i) small capitalisation stocks (vs large capitalisation stocks) and (ii) companies with smaller price to book ratios (value stocks vs growth stocks).
The CAPM is frequently used as a way to calculate the cost of capital as well as minimum required investment returns. It is also used to evaluate a portfolio's performance by attempting to isolate performance from risk. It is less useful as a predictor model, as well outlined by Fama and French[34]. Fama and French[35] more recently expanded their three factor to a five factor model by adding as factors (i) profitability momentum and (ii) company investment level.
With the advent of powerful computers, Barr Rosenberg, a professor at UCLA, started a company BARRA in which he would calculate 96 factors[19] in addition to a stock's beta. Those factors and its analysis were sold in a consulting mode to global asset managers. Those factors were used to guide portfolio construction and find a company's sensitivity to each factor, e.g. to build a size neutral stock portfolio or to overweight value stocks.
The advent of Machine Learning (ML) brought very powerful computers and flexible libraries within the reach of most portfolio managers. They are often called "quants" for quantitative portfolio managers, and they use factors as a way to enhance their portfolio's return. Given that they know beta and some other factors, their objective is to optimise alpha. The new phase of financial investment is working with as many factors as possible and then feeding them into ML models searching for optimal investment portfolios. Given the profusion of factors some of these models have been termed "factor zoo". Most of the proprietary quantitative trading firms are very secretive about the methodologies employed yet there's ample academic literature of experiments being made to attempt to model security pricing.
One of the main pitfalls of the many implementations is that the stock markets are so noisy (pricing wise) that a lot of the literature picks up noise as trend signals which tend to give positive results during a certain time yet will fail in the next period. We do not attempt to give a definitive answer to the stock market prediction challenge (indeed a quasi impossible challenge) but we aim to develop a toolkit which can be used to augment a trader's accuracy and help the systematisation of the investment process.

2.1 State-space Models

2.1.1 From Wiener to Kalman

One of the methods widely used in state-space modeling is the Kalman filter. In [51], where the method was derived, Kalman proposed a recursion scheme in order to improve on an earlier study conducted by Norbert Wiener. There, Kalman argued that Wiener's approach was inflexible, limited in scope, and that his method presented some significant weaknesses. In Kalman's approach, his idea is to model a "system" that gives "output" (we call this observations) based on "states". The relationship is that the output is a linear function of the states perturbed by a Gaussian noise. He also assumed the states are a linear function of the previous states, also perturbed by a random Gaussian noise. This normality assumption is one of the key assumptions in the process of Kalman filter, and this algorithm was used in two aerospace programs, the Polaris and the Apollo [50].

2.1.2 Motivation

In classical time series modeling, one typically assumes that the observations can be modeled as past observations of itself, along with some noise. For example, in the case of the Autoregressive Moving Average ARMA, or its counterpart ARIMA, it can be proposed that current observed value can be explained using the past p values and some sequence of noise process $\{Z_t\}$. Mathematically, we have that, for ARIMA$(p,0,q)$

$$X_t = \sum_{i \le p} \phi_i X_{t-i} + \sum_{j \le q} \theta_j Z_{t-j} \tag{1}$$

Two parameters that govern the above process need to be chosen – the number of lags p and the number of past value of noise processes q. These are determined by examining the sample partial autocorrelation function (PACF) and sample autocorrelation function (ACF) respectively, and a cut-off on those plots are used to determined p and q.
An alternative view we could take is to view observations as being governed by states and noise, similar to our description above. To see this, let the states be

$$S_t = \sum_{i \le p} \phi_i X_{t-i} \tag{2}$$

We also let the white noise process be given by

$$W_t = \sum_{j \leq q} \theta_j Z_{t-j} \tag{3}$$

It follows by substitution that

$$X_t = S_t + W_t \tag{4}$$

It can be seen that the current observation depends on the underlying state of the model, perturbed by white noise process. This motivates the state-space model.
The above Autoregressive Moving Average model is one special case of the state space model, where one assumes that observations at a time point is a function of the states. In turn, the states are also a function of the past states. The aim of the state-space model is to infer the states based on our knowledge from the observations. There are two approaches to do this task, the Kalman Filter and the hierarchical Bayesian model (information can be referred to [32] and [41]).
In Kalman filter, we have to first assume that the noise processes are Gaussian with mean zero, the states "move" through time according to a linear function, and the observations are governed by the states in a linear fashion below (see equations 6.1 and 6.2 on page 319 of [72]).

$$Y_t = A_t X_t + V_t \tag{5}$$
$$X_t = \Phi X_{t-1} + W_t \tag{6}$$

One initializes the algorithm by choosing a pair of initial guesses, one for thestate mean, and the other for the state variance (or covariance). Then, these are used to compute the Kalman Gain matrix, which can be thought of as a transformation that "corrects" for the error, whose expression is given as the amount of deviation an observation is from the state estimates at current time point.

$$e_{t|t-1} = Y_t - A_t \widehat{X_{t|t-1}} \tag{7}$$

The update for the new state estimates (both mean and covariance) will take into account this "residual" and correct for this in the new state estimates. This is iterated until convergence. A diagram depicting this flow can be found in [79]. Note that in our case, the Dynamic Linear Model (DLM) will be used. Here, we treat states as "weights", which can be thought of an the amount of importance of the predictors. These will be the coefficients of the predictors, and we postulated that they are unbiased in the sense that they are changed by some normally distributed variable ν_t given below. The observations are perturbed by some normally distributed variable ε_t, but on average they vary about the mean, represented as the linear combination of states (weights) and predictors. In other words,

$$y_t = x_t^T \beta_t + \varepsilon_t \qquad \varepsilon_t \sim N(0, \sigma_\varepsilon^2)$$
$$\beta_t = \beta_{t-1} + \nu_t \qquad \nu_t \sim N(0, \sigma_\nu^2)$$

A second approach undertakes a more general Bayesian perspective, where the notion of prior, likelihood and posterior are prominent. In this scenario, one has that the observations takes on a probability distribution with the parameters being the states. In turn, the states also have its own "hyperparameters" given by $\{\alpha, \beta, \gamma, \omega, \xi, \eta\}$. This is displayed below:

$$Y \sim F(A, B, C)$$
$$A \sim G(\alpha, \beta)$$
$$B \sim H(\gamma, \omega)$$
$$C \sim J(\xi, \eta)$$

Note that each of A, B, C are the states, and F, G, H, J are probability distributions.
In order to infer the states based on observations, one utilize the Bayes' theorem iteratively. More specifically, one finds the value for the posterior probability density $p(\text{States}|\text{Past data})$ based on the prior $p(\text{States})$ and the likelihood given the states, $p(\text{Past data}|\text{states})$. As more data points were fed into the iterative scheme, the posterior variance will eventually lessen and "centered" around the state mean. The overall effect is that as more and more information was "known", we are more confident that the states have certain mean and certain variance.
To perform prediction of future data based on past data, one expresses the conditional distribution of future observations based on present and past observations, as well as the states (the integrand in the right-hand side of the equation below). Second, one marginalizes the states via integration to obtain predictive distribution (given by the left-hand side). The textbook [62] page 643 gave explanations on this. In later section the marginalized posterior distribution will be needed to implement state-space modeling.

$$p(Y_{t+1}|Y_1, ..., Y_t) = \int\int\int p(Y_{t+1}|A_t, B_t, C_t) \\ p(A_t, B_t, C_t, Y_1, ..., Y_t) dA_t dB_t dC_t$$

The resulting distribution will be normal because, firstly, the distribution of the states are normal. Second, by assumption, the conditional likelihood of the observations $\{y_t\}$ given the states are also normal. By the property of normal distribution, marginalization will still retain the normal distribution properties. Furthermore, the maximum likelihood of the conditional normal distribution is its conditional mean, and it can be shown that this minimizes the square error loss (MSE) (see **Result B**). Because the use of conditional expectation not only maximizes the normal likelihood function, but also on minimizing the mean squared error loss (MSE). These desirable

properties are the reasons for using conditonal expectation as predictions. **Result A** below will show that for our prediction, which is the conditional expectation taken with respect to the predictive distribution $p(y_{T+k}|y_1,...,y_T)$, it is enough to generate predictions for future time points $\{y_{T+k} : k \in \mathbb{N}\}$ using the latest state estimates.

2.1.3 Application of State-space model in finance

State space models have been used in the financial modeling problems – volatility modeling, and stock price forecasting [31], [84]. In Robert Engle's 1982 paper, he used the ARCH method to model inflation rate in the UK. On the other hand, Bayesian hierarchical model was used in [84] to forecast stock prices, which they claim to outperform the method of ordinary least squares.

The Autoregressive Conditional Heteroscedastic model (ARCH), proposed by Robert Engle [31], takes the following form (see page 281 of [72] for the formula for ARCH(1)) with $\{\varepsilon_t\}$ the noise process:

$$Y_t = \sigma_t \varepsilon_t \tag{8}$$

$$\sigma_t^2 = \alpha_0 + \sum_{k \geq 0} \alpha_k Y_{t-k}^2, \quad \alpha_0 \geq 0 \tag{9}$$

This equation tells us current volatility, represented by σ_t using past values of the series. Here, the σ represents the volatility [27] and the Y's being the observed time series values. The first equation above can be regarded as the observation equation, and the second the state equation, and thus agrees with the general form of the state space equation. Tim Bollerslev [20] cited and extended Engle's idea to include the "moving average" part to the original state equation of the ARCH model. This was named as Generalized Autoregressive Conditional Heteroscedastic model (GARCH), and takes the similar form to ARCH and can also be viewed as state-space (see [72] page 285 for the equations below)

$$Y_t = \sigma_t \varepsilon_t \tag{10}$$

$$\sigma_t^2 = \alpha_0 + \sum_k \alpha_k Y_{t-k}^2 + \sum_j \beta_j \sigma_{t-j}^2 \tag{11}$$

The Hierarchical Bayesian model was used by Ying et al [84] in the problem of stock price prediction. In this paper, stock prices, being the observations, were modeled as a linear regression equation with AR(1) noise. However, unlike multiple linear regression, the states β and the coefficient for the AR(1) noise ρ, were also governed by their own distribution given by other set of hyperparameters. A diagram of the hierarchical relationship can be found in Ying [84], Figure 1. In their setting, the β was assumed to be a mixture of Normal distributions, and ρ was assumed a truncated Normal distribution. This was compared against the method of least squares. In the results table (Table 55 of Ying [84]), they claimed that using Hierarchical Bayes method was superior to the least squares method.

2.1.4 Usefulness of state space models

Recall that in ARIMA model, which is a special case of the state space model, it assumes that the sizes of the coefficients (ϕ and θ in the first equation) are less than one in magnitude, and that the time series observations are supposed to have a constant mean, and variance with covariance changing only with time lags. These assumptions were imposed to enable us to talk about prediction into the future, and because the statistical inference on parameter estimates would not be possible without the size constraints. In state space model, the assumption of stationarity and invertibility are not assumed, but it is essential that normality of the observation noise and state noise be supposed.

2.1.5 Some facts

The following lists some facts mentioned, which is used in prediction.

Result A

For each t such that $0 \leq t \leq T$, define the following:

- The prediction of $Y_t \in \mathbb{R}$ given information up to current time point T as $\widehat{y_{T+k|T}}$
- The states $\beta_t \in \mathbb{R}^p$
- The data matrix $X \in \mathbb{R}^{T \times p}$
- The data point $x_t^T \in \mathbb{R}^p$

The aim is to show that, to obtain predictions for future time periods $T+k$, it is enough to use the latest state estimate β_T, the corresponding data point for the period $T+k$, and perform matrix multiplication. Using the above formulation for $y_t \in \mathbb{R}$, one has, for each $k \in \mathbb{N}$

$\widehat{y_{T+k|T}} = E(Y_{T+k}|Y_1,...,Y_T)$
$= E(E(Y_{T+k}|Y_1,...,Y_T)|Y_T)$ (Repeated conditioning)
$= E(Y_{T+k}|Y_T)$ (Tower property, see [7] page 336)
$= E(x_{T+k}^T \beta_{T+k} + \varepsilon_{T+k}|Y_T) = E(x_{T+k}^T(\beta_T +$
$\sum_{i=1}^k \nu_{T+i})|Y_T)$ (Repeated substitution in the state equation)
$= E(x_{T+k}^T \beta_T|Y_T)$ ($E(\nu_t) = 0$ by model definition)
$= x_{T+k}^T \beta_T$ (x_{T+k}^T is known for all k)

It can be seen that the prediction $\widehat{y_{T+k|T}}$ depends on only β_T and the data point for $T+k$, as required.

Result B

Let L be the squared error loss function, and let x by an arbitrarily chosen value X can take. We must show that, by letting u be the minimizer of the loss, this results in $u = E(Y|X=x)$.

$$\begin{aligned} L(Y,u) &= E((Y-u)^2|X=x) \\ &= E(Y^2 - 2Yu + u^2|X=x) \\ &= E(Y^2|X=x) - 2uE(Y|X=x) + u^2 \end{aligned}$$

Taking the derivative with respect to u and set this to zero one then has,

$$0 = \frac{dL}{du} = -2E(Y|X=x) + 2u$$
$$\Rightarrow u = E(Y|X=x)$$

In addition, upon applying second derivative:

$$\frac{d^2L}{du^2} = 2 > 0 \qquad \forall u \in \mathbb{R}$$

Therefore, the conditional expectation $E(Y|X=x)$ minimizes the squared error loss for arbitrarily chosen x. Also, the convexity of L on $\mathbb{R}$ implies that not only is $E(Y|X=x)$ a minimum, but it is also unique. Therefore, $E(Y|X)$ is *the* minimum of L.

2.2 Logistic Regression

Logistic regression is a type of discriminative classifier with linear parameters which extend the machinery of linear regression to a classification type setting. We do that extension by replacing the distribution of the dependent variable y with a Bernoulli distribution, which has a binary response, i.e. $y \in (0,\ 1)$. We thus use:

$p(y|x,w) = Ber(y|\mu(x))$

where $\mu(x) = sigm(w^T x)$

The *sigm* function is the sigmoid function, also known as logistic or logit function. It is defined as:

$sigm(\eta) = \frac{1}{1+exp(-\eta)} = \frac{e^{\eta}}{e^{\eta}+1}$

The sigmoid (called as such due to its S-shaped form) is a squashing function, i.e. any input to it will output a number between [0,1] as seen in the figure 1.

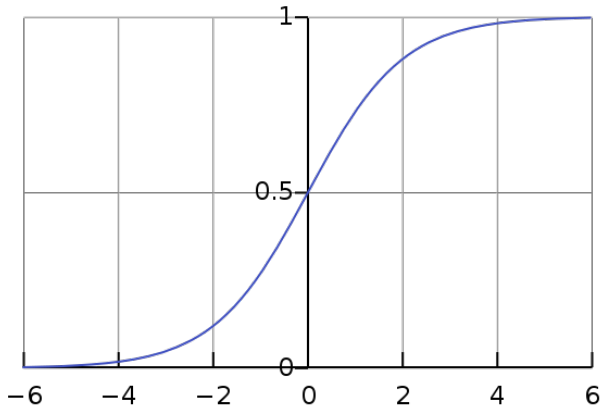


Figure 1. Sigmoid function

Putting the two equations above we get:

$p(y|x,\ w) = Ber(y|sigm(w^T x))$

The equation above is called the logistic regression due to its similarity with linear regression, even though it is in reality a form of classification, not of regression.

From the logistic regression equation we can set a rule based on output probability, e.g. if the probability is set at 0.5, then:

$\hat{y}(x) = 1 \iff p(y=1|x) > 0.5$

The solution to a logistic regression is done by negative log-likelihood, NLL. We apply the log to the likelihood and find the maximum likelihood estimator, MLE, by using a technique called gradient descent, in which we move in the function towards the solution along its steepest descent path.

Logistic regression can be extended to multi-class problems and this is what we did in this project, by creating a 3 class classification problem.

2.2.1 Logistic Regression applications in Finance

The initial applications to finance have always been regression application, in which the impetus was to find the future price through a linear regression algorithm by chasing factors that affected future prices.

Logistic regression has been quite successfully used in finance for problems like credit default forecast, fraud detection, credit card usage, and other classifications tasks in which the objective is, given a large number of parameters, classify the probability of an event.

Dutta, Bandopadhyay and Sengupta [30] studied actively traded stocks in the Indian market using Logistic Regression with success as an enhancer to portfolio managers toolkit. Most of the stock price prediction was run as regression but we believe we can successfully attempt Logistic Regression in our classification project.

2.3 Gradient Boosting

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalises them by allowing optimisation of an arbitrary differentiable loss function.

The idea of gradient boosting originated in the observation by Leo Breiman that boosting can be interpreted as an optimisation algorithm on a suitable cost function.[22] Explicit regression gradient boosting algorithms were subsequently developed by Jerome H. Friedman,[38][39] simultaneously with the more general functional gradient boosting perspective of Llew Mason, Jonathan Baxter, Peter Bartlett and Marcus Frean.[56][57] The latter two papers introduced the view of boosting algorithms as iterative functional gradient descent algorithms. That is, algorithms that optimise a cost function over function space by iteratively choosing a function (weak hypothesis) that points in the negative gradient direction. This functional gradient view of boosting has led to the development of boosting algorithms in many areas of machine

learning and statistics beyond regression and classification.

- Introduction
  As per the exposition of gradient boosting by Li[28].
  Like other boosting methods, gradient boosting combines weak "learners" into a single strong learner in an iterative fashion. It is easiest to explain in the least-squares regression setting, where the goal is to "teach" a model F to predict values of the form $\hat{y} = F(x)$ by minimising the mean squared error $\frac{1}{n}\sum_i (\hat{y}_i - y_i)^2$, where i indexes over some training set of size n of actual values of the output variable y.
  At each stage m, $1 \leq m \leq M$, of gradient boosting, it may be assumed that there is some imperfect model F_m (at the outset, a very weak model that just predicts the mean y in the training set could be used). The gradient boosting algorithm improves on F_m by constructing a new model that adds an estimator h to provide a better model: $F_{m+1}(x) = F_m(x) + h(x)$. To find h, the gradient boosting solution starts with the observation that a perfect h would imply
  $:F_{m+1}(x) = F_m(x) + h(x) = y$
  or, equivalently,
  $:h(x) = y - F_m(x)$.
  Therefore, gradient boosting will fit h to the "residual" $y - F_m(x)$. As in other boosting variants, each F_{m+1} attempts to correct the errors of its predecessor F_m. A generalisation of this idea to loss functions other than squared error, and to classification and ranking problems, follows from the observation that residuals $y - F(x)$ for a given model are the negative gradients (with respect to $F(x)$) of the squared error loss function $\frac{1}{2}(y - F(x))^2$. So, gradient boosting is a gradient descent algorithm, and generalising it entails "plugging in" a different loss and its gradient.

