Stock Price Predictions from Financial Statements using Machine Learning and Deep Learning algorithms augmented with Knowledge Graph Embeddings

CONOR MELODY

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTERS OF SCIENCE IN COMPUTER SCIENCE (ARTIFICIAL INTELLIGENCE)

Author:
Conor Melody

Supervisor:
Dr. Frank Glavin

August 2020

Declaration

I HEREBY CERTIFY THAT THIS MATERIAL, WHICH I NOW SUBMIT FOR ASSESSMENT ON THE PROGRAMME OF STUDY LEADING TO THE AWARD OF MASTERS IN SCIENCE IS ENTIRELY MY OWN WORK, THAT I HAVE EXERCISED REASONABLE CARE TO ENSURE THE WORK IS ORIGINAL, AND DOES NOT TO THE BEST OF MY KNOWLEDGE BREACH ANY LAW OF COPYRIGHT, AND HAS NOT BEEN TAKEN FROM THE WORK OF OTHERS SAVE AND TO THE EXTENT THAT SUCH WORK HAS BEEN CITED AND ACKNOWLEDGED WITHIN THE TEXT OF MY WORK.

SIGNED: Conor Melody

ID NUMBER: 15403058

DATE: August 2020

Acknowledgements

First and foremost, I would like to sincerely thank my supervisor Dr Frank Glavin for his continued support, advice and guidance throughout this project. I would also like to thank my friends and family who have supported and encouraged me throughout the year while completing this Masters.

Abstract

The challenge of predicting individual company stock prices has been the goal of investors for decades. The stock price of a company is affected by many different factors; news, sentiment towards the company, the current political climate and the state of the economy can all have an effect on prices. As such, being able to predict prices in such a chaotic environment represents a significant challenge. However, given the rewards possible for successful prediction, many have tried to develop models, using different methodologies, to do exactly this. Much of the literature relating to the prediction of stock prices focuses on short term predictions of trends (increase or decrease in price) and price, such as the change in price within days, between days and for other periods of time less than one month.

This project investigates the use of Machine Learning, Deep Learning and Knowledge Graphs and Embeddings in uncovering the relationship between the financial performance of companies listed on US Stock Exchanges and their share price. Specifically, this work is concerned with attempting to generate price predictions from financial statements, as well as predict the trend upward or downward and the magnitude of the change of individual company stock prices between each company's annual 10K report. This report provides a comprehensive summary of a company's financial performance throughout its trading year. The use of Knowledge Graph Embeddings representing non-numerical information related to each company to improve the accuracy of predictions is also investigated.

This work does not aim to provide a profitable trading framework for readers, rather it seeks to assess the effect, if any, the state of and changes to the financial filings of individual companies have on each company's share price. This in turn may provide financial decision support to investors. This work has also resulted in the production of new datasets which may be further explored by other researchers. This data is available on GitHub.

https://github.com/conor-mel1/StockPricePredictionFromFinancialFilings

Contents

1 Introduction

1.1 Research Topic and Motivation

The work carried out in this project is concerned with utilising the information found in financial statements published by companies on an annual basis to make predictions on a company's stock price. Specifically, this work is concerned with the trend upward or downward in the price of individual company stock prices between each company's annual 10-K report, which provides a comprehensive summary of a companies financial performance, as well as the magnitude of the change in price. This analysis does not aim to provide a profitable trading framework for readers, but rather asses the effect, if any, the state of and changes to a company's financial statements have on its stock price.

1.2 Research Aims

This work aims to investigate the following questions:

1. How accurately can a company's financial statements predict that company's share price? And which statements in particular are good predictors of the price?

2. Can the change in the state of a company's financial statements be used to predict whether the price of that company's shares has increased or decreased? And how accurately can this change in price be predicted?

3. By taking into account additional non-numerical information related to individual companies unrelated to their financial performance, such as the industry and sector within which the company operates, in the form of a Knowledge Graph, can this added information be used in the form of Knowledge Graph Embeddings to improve predictions of price over the performance achieved using only financial statements?

1.3 Structure of Thesis

This thesis is divided into 7 chapters. Chapter 1 outlines the research topic of the thesis as well as the research questions which will be investigated. Chapter 2 provides some background information on the Machine Learning and Deep Learning algorithms used in the analysis, as well as introducing the concept of a Knowledge Graph and Knowledge Graph Embeddings. This chapter also reviews relevant literature related to the application of Machine Learning, Deep Learning and Knowledge Graphs and Embeddings to stock market forecasting. Chapter 3 gives an overview of the data used in the analysis as well as a description of the data collection process. Chapters 4-6 make up the main investigative sections of the thesis and outline the methodology used and the results of analysis of each of the research questions outlined in Section 1.2. Each of these chapters contains an introduction to the research question(s) being addressed, an explanation of the methods used and discussion and analysis of the results obtained. Finally, Chapter 7 provides a final

discussion of the results obtained through investigation of the research question stated at the beginning of the thesis, and also discusses potential future work which may arise from this project's research.

2 Background and Literature Review

2.1 Relevant Considerations

The following sub-section contains some important concepts and ideas which are worth considering when dealing with stock market forecasting.

2.1.1 Random Walk Hypothesis

The Random Walk Hypothesis dates back to the 1800's and was further developed by Paul Cootner[1] at MIT. The hypothesis was popularised by Burton G. Malkiel in his book 'A Random Walk Down Wall Street[2]. The hypothesis states that the movement of stock prices is essentially random and cannot be predicted. Malkiel conducted an experiment whereby he created a chart which displayed the value of a stock with an initial value of $50. Malkiel then proceeded to flip a coin, moving the stock price up in value when the coin returned heads, and down in value when the coin returned tails. When Malkiel showed this chart to a chartist, an individual who employs the use of technical analysis to predict stock prices based on the assumption that past stock market patterns tend to repeat, the chartist informed Malkiel that he should immediately buy the stock. The argument being that stock prices behave in a similarly random manner. The random walk hypothesis is closely related to another hypothesis concerning financial markets, known as the Efficient Market Hypothesis.

2.1.2 Efficient Market Hypothesis

Informally, the Efficient Market Hypothesis (EMH) states that a company's stock price correctly reflects all known, available information about that company. In essence, the market always prices stocks correctly and any changes in price are as a result of new information being made available. This means that no company's stock is ever undervalued or overvalued but always trades at the fairest value. The origins of the EMH are often traced back to Fama[3]. The EMH comes in three forms: Weak Form, Semi-Strong Form and Strong Form.

- **Weak Form EMH**. The weak form of the EMH says that the current stock price of a company reflects all the data of past prices and no analysis of this past data (known as Technical Analysis) can be used to help investors in their investing decisions. It should be noted that advocates for the Weak Form EMH theory believe that if Fundamental Analysis is used, where investors research companies' financial statements and other non-price related aspects of a company, undervalued and overvalued stocks can be determined, and investors can boost their likelihood of achieving returns which are above the market average.[4]
- **Semi-Strong Form EMH**. The semi-strong form of the EMH says that because all relevant information relating to publicly listed companies is by definition public, and because this information is accounted for in the calculation of a company's stock price, investors cannot make use of either Technical Analysis or Fundamental Analysis to achieve higher returns in the stock market. Advocates for this version of the theory believe only information which is not readily available to the public can help investors achieve returns above those of the broader market.[4]

- **Strong Form EMH**. The strong form of the EMH says that all information, whether publicly available or not, is completely accounted for in the current stock price of a company. There is no type of information which gives any investor an advantage over the rest of the market. Advocates for the strong form theory believe that investors cannot make returns on investments that exceed normal market returns, regardless of information retrieved or research conducted.[4]

Despite its supporters, the EMH remains controversial, with several anomalies existing which are unexplained by the theory. The Neglected Firm Effect[5] is one such anomaly, whereby companies not covered extensively by analysts are sometimes priced incorrectly and may offer greater potential returns. Furthermore, the January Effect[6], first noticed around 1942 by investment banker Sidney B. Wachtel, indicates that historical evidence shows that stock prices tend to experience an upsurge in January. These anomalies indicate that the EMH, at least in its strong form, may well be inaccurate.

2.1.3 'The Superinvestors of Graham-and-Doddsville'

Arguably the most forceful rebuttal of the Efficient Market Hypothesis is an article written by Warren E Buffet[7], one of the greatest investors of all time. In the article, Buffett highlights the above average performance consistently achieved by disciples of the value investing philosophy outlined by Benjamin Graham and David Dodd in their book *Security Analysis*[8].

He rejects the idea that there are no *"undervalued stocks"* because of analysts who ensure *"unfailingly appropriate prices"*, as well as the idea that investors who consistently beat the returns of the wider market are simply lucky. He refers to the Random Walk Hypothesis by describing a scenario where 225 million Americans flip a coin every morning and bet $1 that they will correctly predict the outcome of the flip. Those who predict correctly win a dollar from those who predict incorrectly. Each day the stakes build until after 20 days there are approximately 215 people who have correctly predicted 20 coin flips in a row. This would be the case whether it had been people flipping coins for 20 days in a row, or Orangutans. 215 people (or Orangutans) would end up having correctly predicted 20 coin flips in a row. His argument, however, is that if one had taken 225 million people distributed roughly similar to the U.S. population, and of the 215 winners remaining after 20 days, 40 of these winners came from a particular town in Nebraska, one would be *"pretty sure (they) were on to something"*.

This disproportionate number of coin-flippers, according to Buffett, come from the same *"intellectual origin"* as opposed to a geographical origin; this intellectual village is called Graham-and-Doddsville, and contains a group of winners which simply cannot be explained by chance. The common methodology followed by investors of this philosophy is to search for differences between the *value* of a business and the *price* the business trades at in the stock market. Buffett bemoans the fact that *"no interest has been evidenced in studying the methods of this unusual concentration of value-oriented winners"*. He goes on to outline the numerous individuals who have consistently outperformed the wider stock market year after year, and provides tables of results to attest to this.

Buffett specifically refers to Walter Schloss, a man who never went to college and took a night course from Benjamin Graham at the New York Institute of Finance. Schloss' investment firm achieved a 28 year compounded gain of 23,104%, over the period from 1956 - 1984. This is compared with the Standard & Poor's overall market index's gain of 887% over the same period. In an assessment of Schloss by an associate of Buffett's, his investment methodology involved *"look(ing) up the numbers in manuals and send(ing) for the annual reports, and that's about it"*.

2.1.4 Conclusion

If the Random Walk Hypothesis is to be believed, there is no hope of being able to produce even approximate predictions of price from company financial statements as these changes in price will be completely random. It could be argued that the Efficient Market Hypothesis essentially states that the information contained within a company's financial statements is in some way accounted for within that company's stock price. This hypothesis will be tested somewhat through the investigations carried out in this work. Warren Buffett's compelling evidence offers some hope and also demonstrates the link between the price and value of a company, and the information found in its financial statements. This evidence makes the idea of trying to predict stock prices from company financial statements worth investigating.

2.2 Machine Learning Algorithms

The Machine Learning algorithms used in investigating each of the research questions posed in Chapter 1 are described below.

2.2.1 k-Nearest Neighbour Algorithm

The k-Nearest Neighbour algorithm is a supervised learning algorithm which can be used for both classification and regression tasks. Each training instance $\hat{\mathbf{x}}$ is treated as a point in n-dimensional space, with the value of n determined by the number of attributes of each instance. The idea behind the algorithm is that if two points are close to each other in space, they should be close to each other in their target values.

Data instances are simply stored by the model. When a new query instance $\hat{\mathbf{q}}$ is presented, the distance between the new query instance and all other data points is calculated. The k-nearest neighbours to the new data instance are chosen, and the corresponding target value for the query case is calculated. In the case of a binary classification task, a vote is taken of the k-nearest neighbours, with the majority class label of the k-neighbours assigned to the query case. In the case of a regression task, the average value of the k neighbours is computed and assigned to the query case.

The value of k, the distance metrics used to calculate the distance between instances and weights associated with these distances are hyper-parameters that must be tuned. Common distance metrics used are Euclidean distance ($\sqrt{\sum_{i}^{n}(x_i - q_i)^2}$) and Manhattan distance ($\sum_{i}^{n}|x_i - q_i|$), sometimes called the Taxicab metric.

2.2.2 Decision Tree

A Decision Tree learns a series or 'tree' of rules which it applies to data in order to map each data instance to its target value. This may be a binary classification (0 or 1) or a number representing a quantity; as such they can be used for both classification and regression tasks. Many possible decision trees may be consistent with the data, however, the goal is to select the tree which is the most compact representation of the data. This principle is known as Ockham's Razor[9].

A Decision Tree creates rules according to attributes of the data instances. It splits the data according to the values of certain attributes; these splits result in the highest information gain in the case of classification tasks, and result in the smallest error in the case of regression tasks. This means that groups which are split apart will be as different from each other as possible. A common method used to calculate information gain involves calculating the reduction in entropy achieved by a certain partition. In the case of regression tasks, an error metric such as Mean Squared Error is used and a split is chosen which minimizes error.

Each branch of a decision tree results from splitting the data based on a particular parameter. These branches eventually end in leaves which either give classifications or numerical values. When a new query instance $\hat{\mathbf{q}}$ is presented, the prediction of its target value will be an estimate based on the data the decision tree was trained on.

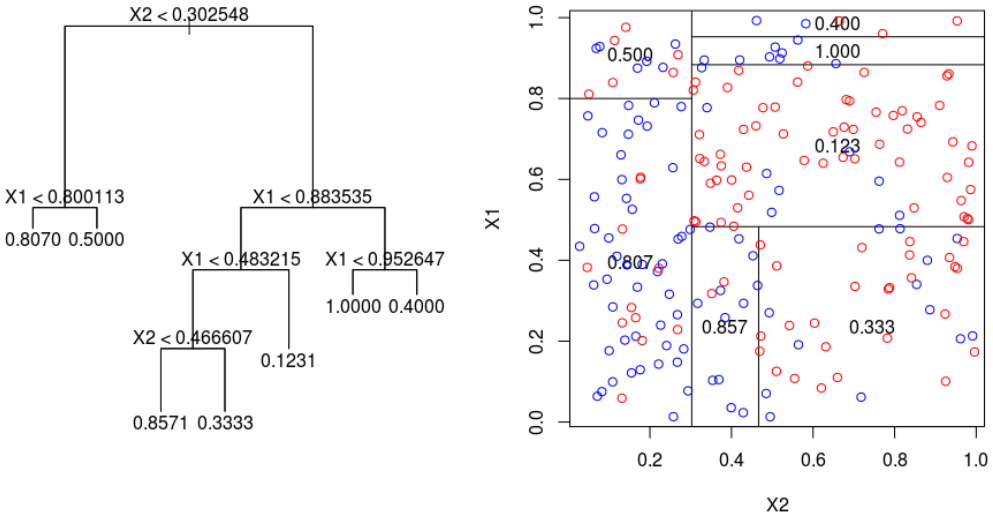

Figure 1: Decision Tree regressor example [10]

2.2.3 Random Forest

The basic building block of a Random Forest is a Decision Tree. A Random Forest classifier or regressor builds multiple Decision Trees and merges their predictions to obtain more accurate and stable predictions. Random Forests are very effective for high dimensional data as they are essentially a large ensemble model.

Each decision tree is trained on different random subsets of the data in a process called Bagging. Parameters associated with Random Forests include the number of estimators (i.e. the number of trees used) and the maximum depth of these trees.

2.2.4 Support Vector Machine

In the context of a binary classification problem, a Support Vector Machine (SVM) seeks to find the maximum margin hyper-plane separating both classes. When dealing with data with just two dimensions, a hyper-plane is represented by a line. Thus the maximum separating hyper-plane in 2D is simply the boundary line that separates the classes with the greatest separation. When scaled to multiple dimensions, this boundary line becomes a plane in n-dimensional space which separates classes with the greatest separation. This dividing line/hyper-plane is called a Support Vector Machine.

The margin of the hyper-plane is the distance either side of the hyper-plane by which the hyper-plane could be moved before hitting a data point. A wider margin between two classes implies a better generalisation. The intuition being that we are placing the hyper-plane as far as possible from known data of both classes.

In many cases, two classes may not be entirely separable by a hyper-plane. In this case, some 'slack' is tolerated and the constraint of a perfectly separating hyper-plane between classes is relaxed to allow for some instances falling inside the margins and indeed beyond the separating hyper-plane. This is done through the use of slack vectors denoted ϵ_i, which represent the errors associated with data points either inside the margin or on the wrong side of the separating hyper-plane. The sum total of these errors is added to the SVM cost function and multiplied by a parameter C, which controls the tolerance for misclassified samples. This parameter C may be tuned to achieve good performance, with large values of C placing more importance on reducing error at the cost of margin width and vice versa.