- Algorithm
  In many supervised learning problems one has an output variable y and a vector of input variables x described via a joint probability distribution $P(x,y)$.¡!–(at least for the purposes of theoretical analysis)–¿ Using a training set $\{(x_1,y_1),\ldots,(x_n,y_n)\}$ of known values of x and corresponding values of y, the goal is to find an approximation $\hat{F}(x)$ to a function $F(x)$ that minimises the expected value of some specified loss function $L(y,F(x))$:
  $: \hat{F} = \underset{F}{arg\,min}\, \mathbb{E}_{x,y}[L(y,F(x))]$.
  The gradient boosting method assumes a real-valued y and seeks an approximation $\hat{F}(x)$ in the form of a weighted sum of functions $h_i(x)$ from some class $\mathscr{H}$, called base (or weak) learners:
  $: \hat{F}(x) = \sum_{i=1}^{M} \gamma_i h_i(x) + \text{const}$.
  In accordance with the empirical risk minimisation principle, the method tries to find an approximation $\hat{F}(x)$ that minimises the average value of the loss function on the training set, i.e., minimises the empirical risk. It does so by starting with a model, consisting of a constant function $F_0(x)$, and incrementally expands it in a greedy fashion:
  $:F_0(x) = \underset{\gamma}{arg\,min} \sum_{i=1}^{n} L(y_i,\gamma)$,
  $:F_m(x) = F_{m-1}(x) + \underset{h_m \in H}{arg\,min} [\sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + h_m(x_i))]$,
  where $h_m \in \mathscr{H}$ is a base learner function.
  Unfortunately, choosing the best function h at each step for an arbitrary loss function L is a computationally infeasible optimisation problem in general. Therefore, we restrict our approach to a simplified version of the problem.
  The idea is to apply a steepest descent step to this minimisation problem. If we considered the continuous case, i.e. where $\mathscr{H}$ is the set of arbitrary differentiable functions on $\mathbb{R}$, we would update the model in accordance with the following equations:
  $:F_m(x) = F_{m-1}(x) - \gamma_m \sum_{i=1}^{n} \nabla_{F_{m-1}} L(y_i, F_{m-1}(x_i))$,
  $\gamma_m = \underset{\gamma}{arg\,min} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) - \gamma \nabla_{F_{m-1}} L(y_i, F_{m-1}(x_i))\right)$,
  Where the derivatives are taken with respect to the functions F_i for $i \in \{1,..,m\}$. In the discrete case however, i.e. when the set $\mathscr{H}$ is finite, we choose the candidate function h closest to the gradient of L for which the coefficient γ may then be calculated with the aid of line search on the above equations. Note that this approach is a heuristic and therefore doesn't yield an exact solution to the given problem, but rather an approximation.

In pseudocode, the generic gradient boosting method is:

- Input: training set $\{(x_i,y_i)\}_{i=1}^{n}$, a differentiable loss function $L(y,F(x))$, number of iterations M.
- Algorithm:
  1. Initialise model with a constant value:
     $: F_0(x) = \underset{\gamma}{arg\,min} \sum_{i=1}^{n} L(y_i,\gamma)$.
  2. For $m = 1 to M$:
  3. Compute so-called "pseudo-residuals":
     $: r_{im} = -\left[\frac{\partial L(y_i,F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ for $i = 1,\ldots,n$.
  4. Fit a base learner (e.g. tree) $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i,r_{im})\}_{i=1}^{n}$.
  5. Compute multiplier γ_m by solving the following one-dimensional optimisation problem:
     $: \gamma_m = \underset{\gamma}{arg\,min} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right)$.
  6. Update the model: $: F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$.

  - Output $F_M(x)$.

- Gradient tree boosting:
  Gradient boosting is typically used with decision trees (especially CART trees) of a fixed size as base learners. For this special case Friedman proposes a modification to gradient boosting method which improves the quality of fit of each base learner.
  Generic gradient boosting at the "m"-th step would fit a decision tree $h_m(x)$ to pseudo-residuals. Let J_m be the number of its leaves. The tree partitions the input space into J_m disjoint regions $R_{1m},\dots,R_{J_m m}$ and predicts a constant value in each region. Using the indicator notation, the output of $h_m(x)$ for input "x" can be written as the sum:
  : $h_m(x) = \sum_{j=1}^{J_m} b_{jm}\mathbf{1}_{R_{jm}}(x)$,
  where b_{jm} is the value predicted in the region R_{jm}.[8]
  Then the coefficients b_{jm} are multiplied by some value γ_m, chosen using line search so as to minimise the loss function, and the model is updated as follows:
  : $F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$,
  $\gamma_m = \underset{\gamma}{arg\,min} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))$.
  Friedman proposes to modify this algorithm so that it chooses a separate optimal value γ_{jm} for each of the tree's regions, instead of a single γ_m for the whole tree. He calls the modified algorithm "TreeBoost". The coefficients b_{jm} from the tree-fitting procedure can be then simply discarded and the model update rule becomes:
  : $F_m(x) = F_{m-1}(x) + \sum_{j=1}^{J_m}$
  $\gamma_{jm}\mathbf{1}_{R_{jm}}(x), \quad \gamma_{jm} = \underset{\gamma}{arg\,min} \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma)$.

- Size of trees:
  J, the number of terminal nodes in trees, is the method's parameter which can be adjusted for a data set at hand. It controls the maximum allowed level of interaction between variables in the model. With $J = 2$ (decision stumps), no interaction between variables is allowed. With $J = 3$ the model may include effects of the interaction between up to two variables, and so on.
  Hastie et al.[46] comment that typically $4 \le J \le 8$ work well for boosting and results are fairly insensitive to the choice of J in this range, $J = 2$ is insufficient for many applications, and $J > 10$ is unlikely to be required.

- Shrinkage
  An important part of gradient boosting method is regularisation by shrinkage which consists in modifying the update rule as follows:
  : $F_m(x) = F_{m-1}(x) + \nu \cdot \gamma_m h_m(x), \quad 0 < \nu \le 1$,
  where parameter ν is called the "learning rate".
  Empirically it has been found that using small learning rates (such as $\nu < 0.1$) yields dramatic improvements in model's generalisation ability over gradient boosting without shrinking ($\nu = 1$).[46] However, it comes at the price of increasing computational time both during training and querying: lower learning rate requires more iterations.

- Number of observations in leaves
  Gradient tree boosting implementations often also use regularisation by limiting the minimum number of observations in trees' terminal nodes. It is used in the tree building process by ignoring any splits that lead to nodes containing fewer than this number of training set instances.
  Imposing this limit helps to reduce variance in predictions at leaves.

2.3.1 Gradient Boosting applications in Finance

There exists numerous papers and other literature that consider the problem of financial time series forecasting. Gradient boosting has been recently used as a powerful tool in many of these literatures.

- Kolanovic and Krishnamachari (2017)[60] from J.P Morgan discuss and compare many algorithms as well as useful features and preprocessing techniques. They also provide an introduction to many algorithms, such as Support Vector Machines (SVM) Artificial Neural Networks (ANN), Convolutional Neural Networks (CNN), Long Short-Term Memory networks (LSTM) and XGboost. Several features and algorithms that they describe are also used in this report.
  Although they have not tested all mentioned algorithms, their results indicate that XGboost gives the best performance.

- Evolution of high-frequency systematic trading: a performance driven gradient boosting model[63]: This paper proposes a performance-driven gradient boosting model (pdGBM) which predicts short-horizon price movements by combining nonlinear response functions of selected predictors. They tested this trading system on the high-frequency data of SPDR S&P 500 index ETF (SPY). In the out-of-sample period, it generated an average of 0.045% return per trade and an annualised Sharpe ratio close to 20 after transaction costs. Various empirical results also showed the model robustness to different parameters.

- Linear and Nonlinear Trading Models with Gradient Boosted Random Forests and Application to Singapore Stock Market[67]: This paper presents new trading models for the stock market and test whether they are able to consistently generate excess returns from the Singapore Exchange (SGX). The experimental data, the gradient boosted random forest yields highest total return with a value of 25.14%, followed by random forest whose total return is 24.32%, while the "buy and hold" strategy makes a loss at 1.94% every year.

2.4 Neural Networks

The earliest progress towards modern day neural networks arose from Frank Rosenblatt's 1958 paper *'The perceptron: A probabilistic model for information and organisation in the brain'*[68]. The 'perceptron' that Rosenblatt proposed was the simplest form of a neural network containing only a single neuron. Significantly however, Rosenblatt established that the weights and biases of this neuron should be adjustable in accordance to some calculation of error. Although this discovery remains central to modern neural networks, several limitations impeded the applicability of Rosenblatt's proposal. The primary limitation was that the perceptron could only classify linearly separable patterns and therefore the method was unsuitable for more complex tasks. Additionally, the original proposal was restricted to binary classification tasks which only reduced its potential applications further. In 1969 Minsky and Papert expressed these concerns for the perceptron model in their paper 'Perceptrons. An Introduction to Computational Geometry'[59]. Moreover, the paper identified that Rosenblatt's flaws arose from the single perceptron and instead proposed networks with multiple layers of perceptron's. Ultimately it remained clear that to remove the limitations and awaken the potential of modern day neural networks further advancements remained necessary.

The Backpropagation Algorithm was derived by multiple researchers in the 1960's and provided the foundation for resolving the issues in the perceptron model. Although it was implemented to run on computers by 1970 it was not until 1974 that Paul Werbos proposed that the algorithm had application in neural nets via his thesis *'Beyond regression: new tools for prediction and analysis in the behavioural sciences'*[80]. The findings of Werbos and proposal of multiple neuron layers by Minsky and Papert cultivated in 1985 when both David Parker and Yann LeCun developed the backpropagation algorithm for multilayer neural networks [65][52]. Most significantly it was found that neural networks with multiple layers and backpropagated error could classify even linearly inseparable data. Shortly after these discoveries, David Rumelhart, Geoffrey Hinton, and Ronald Williams published the 1986 paper 'Learning representations by back propagating error' which received significant exposure for its clear and apprehensible explanations [69]. Such exposure ignited a resurgence of interest to the field of neural networks which would continue to bring about further advancements.

This resurgence of interest in neural networks resulted in the development of several subfields. As early as 1989 Yann LeCun introduced the subfield of image recognition using Convolutional Neural Networks (CNN's) in his paper 'Backpropagation applied to handwritten zip-code recognition' [52]. Unsupervised neural nets, called autoencoders were developed as a form of data compression by Hinton and Zemel in the paper 'Autoencoders, Minimum Description Length and Helmholtz Free Energy' [48]. Recurrent neural networks such as LSTM's and neural networks which made decisions through reinforcement learning were also developed and are discussed at length in sections 2.4 and 2.5 respectively.

Moreover, in recent years several optimisations and techniques have been developed to enhance the performance of existing neural networks. Activation functions were refined causing a shift from sigmoid functions to ReLU or leaky ReLU functions and SoftMax functions on the output layer. More recently for specific convolution neural networks, parametric ReLU's have found significant success [77]. Weight and bias initialisations were found to be a crucial issue for neural network performance and guidelines were published on how to optimise these [44] 'Dropout' was developed as a method for promoting independence between neurons to reduce overfitting in the paper 'Dropout: A Simple Way to Prevent Neural Networks from Overfitting' [75]. Regular stochastic gradient descent became outdated as Netervov Momentum and other superior optimisers such as Adagrad and Adadelta were developed to provide a method for adaptive learning rates. The emergence of minibatch sampling was a significant breakthrough as it allowed a far greater number of epochs in the training phase due to a higher computational efficiency. Finally, the designated structure for neural networks altered dramatically with a particular emphasis on making the networks deeper by significantly increasing the number of hidden layers. One such example of this is the modern day convolutional neural network 'ResNet' which contains upwards of 150 layers.

2.4.1 Key Theory for Neural Networks

- The perceptron: As shown in figure 2([43]). The perceptron proposes that the state of a neuron is determined by its inputs, the weights connecting these inputs and a bias. The inputs are multiplied by the weights which allows them to be 'scaled' according to an error function. The state of a neuron is defined as the sum of the products of the weights and inputs plus the bias subject to an activation function.

  $V = \sum_i W_i X_i + b$

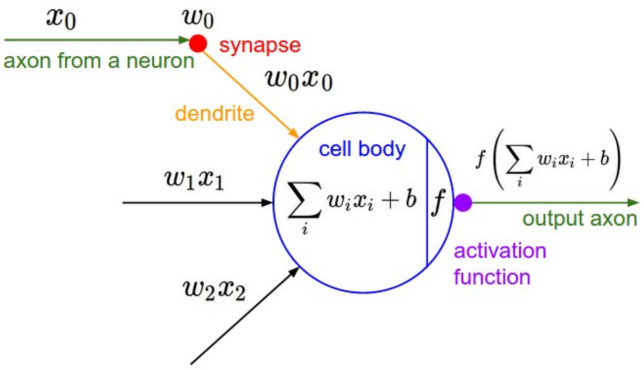


Figure 2. Perceptron

- Activation Functions: Activation functions introduce non-linear properties to the network by converting the input or value of a neuron to an output signal which can

be used as an input to the next layer. Although there are a wide variety of activation functions some have been proven to be more successful than others as shown in figure 3. More specifically when compared to the other sigmoid based functions, the ReLU activation function has the advantage of significantly reducing the likelihood of vanishing gradients. Additional advantages of the ReLU activation function include the sparsity which arises from the inputs that are less than zero as well as faster computational time. A limitation of ReLU however is that it cannot be used on the output layers and thus a SoftMax function is required. Furthermore, under ReLU often some of the gradients are very sensitive and can 'die' during the training stage and therefore a Leaky ReLU is sometimes required. Nevertheless, ReLU remains highly effective and is the most popular activation function used in modern day neural networks

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1+e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

Figure 3. Activation Functions

- Backpropagation algorithm: The backpropagation is essentially extended application of the 'chain rule' seen in calculus. More specifically as seen in 'Learning representations of backpropagated error' [69] the backpropagation algorithm begins with consideration of an error function:

$$E = \frac{1}{2}\sum_c \sum_j (y_{j,c} - d_{j,c})^2$$

Differentiating the error function, we get the change in error (E) with respect to the output layer (y).

$$\frac{dE}{dy_j} = y_j - d_j$$

Using the chain rule we find an expression for the change in error with respect to the input layer (x)

$$\frac{dE}{dx_j} = \frac{dE}{dy_j}\frac{dy_j}{dx_j}$$

Differentiate the activation function to find an expression for $\frac{dy_j}{dx_j}$ and then substitute it into the equation. Assuming a sigmoid activation function:

$$\frac{dE}{dx_j} = \frac{dE}{dy_j} y_j(1-y_j)$$

We have an expression for the change in error with respect to the input layer. However, the value of the input layer is a linear function of the weights and bias. Therefore, considering inputs 'x' as a linear combination of weights (w) and bias (b) we get:

$$\frac{dE}{dw_j} = \frac{dE}{dx_j}\frac{dx_j}{dw_j}$$

$$= \frac{dE}{dx_j} y_j$$

From this:

$$\frac{dE}{dy_j} = \sum_j \frac{dE}{dx_j} w_j$$

This process will need to be repeated for each layer in the neural network up until the very first input layer. From this we get a simple update rule for the weights such that the error is minimised. Note (e) here acts as a fixed learning rate. This process is repeated in a similar manner to determine the update rule for the bias.

$$\Delta w = -e\frac{dE}{dw}$$

- Dropout: As shown in figure 4, dropout is the process of ignoring' a set of neurons and their incoming and outgoing edges at each epoch in the training phase. The 'ignoring' of neurons is performed by assigning a specified probability (usually 0.5) that the neuron is removed from the network. A common problem during the training phase is that *'neurons develop co-dependency which reduces the individual power of each neuron'*[75]. Dropout is implemented to lessen this co-dependency and ultimately limit the amount of overfitting occurring when the model is trained. For this reason, it is expected that a model incorporating dropout will see reduced performance on the training data however increased performance on unseen test data. An additional benefit of dropout is that it promotes neural networks to 'learn more robust features as more combinations and subsets of neurons are considered' [75]. Note that implementing dropout at a probability of 0.5 would require approximately double the amount of training time (epochs) to reach convergence. This is because the dropout causes a reduced version of the network for each iteration of the training process. A visual depiction on the effects of implementing dropout on a neural network is available below [75].

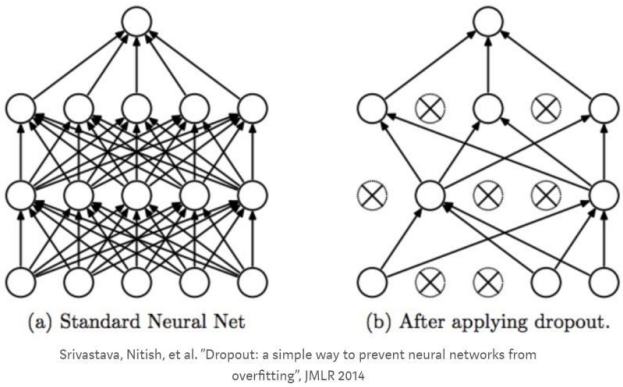


Figure 4. Dropout in Neural Network

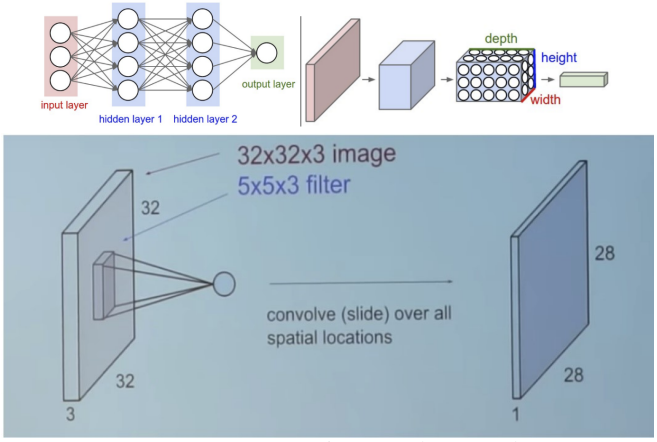


Figure 5. Dropout in Neural Network

- Convolutional layers: Convolutional neural networks share many similarities with standard neural networks in the sense that they contain several layers each housing a certain number of neurons each with learnable weights and biases. The value of each neuron in a layer is the weighted sum of all its incoming weights and biases and this value is used to perform a forward pass until an output layer is produced. Once again as is the case in regular deep neural networks a loss function and backpropagation of weights and biases are employed to train the model. As shown in figure 5 the defining difference of Convolutional Neural Networks however is that they can operate over 2 or 3-dimensional objects which supports the processing of images as inputs to the network. To process these images a convolutional layer slides a filter of designated size over the image and takes the dot product between the image chunk and the filter values for each possible position of the filter on the image. It is commonplace for the outputs of this convolutional layer to be subject to both activation functions and then pooling layers. Finally, convolutional neural networks can be extended with deep fully connected layers (as found in regular neural networks) by flattening the outputs of the final convolutional layer.

- Minibatch sampling: Minibatch sampling is an extremely practical variation of the gradient descent algorithm as it involves processing the training data in 'batches' of a specified size instead of one row at a time. Consequently, the error is calculated over the entire batch and the weights and biases are updated at the end of each batch instead of for every row. The primary advantage of training the model with Minibatch sampling is the immense increase in computational efficiency which is key in tackling problems involving large datasets. Additionally, the increase in computational efficiency allows for a far greater number of epochs in the training phase often results in a significant increase to performance. Despite this, the minibatch sampling method still 'maintains an update frequency which is enough to avoid local minima when achieving convergence' [40]. Ultimately, mini-batch sampling aims to 'provide a compromise between the robustness of regular stochastic gradient descent and the computational efficiency of other common methods such as batch gradient descent' [24].

2.4.2 NN for Financial Trading

The concept of utilising Neural Networks for the application of Machine intelligent financial trading emerged in the early 1990. The concept was developed for the Australian stock market specifically by Vanstone, J. (2005) in the paper 'Trading in the Australian Stock market using Artificial Neural Networks' [78]. The paper discusses that success with neural networks depends upon intelligent feature extraction within the context of financial trading. Consequently, data acquisition and merging from a variety of sources was conducted to develop features such as 'Price to earnings ratio', Current Ratios, Payout ratios and annual per share statistics. Moreover, the paper proposed that implementing neural networks with financial data required the output of the network to be a signal which determined when to enter or exit trades. A 'buy and sell rule' was developed based upon the strength of the signal outputs and is outlined below [78]. The paper acknowledges however that following a simple buy and sell rule such as this creates vulnerability to risk and therefore risk management strategies are mandatory. Additionally, an extensive range of evaluation metrics are suggested to facilitate continual improvements to the networks. The paper concludes that the results from implementing neural networks on financial data suggest they are a viable method however they do not yet provide a complete solution to the complex problem of machine intelligent trading.

The complexity of the financial trading problem is discussed at length in 'Neural Networks, Financial Trading and the Efficient Markets Hypothesis' Skabar, A., Cloete, I. (2002) [74]. The paper aims to determine the correctness of the 'efficient market hypothesis' which states that "the price of an asset reflects all of the information that can be obtained from past

Buy: Buy tomorrow when neural signal output(today) > x, and neural signal output(today) > neural signal output(yesterday)

Sell: Sell tomorrow when neural signal output(today) <= x, and neural signal output(today) < neural signal output(yesterday)

Figure 6. Trading Strategy for NN

prices of the asset." To accomplish the aim the paper proposes a comparison between neural network strategies and a simple buy and hold strategy used as a benchmark. The specific architecture for the neural network was relatively simple consisting of only one hidden layer with sigmoid activation functions and featured extracted inputs such as moving averages or previous prices. A visual representation of the architecture of the selected neural network is presented below [74]. Despite the simple architecture of the network the results of the paper were in favour of the efficient market hypothesis once again suggesting that neural networks are a viable method here.

Academic research from the early 2000's suggested that

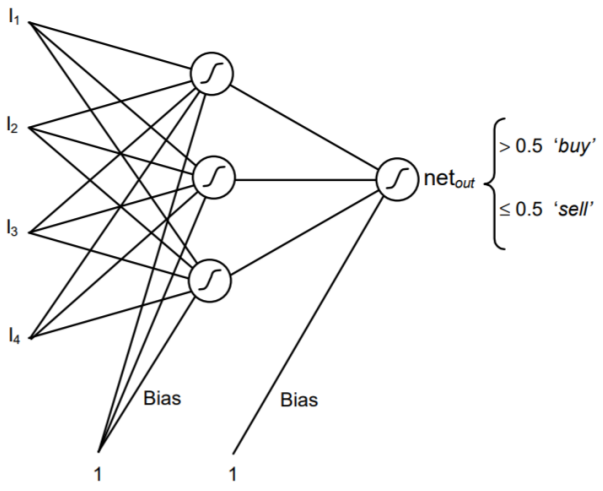


Figure 7. Neural network to decide trade action

neural networks were suitable for financial trading however the specifics of implementing such a method were yet to be refined. JingTao Yao and Chew Lim assisted the process of refining neural networks in a financial context in the paper 'Guidelines for Financial Forecasting with Neural Networks' [83]. The paper suggests that in terms of architecture, deeper networks are not always better in the financial context. Additionally, it is stated that having too many neurons in the input layer can lead to overfitting as the model is more likely to learn by example than a generalised approach. The paper also stresses the importance of insightful feature engineering within the financial context and suggests sensitivity analysis to eliminate the less sensitive features. Finally, it is recommended to consider metrics for success such as goodness of fit as well as accuracy when evaluating the performance of a neural network.

Recent years have brought about further improvements for neural networks within the context of financial trading. Ozbayoglu, A.,& Sezer, O. (2018) developed a method in which 15 technical features were analysed over a 15-day period to create 15x15 2-Dimensional 'images' [64]. These images are then fed through a Convolutional Neural Network to obtain a signal output which determines whether to buy, sell or hold. The results of the study indicate that this convolutional neural network approach outperformed standard buy, sell or hold strategies as well as various other methods such as LSTM's, and standard neural networks over long terms. Siripurapu, A. (2018) also attempted to implement this method however comments that in previous network architecture there was too much pooling, resulting in a final layer which could not distinguish between distinct images due to blur[73]. Consequently, Siripurapu recommends avoiding pooling layers until several cycles of convolution and ReLU activation have been completed.

2.5 LSTM

2.5.1 LSTM origins and previous work

The appearance of Recurrent Neural Networks took place in 1980s and they helped make significant improvements on sequence recognition and prediction. These networks allow each neuron to connect to it's previous state, therefore storing this information in memory. However, this short-term memory feature lacked the capability to bridge longer time lags and dependencies between inputs. In 1997, Long Short-term memory neural networks first appeared as an improvement to recurrent neural nets[49]. Mainly, RNNs where limited in their scope due to very slow learning or poor results when dealing with long term dependencies. This was, in general, due to the vanishing gradient problem, where the error signals approach to 0 while the backpropagation updates the different layers of the net. The authors claim that LSTMs solved this issue and can even "bridge time intervals in excess of 1000 steps". As consequence, they remark that this type of network can grasp both long and short-term behaviour and influence to make predictions.

The solution is described as constraining the error flow through the network. Starting from conventional backpropagation through time, where the error signal is:

$$r_k(t) = f'_k(net_k(t))(d_k(t) - y^k(t))$$

and $d_k(t)$ is the output of unit k. Also, this error signal will be propagated back in time for q steps according to:

$$dr_v(t-q)/dr_u(t) = f'_v(net_v(t-1))w_{uv} \quad / \quad q = 1$$

or

$$f'_v(net_v(t-q))\sum_{l=1}^{n} dr_l(t-q+1)w_{lv}/dr_u(t) \quad / \quad q > 1$$

Thus, if $|f'_l(net_l(t-q))w_{l,l-1}| > 1$ the error flow will increase exponentially as q gets larger producing "oscillating weights and unstable learning".

On the other hand, if $|f'_l(net_l(t-q))w_{l,l-1}| < 1$ the error vanishes as q gets larger and nothing gets learnt by the network.

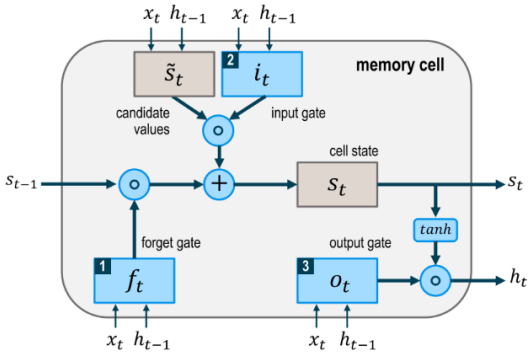
Considering a single neuron connected to itself and to avoid both error vanishing and exploding the authors demand that:

$f_j'(net_j(t))w_{jj} = 1$

Integrating this expression they reach to $f_j(net_j(t)) = net_j(t)/w_{jj}$ and this requires f (the activation function) to be linear which authors refer to as "constant error carousel". This is acknowledged as the main feature of the LSTM development. The concept of gates is introduced to avoid perturbations from irrelevant inputs, called input gates, and to protect other units by irrelevant memory content, called output gates. These gates allow to access and store information in the memory cells about the state of the network. By utilising the tanh function, they effectively bound the output of the network.

Later in 2000, other authors added a forget gate to the architecture. This addressed the problem that the used memory in the cells continued to grow indefinitely and this could cause these networks to stop working[42].

The mathematical formulation for the gates architecture is



1 Forget gate:
Defines which information to remove from the memory (cell state)

2 Input gate:
Defines which information to add to the memory (cell state)

3 Output gate:
Defines which information from the memory (cell state) to use as output

Figure 8. Structure of LSTM memory cell following **Graves(2013)** and **Olah(2015)**

described next:

$f_t = (W_f.[h_{t-1}, x_t] + b_f)$ (1)

$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i)$ (2)

$Ct = tanh(W_C.[h_{t-1}, x_t] + b_C)$ (3)

$C_t = f_t * C_{t-1} + i_t * Ct$ (4)

$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o)$ (5)

$h_t = o_t * tanh(C_t)$ (6)

Source of formulas[53].

First, in equation (1) the forget gate determines which previous state information will be deleted. Here, the σ or sigmoid function represents the gates described previously. Next, the input gate determines which information will be used to update the cell state. In equation 3, vector values are generated as potential candidates for storing. In the following step the new cell state is calculated considering both the forgotten and the new input information. Later, the output gate determines which parts of the cell state will go to the output and a tanh gate bounds the values of the output between -1 and 1.

A couple of shortcomings are mentioned like the network using the memory cells as bias cells on early stages of training or saving redundant information (more than one cell has stored the same information). The authors overcome these problems by activating the memory cells when error stops decreasing and by initialising the gates with a negative bias that pushes the activation of the memory cells towards 0. In their experiments, the authors achieve an improved performance versus simple recurrent networks in different settings. Finally, the authors remark that LSTM network might face issues when trying to solve XOR problems since both states in the XOR clause would have to be stored by the network. Also, LSTM networks include more weights since they may need input and output gates with their respective weights. Among the mentioned benefits, these networks can gap large distances between relevant input values, they have a similar complexity of O(1) like backpropagation algorithm and they don't need significant hyper parameter tuning.

Since their first appearance LSTM neural networks have been utilised for several problems including: time series prediction, speech recognition and music compositions among others. Particularly, we are going to focus in time series prediction for finance. Considering this, we will review in the next section the application of LSTM networks to stock value prediction.

2.5.2 LSTM applications in Finance

Many authors acknowledge the LSTM architecture being suitable for financial predicting problems given the benefits mentioned previously[47]. Since these networks can learn more complex and long-term relationships between data and output. The potential to improve predictions by memorising patterns from the past is frequently remarked. Overall, the acknowledged capacity of LSTM networks on learning long range dependencies is a key feature in the problem we are tackling.

- The LSTM architecture has been applied to the US S&P 500 constituents and has shown to outperform other memory free models like random forest, considered among the best performer models in the past[36]. The authors also develop a rule-based strategy given the model's results on return indices. If a certain share has a better return that the median value of return for that day, it is classified as 1 otherwise as 0. The authors use a sequence of 240 data points of the stocks returns and classify the next data point in a binary class containing those stocks that overperformed vs the cross-sectional median of all the evaluated stocks and the remaining lower performers in the other class. A portfolio strategy of go long with the top k stocks in terms of probability

and shorting bottom k is tested for LSTM and the other methodologies. The LSTM network is stated to outperform significantly the other methods and obtain higher daily returns.

- On other approaches, an LSTM network is mounted on top of an Auto-encoder using price, volume and other macroeconomic features as input[17]. Their objective is to produce sell/buy signals and later evaluate the profitability performance. Using data from different markets (USA, China, Japan and India) they reach to the conclusion that the combination of an auto-encoder and an LSTM model outperforms other simpler models like LSTM or a simple buy and hold strategy.
- Within the portfolio optimisation problem recent studies have remarked the potential of LSTM networks to predict the correlation between a large set of US companies. A combination of ARIMA modelling with LSTM network is stated to outperform other simpler methods[29]. Finally, authors have also combined the LSTM architecture with technical indicators, like MACD and moving averages, as input to build higher performing trading strategies[76]. The application of this approach to the 30 Dow Jones index companies is a viable approach as described by it's authors.

3. PROJECT PROBLEMS

The aim of our project is the prediction of future financial market prices. Given the broad characteristics of this problem, that includes both internal and external factors, we will explore different methodologies, some traditional in the financial context and some of them novel and recently developed. The complexity of the problem is great as there are many players with different pricing expectation methodologies. Much has been researched in the past yet there are no definitive answers on the best approaches.. From fundamental and technical analysis to sentiment or mood analysis, many attempts have been made in the field with various outcomes as described in the literature.

Many different models have been used in the history of financial markets to try to predict a future asset's price, yet often models work well during certain times until a reversion of a trend, a one-off event (a "black swan" event) or just pure luck that the model worked initially. Apart from being able to properly predict the market, one also has to implement trading strategies for locking gains (when the price goes in the forecasted direction) and stop losses (when the price goes again the prediction).

3.1 Project Aims & Objectives

Our objective in this project is to explore and critically review the methodologies described in the previous section to the financial price prediction problem. We will benchmark different methodologies and approaches including State Space models, Gradient Boosting, Logistic Regression and several architectures of the Deep Learning framework. By comparing the accuracy of the predictions of our methods we will evaluate the appropriateness, advantages and disadvantages. The nature of our predictions is such that we have built it as a classification challenge, in which we aim to predict whether a stock variation will occur, but not its magnitude. As an example, if the classification is for 1 day change of 0.5%, then we classify as +1 if the stock went up by more than 0.5% in that period, -1 if it went down by more than 0.5% and 0 if it stayed within that range. Thus a +1 would indicate a buy, a -1 a sell (or short) and a 0 would indicate neutrality.

3.2 Project Questions

The basic question is whether we can use machine learning to augment an investor's toolkit to allow him to do better investment predictions. Our objective is not to create a system that runs autonomously, but to create better indicators that allow an investor to choose whether to buy or sell stocks using past information encoded in pricing, macroeconomic and fundamental company data.

- The most significant question that will raise from this project is whether it is possible to model a problem with this level of complexity and develop it in a way that allows to outperform traditional methodologies.
- We will try to identify and explain which are the most significant factors that contribute to make accurate predictions.
- It will also be important to investigate which configurations and parameters in the developed models bring better results and why.
- Traditional procedures developed through expertise will be analysed and it will be interesting to compare them to novel techniques that require less pre-processing and understanding of the financial markets.

3.3 Project Scope

The scope will be to find out whether predictions have a better than random chance of being right. Random guess, i.e. random stock selection, will be the baseline for investing. Given that our project problem is centred on a three class system (buy, sell or hold) then a random chance would be to have an accuracy of 33%. A result above that would indicate a high confidence in the methods developed. The stocks to be used in the analysis will be the most liquid in the Australian and American markets. Thus we will focus in the 200 companies that form the ASX and 500 stocks of the S%P500 index. We will use historical data, and factors derived from it, to train and evaluate our models. Most of the described techniques have been applied to more developed markets like the U.S. We will also consider an adequate timeframe for all necessary tasks, that includes from data acquisition to modelling the various methods and running the relevant experiments. This project doesn't intend to develop any novel technique, it will

be focalised on the application of existing techniques to the described problem. Even though we expect to perform a simple portfolio strategy evaluation, that will include benchmark against basic trading strategies like "buy and hold", this will not be the centre of our focus and it will be possible to explore more on this on future investigations.

4. METHODOLOGIES

4.1 Machine Learning Models

- **State Space**: In using the state-space model, we chose the state estimates (mean and covariance). Once that was performed, a Kalman recursion commences where iterative updates are performed until convergence, and we can then find out the important predictors based on the state estimates. We will propose a "linear regression" type model with states being the β coefficient, and a Gaussian noise for the observation equation. We also proposed a state equation where the current β estimate is the previous β perturbed by some Gaussian noise (with mean zero).

  Our performance analysis consisted of prediction accuracy. The prediction is in the form of conditional expectation of the posterior predictive Gaussian distribution, given the latest knowledge of the observations. The metric used to quantify the performance of state space model was root mean squared error (RMSE), mean absolute prediction error, and other relative measures. This was used against the pre-specified benchmarks in order to determine whether using Kalman recursions outperformed that of the benchmarks.

- **Logistic Regression**: The logistic regression model was developed as a baseline model to benchmark our other models. This type of model is known to be appropriate for classification as it outputs a probability and derives it's coefficients from the logged odds of each feature. Nevetheless, we explored different hyperparameter configurations like the optimization algorithm, the type of regularization (lasso or ridge) and regularization strength coefficient. The model will output a classification report as well as the corresponding confusion matrix to interpret and compare the resuts.

- **Gradient Boosting**: We use gradient boosting to build an ensemble of decision trees for the classification problem. This combination of weak learners was expected to be between the best performers of traditional machine learning models. We explored different hyperparameter configurations like the number of trees, the learning rate, the max depth of the branches of the trees and the minimum samples per leadf or terminal node. This type of model will be less prone to overfitting since the predictions are product of the average of all the trees.

- **Deep Neural Networks**: As with the other supervised learning methods, deep neural networks require extensive and intuitive feature engineering to be successful. These engineered features form the input layer of the neural network. This data is then processed through the hidden layers and activation functions of the neural network until the output layer is reached. The output layer consists of 3 nodes measuring the output signals for the 3 options (buy, sell or hold) given the feature engineered data for that stock. The backpropagation algorithm is applied and a variety of optimisers including Adagrad, SGD and Adadelta were tested.
  Predictive models in a financial context often learn in a case-by-case manner instead of developing a more generalised approach and thus dropout was used as a method to reduce such overfitting. Minibatch sampling also took part to grant the computational efficiency required to process the vast amount of time series data surrounding stock prices. Finally l different architectures were tested, including both shallow and deeper networks with various subsets of engineered features as the input layer.

- **LSTM**: Before the training stage, LSTM networks require data pre-processing including scaling and sub setting random sequences of the data for testing the validity of our results in any give time and for different stocks. Various features were considered as input as well as the predicted variable itself. This methodology is capable of finding valuable features and patterns that allow it to predict the price of different stocks with accuracy. Various hyperparameter are involved in the architecture development of this model, from the number of neurons, number of layers, length of input sequence, number of training epochs among the most important ones. The validation accuracy is be measured during the training stage to stop training before overfitting becomes evident. Given the multiple configuration options available, finding an optimal solution was challenging and we believe there is room for more improvement in considering more hyperparameter optimisation.

4.2 Data collection

Our project intends to apply machine learning techniques to predict asset prices. For our project we will work with stocks from the ASX200 listing, which comprise the 200 largest companies listed in the Australian Stock Exchange. Furthermore, to test generalisation capacity between different markets we also use data from the SnP500, which includes 500 large USA companies, in terms of market capitalisation. The index price data from both these sources was used in our experiments for the same purposes. Various features will be engineered from the price and volume trading data and this process will be described in depth in further sections.
For economical features we include in the analysis vari-

ous factors with the World Bank as source. This organism collects more than 1800 macro-economical factors for more than 200 countries. The factors used for this project are described in the following section.
Given our objective of obtaining a generalisable model for predicting asset prices, we developed python scripts that use the Alpha Vantage API[3] to download price and volume trading data. This API enables getting both historical and current data for stocks, forex and some cryptocurrencies. For this project we focused on stock price data and our scripts allow to get data from any of the available markets. This development can help in the future to adapt our methods to any of the mentioned markets in very little time. Moreover, the scripts allow us to update our data on a regular basis to keep it up to date and commence a live backtesting of our tool in the near future.
The historical data for ASX200 companies accounts

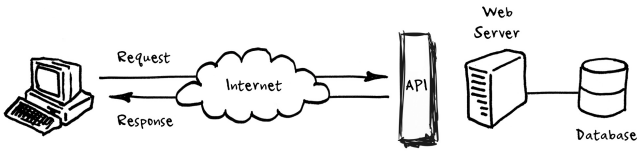


Figure 9. API request schema

for more than 700,000 records of daily Open, High, Low, Close, Adjusted Close and Volume data. For historical companies, the data goes as back as 1999 while for some newer companies only a few hundred records are available. For the SnP500 data we included in the analysis more than 2.5 million records of data with the same features.
Likewise, we developed a python script to connect to the World Bank API that allows to retrieve the mentioned features and update the data when new releases come available. This also let's us acquire a large amount of macroeconomic data from the countries of the markets involved and for some of their most important trading countries too. Since the periodicity of the releases of the World Bank data is different than the daily trading price data, we had to merge it to be able to use it as input for our Machine Learning models. For this, we used the value provided by the source as constant for the corresponding period.