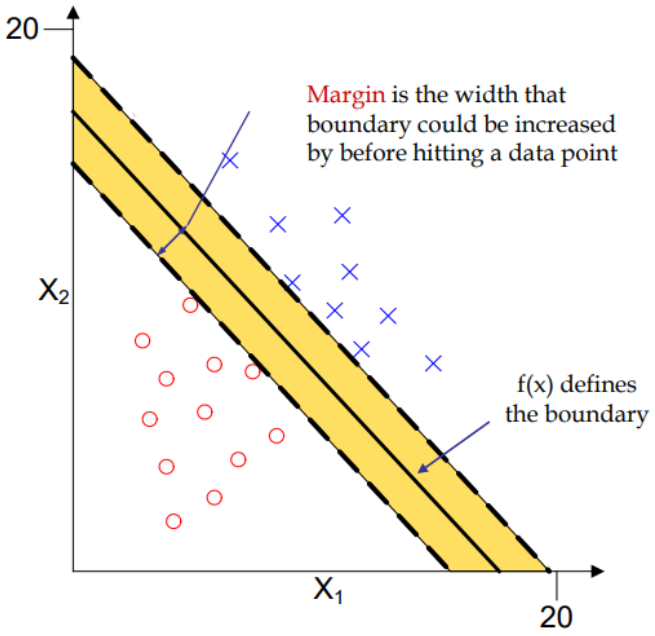

Figure 2: Linear SVM[11]

In the case where two classes are not linearly separable, SVM's may still find the maximum margin hyper-plane between classes after using a transformation to a new feature space where the data is linearly separable. This is known as the kernel trick. Different kernel functions are used to compute the separating hyper-plane without explicitly mapping the data to a new feature space. Examples of commonly used kernel functions include the Radial Basis Function (rbf), the linear kernel and polynomial kernels of varying degrees.

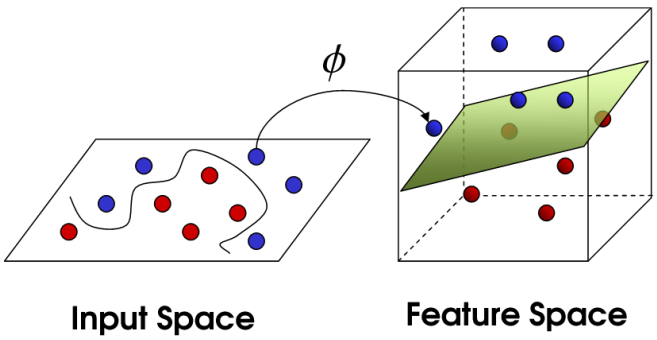

Figure 3: Visualisation of Kernel Trick for SVM [12]

2.3 Deep Learning

Deep Learning is the branch of Machine Learning which deals with Artificial Neural Networks (ANN's). ANN's are a type of learning algorithm which are inspired by the way the human brain learns. They are made up of nodes connected by edges, similar to neurons and synapses in the human brain. Each ANN is organised in layers, each layer is made up of a certain number of nodes, which are connected to nodes in the preceding and following layers. The first and last layer of an ANN are called the input and output layers respectively. Layers between the input and output layers are referred to as hidden layers.

The activation of the i^{th} node in the l^{th} layer is denoted A_i^l. This activation is calculated according to the formula $A_i^l = f(Z_i^l)$. Z_i^l is defined as the sum of inputs into the node, or sometimes called the z-value of a node. This is given by:

$$Z_i^l = \sum_{j=1}^{n} W_{ij}^{l-1} * A_j^{l-1} + b_i^{l-1}$$

where W_{ij}^{l-1} denotes the edge weight from node j in layer l-1 to node i in layer l, A_j^{l-1} denotes the activation of node j in layer l-1 and b_i^{l-1} denotes the bias value from layer l-1. $f(Z_i^l)$ is the value obtained when Z_i^l is passed through an activation function, such as a sigmoid function.

The set of weights and biases associated with an ANN must be learned through training, with the Backpropagation algorithm used to backpropagate errors from the output layer back though the network. The weights are then adjusted according to the portion of the error each node is responsible for. ANN's may be trained for use in Classification tasks to produce a classification, or for Regression tasks which involve predicting a numerical quantity.

The ANN architecture used in this analysis in investigating the research questions posed is the Fully Connected Neural Network, described below.

2.3.1 Fully Connected Neural Network

The most common type of ANN architecture is a Fully Connected Neural Network (FCNN), where all the nodes in each layer from the input layer, through each of the hidden layers to the output layer, are connected to each other by edges. These fully connected layers are sometimes called 'Dense' layers.

Fully Connected Neural Networks may be used for both classification and regression tasks, with the outputs given as 0 or 1 in the case of a binary classification task, or a number in the case of a regression task. Parameters associated with ANN's include the number of hidden layers used, the number of nodes in each layer and the activation functions used.

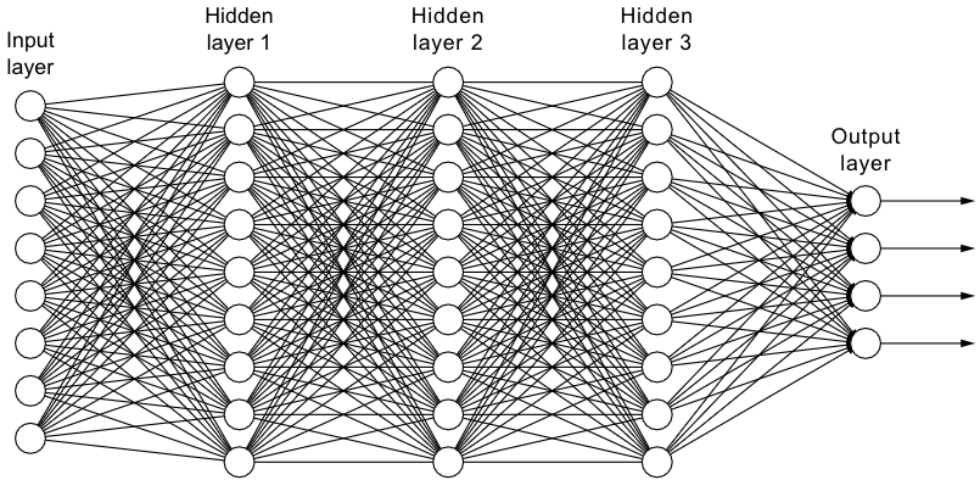

Figure 4: Fully Connected Neural Network architecture[13]

2.4 Knowledge Graphs and Embeddings

2.4.1 Knowledge Graphs

Knowledge Graphs are a form of relational knowledge representation. A Knowledge Graph is a network consisting of interlinked entities, with the links between entities defined by particular relations. Entities represent things such as objects, people and abstract concepts. Relations define how entities are linked to one other. For example, the relation "is a member of" could be used as the relation linking Ireland (an entity) and the EU (another entity) in a Knowledge Graph. KGkg, the Knowledge Graph about Knowledge Graphs, defines a Knowledge Graph as *"a digital structure that represents knowledge as concepts and the relationships between them (facts). A knowledge graph includes an ontology that allows both humans and machines to understand and reason about its contents."*[14]

Formally, a Knowledge Graph is defined as a set of triples (sometimes called 'facts') $\{s, r, t\}$ where s represents the 'source' entity, r represents the 'relation' linking the entities, and t represents the target entity. Furthermore, $s, t \in \mathbf{E}$, $r \in \mathbf{R}$, where $\mathbf{E}$ represents the set of all entities in the graph, and $\mathbf{R}$ represents the set of all relations between entities.

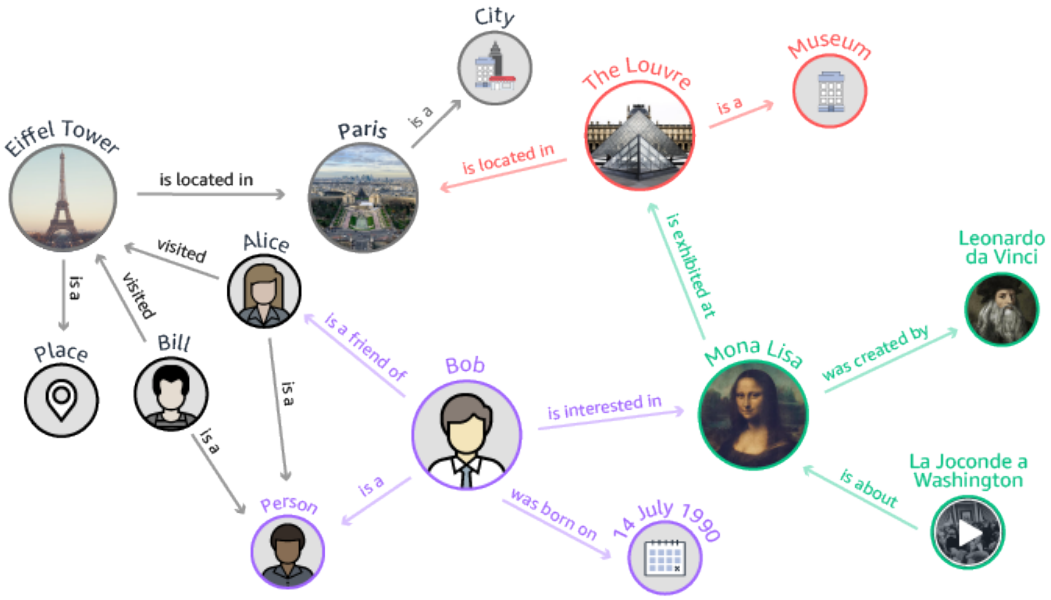

Figure 5: Knowledge Graph with connected entities and relations[15]

A Knowledge Graph may be likened to a database, as data may be queried via structured queries, however a Knowledge Graph adds additional meaning or 'knowledge' to the data within it by its very structure.

The structure of the graph represents how different entities are related. This added 'knowledge' cannot be obtained using traditional database methodologies.

2.4.2 Knowledge Graph Embeddings

Knowledge Graph Embeddings are the numerical representations which result from embedding components of a Knowledge Graph such as entities and relations into continuous vector spaces. Embeddings in this scenario are created in such a way as to preserve the structure of the original Knowledge Graph. Wang et al.[16] provide a systematic review of existing approaches to KG embeddings and their applications. The authors group the different types of embeddings roughly as follows:

- *Translational Distance Models*. These models use distance based scoring functions to assess the accuracy of embeddings. Translational Distance Models measure the likelihood of a fact (a triple consisting of a source entity, relation and target entity) as the distance between the two entities normally after a translation has been carried out by the relation[16]. Entity and Relation embeddings are learned through training. Examples of Translational Distance Embedding Models include the TransE and TransH models.

- *Semantic Matching Models*. These models use similarity-based scoring functions to measure the plausibility of facts[16]. This is done by comparing latent semantics of entities and relations in their vector space representations[16].

  Latent Semantic Analysis is a technique in Natural Language Processing which assumes that words which are close in meaning will occur in similar pieces of text. This results in a matrix being created with individual words represented as rows and the number of occurrences of words in each piece of text given in each of the columns. Each word therefore has a vector representation in an associated vector space. Similar words are found by using a similarity measure such as cosine distance and computing the angle between the vector representations of two words. This is similar to the way Semantic Matching Models operate. In the case of facts derived from a Knowledge Graph, each fact will have a unique vector space representation, and the plausibility of each fact is assessed via a similarity-based scoring function between other facts. Entity and Relation embeddings are again learned through training. Examples of Semantic Matching Models include RESCAL and DistMult.

Knowledge Graph Embeddings may be used in a variety of tasks, these include *Link Prediction*, the task of predicting an entity that has a specific relation with another given entity. This adds missing knowledge to the graph. As well as this, another important use is in *Entity Classification*. Entity Classification aims to categorize entities into different semantic categories, e.g. Barack Obama is a *Person* and the Mona Lisa is a *Work of Art*.

Most recently, the Node2Vec algorithm developed by A. Grover and J. Leskovec[17] has given state-of-the-art performance in learning continuous representations of nodes in networks. This paper and the Node2Vec algorithm will be discussed in Section 6.2.2.

2.5 Literature Review

This review refers to publications where Machine Learning, Deep Learning and Knowledge Graphs and Embeddings have been used in stock market forecasting in a variety of different scenarios using numerous different methodologies. The review is structured roughly according to the nature of the models used for predictions. These are Single Algorithm Models, Ensemble Models and Deep Neural Networks.

2.5.1 Single Algorithm Models

Kyoung-jae Kim[18] applies Support Vector Machines to the problem of predicting the price of a stock index (an index that measures the performance of a stock market, or a subset of the stock market), as well as comparing SVM's to Back Propagation Neural Networks (BPNN) and case-based reasoning (CBR). Kim describes how SVM's can offer a better alternative to a BPNN as the solution of an SVM is generally a global optimum while many Neural Network models tend to get stuck in locally optimal solutions due to the nature of the gradient based back propagation algorithm. The author notes the relatively few parameters which require tuning with an SVM model as opposed to with a NN. The index used is the Korean composite stock price index (KOSPI), with 12 technical indicators chosen as input variables. The daily changes up or down in the price of the index are categorized as 1 or 0 respectively. Various configurations of the Support Vector Machine are trialled, with different kernel functions used, as well as different hyper-parameter values. The results of experiments conducted show that the SVM outperforms BPNN and CBR on hold out data, with the SVM also producing a higher prediction accuracy on training data than the BPNN by over 6%. Following these experiments, statistical tests are conducted which conclude that SVM performs better than CBR but does not significantly outperform BPNN.

Huang et al.[19] also use a SVM in attempting to forecast the future movement of the Japanese NIKKEI 225 index. The SVM is compared against Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA) and Elman Back Propagation Neural Networks (a partially recurrent NN where connections are mainly feed forward but also involve connections that allow the network remember past information). The model inputs consist of the value of the US S&P 500 index, as well as the exchange rate of US Dollars to Japanese Yen. The results of experiments demonstrate the SVM outperforms all other non-ensemble models. However, an ensemble model consisting of SVM combined with other classification algorithms performs best among all models constructed and gives the best predictive accuracy of all forecasting models tested.

Shen, Jiang and Zhang[20] explore the use of Machine Learning models, in particular SVM's, in the prediction of daily and longer term trends associated with US stock indices such as the NASDAQ, Dow Jones and S&P 500. They hypothesise that the "temporal correlation" or correlation over time between other stock markets across the globe should be considered when building a predictive model, and that this information should prove useful to a predictive Machine Learning model. Their hypothesis is supported by the results of their experiments, using a forward feature selection algorithm, which indicate a predictive accuracy for daily trends of over 74%, rising to 77.6%, for the previously mentioned indices. This accuracy rises to 85% over a 30-day period. As well as this, they attempt to predict the exact daily movement of each index using

linear regression, a generalized linear model (GLM) and a SVM, with the SVM model producing the best predictive performance. Furthermore, they outline a trading model based on their previously constructed Machine Learning models and compare this model against other chosen benchmarks in a simulation over different periods of time. This trading model performs much better than other benchmarks on average, with the approximate annual rate of return of money invested reported as 30%.

2.5.2 Ensemble Models

Patel et al.[21] use a fusion of Machine Learning algorithms to make predictions on the future values of two Indian Stock Indices, the CNX Nifty and S&P Bombay Stock Exchange (BSE) Sensex, over varying time horizons. The predictive performance of these hybrid ensembles is compared against the performance of the individual algorithms which make up the ensemble. Ten technical indicators are chosen as input data for each of the models. According to the authors, the motivation for the fusion of several Machine Learning models together comes from their observance that existing methods available at the time of writing involved only a single stage of prediction, taking in the input parameters and producing a final output. It is the authors' belief that as the time horizon of predictions increases, predictions are based off older parameters which make any model predictions less accurate. To deal with this issue, a multi-stage prediction scheme is devised to increase the accuracy of predictions. Each of the hybrid models constructed involve the use of Support Vector Regression (SVR) at the first stage, with the second stage of the model using either an ANN, Random Forest or SVR again. This results in three hybrid prediction models, SVR-ANN, SVR-RF and SVR-SVR. The performance of these hybrid models is then compared with the one stage scheme where an ANN, RF or SVR is used on its own. The authors observe that the two stage models outperform the corresponding single stage models for almost all prediction tasks over various time periods, with the SVR-ANN model performing the best of all models on both Stock Market Indices. The authors' justification for the superior performance observed in two stage models is that the prediction models in the second stage have to identify the transformation from newer, more informative parameters relating to that day outputted from the SVR in stage 1 to that day's closing price. This is much easier than attempting to learn the transformation from older, less informative input parameters from previous days to a future day's closing price.

Ballings et al.[22] compare ensemble methods such as Random Forest and AdaBoost against single classifier models such as NN's, SVM's etc. in the prediction of stock price movements. The authors motivation stems from their belief that at the time of writing, there were very few ensemble methods represented in the literature in this area, despite their distinguished performance in many other areas. The authors set out to benchmark these ensemble methods against single classifier models. Methods are compared using the area under the receiver operating characteristic curve (AUROC) performance measure, with predictions given for one year in advance. All AUROC scores are greater than 0.5, indicating every model constructed is better than random, with Random Forest achieving the highest predictive accuracy of all models used. Added to this, the three ensembles included in the benchmark are ranked in the top 4 performing algorithms, highlighting the effectiveness of ensemble methods.

Hassan et al.[32] combine a Hidden Markov Model (HMM), ANN and Genetic Algorithm's in an ensemble to forecast financial market behaviour. The ANN transforms daily stock prices to sets of values which are used as inputs to the HMM, with the GA then used to optimise the initial parameters of the HMM. The HMM is trained to identify patterns in historical data, which are then used as a basis for making predictions for the next required day. The idea being that if stock behaviour on a certain day corresponds to the behaviour of a stock on a different day, the change in the stock price for the following day in both scenarios should be similar. Predictions are given for several company stocks and are compared against a baseline forecast method with the results demonstrating improved performance of the ensemble model compared to a solitary HMM model, as well as similar performance to that of another popular statistical forecasting method (ARIMA) despite much less analysis of the dataset prior to the prediction.