5. Feature Engineering and Extraction

As introduced earlier in the report, financial markets are extremely dynamic systems with numerous impactful players all driving stock price based on market sentiment and analysis. The dynamic nature of financial markets means that the day-to-day closing price of stocks

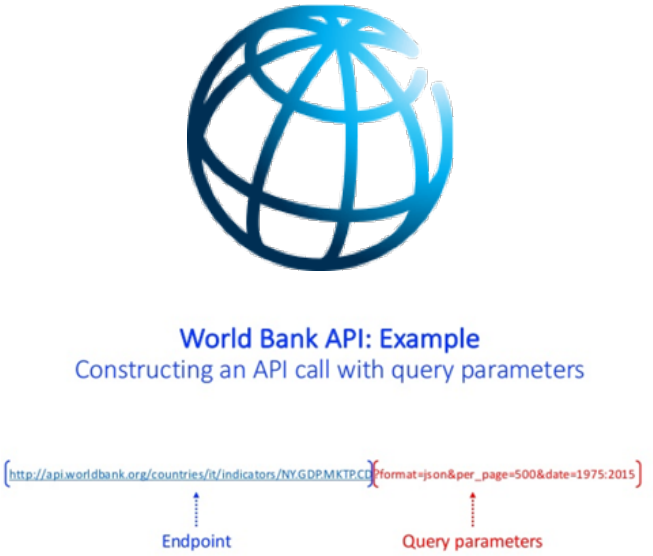


Figure 10. World Bank data query structure

are extremely noisy and the result of a multitude of complex factors. In order to overcome such noise and complexity, extensive and innovative feature engineering is proposed to capture information about trends or patterns in stock price movement.

The possible number of features that could be included in a task such as this is extremely large therefore we must restrict ourselves to a subset of them. We have selected the world bank data features, as well as features related to technical analysis. The world bank data was obtained from Alpha Vantage and consists of economic indicators of many countries. On the other hand, technical analysis was extracted directly from the closing price. This undertaking amounts to incorporating factors that we believe to correlate with closing prices based on our financial intuition along with using engineered features for the technical analysis. By incorporating the two parts together, it is expected that predictive ability for different machine learning methods be improved over merely using one set of indicators.

Our features related to technical analysis are directly manipulated from the closing prices. They were used to describe different aspects of movements in closing price values. Such aspects include the detection of buy/sell signals, possible motives signals such as a bullish/bearish forces, and the movements of the trend. Since different indicators can be used for describing different financial events, we have grouped them based on the aforementioned aspects. Roughly, the first can be described as the "buy/sell" indicators, the second "market motives signal" indicators and the the third "smoothed series" indicators. The second category depends on another stochastic process in addition to the closing prices, unlike the first that took into account only the closing prices.

Below is the total list of engineered or extracted features, grouped into categories based on their nature.

1. **Category 1: Momentum and Buy/Sell Indicators**
   - Momentum
   - MACD
   - Stochastic D and Stochastic K
   - Relative Strength Index (RSI)
   - William's R
   - CCI
   - High/Low
2. **Category 2: Strength Assessment Indicators**
   - Accumulated distribution
   - On-Balance Volume (OBV)
3. **Category 3: Indicators based on Smoothing**
   - Simple Moving Average
   - Weighted Moving Average
   - Exponentially Weighted Moving Average
   - Volume-traded Moving Average
4. **Category 4: Market Stability Indicators**
   - Auto-correation
   - Volatility
5. **Category 5: Macroeconmic Indicators**
   - GDP
   - Debts
   - Inflation
   - Total reserves
   - Industry and Manufacturing
   - Trade
6. **Category 6: Candlestick Chart Feature Engineering**
   - The Hammer
   - Shooting Star
   - Morning Star
   - Three Line Strike
   - Bullish Abandoned Baby

5.1 Category 1: Momentum and Buy/Sell Indicators

The first set of technical indicators consists of those that quantifies gains and losses to measure the movement or momentum of a stock as well as those that establish buy and sell thresholds.

- **Momentum:** The Momentum feature can be thought of as the k lagged difference in price. In other words, we take the difference $Y_t - Y_{t-k}$ where k is pre-specified. The plot of momentum can be accessed from [11] (see also Figure 11). If the plot approaches the line $y = 0$ then this can be a signal for buying. On the other hand, one has possible selling point once the plot starts to "dive" below $y = 0$.

Figure 11. Momentum indicator. Figure obtained from [11]

- **MACD:** The 'Moving Average Convergence Divergence' feature is driven by a MACD signal line, and MACD line, as depicted from Figure 11. The MACD signal line is the exponential moving average (see below) of the closing price with 9-period window. The MACD line is the difference between the 26-period exponential moving average and the 12-period moving average. Of interest will be the difference of the two quantities, which we will call D, and the direction at which this is positive or negative. Noted in [9], when D approaches the $y = 0$ line from above, this possibly can indicate a "selling point", equivalent to the MACD line "diving" below the signal line. On the other hand, a possible "buying point" can occur when D approaches $y = 0$ from the bottom, which is equivalent to MACD line going above the signal line.

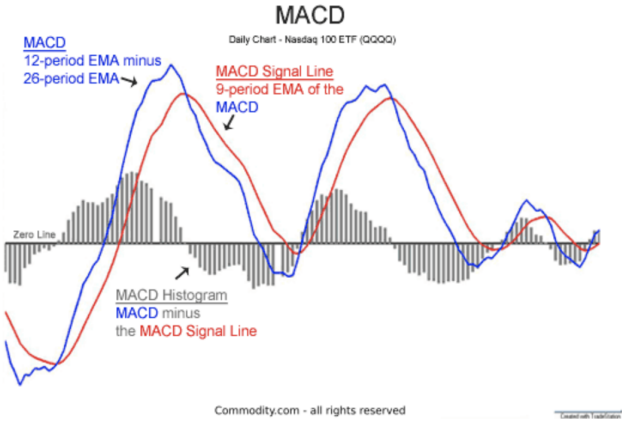


Figure 12. MACD indicator. Figure obtained from [10]

- **Stochastic K and D** These stochastic oscillators measures the amount of deviation a closing price is away from the minimum price, relative to the range. Unlike the previous features the Stochastic K and D are excellent at providing thresholds for determining overbought and oversold points for time series data. The equation for calculating the Stochastic K is available below. Note that Stochastic D is simply the moving average of the Stochastic K.

Figure 13. Stochastics. Figure obtained from [15]

Mathematically, it is expressed as:

$$\text{Stochastic K} = \frac{Y_t - Y_{\text{min over 14 periods}}}{Y_{\text{max over 14 periods}} - Y_{\text{min over 14 periods}}} 100 \tag{12}$$

According to [15], if Stochastic K is above 80, then this can be seen as a sign of overbought. The opposite holds when this quantity is below 20.

- **Williams R:** Williams R measures momentum on a scale of 0 to 100 to potentially indicate any overbought or oversold instances for a stock. Consequently, it is common for investors to include Williams R in their pool of features which determine when to leave and enter the market as it may indicate some degree of trend reversal. The equation for calculating the Williams R is as follows:

$$\text{Williams R} = \frac{\text{highest high} - \text{cloing price}}{\text{highest high} - \text{lowest low}} \cdot -100 \tag{13}$$

- **RSI:** The RSI or Relative Strength Index is another momentum indicator however is particularly concerned with evaluating a stocks recent price performance. More specifically RSI analyses the magnitude of recent price changes to obtain a value within the range of 0 to 100 which can help indicate potential overbought and oversold instances. The equation for calculating the RSI is available below. Note that in this equation 'RS' refers to the ratio of the average of a lag-1 gain to lag-1 losses.

$$\begin{aligned} \text{RSI} &= 100 - \frac{100}{1 + \text{RS}} \\ &= \frac{100 \times \text{RS}}{1 + \text{RS}} \end{aligned}$$

This quantity is then used to check possible overbought or oversell conditions as with Stochastic D. According to [13], if RSI went above 70, it is considered overbought, but it is considered oversold when RSI is lower than 30 (see also Figure 14)

Figure 14. Relative Strength Index. Figure obtained from [13]

- **CCI:** The Commodity Channel Index or CCI is similar to Stochastic K in the sense that it is a stochastic oscillator which measures momentum and can determine overbought and oversold points. It differs however in that it utilises mean deviation in its calculation and is arguably better at recognising cyclical patterns (see [4]). The equation for calculating the CCI is as follows:

$$\text{CCI} = \frac{\text{TP} - \text{20-day MA of TP}}{0.15 \cdot \text{Mean Deviation}} \tag{14}$$

The TP or 'Typical Price' seen in this equation is calculated as:

$$\text{TP} = \frac{\text{high} + \text{low} + \text{close}}{3} \tag{15}$$

- **High/Low:** Also included as features were the highest and lowest points reached by a stock within a given window. The equation for High and Low is given below with u and v indicating the endpoints of the window.

$$\text{High} = \frac{\max\{Y_t : u \le t \le v\}}{Y_T} \tag{16}$$

$$\text{Low} = \frac{\min\{Y_t : u \le t \le v\}}{Y_T} \tag{17}$$

5.2 Category 2: Strength Assessment Indicators

Though not completely separated from the first category in terms of intention, the features in section 2 place a signficant emphasis on evaluating the strength, reliableness and sustainability of stock price increases.

- **Accumulation-Distribution:** The Accumulation Distribution is primarily concerned with assessing the 'justification strength' of closing price increases. If the Accumulation-Distribution indicator is increasing along with the price, we have that the price increase is "justified" and deemed a strong or 'believable' increase. On the other hand, an increase in closing prices accompanied by a decline in Accumulation-Distribution is deemed 'unjustified' and sustained stock price increases are not expected. ([1] (Dr. M. Khushi), [2], and Figure 15). The "non-justified" cases exist in the sense that although prices are rising, there could exist "bullish/bearish" motives behind them. The formulas in [2] show the method of computing Accumulation-Distribution.

Figure 15. Accumulation Distribution. Figure obtained from [1]

- **On-Balance Volume (OBV):** On-Balance Volume is similar to Accumulation Distribution in the sense that both assess the strength and sustainability of stock price increases. Each time the closing price increases, the OBV is increased by the amount equal to the corresponding volume. Similarly, when the closing price decreases the OBV decreases by an amount equal to the volume. In this sense the OBV aims to indicate potential motives and explanations behind stock movement by seeing if these movements are reinforced by the volume traded. The result is that if two peaks in the closing prices are separated by a large trough, as depicted in [12], OBV may be declining despite the later peaks being higher than its predecessor (Figure 16).

Figure 16. On-Balance Volume. Figure obtained from [12]

5.3 Category 3: Indicators based on Smoothing

The third class of predictors all involve the utilisation of moving averages to smooth out the volatile fluctuations of the stock price time series data in order to better capture any underlying trends hidden behind the noise. For each of the smoothing methods discussed below several features were created based on different time periods for example 5 day SMA, 10 day SMA, 25 day SMA and so on.

- **Simple Moving Average (SMA):** Calculating the Simple Moving Average of time series data is one of the most elementary yet effective methods of smoothing and reducing noise. Simple Moving Averages like any moving average are calculated over a given window or time period. It is labelled as 'simple' because this method gives equal weights to all days within the time period when performing the calculation for the moving average. One criticism of the SMA method to smoothing is that observations in a time series often are autocorrelated, therefore weights in the calculation of moving averages can be further adjusted to account for time lags.

Figure 17. SMA for different time periods - From [?]

- **Volume-traded Moving Average:** The Volume-traded Moving Average feature employs the Simple Moving

Average method however it is calculated on the Volume traded instead of the stock price. This smoothing removes the noise in the volume features to be able to identify if any clear trends are occurring. A steady increase in the volume-traded of a stock is a strong indicator that the stock is currently experiencing a bullish market.

- **Weighted Moving Average (WMA):** Weighted Moving Averages address the main criticisms discussed with the Simple Moving Average Methods. In WMA when calculating the moving averages more weight is given to more recent instances and less weight is given to more lagged instances. This ensures that the values returned from the WMA calculations are both smoothed and relevant to the current date in question.

- **Exponentially Weighted Moving Average:** In exponentially weighted moving average method, one takes a convex combination between present observation and predicted observation for the current time point by choosing some number α between 0 and 1, and denote the smoothed value as $\widehat{Y_{t+1}}$ below. The implication of exponential smooting is similar to weighted moving average in that the further into the past the observations are, the less weight were given. The weights decrease exponentially with time lags, and the speed of the decay is governed by α.

$$\widehat{Y_{t+1}} = \alpha Y_t + (1-\alpha)\widehat{Y}_t, \quad \alpha \in [0,1] \tag{18}$$

where:

- $\widehat{Y_{t+1}}$ denotes the estimated price at the next period based on current information $\{Y_s : s \leq t\}$
- $\widehat{Y}_t$ is the estimation of current price using price information from the past

The equation above, as well as the proof of the above statement related to decay of exponential weights in exponential smoothing, can be found in [8].

5.4 Category 4: Market Stability Indicators

The following set of indicators focus on analysing the stability of the market. These features help determine the level of risk involved in stock price movement due to its potential to fluctuate randomly.

- **Autocorrelation:** describes the "standardised" magnitude of linear relationship a current observation has with its past k lags. In other words it measures whether a lagged version of the time series is similar to the current one. Consequently, time series data with a clear trend will have high auto-correlation where as very noise or volatile time series data will tend to have lower auto-correlation. The auto-correlation always takes values in $[0,1]$ and is also the ratio of sample covariance evaluated at s and t, $\widehat{\text{Cov}(X_s, X_t)}$, to products of sample standard deviations, $\sqrt{\widehat{\text{Var}(X_s)}}$, and $\sqrt{\widehat{\text{Var}(X_s)}}$, which is given by:

$$\widehat{\rho_{s,t}} = \frac{\widehat{\text{Cov}}(X_s, X_t)}{\sqrt{\widehat{\text{Var}}(X_s)}\sqrt{\widehat{\text{Var}}(X_t)}}, \quad \forall s,t \in \mathbb{N} \tag{19}$$

- **Volatility:** The volatility of a stock is a highly significant feature in its ability to quantify risk and determine the stability of the market. It is defined as the standard deviation of the observations confined within some interval of time, n_t. Several features were created out of calculating volatility within different windows (10-day volatility, 20 day volatility and so on).

$$\widehat{\sigma_t} = \sqrt{\frac{1}{n_t - 1}\sum_{i=1}^{n_t}(Y_i - \overline{Y})^2} \tag{20}$$

5.5 Category 5: Macroeconomic indicators

The following macroeconomic features were not necessarily engineered but instead were extracted from the World Bank data. A large excess of macroeconomic data was available for extraction and consequently selecting a particular subset was required. The finalised subset of macroeconomic factors that were selected are:

- GDP
- Debts
- Inflation
- Total reserves
- Industry and Manufacturing
- Trade

It is common knowledge that currently Australia's largest trading Partner is China. Consequently, there is strong reason to believe that China's economic factors should correlate with Australian economy and in turn, Australian stock prices. For this reason macroeconomic data from both the Australian and Chinese economy were included in the feature extraction. In terms of the specific economic indicators, **Gross Domestic Product (GDP)** is an obvious choice because it encompasses broad areas of economic activities such as expenditure consumption or government investments which demonstrate the overall health of an economy. **Debts, Inflation**, and **Total Reserves** were included since these factors are directly related to how much a country can spend which greatly impacts

stock price. Additionally, according to page 8 of [16], it could be seen from the barcharts that Australian **Industry and Manufacturing** sectors was among the most productive from a global perspective and therefore it was chosen as a macroeconomic factor. Finally, **Trade** was included as it directly impacts stock price through the buying and selling of associated products or services.

5.6 Category 6: Candlestick Chart Feature Engineering

- **'The Hammer'**: The Hammer is one of the most famous and recognisable candlestick patterns observed by technical analysers when attempting to predict stock price. For a hammer to occur a stock must first be on a downtrend. Then on a given day a stock must drop significantly lower than its opening yet recover to close significantly above its opening. More specifically, the 'tail' created by the low of the stock for the day must be twice the length of the 'real body' of the stocks candlestick. The Hammer often signifies that the downtrend has reached its bottom and that the market is reversing from bearish to bullish. The occurrence of the hammer is confirmed by the following days stock displaying a higher low than the previous day which often results in further buyers entering the market.

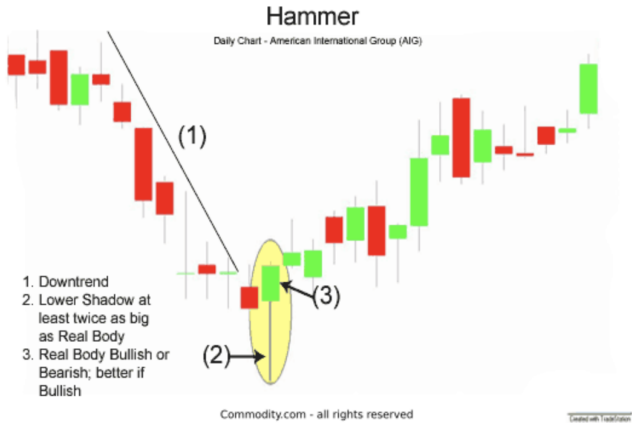


Figure 18. 'The Hammer' [45]

- **'Shooting Star'"**: The Shooting Star is also an iconic candlestick pattern in the world of technical analysis and can be thought of as somewhat of an opposite to the hammer. This is because the conditions required for the existence of a shooting star are opposites to those required to produce a hammer. For example where a Hammer requires a downtrend the Shooting Star instead requires that the stock is currently on an uptrend. Furthermore the Shooting Star requires that the stock rises significantly above its opening price throughout the day however eventually crashes to a point lower than its open resulting in a bearish candle. The occurrence of the shooting star in this manner signifies that the stock is determining its potential 'price top' and that a reversal from a bullish to bearish market is likely.

Figure 19. 'Shooting Star' [71]

- **'Morning Star'**: The Morning Star is a candlestick pattern that involves 3 consecutive candles after coming off a clear downtrend. The first 2 of these 3 bars must be bearish however the final bar must show a resurgence and become bullish. Further specifics on the very particular requirements necessary for the existence of a morning star are outlined in the diagram below. Detecting the existence of a Morning star is important as they are often an indication of a trend reversal from bearish to bullish markets and hence a strong signal that it is a good time to enter the market.

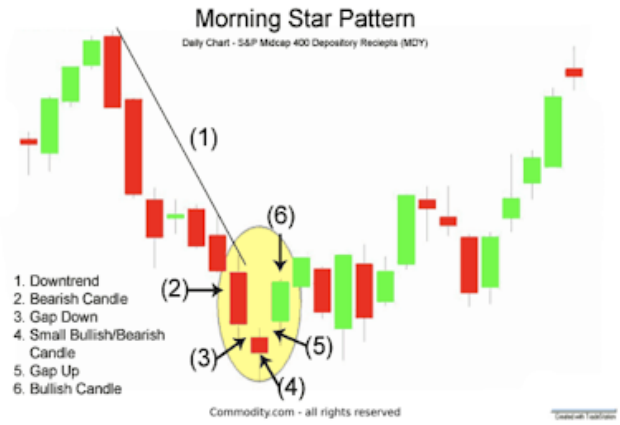


Figure 20. 'Morning Star' [61]

- **'Three Line Strike'**: The Three Line Strike is a less well known candlestick pattern however results have shown it to be one of the most powerful in predicting stock price movement. It is characterised by its 4 consecutive candles each containing lower 'lows' than the next. Most notably however the fourth candle must have a lower 'low' than its preceding candle but must also close for higher than the first candle in the series.