2.5.3 Artifical Neural Networks

Yudong and Lenan[23] propose an improved bacterial chemotaxis optimization (IBCO), incorporated into an Artificial Neural Network (ANN) with back propagation in developing a model for prediction of stock indices in both the short term (next day) and long term (15 days). This method is shown by experimental results to perform better than other methods in its learning ability as well as its generalization. The central motivation behind their work is that conventional back propagation results in a set of weights which are a local optimum of all possible sets of weights due to the nature of gradient descent, in the sense that it is possible to find a more optimal set of weights for the ANN which produce more accurate predictions. The solution, according to the authors, is to optimize the configuration of the ANN while training using global search algorithms, in their case by creating an improved bacterial chemotaxis optimization (IBCO). Their improvement to traditional BCO allows the ANN to move out of locally optimum configurations and helps maintain its global search ability over the entire search space. The IBCO-Back Propagation model is found to perform much better than the simpler Back Propagation model, with greater predictive accuracy as well as faster computation time for predictions.

Atsalakis et al.[24] review and classify multiple published articles focusing on neural and neuro-fuzzy techniques applied to stock market prediction, classifying models based on the nature of their construction. These classifications are made according to the type of input data used, the prediction methodology, and the performance measures used. The authors outline their belief that the most important idea in successful stock market prediction is achieving the best results using the simplest model and using the minimum required input data. The central contribution of the paper is the comprehensive presentation and classification of all the varied methods and techniques that have been applied to the problem of stock market forecasting, with the authors encouraging further analysis, evaluation and comparisons of each of the methods. The conclusions drawn by the authors are that neural networks and neuro-fuzzy models are useful for stock market prediction, however the authors also draw attention to the issues encountered when deciding on the optimal network architectures or configurations of neural models, with the optimal structure only obtainable through trial and error.

Kayakutlu et al.[25] evaluate the effectiveness of several different Neural Network models in stock market index prediction. The motivation of the authors is to compare different variations of NN models against one another as opposed to comparing them against other techniques. The models chosen for analysis are the multi-layer perceptron (MLP), Dynamic Artificial Neural Network (DAN2) and hybrid NN's which use generalized autoregressive conditional heteroscedasticity (GARCH) to extract new input variables. The stock exchange used to analyse predictions is the NASDAQ and models are compared using Mean Squared Error (MSE) and Mean Absolute Deviation (MAD). The authors note the unexpectedly high error results of the GARCH-DAN2 model and highlight and summarize the inconsistencies within the model, with the conclusion drawn that the DAN2 method behaves like a statistical method rather than a ANN. As a result, the most reliable and best performing predictive model is found to be the classical MLP model with hybrid methods failing to improve predicted results.

Boyacioglu et al.[26] also use ANN's and SVM's in attempting to predict the direction of daily movement (positive or negative) in the Istanbul Stock Exchange (ISE) National Index. 10 technical indicators are chosen as the inputs to both models. The paper seeks to highlight and verify the predictability of the index direction using an ANN or SVM, as well as compare the performance of both models against each other. The authors conducted a comprehensive search of possible configurations of parameters for both models using a small subset of the available data to find the most optimal parameter settings for each model, with the training performance of each configuration recorded. The experimental results on test data highlight the better performance on average of the ANN model relative to the best SVM model. The accuracy obtained with the ANN model (75.74%) was superior to all other previous models constructed based on the ISE National 100 Index at the time of writing. As well as this, the authors best SVM model also outperformed similar studies based on the index in the literature.

Wang et al.[27] propose a novel Wavelet De-noising-based Back Propagation Neural Network (WDBPNN). This network can deal with the noise in available data using a wavelet transform to predict the change in the price of a company stock or index. The network is applied to data available on the Shanghai Composite Index. The performance of the WDBPNN is assessed by comparison to an ordinary Back Propagation Neural Network (BPNN). The results of experiments show the WDBPNN significantly outperforms the BPNN used without de-noising processing.

Chen et al.[28] use an LSTM RNN model to predict the returns of the Chinese stock market. The raw data used was transformed into 30-day sequences to make use of the sequence learning power of the LSTM architecture. These sequences were encoded with 10 learning features as well as an earning rate label associated with each sequence. The results of experiments conducted demonstrate the improved prediction accuracy obtained by using an LSTM model, outperforming other methods tested and with an increase of 12.9% in terms of accuracy compared to the random prediction by chance model.

JB Heaton et al.[29] investigate the use of deep learning for problems involving financial prediction and classification. These prediction problems include pricing securities, constructing portfolios and risk manage-

ment. They highlight the benefit in using deep learning models to exploit information and trends hidden in data which would otherwise be unobtainable with conventional financial economics and/or other methods of statistical arbitrage or quantitative asset management. The authors present a comprehensive outline of the architecture of deep neural networks, as well as the steps required to train such a network. They also outline the theory, as well as the usefulness of, stacked auto-encoders for certain problems. A four-step deep learning algorithm is outlined with the goal of *"constructing a procedure which outlines the trade-offs involved in constructing portfolios to achieve a given goal"*, with the example given of beating a relative index in terms of its performance by a specified level. This model is then used to construct an optimal portfolio of stocks using a subset of the stocks contained in the biotechnology IBB index with the goal of outperforming the returns of this index by 1% per year. This is achieved by uncovering performance improving "deep features" within the training data and utilising these features in the optimized portfolio.

Kraus et al.[30] investigate the use of deep learning, specifically Natural Language Processing and LSTM RNN networks, on company financial disclosures to aid decision making, as well as predict stock price movements in response to these disclosures. The authors investigate whether sequence modelling with deep neural networks provides better predictions of stock price movements after company financial disclosures when compared to bag-of-words methods. The authors also explore the use of transfer learning to further improve predictive performance. The authors utilize sequence modelling based on deep neural networks due to its ability to consider long term dependencies between words. The authors experiments involve predicting the direction up or down of the stock price related to the company disclosure, as well as the magnitude of the change in both cases. The results of these experiments give a strong indication that deep learning methods are superior to traditional bag-of-words approaches, as well as consistently highlighting the added benefit of employing transfer learning to further boost performance.

Vargas et al.[31] use financial news articles as well as some technical indicators as inputs to a deep learning model to predict the intraday (daily) directional movements of the S&P 500 index. The authors make use of Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) architectures in their experiments. The resulting RCNN model aims to utilise the benefits associated with both CNN and RNN models, with CNN's better able to extract semantic information from text and RNN's better able to capture the overall context of the text, with the results obtained compared against those achieved by Ding et al. [15] The hybrid model proposed by the authors, which takes in both financial news articles as well as numerical technical indicators, is compared against a model which uses only news articles as inputs. As well as this, the authors two different methods of representing text sentences are compared, word embeddings and sentence embeddings. The results of the performance of each model are then compared against other benchmark models reported in the literature. The results of these experiments reveal that sentence embeddings are better than word embeddings as inputs to the model, RCNN is a more optimal structure than CNN for index prediction, and the inclusion of technical indicators in addition to news articles leads to improved performance.

Kim et al.[33] propose a Genetic Algorithm approach to feature discretization and for calculating the associated weights of an ANN which is then used to predict the stock price volatilities of the Korean stock price

index (KOSPI). In this study, the GA is used not just to optimize the learning algorithm, but to optimize the feature space by reducing its dimensionality and removing irrelevant features. Feature discretization involves *"transform(ing) continuous values into discrete ones in accordance with certain thresholds"* and is closely related to dimensionality reduction. By discretizing input data in this way, the author hopes to simplify the learning process as well as the generalizability of the learned model by eliminating uninformative features. Experiments are conducted comparing three models using 12 technical indicators as inputs, the author's genetic-algorithm feature discretization (GAFD) model, back propagation neural network with linear transformation (BPLT) and ANN trained with GA with linear transformation (GALT). The results demonstrate that the GAFD model outperforms both comparative models by 10-11% on hold out test data, with the author noting a slight difference in the prediction accuracy between the training and test data for the GAFD model and a larger difference in the case of both other models. The authors hypothesize that this may be because the discretization helps in the simplification of the learning process which eliminates irrelevant patterns and in turn may help the model from overfitting.

Ding et al.[34] propose a deep learning method using NLP techniques for event driven stock market prediction. Events are extracted from text such as news articles and financial news, with these events then represented as dense vectors via a Neural Tensor Network (NTN). By using dense vectors, the authors address the issue of sparsity associated with structured representations of events, with these dense vectors, called event embeddings, representing structured events. A deep CNN is then used to predict how events affect both short and long-term price movements of the SP 500 index as well as of individual stocks. The authors report that their model outperforms all other state-of-the-art baseline methods tested in terms of accuracy of prediction of the S&P 500 index, as well as accuracy of prediction for individual stocks, with the conclusion drawn that event-based models are better at representing text documents used for prediction than word-based models. The constructed model is then used in a market simulation as part of a trading strategy, with the model achieving consistently better returns than the associated benchmark on money invested in 7 of the 8 companies chosen.

More recent AI techniques employed in stock forecasting include Knowledge Graphs and Knowledge Graph Embeddings. Ding et al.[35] build on their previous work from 2015 by utilising a Knowledge Graph to provide additional background knowledge on entities and relations extracted from financial news text corpora. They propose to combine the information from the Knowledge Graph into the objective function of an event embedding learning model. Event embeddings are numerical vector representations of events, with the idea being that events which are similar will also have similar event embeddings, and vice versa. The aim of this model is to obtain improved event embeddings by utilising available Knowledge Graphs and produce more accurate predictions of stock market changes. In their own words, their work *"integrates knowledge into vector representations of events"*. Event embeddings are learned using a Neural Tensor Network (NTN). The performance of the model in producing accurate event embeddings is assessed, with predicted event embeddings compared against human-labelled event similarity. The performance of the constructed event embedding model combined with a CNN on S&P 500 Index prediction is given as 66.93%, outperforming all other representation models tested. This is also the case for Individual Stock prediction, reinforcing the

conclusion that event-based models are better at representing news documents for prediction than word-based models.

Liu et al.[36] explore the use of Knowledge Graphs and Knowledge Graph Embeddings in improving the accuracy of stock price predictions. They propose a joint model combining the TransE model for representation learning combined with a Convolutional Neural Network (CNN) to improve the accuracy of feature extraction from news articles related to Apple Inc. This joint model helps with the information loss observed when text is constructed into a structural vector. As well as this, within the same model, the authors make use of daily trading data and other technical indicators. Their approach is then evaluated using a SVM and an LSTM RNN, as a comparison between traditional machine learning in the case of the SVM and a deep learning method in the case of the LSTM RNN. The joint model constructed achieves the highest performance of all models in predicting the movement of Apple stock for the next day. In addition, the classification accuracy of the joint learning model on news article sentiment is also reported as 97.66%.

Liu et al.[37] propose to combine the use of a Knowledge Graph with news sentiment analysis to predict stock prices. Central to their methodology is improving the way in which the correlation between stocks is measured by incorporating the various relationships between companies to predict stock price volatilities. To do this, the authors calculate the correlation using embeddings derived from an Enterprise Knowledge Graph, which considers various relationships between companies, with the premise being that news affecting one company will also have an effect on the share price of companies associated or related to this company. News sentiment vectors were obtained and constructed for each company, as well as information regarding quantitative features of the company's stock price. The final predictive Gated Recurrent Unit (GRU) model created takes as inputs news sentiment vectors, the correlation matrix created using the relationship embeddings from the Enterprise Knowledge Graph and stock quantitative data. The results of experiments conducted using this model highlight the superior accuracy of the model when compared against other benchmarks, with the Enterprise Knowledge Graph model producing 8.1% greater average accuracy than other comparative models.

Most recently, Long et al.[38] have proposed an innovative approach by integrating deep learning and a Knowledge Graph into a framework with the goal of predicting stock price movements and longer term trends, with the example of the Chinese stock market used. Their model uses both market data as well as trading information, in the form of desensitized transaction records, together with the help of a Knowledge Graph and Knowledge Graph Embeddings to account for the correlation between stocks. The belief of the authors is that the stock market is influenced by trading behaviours (individual traders buying or selling). The authors make use of the Knowledge Graph by applying the Node2Vec algorithm to obtain vector representations of company nodes from the Knowledge Graph, with these vectors then used by the deep learning architecture to extract features. The Deep Stock-trend Prediction Neural Network (DSPNN) constructed is an attention-based, bidirectional LSTM RNN network which predicts price trends, with experiments demonstrating that the model outperforms other prediction baselines tested. Experiments performed comparing the DSPNN to other techniques as well as to variations of its model makeup demonstrate the superior performance of the DSPNN model. The authors note the consistent improvement in accuracy obtained because of using trading

data as well as public market information, with investor clustering and Knowledge Graph Embeddings used to mine these useful features.

3 Data

3.1 Overview of Data Used

In order to be listed on a US Stock Exchange, companies are required by law to provide yearly and quarterly financial disclosures to the Securities and Exchange Commission (SEC)[53]. This yearly statement is known as a 10-K filing, and provides a comprehensive summary of a company's financial performance. The quarterly statement is known as a 10-Q filing. Both filings allow investors to assess the financial state of a company before deciding to invest in shares of that company. The nature of the filings is unstructured, with some sections of these disclosures containing text, and others containing the financial data required for this analysis. Fortunately, some websites provide access to these filings in a structured, easy to use format. The website used to gather data in the case of this project is Seeking Alpha[52], which provides these financial filings in CSV file format.

This analysis aims to investigate whether it is possible to predict an individual company's share price based off that company's financial statements, and also whether the change in the state of a company's statements can be used to predict a corresponding change in price. The idea behind this is that as companies operate throughout the year, making profits or losses, acquiring assets etc., their performance is reflected in their share price. Thus, the financial performance from a previous trading year (outlined in a company's filings) should be reflected in a company's price at the end of the trading year.

For each company in the analysis, the Balance Sheet, Income Statement and Cash Flow statement associated with every 10k filing over the past 10 years were downloaded. In total, 500 different companies make up the analysis, giving a total of 5000 data instances (one instance per company per yearly filing). Companies were chosen from 10 different economic sectors, which are described in Section 6. As well as this, only companies listed on US Stock Exchanges from the 31st of December 2008 were selected for analysis. This was done in order to ensure that all companies in the analysis had produced 10-K filings in each of the last 10 years.

As well as this, daily price data associated with each company was downloaded for the last 10 years, this data contained information such as the date, opening share price, daily high, daily low, closing price, volume and price change percentage. All price data had been adjusted for stock splits. The price associated with each company and each 10-K filing was chosen as the closing price on the last day of the month the 10-K filing was filed in. The idea being that all changes in price resulting from a company's financial performance have taken place by this point.

In order to investigate the research questions posed, it was necessary to modify the data to answer the particular question being investigated. These modifications fall under the following headings:

- ***Raw Filings with raw price data***. This data is used to attempt to answer the question of how accurately a company's stock price can be predicted from its raw financial statements, and which statement(s)

produce the most accurate predictions. This was done using each companies Balance Sheet, Income Statement, Cash Flow statement and also using these three statements merged together in a 'Merged Statements' dataset with the corresponding price. Each dataset used in answering this question contains 5000 data instances, one instance per company per 10-K filing.

- ***Change in Filings with change in price***. This data is used to attempt to answer the question of how accurately the change in a company's stock price can be predicted from the change in its financial statements, and which statement(s) produce the most accurate predictions. The change in statements is calculated between consecutive 10-K filings. This was done using each company's Balance Sheet, Income Statement, Cash Flow Statement and also using these three statements merged together in a 'Merged Statements' dataset with the corresponding price. Each dataset used in answering this question contains 4500 data instances. This is because the change in statements can only be calculated 9 times for each set of 10 company statements.

- ***Change in filings with price trend classification***. This data is used to attempt to answer the question of how accurately the trend upward or downward in a company's price can be predicted from the change in its financial statements, and which statement(s) produce the most accurate predictions. In this case, an increase in price between successive filings was denoted as 1 and a decrease in price denoted as 0. There were found to be 1462 instances of a price decrease (Class 0) and 3038 instances of a price increase (Class 1). For the same reasons as stated above, each dataset used in answering this question contains 4500 data instances.

Four datasets were created in answering each research question, one for each Balance Sheet, Income Statement, Cash Flow Statement and 'Merged Filings' Statement. Datasets associated with a change in price will be referred to as Delta Filings from now on for simplicity.

3.1.1 Data pre-processing

Upon downloading the data, several Python scripts were created in order to modify, clean and arrange the data in a way suitable for input to predictive machine learning models. This included steps such as removing dollar signs, percentage signs and dashed lines, transposing the initial format of the filings to make each yearly filing into a row instead of a column. The pre-processed data was originally stored in tables in MySQL before being converted to CSV files. This pre-processed data is available at:
`https://github.com/conor-mel1/StockPricePredictionFromFinancialFilings`.

3.2 Financial Statements

3.2.1 Income Statement

Following pre-processing, the parameters chosen from each Income Statement were:

- ***Total Revenues***; this figure represents the total amount of a company's sales and other sources of income.

- ***Total Operating Income***; this figure represents the amount of profit realized from a business's operations after deducting operating expenses such as wages, depreciation and cost of goods sold.[41]
- ***Total Net Income***; Net income is calculated as total revenues minus expenses, interest, and taxes.
- ***Revenue per Share***; This figure represents the total revenue earned per share.
- ***Normalized Basic Earnings per Share***; Normalized earnings remove one-off events and smooth seasonal affects on revenue. Normalized earnings better represent the true health of a company's core business.[42]
- ***Dividend per Share***; Dividend per share is the sum of declared dividends issued by a company for every ordinary share outstanding. The figure is calculated by dividing the total dividends paid out by a business over a period of time by the number of outstanding ordinary shares issued.[43]
- ***EBIT (Earnings before Interest, Taxes)***; This figure represents a company's net income before income tax expense and interest expenses are deducted.
- ***EBITA (Earnings before Interest, Taxes and Amortization)***; This figure represents a company's net income before income tax expense, interest expenses and amortization expenses are deducted.

These parameters were chosen as they were common to all companies from all 10 sectors.

3.2.2 Balance Sheet

Following pre-processing, the parameters chosen from each Balance Sheet were:

- ***Total Assets***; This figure represents the combined amount of a company's fixed assets and current assets.
- ***Total Liabilities***; This figure represents the combined debts and obligations the company owes to others.
- ***Total Equity***; This figure represents the amount of money that would be returned to a company's shareholders if all of the company's assets were liquidated and all of the company's debt was paid off.[44]
- ***Total Shares Outstanding on Filing Date***; The number of company shares which are outstanding at the time of the company's filing.
- ***Book Value per Share***; The book value of a company is the difference between that company's total assets and total liabilities. As such Book Value per Share is the book value of a company divided by all shares outstanding.
- ***Tangible Book Value***; This figure measures a company's equity without the inclusion of any intangible assets such as brand recognition and intellectual property.
- ***Tangible Book Value per Share***; This figure represents the tangible book value of a company divided by all shares outstanding.

- ***Total Debt***; This figure represents the total debt owed by a company.
- ***Net Debt***; Net debt is a liquidity metric used to determine how well a company can pay all of its debts if they were due immediately.[45]

These parameters were chosen as they were common to all companies from all 10 sectors.

3.2.3 Cash Flow Statement

Following pre-processing, the parameters chosen from each Cash Flow Statement were:

- ***Net Income***; Defined in section 3.2.1.
- ***Cash from Operations*** This figure represents the amount of money a company brings in from its ongoing, regular business activities.
- ***Cash from Investing*** This figure represents how much cash has been generated via investment-related activities in a specific period.
- ***Cash from Financing***; This figure represents the net flows of cash that are used to fund the company.
- ***Net Change in Cash***; This figure represents the amount by which a company's cash balance changes in an accounting period.
- ***Levered Free Cash Flow***; This figure represents the amount of cash a company has after paying its debts.
- ***Unlevered Free Cash Flow***; This figure represents the amount of cash a company has before paying its debts.
- ***Free Cash Flow per Share***; Free cash flow represents the cash a company generates after accounting for cash outflows to support operations and maintain its capital assets.[46] Free Cash Flow per Share is simply this value divided by the total number of shares outstanding.

  These parameters were chosen as they were common to all companies from all 10 sectors.

3.3 Limitations of the data

Some limitations of the data include the fact that each of the 10-K filings associated with a company are not always equally spaced apart in time. Some companies may decide to change the time of their yearly filings and as such there may be some overlap in the figures contained in consecutive filings. As well as this, it is difficult to reconcile each filing with a price, as it could be argued that a more appropriate price would be to use the price on the exact date of the filing. As the exact date of each filing was not given with the data which was downloaded, the closing price on the last day of the filing month seems most appropriate to use.

Furthermore, not all parameters from every statement are used in this analysis. Parameters were chosen on the basis that they were common to each company in the analysis. This means that not all potentially

useful parameters have been used. Using the specific example of banks, some parameters which would be common to all banks but not companies from other industries have not been used. These parameters may be important in assessing the price changes in banks specifically, and by not utilising them valuable information may be lost. This is a possible area for future work and is discussed in more detail in Section 7.

Finally, there is a substantial amount of noise present in stock price data. These prices can be affected by breaking news, the current political climate and other world events on a daily basis. Even though there may be a definite relationship between the financial statements and the share price, the noise in the data will make it difficult to uncover this relationship.

4 Raw Filing Data - Comparison of Regression Models

4.1 Introduction

This section will investigate the first of the research questions posed in the introduction.

1. How accurately can a company's financial statements predict that company's share price? And which statements in particular are good predictors of the price?

4.2 Regression Algorithms Used

The regression algorithms used include the k-Nearest Neighbour Algorithm, the Decision Tree algorithm, the Random Forest algorithm as well as a Fully Connected Neural Network (FCNN). Descriptions of these algorithms can be found in Section 2.2/2.3.

4.3 Methods

This section aims to calculate the accuracy of price predictions using raw unadjusted company statements and their associated prices. This is done using each companies Balance Sheet, Income Statement, Cash Flow Statement as well as using each of these statements together in a Merged Statements dataset. These predictions are made using both Machine Learning and Deep Learning predictive algorithms.

4.3.1 Model Training and Testing

Machine Learning Models

Each Machine Learning model is evaluated using 10-Fold Cross Validation. K-Fold Cross Validation is a technique used to assess the accuracy of Machine Learning algorithms. The available data is split into K 'folds'. This is done to account for differences in the performance of the algorithm based on the specific data the algorithm was trained on. The average accuracy obtained by the algorithm over the whole dataset is obtained as follows. For K iterations, the algorithm is trained on K-1 folds and tested on the remaining fold. As such, each fold is used for training K-1 times and used for testing once. The accuracy of the model for each iteration is recorded and the average performance of the algorithm over all folds is used as a measure of the overall performance of the algorithm.

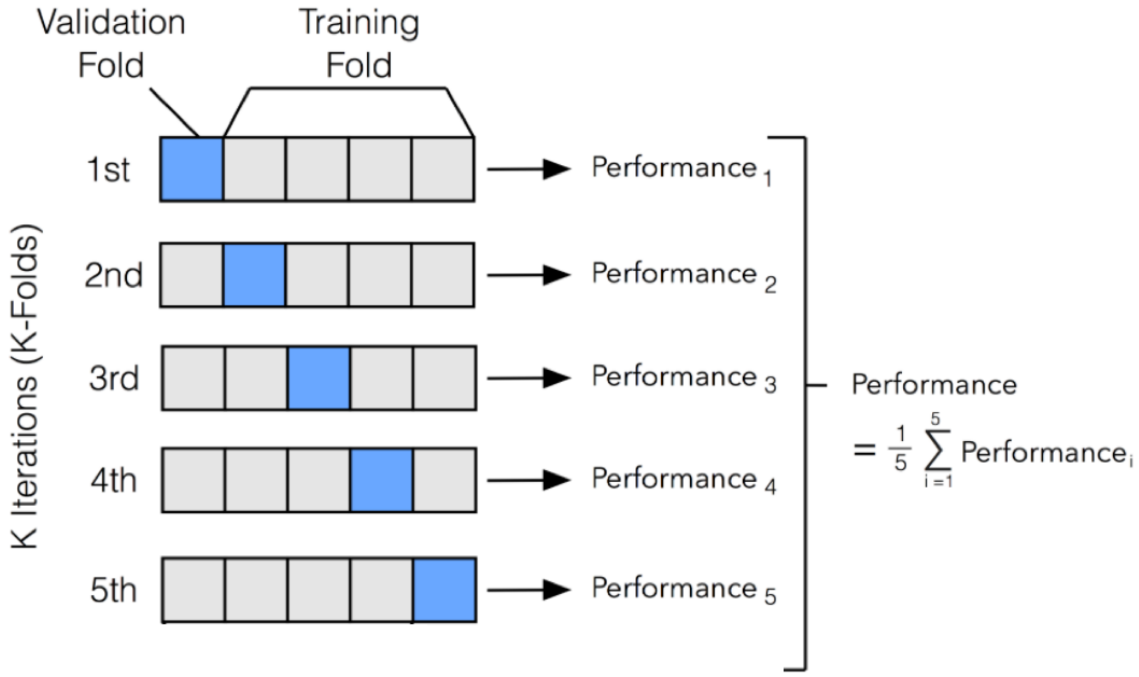

Figure 6: Visualisation of 5-Fold Cross Validation [39]

All models are trained on each of the 10 training folds and tested on the respective test folds. Mean Absolute Error (MAE) is the metric used to assess the accuracy of model predictions. MAE is calculated as:

$$MAE = \frac{\sum_{i}^{n} |x_i - y_i|}{n}$$

Where x_i is the predicted price given by the model, and y_i is the true price value.

The average MAE and associated standard error of these results over all 10 folds are reported. The KNN, Decision Tree and Random Forest algorithms are implemented using the Scikit-Learn[48] Python library. Optimal hyper-parameters for each algorithm are obtained via a grid search, with the hyper-parameters which produce models with the lowest MAE chosen. The results of this hyper-parameter testing are given in Tables 10 and 11 found in the Appendix.

Deep Learning Models

Each FCNN architecture is evaluated using 5-Fold Cross Validation. 5-Fold Cross Validation is used instead of 10-Fold due to the training time associated with each FCNN model. Each model is trained using 1000 epochs and a batch size of 5. The architectures tested were as follows:

- One Hidden Layer; hidden layer contains the same number of nodes as input parameters.
- One Hidden Layer; hidden layer contains the squared value of input parameters as the number of nodes.
- Two Hidden Layers; hidden layers each contain the same number of nodes as input parameters.
- Two Hidden Layers; hidden layers are arranged in a triangular structure with the nodes in the first layer equal to the number of input parameters, and remaining layers containing approximately half the number of nodes as the previous layer.
- Three Hidden Layers; hidden layers each contain the same number of nodes as input parameters.
- Three Hidden Layers; hidden layers are arranged in a triangular structure with the nodes in the first layer equal to the number of input parameters, and remaining layers containing approximately half the number of nodes as the previous layer.

The ReLU[49] activation function is used at each hidden layer as well as at the output layer for each network architecture. Dropout of 20% is also applied at each hidden layer, this helps to prevent the model from overfitting on training data by randomly assigning a value of 0 to the inputs to selected nodes. This forces other nodes to compensate for this loss and reduces the networks reliance on any particular set of nodes. The Adam optimizer[50] is also used for each architecture. Each FCNN is implemented using the Tensorflow[51] package. The performance of each FCNN model is reported in Tables 10 and 11 in the appendix.

Note on Deterministic Versus Non-Deterministic Models

Each of the Machine Learning algorithms used are deterministic. This means that each algorithm will arrive at the same model each time given particular training data. This is not the case for the FCNN, which is non-deterministic. The cost function associated with a FCNN is non-convex, meaning that there is no absolute best configuration of weights for the network. As such, during training the FCNN will converge to a set of locally optimal weights, of which there may be many. As well as this, because the weights associated with a FCNN are initialised randomly, each time a FCNN is trained using a particular architecture it will produce a new model with a unique weight configuration.

4.4 Results

The average Mean Absolute Error (MAE) of each algorithm as well as its standard error is reported. This is averaged over 10 folds of Cross Validation in the case of KNN, Decision Tree and Random Forest, and over 5 folds of Cross Validation in the case of the FCNN model. The optimal hyper-parameter configurations for each algorithm were found using a grid search (where possible). The results of this grid search can be found in Tables 10 and 11 in the Appendix.

Statement	Algorithm	Hyper-Parameters	Average MAE	Std. Error
Balance Sheet	Decision Tree	Max Depth = 31, Splitter = Best	35.11	2.60
Balance Sheet	k-Nearest Neighbour	K = 2, Weight = Distance , Metric = Euclidean	26.84	3.05
Balance Sheet	Random Forest	Max Tree Depth = 14, No. Estimators = 26	29.86	2.33
Balance Sheet	FCNN	Hidden Layers = 1, No. Nodes = 81	39.96	2.54
Cash Flow	Decision Tree	Max Depth = 6, Splitter = Best	35.72	2.18
Cash Flow	k-Nearest Neighbour	K = 24, Weight = Distance, Metric = Manhattan	48.39	3.83
Cash Flow	Random Forest	Max Tree Depth = 11, No. Estimators = 29	32.91	2.02
Cash Flow	FCNN	No. Hidden Layers = 1, No. Nodes per Layer = 64	46.01	3.02
Income Statement	Decision Tree	Max Depth = 8, Splitter = Best	28.95	2.77
Income Statement	k-Nearest Neighbour	K = 3, Weight = Distance, Metric = Manhattan	36.58	5.91
Income Statement	Random Forest	Max Tree Depth = 14, No. Estimators = 22	25.91	2.40
Income Statement	FCNN	No. Hidden Layers = 1, No. Nodes per Layer = 100	38.37	6.64
Merged Statements	Decision Tree	Max Depth = 17, Splitter = Random	30.24	3.53
Merged Statements	k-Nearest Neighbour	K = 2, Weight = Distance, Metric = Manhattan	25.12	1.82
Merged Statements	**Random Forest**	**Max Tree Depth = 15, No. Estimators = 28**	**24.26**	**2.19**
Merged Statements	FCNN	No. Hidden Layers = 1, No. Nodes per Layer = 676	32.26	2.82

Table 1: Raw Filings Price Prediction Results.

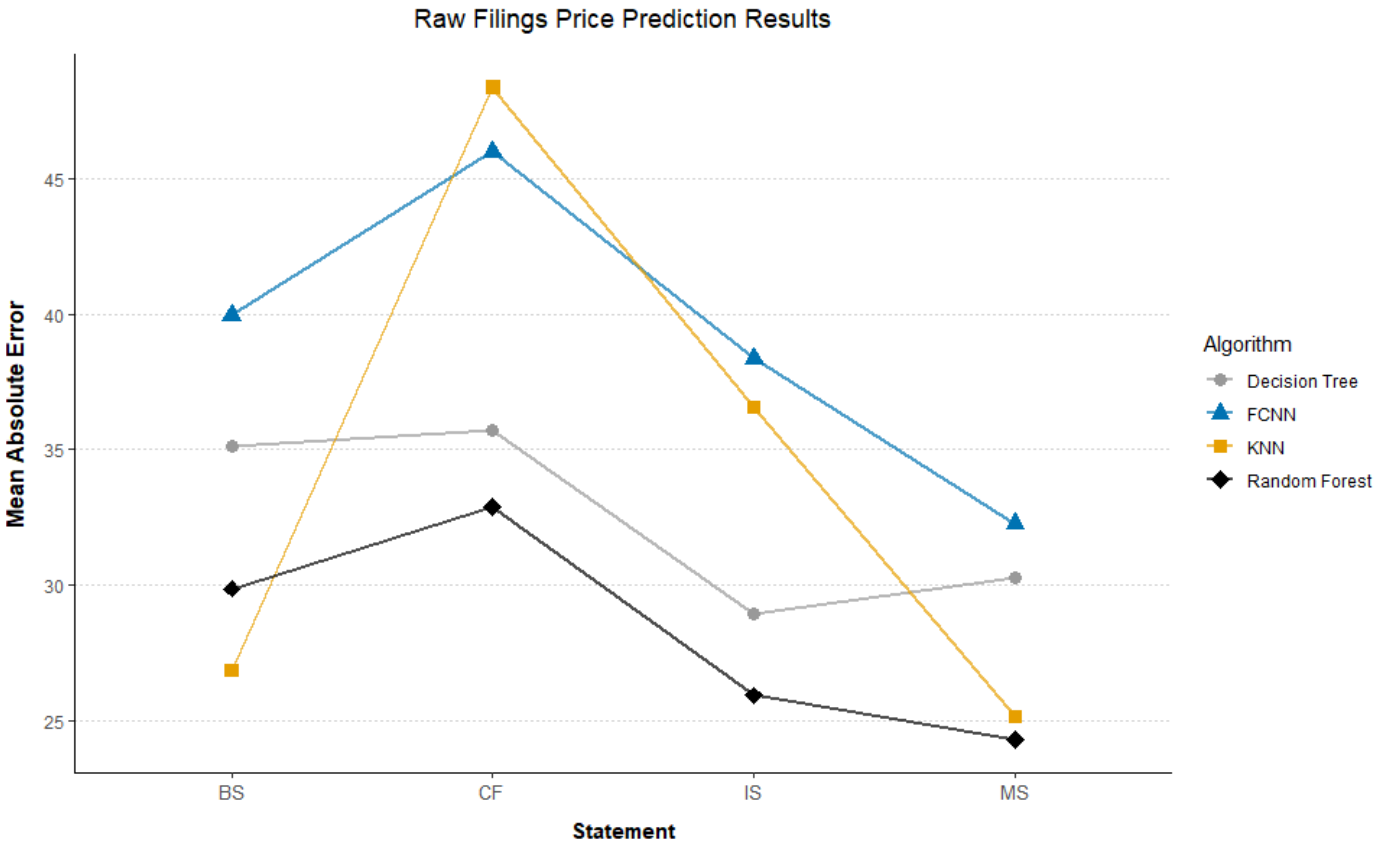

Figure 7: Raw Filings Price Prediction Results

The Random Forest model trained on the Merged Statements data shows the best performance of all models tested. This demonstrates the benefit of using Ensemble models for regression tasks, as the Random Forest is able to make use of 28 different Decision Trees and obtain its prediction by averaging the predictions of these 28 individual Decision Trees. The Decision Tree algorithm itself performs well relative to the other algorithms and achieves the 2^{nd} lowest MAE for two of the statements.