The occurrence of a Three line Strike is a signal that the market trend is reversing to bullish behaviour. According to some research, identifying a Three Line Strike results in a stock price increase on the next day in 84 per cent of cases [26].

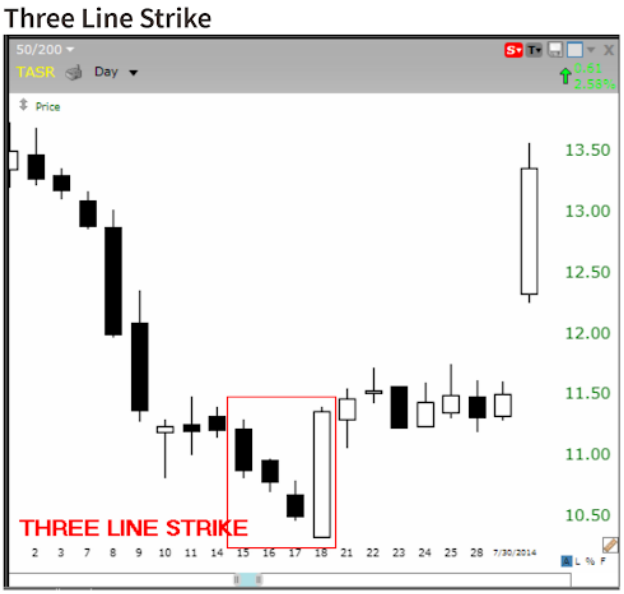


Figure 21. 'Three Line Strike'

- **'Bullish Abandoned Baby'**: The final candlestick formation that was considered for feature extraction was the somewhat oddly named 'Bullish Abandoned Baby.' This particular formation consists of three candles; the first bearish, the second a gapped doji candle and the third a bullish candle that has gapped upwards. The Bullish Abandoned Baby often comes off of a downtrend and indicates that the low of such a downtrend has been established. Consequently the Bullish Abandoned Baby is an indication of trend reversal from Bearish to Bullish which predicts a further stock price increase on the following day with 70 per cent accuracy [26].

Figure 22. 'Bullish Abandoned Baby'

5.7 Experiments

For this project we performed several experiments with different subsets of the data and with all the Machine Learning models developed. The aim of these experiments is both to test the generalisation capabilities of our models and to benchmark them against each other. As described in previous sections, the developed framework allows us to perform these tests in a reproducible and reliable manner assuring comparability among the results. Further experiments will be conducted in the future to comprise other markets and even securities.

Since our data is separated in each stock we can randomly choose stocks to train with and avoid any sort of bias in that aspect. The developed framework has the capability to easily create and incorporate new experiments. This will allow us in the future to incorporate new data, features and time horizons for our models.

We performed these experiments with different variations on the target variable, in order to test the prediction capacity on different time horizons. The baseline configuration was price change for one step (trading day) split into 3 categories, up by 0.5 percent or more, down by 0.5 percent or more and the third with the values in between those categories.

We performed 4 general types of experiments as described next:

- Experiment 1: Train model with S&P500 index historical data Test model with ASX200 index historical data
- Experiment 2: Train model with a subset of S&P500 stocks Test model with a subset of ASX200 stocks

  This experiment consisted on 30 randomly selected stocks from the American stock market, equivalent to 150930 daily price records with their engineered features from the following companies that conformed the

training set: Likewise, the test set consisted of 5 ran-

Ticker	Company	Ticker	Company
A	Agilent Technologies	KORS	Michael Kors Holdings
ADM	Archer-Daniels Midland	LUV	Southwest Airlines
AET	Aetna	MAS	Masco
ANSS	Ansys	NLSN	Nielsen Holdings
BLL	Ball	PNC	PNC Financial Services Gr
CL	Colgate-Palmolive	PPG	PPG Industries
DG	Dollar General	QRVO	Qorvo
DISCK	Discovery	RE	Everest Re Group
EQIX	Equinix	RHI	Robert Half International
FLIR	FLIR Systems	SBUX	Starbucks
GPS	Gap	SHW	Sherwin-Williams
IBM	IBM	SRCL	Stericycle
INTU	Intuit	TSN	Tyson Foods
IPG	The Interpublic Group	V	Visa
KO	Coca-Cola	XEL	Xcel Energy

Figure 23. Training set: S&P randomly selected stocks

domly selected stocks from the Australian stock market or 21222 daily price records that comprised the test set.

Ticker	Company
AWC	Alumina Limited
CMW	Cromwell Prop Ordinary
QBE	QBE Insurance Group
RHC	Ramsay Health Care
SWM	Seven West Media Limited

Figure 24. Test set: ASX randomly selected stocks

- Experiment 3: Train model with a subset of ASX200 stocks Test model with a different subset of ASX200 stocks
- Experiment 4: Train model with a subset of S&P500 stocks Test model with a different subset of S&P500 stocks

By restricting both stocks and indices to be able to be part either of the training or testing data during each experiment, we make sure of the generalisation capability of our models and, at the same time, that we don't incur in any sort of errors related to mixing train and test data. The nature of the data and the methodologies used are very conducive to short term predictions.

These experiments were run on features extracted, which conform a large matrix of date and features, where the dependent variable is return 1 days forward. We used the following modelling techniques on the data extracted:

- Logistic regression (with regularisation?)
- Artificial Neural networks
- Gradient Boosting
- LSTM
- State space with Kalman filter

5.8 Results

Success in using the techniques will be measured by:

- Whether model correctly predicted market direction for next 1 to 5 days.
- Whether model correctly predict price changes for the next 1 to 5 days.

If we see that we can find reliable stock prediction models in steps 1 and 2 above, then we'll attempt to build a diversified portfolio which would be rebalanced on a daily basis – taking into account transaction costs – and compare it to the benchmark index.

6. RESOURCES

6.1 Software

Our system was run on Python 3 using Spyder as a coding tool. We used Python 3 as a platform, with Scikit-Learn, Tensorflow, keras, and pydlm as packages for the analyses. Processing platform was local laptop computers of members of the group and the units with GPU had a much better performance. We used Github as a repository platform so we could all collaborate on the code simultaneously. Communication was mostly done through Slack and Whatsapp and video conference through Zoom and Google Hangouts. Final report was produced in LaTeXusing TeXworks.

6.2 Materials

Data sets were obtained from public information data. Pricing data was obtained from Alphavantage [3], which has extensive APIs to download stock price data. Macroeconomic information was obtained from the Worldbank database [82]. Additional code for preprocessing feature engineering was generously ceded to our group by Dr. Matloob Khushi [58].

6.3 Roles & Responsibilities

Our group split the work according to most personal competencies, yet everyone was always following the whole group. All the sessions had a primary responsible person with full group collaboration. The main roles of each members were:

- Aditya Kant Sharma – Create the object oriented Python code and apply the Gradient Boosting methodology
- Ignacio Colino – Extract price and macroeconomic data and apply the Long Short-term memory methodology
- Pote Pongchaikul – Extract the feature engineering and apply state-space modelling as a regression problem
- Sergio Kulikovsky – Obtain fundamental analysis data and apply logistic regression as a classification problem

- Thomas Brown – Extract the technical analysis features from the pricing data set and apply the Neural network modelling

7. Milestones

Following is the schedule of the main activities performed by the group during each of the 14 weeks allocated to the project:

- Week 1 - August 10, 2018
    1. Making sense of the problems.
    2. A draft of steps that could be followed:
        (a) LSTM, Reinforcement learning;
        (b) State-space model;
        (c) Linear regression;
        (d) Proposed data acquisition from Capital IQ - Our first step in the gathering of data was to obtain them from Bloomberg terminals and Capital IQ. These three sources were chosen initially because they provide financial quantities which, based on our financial intuition, should correlate with the stock prices to be predicted.
        (e) Retrieve data from ASX and World Bank - Our second option is to retrieve the macroeconomic data from World Bank API, and stock data from alpha vantage. This provides numerous economic indicators from a large selection of countries, ranging from GDP, government debts, net exports, etc. Because we believe, from our intuition, that economic indicators can correlate with stock prices, we have incorporated them as predictors. For instance, one of our goals is to predict Australian stock prices traded in ASX200. From our knowledge, China is one of the major global trading partners. So, we believe that China's economic factors should correlate with Australian economy and in turn, Australian stock prices. This convinced us that we should include Australian and China's economic indicators as predictors. In addition, other countries' economic indicators were ignored due to computer memory constraints. To get stock prices, we opted for alpha vantage as our choice for obtaining closing prices for ASX200 and S&P500 data, both indices and stocks themselves.
- Week 2 - August 17, 2018
    1. A review and investigation into various methods used in the literature;
    2. applications in literature;
    3. Consideration of factor analysis from Capital IQ data.
- Week 3 - August 24, 2018
    1. Explanation of each method, discussion of expected outcomes, and the preparation of project schedule;
    2. Rejected the idea of using capital IQ data due to risks involved - We must forgo this choice because of the fact that they are difficult and time-consuming to retrieve.
- Week 4 - August 31, 2018
    1. A project proposal detailing deliverables and methodologies.
- Week 5 - September 7, 2018
    1. A dataset consisting of attributes from world bank data and feature engineered data - with functions for feature engineering from Dr. M. Khushi [58]. We have also included features that were related to technical analysis. These consist of weighted moving average, autocorrelations, volatility, MACD, etc. Each of these predictors were computed from the closing prices, and we acknowledge Dr. M. Khushi for providing this source. Our intention in incorporating these predictors are to consider the different "signals" extracted from closing prices, to use them as features, and to investigate the extent to which they can predict stock prices. Together, the features for technical analysis as well as world bank data were used as features.
    2. Functions to implement candlestick patterns devised - Additionally, "candlesticks patterns" were also considered as possible predictors. These patterns correspond to the different stock price movement patterns that were pictorially represented as "candlesticks". We investigated into the addition of these, including "hammer", "shooting star", "morning star", "Three Line Strike", and "Bullish Abandoned Baby".
- Week 6 - September 14, 2018
    1. Analyzed 200 ASX dataset for analyses, and some predictive results from Neural network classification;
    2. Functions to implement candlestick patterns rejected afterwards - We found that each of them rarely occurs, and we believe that incorporating them would not amount to an increase in predictive power. As a result of this, we decided to exclude them as predictors.

3. Switch from Reinforcement learning to Gradient Boosting.

- Week 7 - September 21, 2018 to Week 8 - October 12, 2018

  1. Implementing codes and testing/debugging (creation of final datasets)
     - Since the retrieved world bank data were obtained from outside source (alpha vantage) and technical analysis features were extracted directly from the available closing prices, the obstacles we faced were merging problem, and "data size" problem. To expand, world bank data provide economic data on an annual basis, whereas our stock price data was on a daily basis. If we merged these two directly without preliminary transformations, at least we will obtain inconsistencies in both the number of observations and time inconsistencies.
     - One approach we took is to take the annual observation of the world bank data and to keep this at a constant value through the entire year of interest. Once this was done, we merge them via the merge function available in pandas package. However, this approach was not deemed to be successful since missing values were created as a consequence. We therefore, as our final solution, resort to the removal of some of these features. One of the criteria of removal is if the number of missing values of a feature exceeds a prespecified threshold, then the feature is ignored.
     - Our next problem concerns the size of the data. After our process of merging and features removal, the size of our datasets for S&P500 and ASX200 were far too large for our computers to handle. Therefore, we have to slice the dataset by stocks, and randomly select a specified number of them in a random fashion. Once this is completed, we then concatenate these datasets to form our final datasets. This procedure was done for both ASX200 and S&P500, and randomisation was introduced to avoid possible biased sampling.
     - We found the values for the world bank data were of considerable orders of magnitude larger than the stocks, and this will pose later problems when we train our machine learning algorithms, such as "wrongly regularised" lasso or ridge penalty in the case of logistic regression. We have standardised each of our predictors in order to scale all values to be within a common scale.
     - Since prediction tasks can be taken from two perspectives – regression and classification – it is then necessary to perform transformation on the stock prices prior to applying classification methods. In classification, the aim is to predict buy/sell/hold signals. To obtain these, some criteria will be applied [Insert criteria]. On the other hand, discretisation of the response variable is not necessary for regression model since the aim is to predict the numerical stock values.
  2. Finalise progress report.

- Week 9 - October 12, 2018

  1. Implementing codes and testing/debugging;
  2. Performance analysis on ASX200 stocks data (with training done on ASX200);
  3. Performance analysis on S&P500 index data (with training done on ASX index);
  4. Performance analysis on stocks in stocks in ASX200 data (with training done on stocks in S&P500);
  5. Performance analysis on ASX150 data (with training done on ASX50) Preparation of final report

- Week 10 - October 19, 2018

  1. Create unifying structure - This is to bring together all dataset, all experiments of interest, and methodologies together to analyse data in one place, and to streamline the many implementations. Thus unifying structure will involve feature engineering and data preprocessing as well.
  2. Prepare reports;
  3. Presentation drafts - Included decisions of the presentation content based on the report, and with the supervisor's feedback to further improve clarity.

- Week 11 - October 26, 2018

  1. Finalise report - Elaborate on the findings conveyed in the presentation in detail.
  2. Presentation slides preparation

- Week 12 - November 2, 2018

  1. Finalise presentation slides.

- Week 13 - November 9, 2018

  1. Present project to University.

- Week 14 - November 16, 2018

  1. Demonstrate code to Dr. Matloob Khushi;
  2. Finalise report writing - finalise the writing of methodologies, results, discussions and findings.
  3. Submission of final report.

8. RESULTS

List out project outcomes such as description of experimental results, data analysis, and prototype development. Interpret results to answer project questions raised.
Provides insightful knowledge of project results by identifying and discussing project outcomes. Demonstrate critical knowledge by interpreting results or project outcomes to answer project problems, issues or questions raised.

8.1 Evaluation Metrics

To evaluate the performance of the classifiers in completing each experiment classification reports and confusion matrices have been provided. A classification report presents three important evaluation metrics; Precision, Recall and F-score for each class in the study. An explanation of these evaluation metrics is provided below. Alternatively a confusion matrix indicates the number of errors that exist between each of the classes in the study allowing for an examination into the types of common errors that a model is making.

- **Precision:** The precision metric measures the ratio of correct positive predictions of a class to the total number of guessed predictions for that class. The equation for calculating precision is available below.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \quad (21)$$

- **Recall:** The recall metric captures the ratio of correct predictions of a class to the total number of instances of that class. The equation for calculating the recall is available below.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \quad (22)$$

- **F-Score:** Finally, the F-Score is simply the weighted average of the Precision and Recall metrics. Consequently the equation for calculating the F-Score is as follows.

$$\text{F-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Recall} + \text{Precision}} \quad (23)$$

These evaluation metrics were suitable for classifiers such as the Neural Networks or Gradient Boosting however they are unsuitable for regression methods such as State Space Modelling. Instead, to evaluate the performance of the State Space Modelling, Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) are provided. The MSE measures the average squared difference between the predicted values and the reference value and the RMSE simply takes the square root of this. The equation for calculating the MSE is provided below:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad (24)$$

8.2 ASX/S&P Index Experiment - Results

8.2.1 Logistic Regression

The following classification report depicts the precision, recall and f-score by class for the **Logistic Regression** Method for the ASX/S&P **index** experiment. The figure below depicts

Table 1. Classification Report for Logistic Regression

Class	precision	recall	f-score	support
buy	0.52	0.49	0.51	972
hold	0.61	0.61	0.61	2180
sell	0.38	0.40	0.39	808
avg/total	0.60	0.61	0.60	3960

the confusion matrix caclulated from the **Logistic Regression** Method for the ASX/S&P **index** experiment.

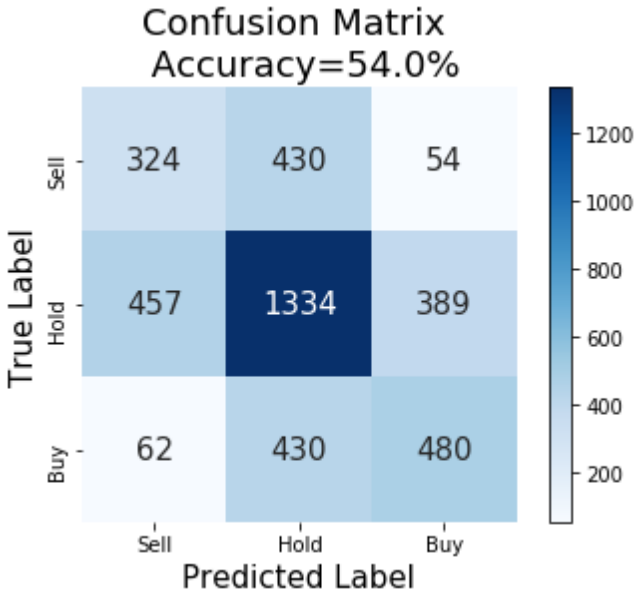


Figure 25. Index data: Logistic Regression Confusion Matrix'

8.2.2 Artificial Neural Networks

The following classification report table:2 depicts the precision, recall and f-score by class for the **Neural Network** Method for the ASX/S&P **index** experiment.

Table 2. Classification Report for Neural Network

Class	precision	recall	f-score	support
buy	0.53	0.57	0.55	972
hold	0.65	0.70	0.67	2180
sell	0.57	0.40	0.47	808
avg/total	0.60	0.61	0.60	3960

The figure:26 depicts the confusion matrix caclulated from the **Neural Networks** Method for the ASX/S&P **index** experiment.