The KNN algorithm performs very well, producing the 2^{nd} lowest MAE of all models tested, and producing two of the top four performing models, along with the Random Forest algorithm. The optimal value of K is found to be 3 or less for three of the four KNN models built, indicating that most test instances are best described by neighbours immediately in their vicinity. Furthermore, the Manhattan distance metric is found to produce the best performing models for three of the four statements.

The FCNN performs poorly relative to the other algorithms, giving the greatest error for all but one statement, the Cash Flow statement. Given the much larger training time associated with the FCNN model, this performance is disappointing. The architecture which was found to be optimal for data from each statement consisted of a single hidden layer with a number of nodes equal to the square of the input parameters. This indicates that the algorithm required a substantial number of nodes to express the mapping from inputs to prices. Perhaps making use of greater numbers of nodes in deeper layers would have resulted in improved performance. However, the nature of the data, as indicated by the low value of K which results in good performance for KNN, may be such that too much noise in the data prevents a mapping being learned from all the data instances, and a better approach when producing predictions for test data instances would be to consider only data within a certain region around the test instance.

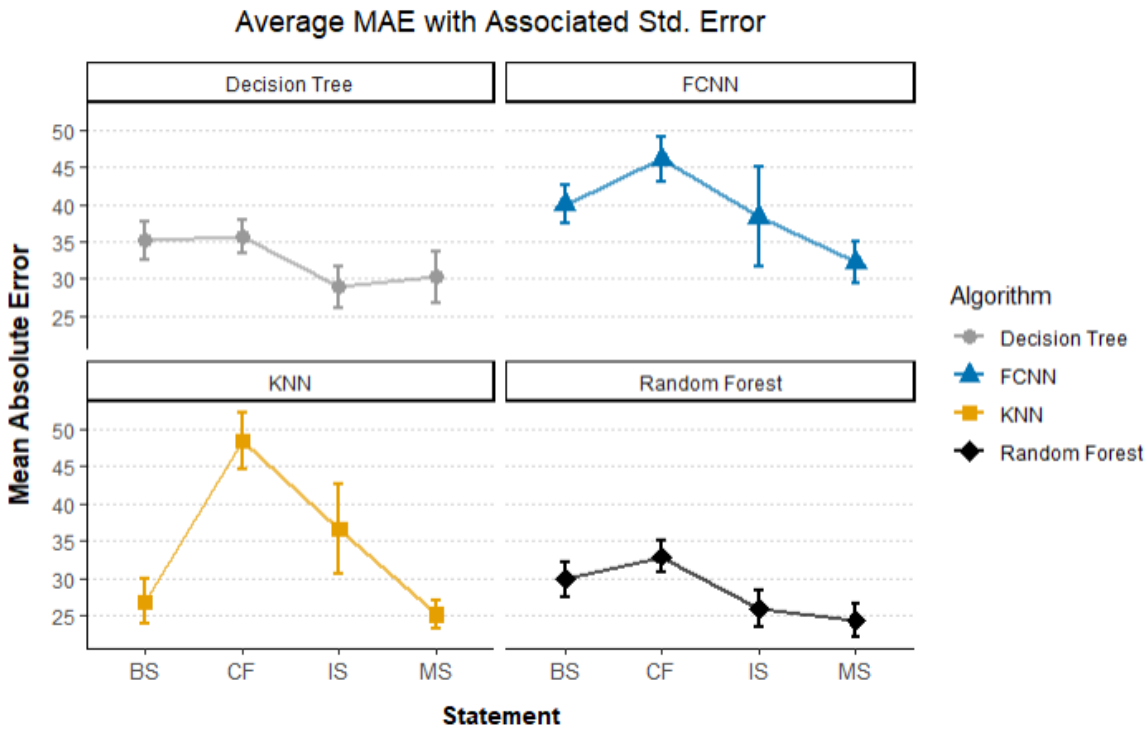

Figure 8: Average MAE with Associated Standard Error of the mean

The average MAE achieved by each algorithm along with the standard error of this MAE over all cross validation folds is displayed above. The standard errors for each of the algorithms in general were quite stable. Despite the FCNN being evaluated on 5-Fold Cross Validation instead of 10-Fold in the case of the Machine Learning models, there is not a large variation in the performance of the model given its non-deterministic

nature. The obvious exceptions being the Standard Error for the KNN and FCNN models trained on the Income Statement data.

Statement	**Average MAE Across All Algorithms**
Balance Sheet	32.94
Cash Flow	40.76
Income Statement	32.45
Merged Statements	**27.97**

Table 2: Average MAE Across All Algorithms

The Merged Statements dataset produces the models with the lowest MAE on average, despite each data instance having more than double the dimensions (26) of other statements. This demonstrates the importance of taking account of all statements in any analysis of a companies financial state. Despite this, a MAE of 24.26 still represents a considerable error and further underlines the difficulties in attempting to predict stock prices accurately. The statement which produced the worst price predictions on average was the Cash Flow statement. One possible reason for this could be the fact that each data instance from this statement had only 8 attributes, the fewest of any statement in this analysis.

5 Delta Filings

5.1 Introduction

This section will investigate two more of the research questions mentioned in the introduction. Namely:

1. Can the change in the state of a companies financial statements be used to predict whether the price of that companies shares has increased or decreased? And,

2. How accurately can this change in price be predicted?

In order to answer both of these questions, it was necessary to create two new datasets from the original data. One dataset containing the change in filings and the the corresponding price change, while the other the change in filings and corresponding price trend classification. To do this, the difference in filing parameters for consecutive years was calculated.

As an example, the attributes which make up a company's Balance Sheet in 2010 were taken away from the corresponding attributes in 2011. This difference in attributes from year to year is then used to make predictions of price trend and the magnitude of the change. Initially, 5000 instances were present in the original data. However, it is not possible to calculate the difference in parameters for the first of 10 filings for each company as the data is unavailable, i.e. it is not possible to find the change in Apple's Balance Sheet from 2009 to 2010 if there is no data for Apple's Balance Sheet from 2009. As such, it is only possible to calculate the change in filings 9 times for each companies 10 filings.

This reduces the size of the dataset to 4500 instances. Each instance represents the change in parameters from year to year as well as the corresponding change in price or price trend classification. Two datasets were then created from this 'Delta Filings' dataset. If the change in price between consecutive years was positive, the instance was classified as 1, and if the change in price was negative, the instance was classified as 0.

In total, as stated in Section 3.1, of the 4500 instances, 1462 instances were given the class label 0 (price decrease), and 3038 were given the class label 1 (price increase).

5.2 Trend Prediction - Description of Classification Algorithms Used

The classification algorithms used include the k-Nearest Neighbour Algorithm, the Support Vector Machine algorithm, the Decision Tree algorithm, the Random Forest Algorithm as well as a Fully Connected Neural Network (FCNN). Descriptions of these algorithms can be found in Section 2.2/2.3.

5.3 Methods

5.3.1 Model Training and Testing

Machine Learning Models

Stratified 10-Fold Cross Validation is used to compare the performance of each model. Stratified Cross Validation is a type of Cross Validation normally used for classification tasks.

The main difference between standard K-Fold Cross Validation and Stratified K-Fold Cross Validation is that Stratified K-Fold Cross Validation aims to ensure that each fold of the data is representative of the dataset as a whole. In a binary classification task, if there are a greater number of data instances of one class over another class, and if normal cross validation is applied in the usual way, some folds may contain many more examples of one class over another. Other folds may not even contain examples of both classes. As such, this makes it much more difficult for the model to learn under-represented classes and causes the accuracy of the model to fluctuate. By using stratified cross validation, each fold of the data is made by preserving the percentage of samples for each class[40]. In the case of this task, there are 1462 instances of Class 0 (32.5% of the overall data) and 3038 instances of Class 1 (67.5% of the overall data). Each fold, in this case, will be made up of approximately 32.5% class 0 and 67.5% class 1.

Each Machine Learning model is evaluated using 10-Fold Stratified Cross Validation. All models are trained on each of the 10 training folds and tested on the respective test folds. Classification Accuracy is the metric used to asses the performance of model predictions.

The kNN, Decision Tree and Random Forest algorithms are implemented using Scikit-Learn[48]. Optimal hyper-parameters for each algorithm are obtained via a grid search, with the hyper-parameters which give the lowest MAE chosen. The results of hyper-parameter testing are reported in Tables 12 and 13 found in the Appendix.

Deep Learning Models

The FCNN models tested have the same architectures as described in Section 4.3.1 and were all trained using 1000 epochs and a batch size of 5. The Adam optimizer[50] was used for each architecture. The ReLU[49] activation function is used at each hidden layer, and the sigmoid activation function used at the output layer for each network architecture. Dropout of 20% is also applied at each hidden layer, as described in Section 4.3.1. Each architecture is evaluated using 5-Fold Cross Validation. The performance of each FCNN model is also reported in Tables 12 and 13 found in the Appendix.

5.4 Results

The mean accuracy of each algorithm as well as its standard error is reported. This is averaged over 10 folds of Stratified Cross Validation in the case of KNN, Decision Tree and Random Forest, and over 5 folds of Stratified Cross Validation in the case of the FCNN model. The optimal hyper-parameter configurations for each algorithm were found using a grid search (where applicable). The results of this grid search can be found in Tables 12 and 13 in the Appendix.

Statement	Algorithm	Hyper-Parameters	Mean Accuracy	Std. Error
Balance Sheet	SVM	C = 100, kernel = Poly, degree 3	0.675	0.001
Balance Sheet	Decision Tree	Max Depth = 5, Splitter = best, Criterion = Gini Index	0.680	0.019
Balance Sheet	k-Nearest Neighbour	K = 28, Weight = Distance , Metric = Manhattan	0.694	0.008
Balance Sheet	Random Forest	Max Tree Depth = 9, No. Estimators = 25	0.706	0.018
Balance Sheet	FCNN	No. Hidden Layers = 1, No. Nodes per Layer = 81	0.677	0.004
Cash Flow	SVM	C = 10, kernel = polynomial, degree 3	0.678	0.008
Cash Flow	Decision Tree	Max Depth = 5, Splitter = random, Criterion = Entropy	0.676	0.004
Cash Flow	k-Nearest Neighbour	K = 54, Weight = Distance, Metric = Manhattan	0.680	0.007
Cash Flow	Random Forest	Max Tree Depth = 7, No. Estimators = 21	0.685	0.014
Cash Flow	FCNN	No. Hidden Layers = 1, No. Nodes per Layer = 8	0.677	0.005
Income Statement	SVM	C = 10000, kernel = polynomial, degree 3	0.680	0.007
Income Statement	Decision Tree	Max Depth = 4, Splitter = Best, Criterion = Gini Index	0.688	0.015
Income Statement	k-Nearest Neighbour	K = 90, Weight = Distance, Metric = Euclidean	0.700	0.018
Income Statement	Random Forest	Max Tree Depth = 6, No. Estimators = 28	0.699	0.015
Income Statement	FCNN	No. Hidden Layers = 1, No. Nodes per Layer = 10	0.682	0.004
Merged Statements	SVM	C = 10000, kernel = polynomial, degree 3	0.682	0.008
Merged Statements	**Decision Tree**	**Max Depth = 2, Splitter = Best, Criterion = Gini Index**	**0.7104**	**0.010**
Merged Statements	k-Nearest Neighbour	K = 26, Weight = Uniform, Metric = Manhattan	0.707	0.016
Merged Statements	Random Forest	Max Tree Depth = 4, No. Estimators = 14	0.7102	0.012
Merged Statements	FCNN	No. Hidden Layers = 1, No. Nodes per Layer = 676	0.688	0.006

Table 3: Price Trend Classification Results.

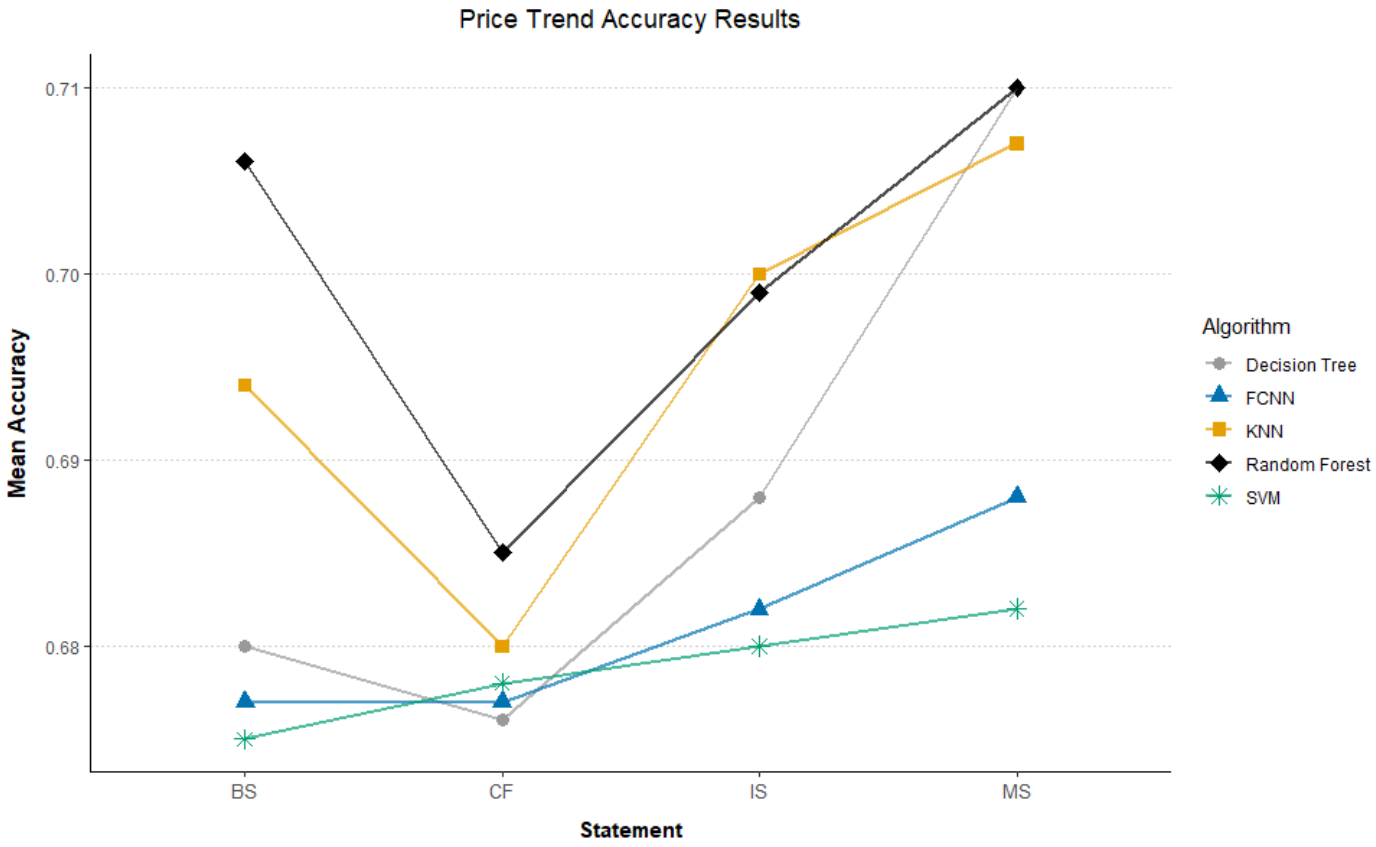

Figure 9: Price Trend Accuracy Results

The Decision Tree algorithm trained on the Merged Statements data achieves the highest mean accuracy of all algorithms tested, marginally outperforming the Random Forest algorithm. This is unsurprising as the Decision Tree is capable of dealing with complex non-linear data. The performance of the Random Forest algorithm once again underlines the effectiveness of using ensemble methods for a variety of tasks.

The KNN algorithm also performs consistently well relative to the other algorithms, finishing in the top 2 performing algorithms for three different statements and producing the 3^{rd} most accurate model overall. The optimal value of K for each statement is quite varied, however the Manhattan distance metric is once again found to produce the best performing models for three of the four statements.

The FCNN model underperforms relative to other algorithms once again. The architecture utilising one hidden layer consisting of a number of nodes equal to the square of the input parameters was once again

found to be the best performing architecture for the FCNN for 2 of the 4 statements.

The SVM woefully underperforms the other algorithms tested, producing the least accurate predictions for three of the four statements. The optimal kernel used in each case was a polynomial kernel of degree 3. A possible reason for this poor performance could be that the SVM requires a more complex kernel mapping than a polynomial of degree 3. The SVM algorithm is capable of dealing with non-linear data, however, the SVM is restricted in making predictions on data by the nature of its kernel structure, which causes the model to make biased assumptions about the way the data is distributed. In this case, the SVM assumes the data can be separated using a polynomial kernel of degree 3. Given the noisy nature of stock market data, this is an unlikely assumption. A more complex kernel mapping may be required to improve its accuracy.

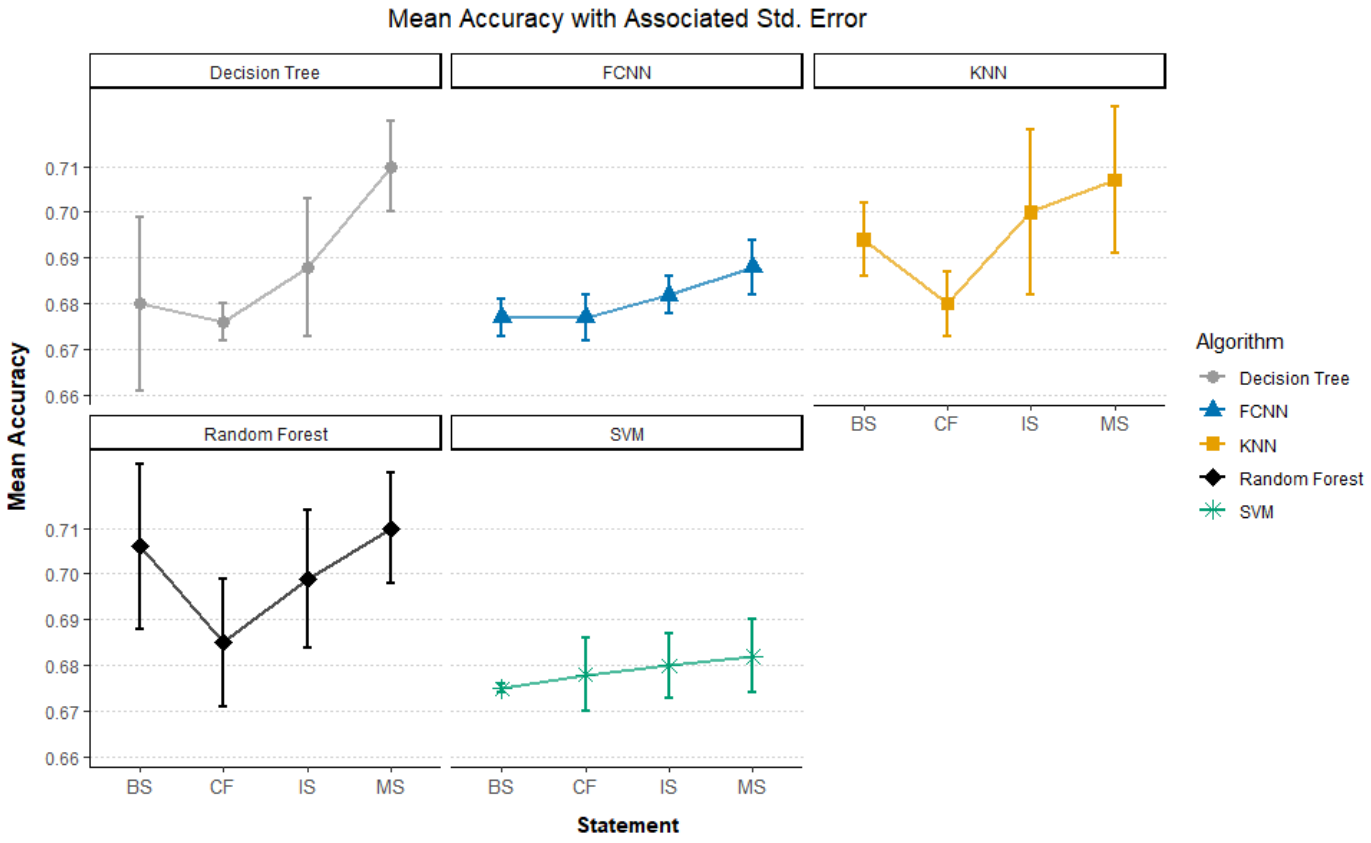

Figure 10: Mean Accuracy with Associated Standard Error

The mean accuracy achieved by each algorithm along with the standard error of this accuracy over all cross validation folds is displayed above. The algorithm which consistently produces the largest standard

error is the Random Forest algorithm, followed by the Decision Tree algorithm. The greater standard error associated with the Random Forest model could be attributed to the fact that its prediction is averaged over 14 estimators, each with their own prediction, which may vary greatly.

Statement	**Mean Accuracy Across All Algorithms**
Balance Sheet	0.686
Cash Flow	0.679
Income Statement	0.690
Merged Statements	**0.700**

Table 4: Mean Accuracy Across All Algorithms

The Merged Statements data once again produces the most accurate predictions, with an average trend prediction accuracy over all algorithms of 70%. The average difference in accuracy between algorithms trained on the worst performing data (Cash Flow statement) and the best performing was 2.1%.

Despite an average accuracy of 70% associated with the Merged Statements data given the difficulties in dealing with the noise present in stock market data, these results must be interpreted with caution. One issue associated with the data is the ratio of positive to negative classes. Approximately 67% of the data instances are Class 1 instances (price increase), with the remaining 33% of the data instances of Class 0 (price decrease). Therefore, if the model always predicts Class 1 for each instance as there are more Class 1 instances than Class 0 (more price increases than decreases), it will achieves an accuracy of approximately 67%. As such, other performance metrics such as precision, recall or F1 score may provide a better idea of the true performance of different models.

5.5 Price Change Prediction - Regression Algorithms Used

This section aims to investigate how accurately the magnitude of the change in a company's stock price can be predicted from the corresponding change in its financial statements, and which statement(s) are good predictors of this change. The regression algorithms used include the k-Nearest Neighbour Algorithm, the Decision Tree algorithm, the Random Forest Algorithm as well as a Fully Connected Neural Network (FCNN). Descriptions of these algorithms can be found in Section 2.2/2.3.

5.6 Methods

Machine Learning Models

Each Machine Learning model is evaluated using 10-Fold Cross Validation. All models are trained on each of the 10 training folds and tested on the respective test folds. Mean Absolute Error (MAE) is the metric used to asses the accuracy of model predictions. MAE is defined in Section 4.3. The average MAE and associated standard error of these results over all 10 folds are reported.

The kNN, Decision Tree and Random Forest algorithms are implemented using Scikit-Learn[48]. Optimal hyper-parameters for each algorithm are obtained via a grid search, with the hyper-parameters which produce models with the lowest MAE chosen. The results of this hyper-parameter testing are given in Tables 14 and 15 found in the Appendix.

Deep Learning Models

The FCNN models tested have the same architectures as described in Section 4.3.1 and were all trained using 500 epochs and a batch size of 5. The 'tanh' activation function is used at the output layer for each architecture, with the ReLU[49] activation function used for each hidden layer. The tanh activation function allows the FCNN to predict negative as well as positive price changes. A smaller number of epochs were used in this case as the tanh activation function used at the output layer causes the models to converge to a weight configuration quickly, requiring less training. Dropout of 20% is applied at each hidden layer, as described in Section 4.3.1. The Adam optimizer[50] is once again used for each architecture. Each architecture is evaluated using 5-Fold Cross Validation. The performance of each FCNN model is reported in Tables 14 and 15 found in the Appendix.

5.7 Results

The average Mean Absolute Error (MAE) of each algorithm as well as its standard error is reported. This is averaged over 10 folds of Cross Validation in the case of KNN, Decision Tree and Random Forest, and over 5 folds of Cross Validation in the case of the FCNN model. The optimal hyper-parameter configurations for each algorithm were found using a grid search (where possible). In cases where two or more FCNN architectures gave identical results, the architecture with the simplest structure is chosen. The results of this grid search can be found in Tables 14 and 15 in the Appendix.

Statement	Algorithm	Hyper-Parameters	Average MAE	Std. Error
Balance Sheet	Decision Tree	Max Depth = 10, Splitter = random	16.94	2.09
Balance Sheet	k-Nearest Neighbour	K = 99, Weight = Distance , Metric = Manhattan	15.94	1.08
Balance Sheet	Random Forest	Max Tree Depth = 5, No. Estimators = 27	16.08	1.95
Balance Sheet	FCNN	No. Hidden Layers = 1, No. Nodes per Layer = 81	15.91	1.07
Cash Flow	Decision Tree	Max Depth = 1, Splitter = random	16.36	2.16
Cash Flow	k-Nearest Neighbour	K = 99, Weight = Distance, Metric = Manhattan	16.24	2.28
Cash Flow	Random Forest	Max Tree Depth = 1, No. Estimators = 8	16.67	1.96
Cash Flow	FCNN	No. Hidden Layers = 2, No. Nodes per Layer = 8, 4	15.91	1.43
Income Statement	Decision Tree	Max Depth = 6, Splitter = random	16.68	1.81
Income Statement	k-Nearest Neighbour	K = 99, Weight = Distance, Metric = Manhattan	15.49	1.60
Income Statement	Random Forest	Max Tree Depth = 12, No. Estimators = 14	16.24	1.67
Income Statement	FCNN	No. Hidden Layers = 1, No. Nodes per Layer = 10	15.90	1.03
Merged Statements	Decision Tree	Max Depth = 4, Splitter = random	16.84	2.61
Merged Statements	**k-Nearest Neighbour**	**K = 99, Weight = Distance, Metric = Manhattan**	**15.46**	**2.20**
Merged Statements	Random Forest	Max Tree Depth = 6, No. Estimators = 25	15.69	1.76
Merged Statements	FCNN	No. Hidden Layers = 1, No. Nodes per Layer = 26	15.90	1.60

Table 5: Price Change Magnitude Regression results.

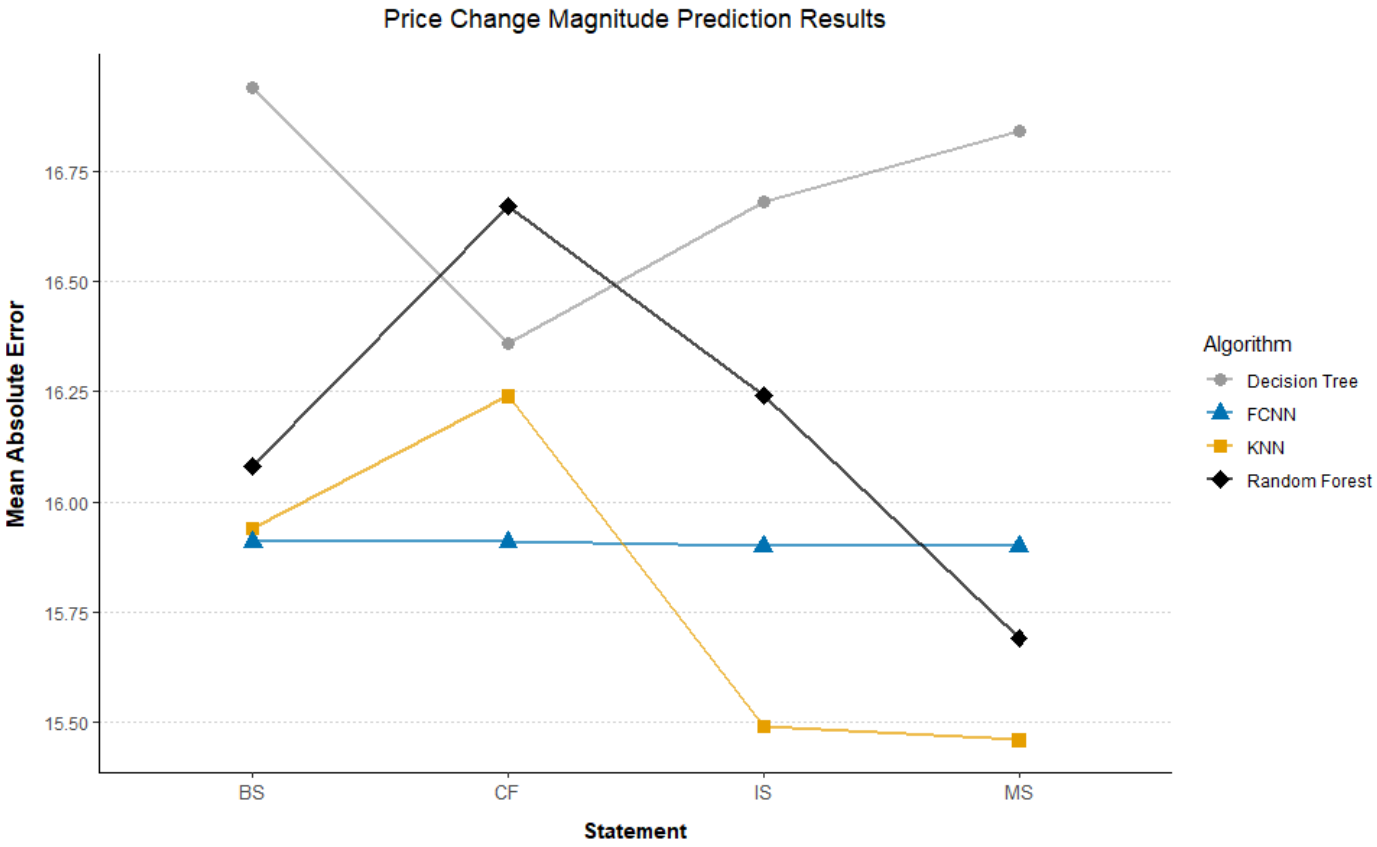

Figure 11: Price Change Magnitude Prediction Results

The KNN algorithm trained on the Merged Statements data achieves the lowest average MAE of all algorithms tested. The same algorithm trained on the Income Statement data also achieves the 2^{nd} lowest average MAE of all algorithms tested. The optimal value of K for each of the statements was found to be 99, suggesting the optimal strategy found by the algorithm involved averaging predictions over a large number of neighbouring data instances. Once again, the Manhattan distance metric was found to produce the best performing KNN models for all statements.

As was the case in previous tasks, the Random Forest algorithm also produces a top performing model, the 3^{rd} best overall with an average MAE of 15.69. Despite this, the performance of the Random Forest is not as consistently high as on previous tasks, producing the worst or second worst model for three of the four statements.

The Decision Tree algorithm unusuallly under performs

The FCNN performs much better than in previous tasks, producing consistently good predictions relative to the other algorithms. The performance of the FCNN for each statement is almost identical despite the varied architectures and data used. This may be related to the tanh activation being used at the output layer, causing early convergence of network weights.

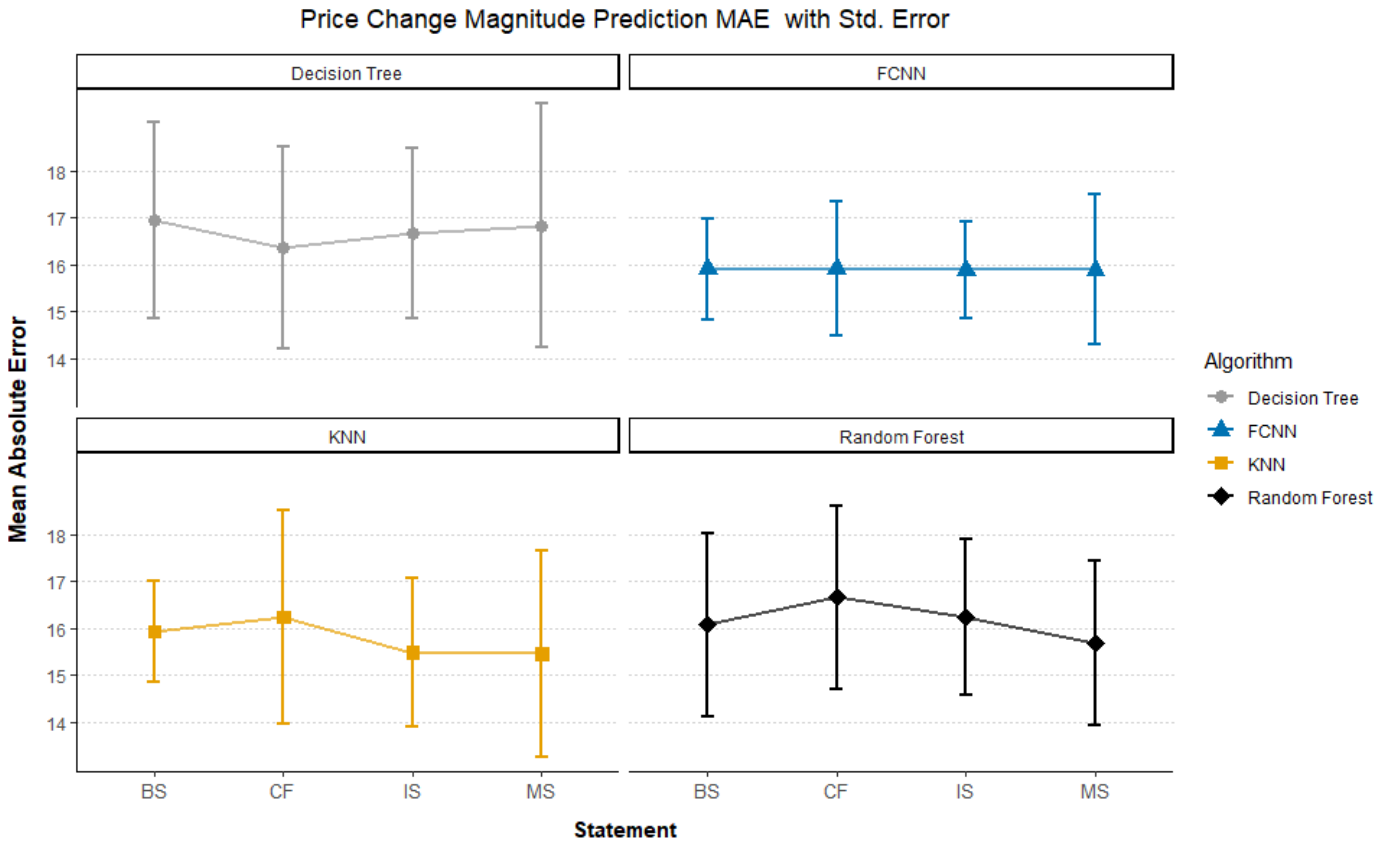

Figure 12: Price Change Magnitude Prediction MAE with Std. Error

The average MAE achieved by each algorithm along with the standard error of this MAE over all cross validation folds is displayed above. The standard errors associated with the Decision Tree, KNN and Random Forest algorithms are approximately equivalent, with the standard errors associated with FCNN models found to be consistently smaller relative to the other models.