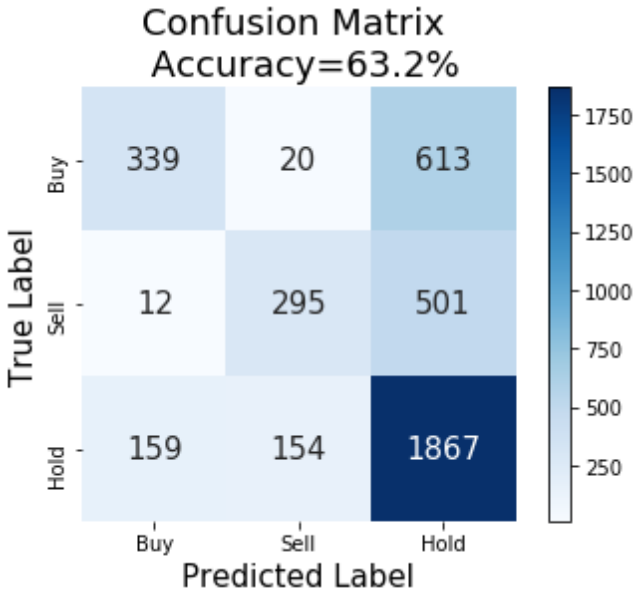


Figure 26. Index Data: Neural Network Confusion Matrix

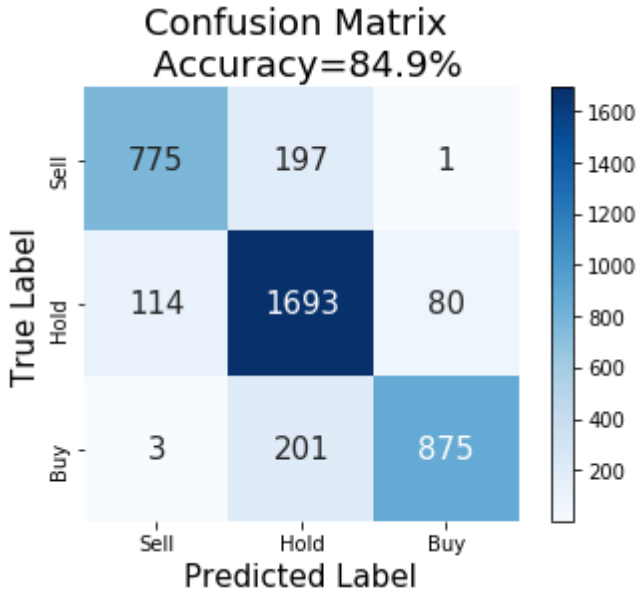


Figure 27. Index Data: LSTM Confusion Matrix

8.2.3 LSTM's

The following classification report table:3 depicts the precision, recall and f-score by class for the **LSTM** Method for the ASX/S&P **index** experiment.

Table 3. Classification Report for LSTM

Class	precision	recall	f-score	support
buy	0.92	0.81	0.86	972
hold	0.81	0.90	0.85	2180
sell	0.87	0.80	0.83	808
avg/total	0.85	0.85	0.85	3960

The figure:27 depicts the confusion matrix caclulated from the **LSTM** Method for the ASX/S&P **index** experiment.

8.2.4 Gradient Boosting

The following classification report 4 depicts the precision, recall and f-score by class for the **Gradient Boosting** Method for the ASX/S&P **index** experiment. The figure:28 depicts

Table 4. Classification Report for Gradient Boosting

Class	precision	recall	f-score	support
buy	0.52	0.51	0.51	972
hold	0.65	0.72	0.68	2180
sell	0.59	0.43	0.49	808
avg/total	0.61	0.61	0.60	3960

the confusion matrix caclulated from the **Gradient Boosting** Method for the ASX/S&P **index** experiment.

8.2.5 State Space Modelling

The following table:5 presents the various evaluation metrics used to assess the performance of the **State Space Modelling** for the ASX/S&P **index** experiment.

Table 5. Evaluation Metrics for State Space

Metric	Result
MSE	20,676,315
RMSE	4547
MSE (Naive)	6,069,425
RMSE (Naive)	2464

8.3 ASX/S&P Index Experiment - Interpretation of Results

Overall, the results for the ASX/S&P Experiment provide significant evidence that machine intelligent financial trading is a potential solution to the stock price prediction problem. This is particularly the case for the LSTM method which outperformed all other methods boasting incredibly high precision and recall statistics of around 80-90 over all three classes. Success was also found through Artificial Neural Networks, Gradient Boosting and Logistic Regression which all hovered at around 60% accuracy which is well above the 33% benchmark expected for a classification task of this manner. The State Space Modelling Method however performed rather poorly on this experiment resulting in significantly high mean squared error to the point where the Naive scores were lower. Additionally, from the collection of classification reports it is evident that over all the different methods some classes were predicted with more success than others. In particular 'hold' was the best predicted class, presenting the greatest f-score of all classes for every single classifier. This result may be due to a class imbalance issue in that the 'hold' class has more than double the support of the other classes which can tend to it being favoured by the algorithms. On the other end of the spectrum 'sell' was the worst predicted class boasting the lowest f-scores of any class across all the methods. Although this may be in part explained by the low support it is also

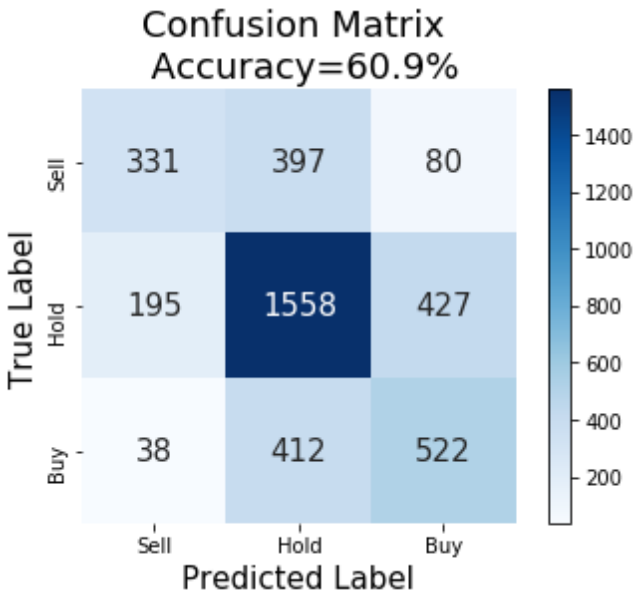


Figure 28. Index Data: Gradient Boosting Confusion Matrix

likely that this is the result of selling points being naturally harder to predict than buying points when analysing stock price.

The confusion matrices provide further evidence that the methodologies implemented in this report are a suitable solution to the stock price prediction problem. In particular it can be seen that the confusion matrix for the LSTM contains significantly little errors between buy/sell and sell/buy. This is of importance as it is far more costly for a trading system to be labelling a selling point as a buy than it is to label a hold as a buy. Analysis of the other confusion matrices reveals that this ability to avoid the worst type of errors is shared by the Neural Network, Gradient Boosting and Logistic Regression methods. Ultimately, the ASX/S&P experiment has provided significant evidence of achieving the primary goal of the report; to beat the market using machine intelligent financial trading. Furthermore these results demonstrate the crucial ability for the algorithms to learn from a specific market to develop generalised knowledge which can then be applied across other markets.

8.4 ASX/S&P Stocks Experiment - Results

8.4.1 Logistic Regression

The following classification report table:6 depicts the precision, recall and f-score by class for the **Logistic Regression** Method for the ASX/S&P **stock** experiment.

Table 6. Classification Report for Logistic Regression

Class	precision	recall	f-score	support
buy	0.48	0.28	0.35	7602
hold	0.31	0.58	0.40	6279
sell	0.37	0.25	0.35	7341
avg/total	0.39	0.36	0.35	21222

The figure 29 depicts the confusion matrix caclulated from the **Logistic Regression** Method for the ASX/S&P **stock** experiment.

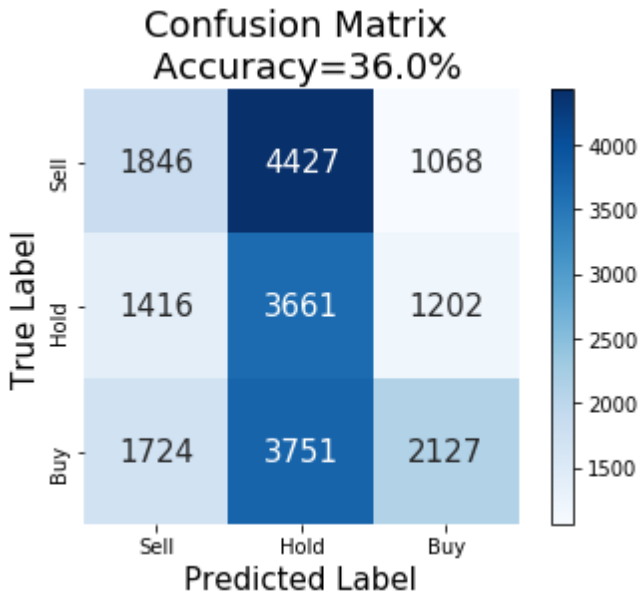


Figure 29. Stock Data: Logistic Regression Confusion Matrix

8.4.2 Artificial Neural Networks

The following classification report table:7 depicts the precision, recall and f-score by class for the **Neural Network** Method for the ASX/S&P **stock** experiment. The figure 30

Table 7. Classification Report for Neural Network

Class	precision	recall	f-score	support
buy	0.42	0.56	0.48	7602
hold	0.29	0.15	0.20	6279
sell	0.43	0.45	0.44	7341
avg/total	0.38	0.40	0.38	21222

depicts the confusion matrix caclulated from the **Neural Network** Method for the ASX/S&P **stock** experiment.

8.4.3 LSTM's

The following classification report table:8 depicts the precision, recall and f-score by class for the **LSTM** Method for the ASX/S&P **stock** experiment. The figure 31 depicts the

Table 8. Classification Report of LSTM

Class	precision	recall	f-score	support
sell	0.83	0.95	0.89	7337
hold	0.88	0.50	0.64	6275
buy	0.79	0.97	0.87	7600
avg/total	0.83	0.82	0.81	21212

confusion matrix caclulated from the **LSTM** Method for the ASX/S&P **stock** experiment.

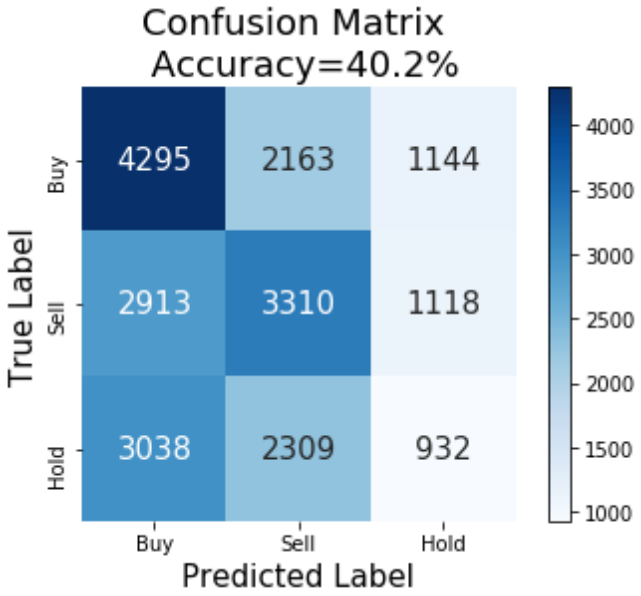


Figure 30. Stock Data: Neural Network Confusion Matrix

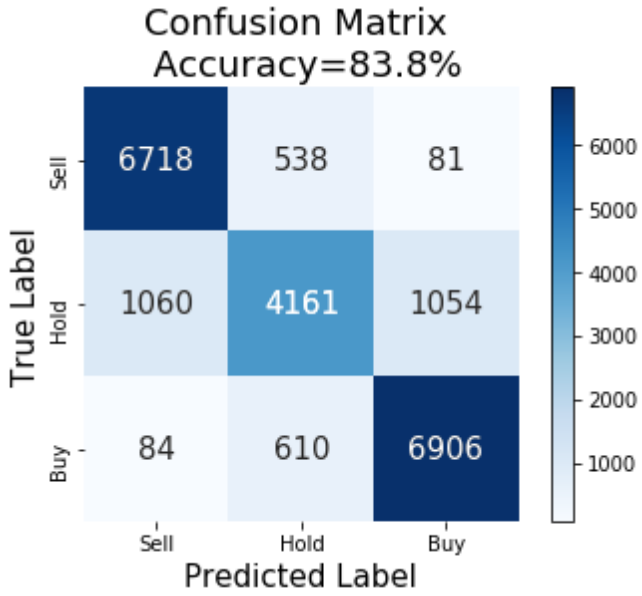


Figure 31. Stock Data: LSTM Confusion Matrix

8.4.4 Gradient Boosting

The following classification report table:9 depicts the precision, recall and f-score by class for the **Gradient Boosting** Method for the ASX/S&P **stock** experiment.

Table 9. Classification Report of Gradient Boosting on stock data

Class	precision	recall	f-score	support
sell	0.33	0.45	0.38	7341
hold	0.30	0.07	0.11	6279
buy	0.35	0.46	0.39	7602
avg/total	0.33	0.34	0.31	21222

The figure:32 below depicts the confusion matrix caclulated from the **Gradient Boosting** Method for the ASX/S&P **stock** experiment.

8.4.5 State Space Modelling

The table:10 presents the various evaluation metrics used to assess the performance of the **State Space Modelling** for the ASX/S&P **stock** experiment.

Table 10. Evaluation Metrics for State Space

Metric	Result
MSE	195,962
RMSE	443.7
MSE (Naive)	1032
RMSE (Naive)	32.1

8.5 ASX/S&P Stocks Experiment - Interpretation of Results

The results for the ASX/S&P Stocks Experiment were far weaker on the whole when compared to the ASX/S&P Index Experiment. In particular the Gradient Boosting, Artificial Neural Net and Logistic Regression methods all reported significantly low accuracies of around 30-40% which is barely an improvement on random guessing. The State Space Modelling performed far better on the stock experiment than the index experiment however its mean squared error results are still yet to outperform the naive counterparts. Crucially however, the LSTM maintained its significantly high performance and once again achieved remarkable f-score statistics over all of the classes. This result is evidence that the memory capabilities of the LSTM allow it to better capture the cyclical trends that underlie time series stock price data.

The classification reports reveal that the methods of Gradient Boosting, Neural Networks and Logistic regression all have significantly low recall for the 'hold' class. This inability to capture the 'hold' class explains the considerably low accuracies obtained by these models in this experiment. Even the LSTM contained a dramatically low recall for 'hold' of 0.50 when compared to the recall of 0.95 and 0.97 for buy and sell respectively. The low recall of hold coupled with its high precision indicate the underguessing 'hold' was the most significant error being made by the model. The inability to capture the 'hold' class may be due to a difference in which factors are likely to indicate a hold between the stock data and index data.

The confusion matrices for the Gradient Boosting, Neural Net and Logistic Regression methods only reinforce that the 'hold' class was a significant issue in this experiment. Additionally, it can be seen that there are a vast number of errors between buy/sell and sell/buy which indicates that even beyond the inability to capture the 'hold' class there are no redeeming aspects to these models for this experiment. The confusion matrix for the LSTM however reveals that once

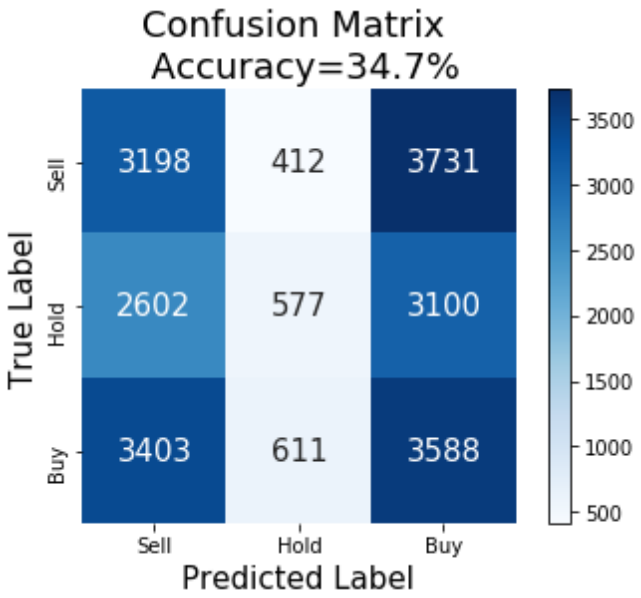


Figure 32. Stock Data: Gradient Boosting Confusion Matrix

again this method is able to avoid making the worst type of errors with only 84 cases of predicting sell into buy and 81 cases of predicting buy into sell. The LSTM's success in learning generalisable trends over different markets for both stock and index data cement the primary goal of this project; that machine intelligent financial trading can beat the market.

9. DISCUSSION

Our initial question, whether machine learning can beat market, has been shown to have quite positive results in our project. Prediction accuracy results were quite above the random, baseline choice.

We worked considerably on learning and applying common feature engineering techniques for this type of problem like real traders do. We focused both on features derived directly from the price time series as well as on candle-sticks pattern recognition. The inclusion of this expert knowledge was fundamental to achieve solid prediction results later on.

Our project took two approaches to the stock price prediction problem. One was through a classification problem using Gradient Boosting, Logistic Regression, Long Short Term Memory Neural Networks (LSTM) and deep Neural Networks. The second one was through a regression problem using State Space and Kalman filter techniques.

Our classification problem was set as a prediction of whether a security would have a variation above a certain threshold over a certain period. Those two parameters could be changed for each of our experiments. As an example a threshold of 0.5% over a 1 day period would classify as (i) +1 if the stock went up by more than 0.5% in 1 day, (ii) -1 if it went down by more than 0.5% and (iii) 0 if it stayed within that range. Thus a random investing choice would have a 33% chance of getting the stock prediction correct.

The best model out of all experiments tried was the LSTM as it was a Neural Network putting more weight into more recent data. This was an expected result given the abundance of data we had available and the acknowledged potential of these models for this type of problem. These are our general findings but other models may adapt better in other types of securities or experiments. Apart from that, we should keep testing the generalisation capability of our models to other markets and back test our results as described in following sections. Apart from our algorithmic results, we focused on

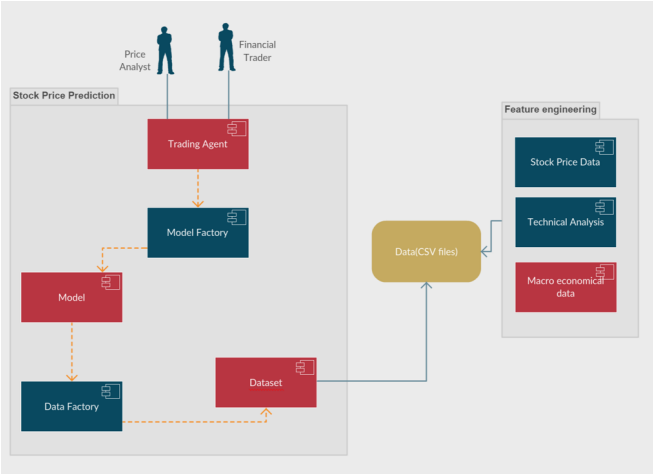


Figure 33. Concept diagram of ML framework

developing a framework that allows us to compare and test different machine learning models with a reliable and flexible approach. Furthermore, it gives the user the possibility to test different algorithms to a variety of experiments previously mentioned. Figure 33 shows the concept diagram of the ML framework. This framework separates the feature engineering and stock price prediction which makes it easy to keep the two components independent. Also the Stock price prediction component is totally abstracted to make it easy for price-analyst or financial-trader to use the framework with different datasets.

As a key feature, the framework is built with independent compartments that deal with:

- data acquisition
- feature engineering
- experiment setup
- modelling
- and testing.

This also means that we can build upon it to expand its usage to different types of securities (like Forex and crypto-currencies) and also continue developing our feature engineering and models. It was paramount to build a flexible data pipeline that permits modifications with minimal impact to the overall framework. This setup will save significant time in terms of

future development and speed up trials too.
The framework itself is the foundation of a product aimed to augment human intuition in this highly complex environment. The code base is hosted in git-hub which enables an agile development of the solution and testing new features prior implementation.
Finally, this framework can be adapted to other machine learning problems outside the scope of finance, like different type of time-series prediction.
Additionally, we studied the different factors involved in our models and how they impact on the predictions of our models. This is an important part of enhancing the interpretability of our models and further comprehending the underlying patterns that the models extract from the data. Considering we included a wide variety of features (more than 80), from macroeconomic to price derived, we explored their influence in the State Space model.

9.1 Factors

One of our aims is to investigate into the attributes that are important in the sense of large weights, in addition to study the performance of various machine learning methods. By assessing these factors, we could gain insight and make recommendations based on the nature of the investments.

9.1.1 Supplementary experiment

It could be seen from Figure 34 that large weights were mostly associated with a few factors, and they mostly are members of the family of "moving average" predictors. The weighted moving average shows positive correlation with the adjusted closing price. On the other hand, the exponentially weighted moving average of various α weights were also significant, but the relationship is in the opposite direction. Since adjusted closing price is our response, we have no doubt that closing price will be important.
The majority of the factors were observed to be extremely weakly correlated with the adjusted price. First, simple moving average has considerably less weight compared to the weighted and exponentially weighted counterparts. For other technical indicators we have included, such those members of Category 1 indicators, do not have significant weights. The attributes we acquired from World Bank data were also seen to have extremely small weights, similar to the mentioned indicators.

9.1.2 S&P500 vs ASX200 stocks experiment

For this experiment, we now see the larger weights of the attributes "low" and "move". Here, "low" and "move"' are engineered by taking the ratio of past prices to current price. In this case, unlike the supplementary experiment, the moving average family of indicators have much less weights compared to this experiment.
One significant similarity we see between Figure 34 and Figure 37 was that World Bank data attributes do not have as much weight as the attributes that incorporates past prices.

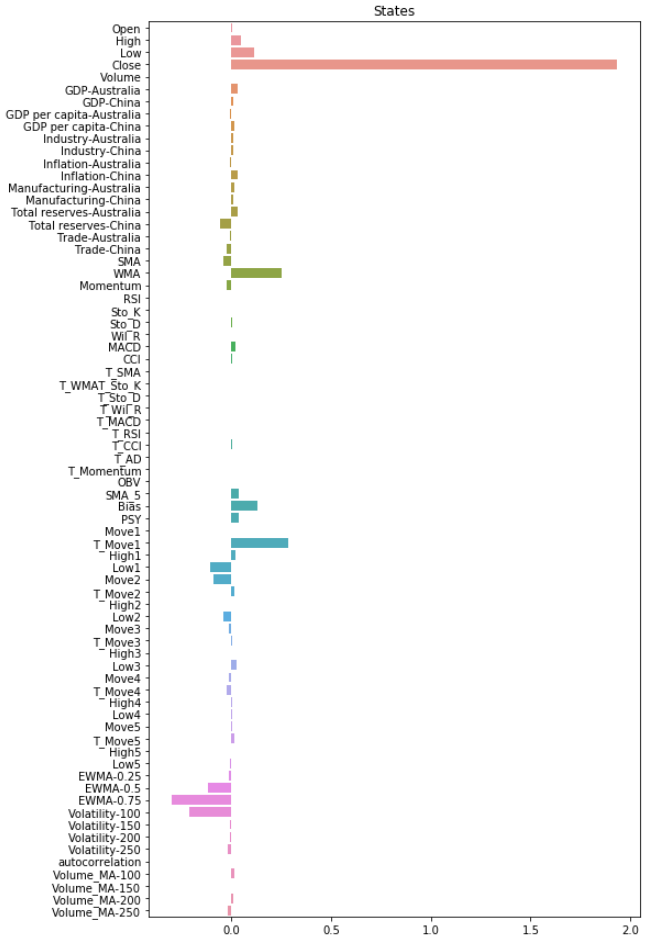


Figure 34. Weights of the predictors for supplementary experiment (train CPU, test XRO)

9.1.3 Indices

In contrast to the previous experiments, the importance of the World Bank attributes has increased substantially from having limited weights to large weights, relative to that of technical indicators in this experiment (Figure 40). On the other hand, those attributes engineered from past prices now have less weights, unlike the previous experiments.

Recall we saw the significant weights given of using attributes that incorporates past prices such as "Low", "Move", and 'the moving average family. These suggest significant importance relative to other predictors. For the supplementary experiment, which consists of training on CPU and testing on XRO, the moving average family, Low and Move, appear to have this property. The World Bank data, on the other hand, almost have no significance due to its almost invisible bars. The World Bank attributes appear to gain greater importance when used them to train on S&P500 indices. This is expected since the World Bank attributes are macroeconomic attributes, and the moving averages, and other attributes engineered from price data, are "micro-level" attributes.

In the scenario of indices experiments, we found that macroeconomics indicators are significant compared to tech-

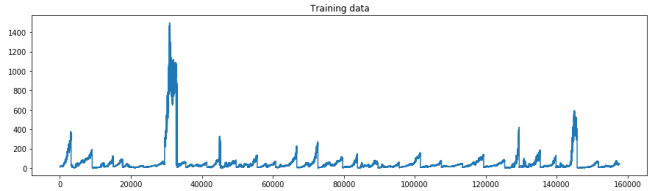


Figure 35. Time series of S&P500 stocks

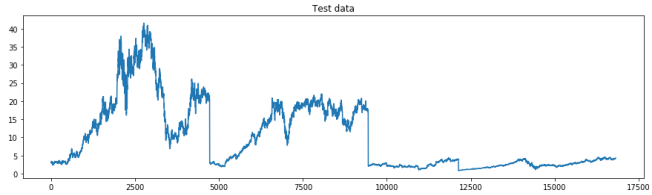


Figure 36. Time series of ASX200 stocks

nical analysis indicators. This is expected once we look at, for example, the definition of GDP which we took from slide 3 of [6], and it states that GDP is "the market value of all final goods & services produced within a country in a given period of time". This means that GDP can be interpreted as an aggregate (i.e. a sum). Recall that ASX200 listing consists of 200 largest companies in Australia. Based on our intuition, we could think of the 200 companies as "powerhouse" for the Australian economy, and the totality of "final goods & services produced within a country" may be assumed to be produced from the 200 companies. This suggests that the total production of the 200 companies is related to GDP, and GDP could be used as a guide to the value of final "production" of the 200 companies, reflected by the price of the stocks. For S&P 500 indices, these are related to the 500 largest US companies (see [14]). Similar to the viewpoint we took for Australian Stocks - assuming the largest 500 companies could make up substantial productivity of the United States, we should not be surprised that GDP should act as an indicator for productivity of the aggregate of large companies. The described correspondence, then, suggests that we should expect that GDP factors, and other macroeconomic indicators, to have large weights in this situation.

On a second note, we should also not be surprised by the lack of predictive ability of the World Bank macroeconomic indicators in ther prediction of two stocks in not only the supplementary experiment we did (CPU and XRO), but also the S&P500 vs ASX200 stocks experiment. Though the supplementary experiment belong to ASX200, they nevertheless are only components of them. The macro-economic indicators take the standpoint of "the whole country" and may not represent "individual parts", which could be different from the "whole". The moving average indicators, such as EWMA-0.75, are extracted directly from a given time series data. Since moving average approximates the underlying trend, and is calculated from the time series itself, we should not be surprised that they have significant weights. This especially is the case for Low and Move, which used past prices directly.

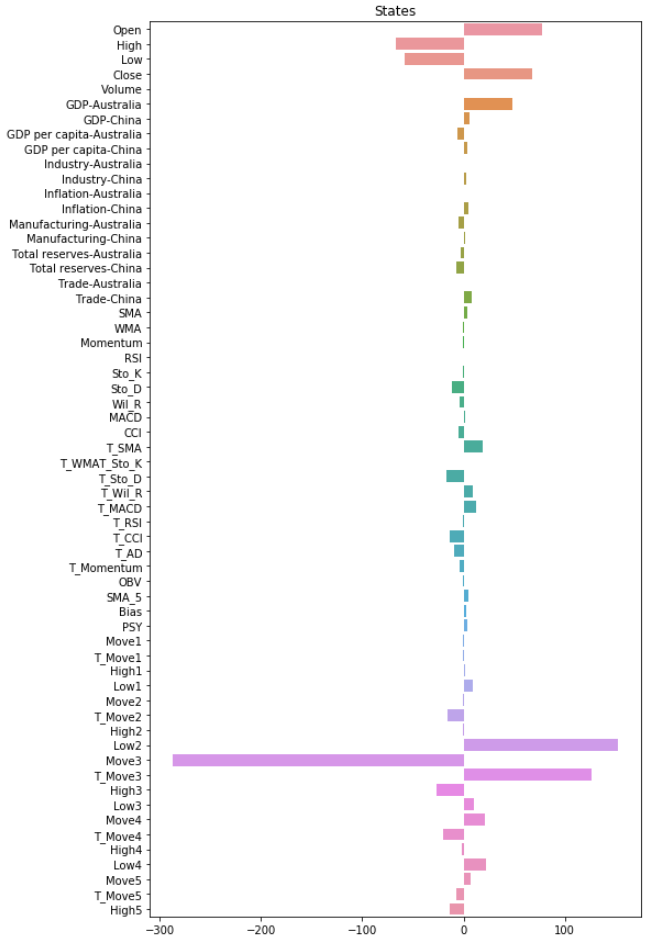


Figure 37. Weights of the predictors for S&P500 vs ASX200 stocks

As a result, we found that the importance of World Bank attributes will depend on the scope of the dataset at hand. If we used the World Bank attributes for data analyses, where such data could be taken from the collection of many large companies as we mentioned, then it is likely the World Bank data would be relevant, and could potentially have large weights. On the other hand, on an individual stock scale, less significance for for World bank attributes could potentially be the case.

9.2 Performance analysis

Recall that state-space models were compared to the naive method via predictions in different experiments described in **Results**. It was found that state-space model was inferior to naive model. To probe into this issue, the **Results** section displayed time series plots for different experiments. We have chose to include:

1. Indices
2. Supplementary experiment

We found that state-space is not adapted to abrupt changes in data trend as seen here, hence it could be an unsuitable

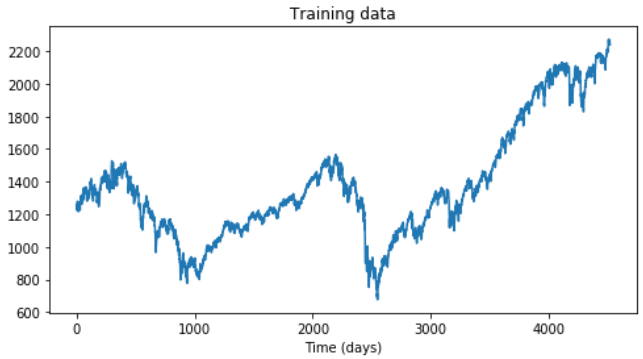


Figure 38. Time series of S&P500 index

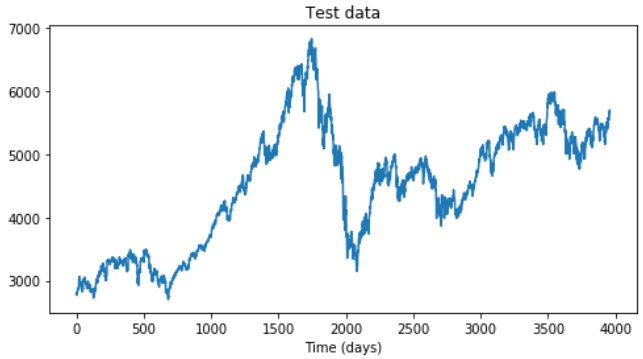


Figure 39. Time series of ASX200 index

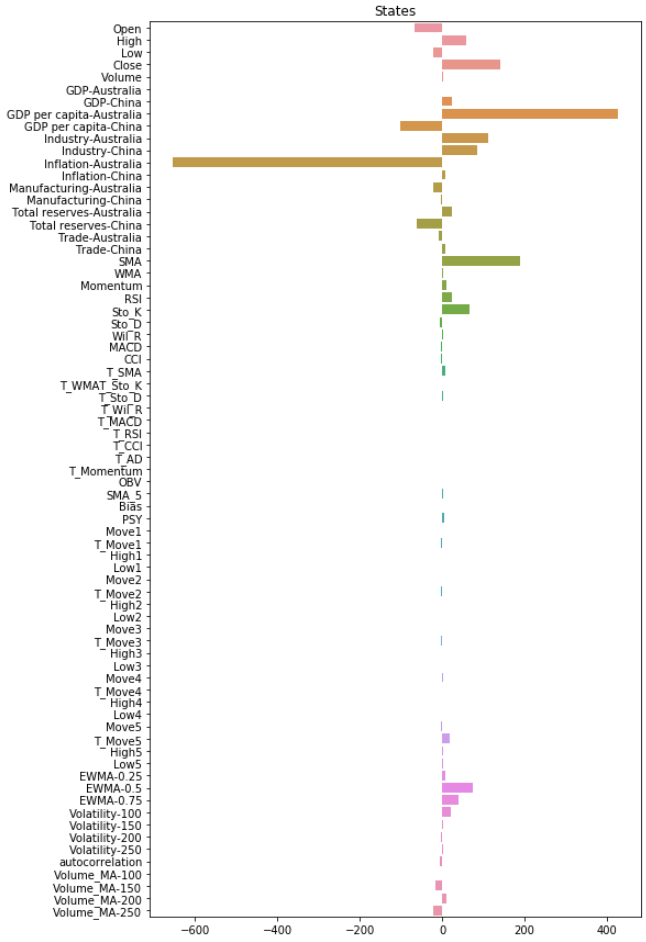


Figure 40. Weights of the predictors for indices experiment

method for transfer learning. We also found that state-space was also not adapted to sudden changes, even in the same direction as seen in Figures 38 and Figure 39. Both have increasing trend, but not of the same rate as seen from the scale - the training data has slower increase in trend. The test data, on the other hand, increases at substantially faster rate. As can be seen from Table 1, it is expected that state-space model will not be a suitable method. Therefore, we found that using the state-space model to predict new situation as described was not appropriate.

The advantage we found for state-space was that if both training data and test data have strong trend of comparable size, then state-space will outperform the naive forecast. This was shown in the supplementary experiment, whose result can be seen from Table 1. Plotting the CPU (the training data) stock and XRO (test data) stock reveals that both have trend with similar slope. This was not the case with the first demonstration, and the second. So, based on the experiments, we found that state-space model perform well if both the training data and testing data have strong and comparable trend.

9.3 Assumptions made

We have also made a number of assumptions, which involves the following:

1. We assumed that the predictors do not interact with one another. Prices are also assumed to be linearly related to the predictors in an additive sense. We assumed this to allow us to interpret weights later when we investigate which predictors could be important. As with linear regression, noise from the data are independent of each other.

2. Our formulation assumes only a simple hierarchy, where there is one equation used for modelling observations (prices, given by $\{y_t\}$) and another for modelling weights β_t. This model is not as complex as that proposed in [84], where multi-level Bayesian approach was assumed.

3. We have not made any attempt to adjust coefficients such as transition matrix in the state equation, or the variance of the perturbation in the state equation. We assumed the default initial state estimates, which is explained in the **Implementation** section.

which we mentioned that they are aimed at simplifying the implementation and to ease interpretation of states (weights). However, it is possible that the assumptions can pose excessive restriction and further limits to the performance of our model in regards to the complexity of the model. One of the

assumptions are the simplicity of the model we proposed, and the second is on the additive property.

In order to address the possible problem of using simplicity, one can propose a more complex alternative. This might require further domain knowledge in order to determine the appropriate levels of the hierarchy, suitable transition matrix in the state equations and magnitude of the perturbation in state variance. One possible advantage of this extension is to allow states to vary with other set of states, which, from domain knowledge could be necessary. The use of a more complex Bayesian hierarchical method for the state-space method was used in financial modelling in [84]. There, they took similar approach but added further hyperparameters to the β coefficients that we have.

Secondly, we do not assume that any pair of attributes interact. One reason is to simplify the execution of our algorithm. Another was that although we have some intuition about the predictors that could be relevant, we nevertheless do not have enough specialised knowledge to decide which parameters that may require tuning prior to the execution of the algorithm. If one was to include more complex terms such as the addition of interactions, we believe that this may require justifications not only on the complexity of the interaction, but also the features involved. This may again involve specialised knowledge.

9.4 Recommendations

Our descriptions regarding the issues of transfer learning, the importance of factors in different circumstances, and the nature of the model formulation lead us to make some remarks.

- The significance of predictors may be negligible for some datasets, but substantial for another. We saw an example earlier that World Bank data was significant when used to analyse indices, but was not when used to analyse individual stocks. We recommend that the nature of the predictors be assessed alongside that of the time series to be analysed.

- It may be necessary to incorporate specialised domain knowledge, especially the financial market behaviour, to improve the performance of the model. Our basic intuition was used to only select the attributes we believe could have predictive power. We believe that further justification is required for introducing further model complexity, such as introducing interactions.

- Whether trend in the data are comparable or not can be a crucial factor to determine the usability of the method. As we saw, the significant differences in training and testing data in terms of trend produced large errors. On the other hand, if the trend for training and testing data are comparable, methods such as state-space is superior to the benchmark. Therefore, it is advisable that investors take the trend of market movement into consideration.

10. LIMITATIONS AND FUTURE WORKS

10.1 Limitations

Stock price prediction is a very difficult task, and there are many players in the same market doing this. Essentially, every investor is doing it, even if unknowingly. A long term investor buys a stock hoping it will go up in the long run yet the short term trader hopes it will go up within the next trade or minutes.

Our project has shown that there is important information into the historical price time series and that information can be very helpful for actual stock price prediction to help in real world trading. We knowingly have introduced a few limitations to our study which we intend to improve in the future. Those limitations were mostly due to time yet we intend to keep working on this project in the near future.

The main limitations from our study were:

- **Backtesting** was performed as a classification matrix on cross-validated data. A better methodology would be to use time-series cross-validation, similar to Leave One Out Cross Validation (LOOCV) respecting time in a sequential time-series. Accuracy may be over-stated in our project as the test data may be using correlated training data.

- **Hyperparameter tuning** was left for further work due to time constraints. All the experiments were done with the same parameters which were chosen with our user sensibility.

- Experiments were performed on **stocks and equity indexes** only. For further work we intend to evaluate the methodologies studied being applied to currency, commodities and cryptocurrencies also. Our trading platform allows for that extension and we will try those types of securities in the future.

- **Technical analysis as binomial variable** may not be the best choice of feature engineering. Since technical analysis patterns don't occur too often, a Gaussian distribution around the point where it occurs may better represent the fact that it occurred for further time points.

- Our data is from the last 20 years, where the **upwards price trend** in the equities markets was very pronounced. There were some short term localised market crashes during the period yet for most of the time the trend was mostly positive upwards. Using data from more volatile times could induce better model training as the testing could possible be biased due to similar upwards trend during testing periods.

The limitations from the project do not invalidate it but they create opportunity for further work, which we'll discuss in the next session.

10.2 Future Work

The first point of action will be to create improved routines for **back-testing**. Back-testing is the term used in finance for cross-validation. Our system did cross validation by training and testing on different time series, e.g. train on S&P500 and test on ASX200. The risk of that is that the two indices may be correlated and thus the testing is actually occurring on data that has already been "seen" by the testing.

We intend to do cross-validation by using time-series cross-validation, in which a series is trained up to a certain point and tested in subsequent points. An example would be to learn using data from the past twenty years up to June 2018 and then test on July 2018 data. Next we would train the model using data until July 2018 and train on August 2018. This method would be closer to real life, where we train on past data and test on future data.