Statement	Average MAE Across All Algorithms
Balance Sheet	16.22
Cash Flow	16.30
Income Statement	16.08
Merged Statements	**15.97**

Table 6: Average MAE Across All Algorithms

The Merged Statements data once again produces the best performing models on average, with the Cash Flow statement once again producing the worst performing models. Again a possible reason for the poor performance of models trained on the Cash Flow statement data could be the low number of attributes present in the data relative to the other algorithms. Indeed, if statements are ranked according to the number of attributes associated with each data instance, the ranking is exactly the same as the ranking based on average MAE across all algoritms. This suggests that the more information available for each statement, the greater the predictive power of models trained on that statements data.

6 Knowledge Graph and Embeddings

This section aims to investigate whether it is possible to improve on the performance achieved in predicting price values and price changes in previous sections by taking into account other non-filing data relating to individual companies, such as the industry and sector within which the company operates, in the form of a Knowledge Graph. The central question being whether this added non-numerical information can be used via Knowledge Graph embeddings to improve the accuracy of predictions from using only filing data.

6.1 Knowledge Graph

The Knowledge Graph used in this analysis was created manually using information available on the Seeking Alpha website[52]. Seeking Alpha provides both the Economic Sector and Industry each company operates in. As well as this, the website provides a 'Company Profile' section which gives an overview of the products and services each company provides. This information is utilised in the form of a Knowledge Graph.

To recap, a Knowledge Graph is a network connecting related objects and concepts. The graph is defined as a set of triples (s, r, t) where s, t $\in$ E, r $\in$ R. E represents the set of all entities in the graph, and R represents the set of relations between entities. Examples of entities include companies, concepts, sectors and industries and relation types include industry member, sector member and associated concept. Examples of concepts include Agriculture, Food, Software etc.

The graph was constructed in the following way; each sector is connected to the industries within that sector, each industry is then connected to all of the companies within that industry. Following this, each company is then connected to the concepts directly associated to the products and services each company provides. In total, there are 1521 triples used in creating the Knowledge Graph. A visualisation of the graph was created using the Network X Python library[54].

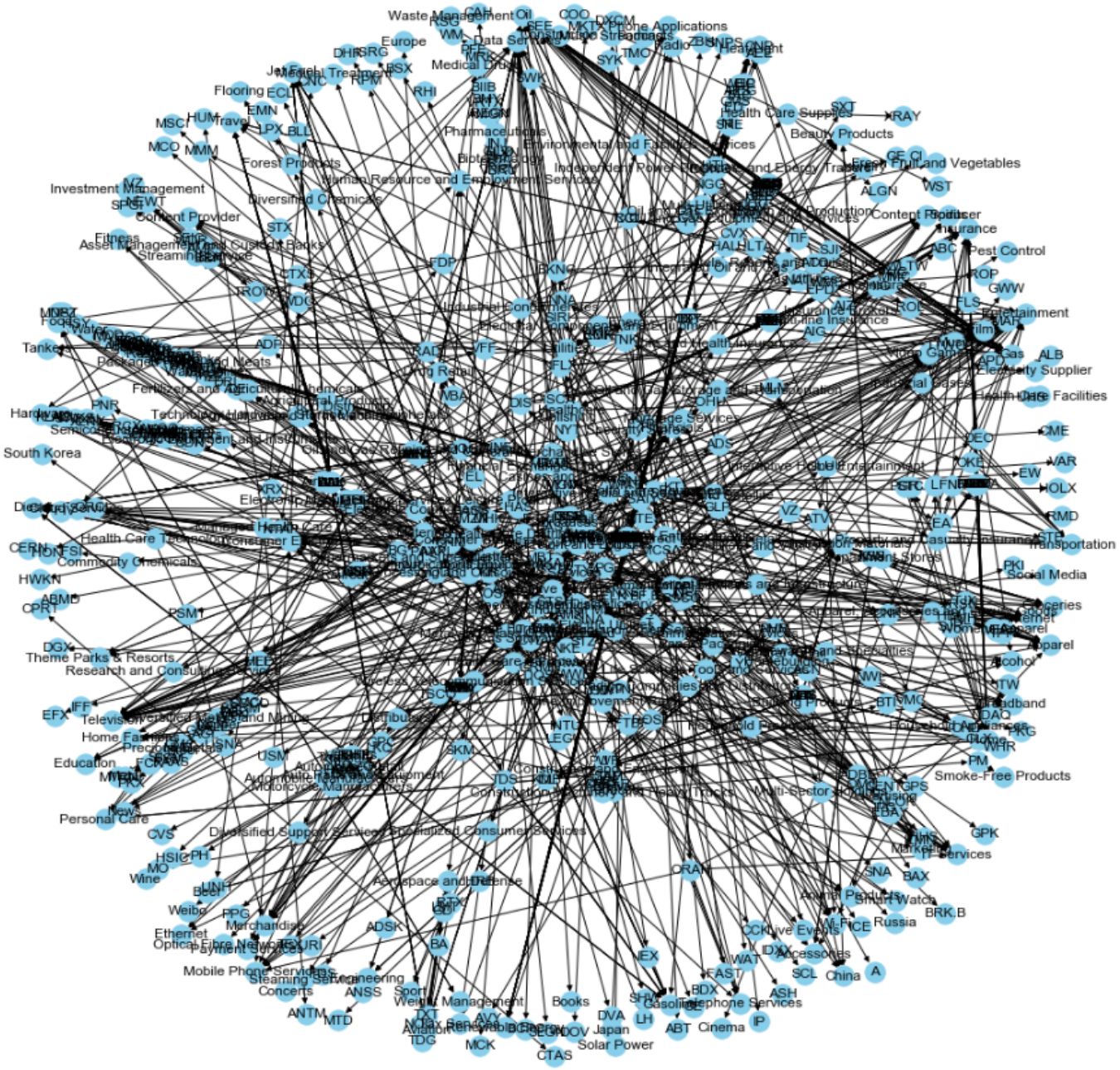

Figure 13: Manually Constructed Knowledge Graph

6.2 Knowledge Graph Embeddings

The Node2Vec algorithm is used to obtain embeddings for each company node. However, in order to provide a comprehensive description of the Node2Vec algorithm it is necessary to first explain the Skip-Gram model.

6.2.1 Skip-Gram Model

The Skip-Gram model is an important model used in Natural Language Processing. The Skip-gram model is based on the distributional hypothesis, which states that words in similar contexts tend to have similar meanings (and by extension similar vector representations). It is a member of the Word2Vec class of models which seek to represent words from a corpus as vectors in n-dimensional space. One hot vector encodings are made for each unique word contained in the corpus, with a unique index assigned to each word. The length of each encoding is the same as the number of unique words in the corpus. As an example, given a corpus containing 8 unique words, with the word 'the' assigned the index 3, the one hot encoding of the word 'the', assuming indexing from 1, would be given by:

$$3 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{1}$$

The embeddings for each word are learned via training on context target word pairs. This is best explained using the visualisation in Figure 14 below.

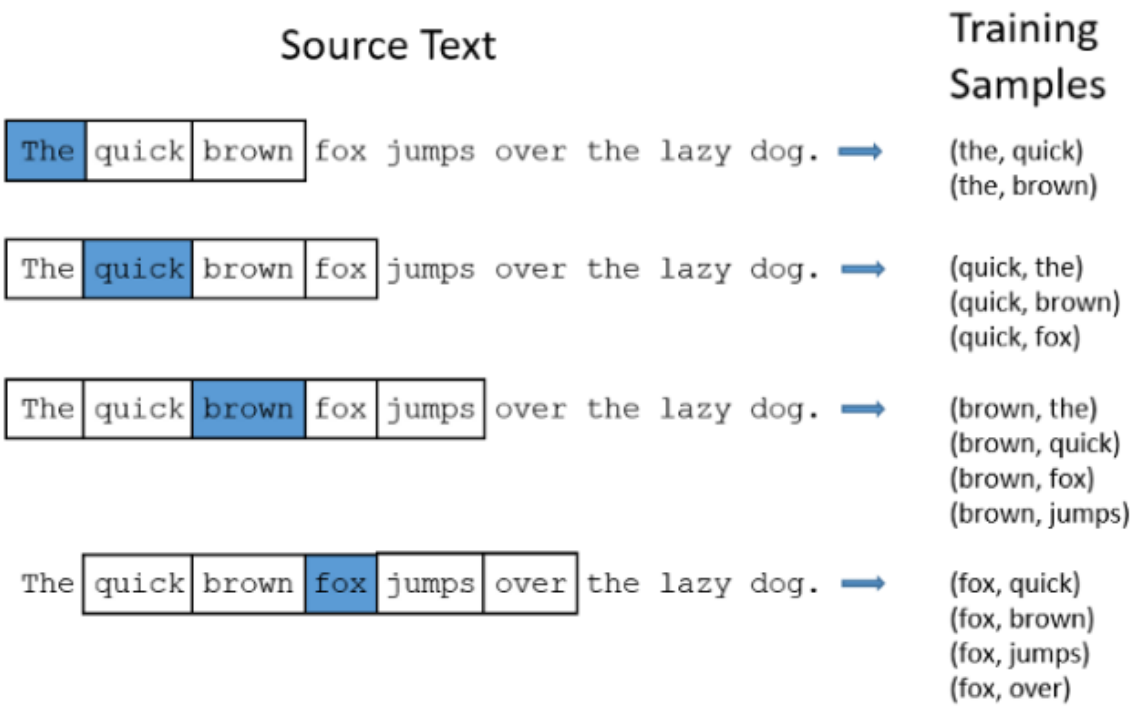

Figure 14: Context-Target pair training samples for Skip-Gram model[55]

Using the context-target word pairs, the embeddings for each word are learned via the training process pictured below. The following piece offers a succinct description of this training process and is taken from [55]:

There is no activation function on the hidden layer neurons, but the output neurons use softmax. When training this network on word pairs, the input is a one-hot vector representing the input word and the training output is also a one-hot vector representing the output word. But when you evaluate the trained network on an input word, the output vector will actually be a probability distribution (i.e., a bunch of floating point values, not a one-hot vector)".

This probability distribution is achieved using the softmax function at the output layer. The entry at each index of the output vector is the probability that the word corresponding to that index is a 'nearby' word (in the case of Figure 14 above, within the 2 word window) to the input word. For words which occur frequently together, this probability will be high, and for words which do not often occur together this probability will be low. Hence the output one-hot vector will be the vector with 1 in the index with the highest probability.

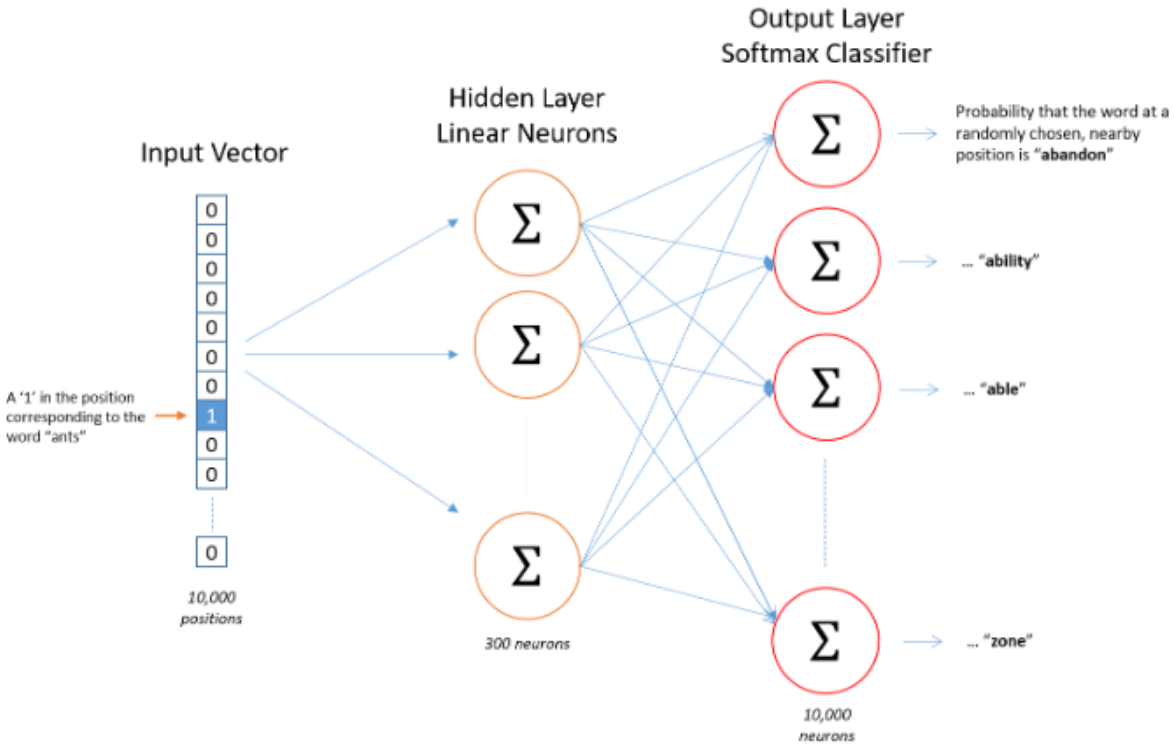

Figure 15: Skip-gram Model with 10,000 word vocabulary and 300 dimensional embeddings [55]

The word embeddings are learned in the following way. Given that 300 neurons were used in the hidden layer, this is the dimension size of the embedding. This hidden layer is represented by a weight matrix with 10,000 rows and 300 columns, with each row of the weight matrix corresponding to the 300 dimensional vector representation of each word.

This is the case as when each one-hot vector enters the network and is multiplied by the hidden layer weight matrix, because every entry is 0 except for one entry with a value of 1 at the corresponding unique word index, this index in the one-hot vector is the row index corresponding to the vector embedding for the word. This is best illustrated by Figure 16 below.

$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

Figure 16: One-Hot Vector multiplied by hidden layer weight matrix producing word embedding [55]

6.2.2 Node2Vec

The Node2Vec[17] algorithm is used to obtain embeddings for nodes for each company from the above Knowledge Graph. Node2Vec is a semi-supervised algorithm which provides a framework for learning continuous feature representations of nodes (or edges) in networks. As such, it is ideally suited to calculating embeddings of entities from a Knowledge Graph. The algorithm learns a mapping from nodes to a *"low-dimensional space of features that maximizes the likelihood of preserving network neighborhoods of nodes"*[17], this means nodes located in specific neighbourhoods have similar embeddings.

The algorithm works by extending the Skip-Gram architecture to networks. It does this by simulating random walks of fixed length through the network. These random walks then act as 'training data' for a neural network which learns the embeddings associated with each node based on the nodes which are 'nearest' to it in the same way as the Skip-Gram model does for word vectors. Experiments conducted by the authors[17] demonstrate that Node2Vec outperforms state-of-the-art methods by up to 26.7% on multi-label classification and up to 12.6% on link prediction.

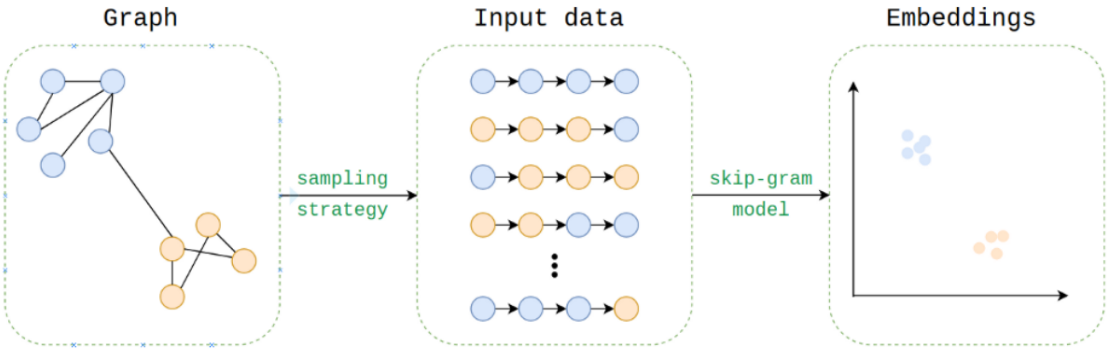

Figure 17: Node2Vec Embedding Process[56]

As can be seen in Figure 17 above, Node2Vec extends the Skip-Gram model by treating each random walk in the same way as the Skip-Gram model treats a sentence, and individual nodes in the same way as the

Skip-Gram model treats individual words.