Time series cross validation is very computational intensive (it approximates a LOOCV) as for each prediction the model has to be retrained using all the past data up to that point so we come to our second point of future works of **hosting the solution on an external** server such as AWS. We intend to use Amazon S3 for data storage and processing warehousing as well as utilise Amazon Lambdas to host the trading system. This will allow us to be more effective at training and testing our models. The actual trading system can then be run in real-time from Amazon AWS.

Once we are externally hosted, apart from improved cross-validation, we intend to do **hyper-parameter training** to effectively improve the models and find the optimal models to be used. The trading system we developed is very flexible and allows for easy expansion to new methodologies. We don't foresee a single methodology to be superior to all the other ones, but we want to increase our modeling toolkit and seek, from the expanded kit, which are the best and with which hyper-parameters they are improved.

We would also like to **expand feature engineering** by seeking further features which can be included into the data, especially fundamental company data. We weren't able to do much with fundamental data in this project due to the difficulty of obtaining the data within the time constraints, yet with time we can work on that.

Improving the **technical analysis pattern recognition** would be useful. Certain patterns, when they occur, may influence a time-series not only when the pattern occurs but around it. So we'd like to work in finding distributions around the point where the pattern occurs. Rather than it being a binomial variable on the date a pattern occurs, it'd become a distribution, e.g. normal, around that point.

We worked mostly on equities by looking at stocks and indexes, but in the future we'd like to look at other securities, such as **currencies and commodities**. Currencies have an advantage from a trading point of view in that their trading is continuous. As an example, the trading pair Australian Dollar vs United States Dollar (AUDxUSD) is very active during Australian and American business hours, contrary to Australian stocks which are only actively trading during Australian Stock Exchange (ASX) hours.

Another interesting aspect of future work would be to **cluster securities according to volatility**. Volatility and correlation to the market (in finance usually referred as Beta) is an essential risk characteristic of securities and their clustering could yield slightly different models according to their volatility levels.

Coming to the real world of portfolio management and stock trading, we need to be wary of **noise** and **trend**. It's very important that our models recognise noise and filter it out, yet the literature has much reference to models which were overfit due to noise. They are not very robust to future data and may work well until a market reversion in which all gains are lost. This also occurs when the market or a security is trending so we need to be very careful with market reversions and be able to properly measure **portfolio risk**. One of the easiest ways to improve investment gains is to assume larger risks, yet when there is a loss it will also be higher. Very often a portfolio manager is not aware of the risk of his portfolio and we'll need to devise appropriate value at risk (VAR) strategies by understanding securities correlations.

Time series cross validation is very computational intensive (it approximates a LOOCV) as for each prediction the model has to be retrained using all the past data up to that point so we come to our second point of future works of **hosting the solution on an external** server such as AWS. We intend to use Amazon S3 for data storage and processing warehousing as well as utilise Amazon Lambdas to host the trading system. This will allow us to be more effective at training and testing our models. The actual trading system can then be run in real-time from Amazon AWS.

Once we are externally hosted, apart from improved cross-validation, we intend to do **hyper-parameter training** to effectively improve the models and find the optimal models to be used. The trading system we developed is very flexible and allows for easy expansion to new methodologies. We don't foresee a single methodology to be superior to all the other ones, but we want to increase our modeling toolkit and seek, from the expanded kit, which are the best and with which hyper-parameters they are improved.

We would also like to **expand feature engineering** by seeking further features which can be included into the data, especially fundamental company data. We weren't able to do much with fundamental data in this project due to the difficulty of obtaining the data within the time constraints, yet with time we can work on that.

Improving the **technical analysis pattern recognition** would be useful. Certain patterns, when they occur, may influence a time-series not only when the pattern occurs but around it. So we'd like to work in finding distributions around the point where the pattern occurs. Rather than it being a binomial variable on the date a pattern occurs, it'd become a distribution, e.g. normal, around that point.

We worked mostly on equities by looking at stocks and in-

dexes, but in the future we'd like to look at other securities, such as **currencies and commodities**. Currencies have an advantage from a trading point of view in that their trading is continuous. As an example, the trading pair Australian Dollar vs United States Dollar (AUDxUSD) is very active during Australian and American business hours, contrary to Australian stocks which are only actively trading during Australian Stock Exchange (ASX) hours.
Another interesting aspect of future work would be to **cluster securities according to volatility**. Volatility and correlation to the market (in finance usually referred as Beta) is an essential risk characteristic of securities and their clustering could yield slightly different models according to their volatility levels.
Coming to the real world of portfolio management and stock trading, we need to be wary of **noise** and **trend**. It's very important that our models recognise noise and filter it out, yet the literature has much reference to models which were overfit due to noise. They are not very robust to future data and may work well until a market reversion in which all gains are lost. This also occurs when the market or a security is trending so we need to be very careful with market reversions and be able to properly measure **portfolio risk**. One of the easiest ways to improve investment gains is to assume larger risks, yet when there is a loss it will also be higher. Very often a portfolio manager is not aware of the risk of his portfolio and we'll need to devise appropriate value at risk (VAR) strategies by understanding securities correlations.
Another very important part of the trading system that will be built is to build **lock-gain and stop-loss** within the system. Lock-gain is a technique in which if the model predicts a 5% gain in 5 days, once the security has gone up by that amount (or a fraction of it, for conservativeness) the trading is closed and the gain is locked. A stop-loss strategy would be to close a trade if it's starting to go bad, e.g. in the wrong direction. For the same trade exemplified before, if the stock goes down by more than a certain threshold, e.g. 3%, the trade is reversed and we stop the loss. The model may then predict further gains, but the portfolio manager will be then more cautious and will verify if there isn't information previously unknown to him.
Our whole study so far has been about classifying absolute returns, i.e. in relation to the US$ or to the AU$. Another great opportunity would be to implement **relative trading strategies**. These strategies could involve, rather than trying to find the direction of a stock's move, the direction of a stock move relative to another. This could be done in a few scenarios:

- Long short between two stocks: in this strategy one could use machine learning to do relative trades among correlated stocks, e.g. Google vs Apple. These are trades that are less obvious to normal investors yet could be easily implemented in this kind of automated trading systems.

- Enhanced index funds: in this strategy one could create an index fund with all the index components and go over or underweight securities which are, relative to the index, classified as over or under-performing relatively. In this case, a small gain could go a long way. Taking into account that 75% of equity mutual funds lose to the index, an extra performance of 3% per year would make this fund among the top performing funds in the market.

- Pairwise currency trading: the same strategy as in between two stocks could be employed among currencies, searching for trading signals among very different currencies.

- Pairwise crypto-currency trading: the largest crypto exchange globally, Binance [18], only trades pairwise crypto trading. One can trade any crypto security for another except for fiat currency. Most of that trading is done on individual sentiment analysis, and we could also try to implement machine learning models for that.

- Given the way that our trading system was built, to implement the four strategies above we would basically need to work on the data-factory component of the system (the part that builds the features, by building the basic data with security vs security as opposed to vs currency) and in building further time-series features.

- Pairwise crypto-currency trading: the largest crypto exchange globally, Binance [18], only trades pairwise crypto trading. One can trade any crypto security for another except for fiat currency. Most of that trading is done on individual sentiment analysis, and we could also try to implement machine learning models for that.

Given the way that our trading system was built, to implement the four strategies above we would basically need to work on the data-factory component of the system (the part that builds the features, by building the basic data with security vs security as opposed to vs currency) and in building further time-series features.
We envision a long, challenging and rewarding path for our group to continue to work on these challenges.

11. CONCLUSION

Financial markets are a very dynamic system. Many different players are active in the market with different expectations on why a security's price may behave in the future, leading to a variety of price making expectations. Different investor expectations create a very dynamic pricing composition which finds equilibrium at market prices. Since everyone's expectation is differently formed, markets tend to move very dynamically which leads to constant price fluctuations.
The statistical analysis of results produced by various machine learning approaches used in the study shows sufficient evidence in favour of using machine learning methods for financial trading. Also it is important to use domain knowledge

of finance to enrich the data before applying machine learning algorithms. This is the reason why we used both technical indicators with fundamental data to create a rich feature set. This dataset can be further enhanced with sentimental analysis to cater for sentiments of investors while they trade.
In our opinion we can deploy the stock-price prediction model to real-time trading system to make better decisions while placing the orders over the stock exchanges. And we would like to investigate further if the price-prediction framework could be utilised for trading of different asset types.

References

[1] https://commodity.com/technical-analysis/accumulation-distribution/

[2] https://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:accumulation_distribution_line (cited in Dr. M. Khushi's code)

[3] https://www.alphavantage.co/

[4] https://commodity.com/technical-analysis/commodity-channel-index or see https://www.mrao.cam.ac.uk/~mph/Technical_Analysis.pdf, page 164

[5] Based on Dr. M. Khushi's Python code

[6] https://www3.nd.edu/~wbrooks/SyllabusS14.pdf

[7] Grimmett, G., & Stirzaker, D. (2001). Probability and random processes. Oxford University Press.

[8] https://otexts.org/fpp2/ses.html

[9] https://www.investopedia.com/terms/m/macd.asp

[10] https://commodity.com/technical-analysis/macd/

[11] https://commodity.com/technical-analysis/momentum/ or see https://www.mrao.cam.ac.uk/~mph/Technical_Analysis.pdf, page 8

[12] https://commodity.com/technical-analysis/on-balance-volume/ or see https://www.mrao.cam.ac.uk/~mph/Technical_Analysis.pdf, page 135

[13] https://commodity.com/technical-analysis/relative-strength-index/ or see https://www.mrao.cam.ac.uk/~mph/Technical_Analysis.pdf, page 126

[14] https://www.investopedia.com/terms/s/sp500.asp

[15] https://commodity.com/technical-analysis/stochastics/ or see https://www.mrao.cam.ac.uk/~mph/Technical_Analysis.pdf, page 131

[16] https://www.austrade.gov.au/ArticleDocuments/3823/Australia-Benchmark-Report.pdf.aspx

[17] Bao, W., Yue, J., & Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. PloS One, 12(7), e0180944. doi:10.1371/journal.pone.0180944

[18] Binance - exchange the world https://www.binance.com/

[19] Blackburn, Mark (1979) *The no-risk portfolio of Barr Rosenberg* The New York Times, June 24, 1979, pp F1.

[20] Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. Journal of Econometrics, 31(3), 307-327. doi:10.1016/0304-4076(86)90063-1

[21] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, *'Backpropagation applied to handwritten zip code recognition'*, *Neural Computation*, vol. 1, no. 4, pp. 541-551, 1989.

[22] Breiman, L. (June 1997). "Arcing The Edge" (PDF). Technical Report 486. Statistics Department, University of California, Berkeley. https://statistics.berkeley.edu/sites/default/files/tech-reports/486.pdf

[23] Brockwell,P.J.,&Davis,R.A.(2002) .Introductiontotimeseriesandforecasting(2nded.). NewYork:Springer.

[24] Brownlee, J., 2017, *'A Gentle Introduction to Mini-Batch Gradient Descent and How to Configure Batch Size*, viewed 10^{th} May 2018,

[25] Budhiraja, A. (2016). *'Learning Less to Learn Better – Dropout in Deep Learning'*

[26] Bulkowski, T. (2008) *"An Encyclopedia of Candlestick Charts"*

[27] Chan, N. H. (2010). Time series: Applications to finance with R and S-plus (2nd ed.). Hoboken, N.J: Wiley.

[28] Cheng Li. "A Gentle Introduction to Gradient Boosting" http://www.chengli.io/tutorials/gradient_boosting.pdf

[29] Choi, H. K. (2018). Stock price correlation coefficient prediction with ARIMA-LSTM hybrid model.

[30] Dutta, Avijan & Bandopadhyay, Gautam & Sengupta, Suchismita (2012) *Prediction of Stock Performance in the Indian Stock Market Using Logistic Regression* International Journal of Business and Information, Vol 7, No 1, 2012

[31] Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. Econometrica: Journal of the Econometric Society, 987-1007.

[32] Escobar, M. D., & West, M. (1995). Bayesian density estimation and inference using mixtures. Journal of the american statistical association, 90(430), 577-588.

[33] Fama, Eugene & French, Kenneth (1992) *Common risk factors in the returns on stocks and bonds* The Journal of Financial Economics, Vol. 33 (1993), pp. 3-56.

[34] Fama, Eugene & French, Kenneth (2004) *The capital asset pricing model - theory and evidence* Journal of Economics Perspectives, Vol. 18, No. 3. (2004), pp. 25-46.

[35] Fama, Eugene & French, Kenneth (2014) *A five-factor asset pricing model* Journal of Financial Economics, Vol. 116. (2015), pp. 1-22.

[36] Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. European Journal of Operational Research, 270(2), 654-669. doi:10.1016/j.ejor.2017.11.054

[37] http://www.people.fas.harvard.edu/~djmorin/waves/Fourier.pdf

[38] Friedman, J. H. (February 1999). "Greedy Function Approximation: A Gradient Boosting Machine". https://statweb.stanford.edu/~jhf/ftp/trebst.pdf

[39] Friedman, J. H. (March 1999). "Stochastic Gradient Boosting" https://statweb.stanford.edu/~jhf/ftp/stobst.pdf

[40] Ge, R., Huang, F., Jin, C., & Yuan, Y. (2015) '*Escaping from Saddle Points-Online Stochastic Gradient for Tensor Decomposition*.' In COLT (pp. 797-842).

[41] Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (1995). Bayesian data analysis. Chapman and Hall/CRC.

[42] Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. Neural Computation, 12(10), 2451-2471. doi:10.1162/089976600300015015

[43] Git: CNN http://cs231n.github.io/neural-networks-1/

[44] Glorot, X., Bengio, Y (2010) *'Understanding the difficulty of training deep feedforward neural networks.'* DIRO, Universite de Montreal, Montreal, Quebec, Canada.

[45] https://commodity.com/technical-analysis/hammer/

[46] Hastie, T.; Tibshirani, R.; Friedman, J. H. (2009). "10. Boosting and Additive Trees". The Elements of Statistical Learning (2nd ed.). New York: Springer. pp. 337384. ISBN 0-387-84857-6.

[47] Heaton, J. B., Polson, N. G., & Witte, J. H. (2016). Deep learning in finance.

[48] Hinton, G., Zemel, R. (1993) *'Autoencoders, Minimum Description Length and Helmholtz Free Energy'*. Advances in Neural Information Processing Systems 6 (NIPS).

[49] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735-1780. doi:10.1162/neco.1997.9.8.1735

[50] Hutchinson, C. E. (1984). The Kalman filter applied to aerospace and electronic systems. IEEE Transactions on Aerospace and Electronic Systems, (4), 500-504.

[51] Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. Journal of basic Engineering, 82(1), 35-45.

[52] Le Cun, Y. (1985) *'A learning scheme for asymmetric threshold networks.'* In Proceedings of Cognistiva 85, pages 599-604.

[53] LSTM Formulas http://colah.github.io/posts/2015-08-Understanding-LSTMs

[54] Hassine, Marlene (2015) *Active funds vs benchmark performance comparison* Lyxor ETF Research

[55] Markowitz, Harry (1952) *Portfolio selection* The Journal of Finance, Vol. 7, No. 1. (Mar., 1952), pp. 77-91.

[56] Mason, L.; Baxter, J.; Bartlett, P. L.; Frean, Marcus (1999). "Boosting Algorithms as Gradient Descent". In S.A. Solla and T.K. Leen and K. Müller. Advances in Neural Information Processing Systems 12. MIT Press. pp. 512–518. http://papers.nips.cc/paper/1766-boosting-algorithms-as-gradient-descent.pdf

[57] Mason, L.; Baxter, J.; Bartlett, P. L.; Frean, Marcus (May 1999). "Boosting Algorithms as Gradient Descent in Function Space" https://www.maths.dur.ac.uk/~dma6kp/pdf/face_recognition/Boosting/Mason99AnyboostLong.pdf

[58] Khushi, Dr Matloob https://sydney.edu.au/engineering/people/matloob.khushi.php

[59] Minsky, M. & Papert, S. (1969) *'Perceptrons, an introduction to Computational Geometry'*. M.I.T. Press, Cambridge, Mass

[60] M. Kolanovic and R. T. Krishnamachari. Big data and ai strategies, machine learning and alternative data approach to investing. J. P. Morgan, 2017.

[61] https://commodity.com/technical-analysis/morning-star/

[62] Murphy, K. P. (2012). Machine learning: A probabilistic perspective. Cambridge, Mass: MIT Press.

[63] Nan Zhou, Wen Cheng, Yichen Qin & Zongcheng Yin Evolution of high-frequency systematic trading: a performance-driven gradient boosting model https://doi.org/10.1080/14697688.2015.1032541

[64] Sezer, O., Ozbayoglu, M. (2018) '*Algorithmic Financial Trading with Deep Convolutional Neural Networks: Time Series to Image Conversion Approach.'*

[65] Parker, D (1985) *'Learning-logic, a Technical Report'* Sloan School of Management, MIT, Cambridge.

[66] https://pydlm.github.io/

[67] Qin Qin, Qing-Guo Wang, Jin Li, Shuzhi Sam Ge: Linear and Nonlinear Trading Models with Gradient Boosted Random Forests and Application to Singapore Stock Market https://file.scirp.org/pdf/JILSA_2013022213381465.pdf

[68] Rosenblatt, F. (1958) *'The perceptron: A probabilistic model for information and organization in the brain.'* Psychological Review Vol. 65, No. 6

[69] Rumelhart, D., Hinton, G,, Williams, J. (1986) *'Learning representations by back-propagating errors'*. Nature 323 , 533-536.

[70] Sharpe, William F. (1964) *Capital asset prices: a theory of market equilibrium under conditions of risk* The Journal of Finance, Vol. 19, No. 3. (Sep., 1964), pp. 425-442.

[71] https://commodity.com/technical-analysis/shooting-star/

[72] Stoffer, D. S., & Shumway, R. H. (2000). Time Series Analysis and Its Applications. With R Examples.

[73] Siripurapu, A. (2018) *'Convolutional Networks for Stock Trading'*. Stanford University Department of Computer Science

[74] Skabar, A., Cloete, I. (2002) '*Neural Networks, Financial Trading and the Efficient Markets Hypothesis.*' School of IT – International University in Germany.

[75] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. (2014) '*Dropout: A Simple Way to Prevent Neural Networks from Overfitting.*' Journal of Machine Learning Research 15

[76] Troiano, L., Villa, E. M., & Loia, V. (2018). Replicating a trading strategy by means of LSTM for financial industry applications. IEEE Transactions on Industrial Informatics, 14(7), 3226-3234. doi:10.1109/TII.2018.2811377

[77] Trottier, L., Giguere, P., Chaib-draa, B. (2018) '*Parametric Exponential Linear Unit for Deep Convolutional Neural Networks'*.

[78] Vanstone, J. (2005) *'Trading in the Australian Stock market using Artificial Neural Networks'* School of Information Technology – Bond University.

[79] http://web.mit.edu/kirtley/kirtley/binlustuff/literature/control/Kalman%20filter.pdf

[80] Werbos, P. (1974) *'Beyond regression: new tools for prediction and analysis in the behavioural sciences'* Thesis (PhD) -Harvard University

[81] https://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:williams_r

[82] World Bank: a vital source of financial and technical assistance to developing countries around the world http://www.worldbank.org/

[83] Yao, J., Tan, C. (2003) '*Guidelines for Financial Forecasting with Neural Networks'*.

[84] Ying, J., Kuo, L., & Seow, G. S. (2005). Forecasting stock prices using a hierarchical Bayesian approach. Journal of forecasting, 24(1), 39-59.