Figure 18 is a 2D plot of the 5 dimensional embeddings obtained for each company node via Node2Vec using tSNE. tSNE[57] is a nonlinear dimensionality reduction technique used for embedding high-dimensional data in a low-dimensional space of two or three dimensions for visualization.

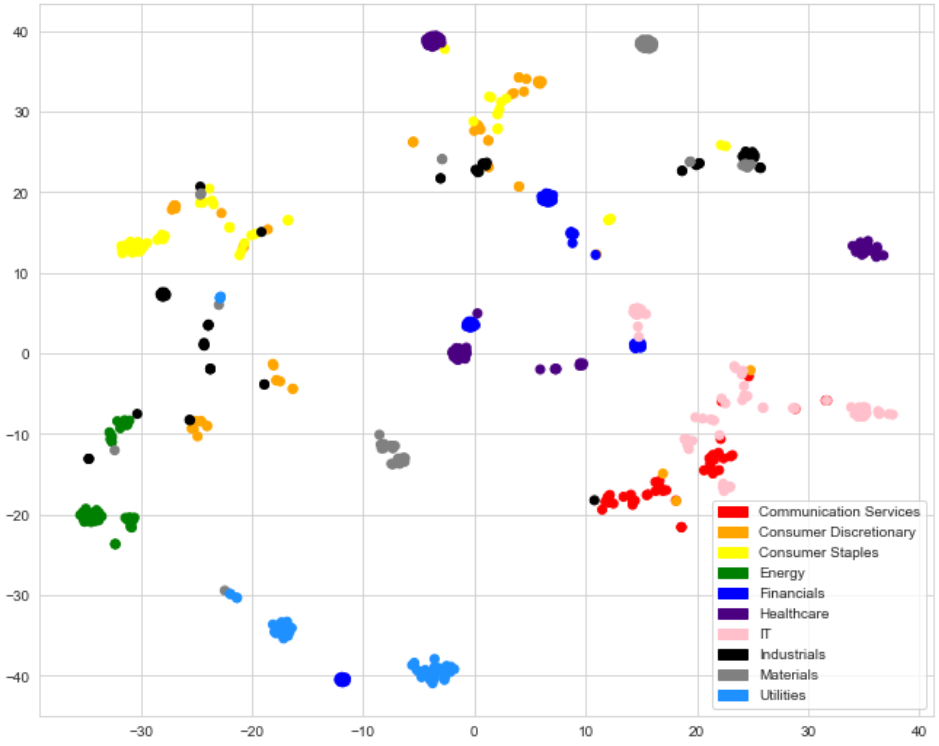

Figure 18: 2D plot of 5 Dimensional Company Embeddings using tSNE

6.2.3 Limitations

One of the limitations of using this Knowledge Graph as it is defined above is the fact that companies may change the products and services they provide over time. This Knowledge Graph is constant for each of the past 10 years, which does not reflect how individual companies have changed the products and services they provide over the past 10 years. In a sense, the embedding used to represent each company should be different as time goes by, however, in this analysis the embedding is exactly the same for each of the past 10 years.

6.3 Results

Embeddings associated with each company were added to each data instance as additional attributes. The idea being that this added information will improve the model by encoding additional information associated with each company. This may help with situations where poor financial performance by a company throughout a trading year is not reflected in their share price at the end of the year because of the industry within which the company operates, a situation which was common among IT companies during the Dot-Com bubble from 1997 - 2000.

Equally so, some companies may have a stigma attached to them, which results in good financial performance throughout the trading year not being reflected in that company's share price at the end of the year. It is hoped that by training models where the data has been augmented with these company embeddings, the models will be better able to deal with discrepancies in the data arising from situations such as those mentioned above.

Price Prediction With Raw Filing Data

The best performing algorithm for predicting company stock prices from raw financial statements was found to be the Random Forest algorithm. This model was trained on the Merged Statements data for this task. The effect on model performance of adding an embedding associated with each company will be assessed using embeddings of dimension 5, 7 and 9. Embeddings are trained via the Node2Vec algorithm using 500 Random Walks each of length 20. The performance of the algorithm will be averaged over 10 folds of cross validation. Optimal hyper-parameters are found using the same grid search used in Section 4.

Statement & Algorithm	Embedding Dimension	Average MAE	Difference	% Improvement
Merged Statements, Random Forest	0	24.26	NA	NA
Merged Statements, Random Forest	5	22.19	-2.07	8.5%
Merged Statements, Random Forest	7	21.95	-2.31	9.5%
Merged Statements, Random Forest	9	21.46	-2.80	11.5%

Table 7: Changes in MAE of Price Predictions due to Company Embeddings

Analysis of Results

The use of company embeddings in this task improves the performance of the Random Forest algorithm substantially. An 8.5% improvement is observed on the raw filing data using an embedding of dimension 5, rising to an 11.5% improvement using an embedding of dimension 9. Using embeddings of dimension greater than 9 may improve the performance of the model further.

Trend Prediction

The best performing algorithm for predicting the price trend from the data was found to be the Decision Tree algorithm. This model was also trained on the Merged Statements data for this task. The effect on model performance of adding an embedding associated with each company will be assessed using the same embeddings as above. The performance of the algorithm will be averaged over 10 folds of cross validation. Optimal hyper-parameters are found using a grid search.

Statement & Algorithm	Embedding Dimension	Mean Accuracy	Difference	% Improvement
Merged Statements, Decision Tree	0	0.710	NA	NA
Merged Statements, Decision Tree	5	0.708	-0.002	- 0.3%
Merged Statements, Decision Tree	7	0.705	-0.005	-0.7%
Merged Statements, Decision Tree	9	0.706	-0.004	-0.6%

Table 8: Changes in Trend Prediction Accuracy due to Company Embeddings

Analysis of Results

The company embeddings in this scenario fail to improve the accuracy of predictions over the accuracy obtained from using only statement data. However, this disimprovement is less than 1% in each case, despite the increase in dimensions of each data instance. Furthermore, as stated in Section 5.4, accuracy may not be the best metric to use to assess the performance of models for this data. It is possible that these embeddings would improve performance metrics such as precision, recall or F1 score.

Price Change Magnitude Prediction

The best performing algorithm for predicting the magnitude of price changes from the data was found to be the k-Nearest Neighbour algorithm. This model was also trained on the Merged Statements data for this task. The effect on model performance of adding an embedding associated with each company will be assessed using the same embeddings as above. The performance of the algorithm will be averaged over 10 folds of cross validation. Optimal hyper-parameters are found using a grid search

Statement & Algorithm	Embedding Dimension	Average MAE	Difference	% Improvement
Merged Statements, KNN	0	15.46	NA	NA
Merged Statements, KNN	5	15.50	+0.04	-0.26%
Merged Statements, KNN	7	15.39	-0.07	0.45%
Merged Statements, KNN	9	15.19	-0.27	1.75%

Table 9: Changes in MAE of Price Change Magnitude predictions due to Company Embeddings

Analysis of Results

Using company embeddings of dimensions 7 and 9 improved the performance of the KNN model over using only the raw statement data. A slight disimprovement of 0.26% compared with the raw statement data is noted when using the embedding of dimension 5. As was the case for the price prediction with raw filing data, the performance of the model seems to improve as the dimensions of the embeddings get larger.

Summary of Results

Of the nine tests conducted using company embeddings, 5 of these tests resulted in improved model performance. In the case of price prediction from raw statement data and the prediction of price change magnitudes, an increase in the dimension size of the embedding resulted in continually improving model performance. The embeddings may continue to improve the performance of models up to a certain dimension size, but the noise in the data will eventually limit possible improvements in the models performance.

In the case of the price trend prediction task, the embeddings failed to improve the accuracy of the model. However, as mentioned previously, due to the imbalanced nature of the data used in this task, using accuracy as a measure of model performance may not be the most appropriate measure of performance for this task. Other metrics such as precision, recall and F1 score may provide a better measure by which to assess the performance of models trained and tested on this data. It is possible, given the improvements noted for the other two tasks, that the company embeddings may improve the performance of models using a different measure of evaluation.

7 Conclusions

7.1 Contributions

The contributions of this work will now be discussed in relation to the research aims stated in Section 1.3.

1. *How accurately can a company's financial statements predict that company's share price? And which statements in particular are good predictors of the price?*

   Four algorithms were tested in investigating this research question. These are the Decision Tree, Random Forest, Fully Connected Neural Network and k-Nearest Neighbour algorithms. The Random Forest algorithm trained on the Merged Statements dataset made up of the combined data from each company's Balance Sheet, Income Statement and Cash Flow statement results in the lowest observed error in price predictions of any model tested. This performance is achieved without any prior feature engineering being applied to the data used. Despite the good performance of the Random Forest algorithm relative to other algorithms tested, the average Mean Absolute Error associated with this models predictions is approximately 24. This error is still significant and once again demonstrates the difficulty in predicting stock prices due to the inherent noise present in the data.

   The Merged Statements dataset, made up of the combined data from each company's Balance Sheet, Income Statement and Cash Flow statement is found to produce the most optimal predictions on average. This underlines the importance of considering each statement contained in a company's annual 10-K filing when assessing whether or not to invest in shares of a particular company.

2. *Can the change in the state of a company's financial statements be used to predict whether the price of that company's shares has increased or decreased? And how accurately can this change in price be predicted?*

   Five Algorithms were tested in assessing how accurately the trend upward or downward in the price of a company's shares based off the changes in their financial statements could be predicted. These include the four algorithms mentioned above, as well as the Support Vector Machine algorithm. The best performing algorithm in predicting this trend was found to be the Decision Tree algorithm trained on the Merged Statements data, very marginally beating the performance of the Random Forest algorithm. The Decision Tree achieved an accuracy of 71% in predicting price trends based off the change in the Merged Statements data, which once again proved to produce on average the most accurate predictions of all statements.

   The algorithm which most accurately predicted the magnitude of this change in price was the k-Nearest Neighbour algorithm, also trained on the Merged Statements data. This model achieved an average Mean Absolute Error of 15.46.

3. *By taking into account additional non-numerical information related to individual companies unrelated to their financial performance, such as the industry and sector within which the company operates,in the form of a Knowledge Graph, can this added information be used in the form of Knowledge Graph Embeddings to improve predictions of price over the performance achieved using only financial statements?*

   Embeddings of 5, 7, and 9 dimensions were created using the Node2Vec algorithm to represent each company. These embeddings represent additional non-numerical information associated with each company. The best performing algorithm for each task and corresponding data augmented with these embeddings were assessed to determine whether an improvement on the original performance was possible. In 5 of the 9 cases tested, the company embeddings imrproved the performance of the best performing model for that task. The embeddings were particularly effective for the raw filing price predictions, improving the best performing model by at least 8.5%, with this improvement increasing to 11.5% using a 9 dimensional embedding. In cases where the embeddings didn't improve model performance, the reduction in model performance was less than 1% in each case, this is in spite of the increase in the dimensionality of the data resulting from adding the embeddings. As discussed in Section 6.3, changes to the evaluation metrics used to assess the performance of models for this task may result in embeddings improving the performance of these models also.

   The results demonstrate the usefulness of using Knowledge Graph Embeddings for encoding non-numerical information, and also underline the effectiveness of the Node2Vec algorithm

7.2 Future Work

The application of feature engineering techniques applied to the data and investigation into whether or not this improves model performance is a potential area of further research. In this work, having chosen parameters common to every company from each statement, each parameter was used in further analysis. Some of these parameters are likely not useful and/or highly correlated with one another and their removal could result in improved model performance due to reduced noise in the data.

Not all potentially useful parameters have been used in this analysis. In order to make the analysis consistent, parameters which were common to each company were used. This may have resulted in potentially vital information being lost. A better approach may be to generate price predictions for companies in a specific sector or industry via a model which has been trained on data only relating to companies within the sector or industry. As an example, banks earn income through interest earned on loans and other financial products, trying then to compare a banks performance with an Energy company, which has a completely different business model, is difficult. A better approach might be to compare a banks performance with the performance of other banks.

Furthermore, a specific focus on Neural Network performance on the data could be assessed in greater detail. Given their well documented high performance for a range of other tasks, their relatively poor performance on the tasks outlined in this work comes as a surprise. The best performing network architectures tended to be those with hidden layers made up of a large number of nodes relative to the number of input parameters. As well as this, experimentation on optimal hidden layer activation functions could result in improved performance. In this work only the ReLU activation function is used in hidden layers.

Finally, augmenting the data with embeddings obtained using other Knowledge Graph Embedding models could be investigated to assess the differences in performance achieved by using different models.

References

[1] P. Cootner, *"The Random Character of Stock Market Prices"*, M.I.T. Press, 1964

[2] Burton G. Malkiel, *"A Random Walk Down Wall Street"*, W. W. Norton & Company Inc., 1973.

[3] E. Fama *"Efficient Capital Markets: A Review of Theory and Empirical Work"*, The Journal of Business , 1970

[4] Investopedia.com , August 2020,
URL : https://www.investopedia.com/ask/answers/032615/what-are-differences-between-weak-strong-and-semistrong-versions-efficient-market-hypothesis.asp#:~:text=Though%20the%20efficient%20market%20hypothesis%20theorizes%20the%20market,no%20form%20of%20technical%20analysis%20can%20aid%20investors

[5] A. Arbel, S. Carvell and P.Strebel *"Giraffes, Institutions and Neglected Firms"*, Financial Analysts Journal, 1983

[6] The January Effect, August 2020,
URL : https://www.investopedia.com/terms/j/januaryeffect.asp

[7] Warren Buffett, *"The Superinvestors of Graham-and-Doddsville"*, Hermes, 1984

[8] B. Graham, D. Dodd, *"Security Analysis"*, McGraw-Hill, 1934

[9] Ockham's Razor, See textbook *"Artificial Intelligence"*, Russel and Norvig, 3rd edition, 2010
URL : https://en.wikipedia.org/wiki/Occam%27srazor

[10] Decision Tree Regressor, July 2020,
URL : https://gdcoder.com/decision-tree-regressor-explained-in-depth/

[11] Prof. M. Madden, Lecture Topic 4: Support Vector Machines, Deep Learning CT5133, NUI Galway

[12] Kernel Trick for Support Vector Machine, July 2020,
URL : https://datascience.stackexchange.com/questions/17536/kernel-trick-explanation

[13] Neural Network Architectures, July 2020,
URL: https://freecontent.manning.com/neural-network-architectures/

[14] KGkg: The Knowledge Graph about Knowledge Graphs,
URL: https://kgkg.factnexus.com/@3782 6.html

[15] Amazon.com, August 2020, URL: https://aws.amazon.com/neptune/

[16] Q. Wang, Z. Mao, B. Wang, L. Guo, *"Knowledge Graph Embedding: A Survey of Approaches and Applications"*, IEEE Transactions on Knowledge and Data Engineering, 2017

[17] A. Grover, J. Leskovec, *node2vec: Scalable Feature Learning for Networks*, KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016

[18] Kyoung-jae Kim, *"Financial Time Series Forecasting using Support Vector Machines"*, Elsevier: Neurocomputing, 2003.

[19] W. Huang, Y. Nakamori and S.Y. Wang, *"Forecasting Stock Market Movement Direction with Support Vector Machine"*, Elsevier: Computers & Operations research, 2005.

[20] S. Shen, H. Jiang and T. Zhang, *"Stock Market Forecasting Using Machine Learning Algorithms"*, Department of Electrical Engineering Stanford University, 2012.

[21] J. Patel, S. Shah, P. Thakkar and K. Kotecha, *"Predicting Stock Market Index Using Fusion of Machine Learning Techniques"*, Elsevier: Expert Systems with Applications, 2014.

[22] M. Ballings, D. Van den Poel, N. Hespeels and R. Gryp, *"Evaluating Multiple Classifiers for Stock Price Direction Prediction"*, Elsevier: Expert Systems with Applications, 2015.

[23] Y. Zhang and L. Wu, *"Stock Market Prediction of S&P 500 via Combination of Improved BCO Approach and BP Neural Network"*, Elsevier: Expert Systems with Applications, 2009.

[24] G. S. Atsalakis and K. P. Valavanis, *"Surveying Stock Market Forecasting Techniques – Part II: Soft Computing Methods"*, Elsevier: Expert Systems with Applications, 2009.

[25] G. Kayakutlu, E. Guresen and T. U. Daim, *"Using Artificial Neural Network Models in Stock Market Index Prediction"*, Elsevier: Expert Systems with Applications, 2011.

[26] Y. Kara, M. A. Boyacioglu and O. K. Baykan, *"Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange"*, Elsevier: Expert Systems with Applications, 2011.

[27] J.Z. Wang, J.J. Wang, Z.G. Zhang and S.P. Guo, *"Forecasting Stock Indices with Back Propagation Neural Network"*, Elsevier: Expert Systems with Applications, 2011.

[28] K. Chen, Y. Zhou and F. Dai, *"A LSTM-based Method for Stock Returns Prediction : A Case Study of China Stock Market"*, IEEE International Conference on Big Data, 2015.

[29] J. B. Heaton, N. G. Polson and J. H. Witte, *"Deep Learning for Finance: Deep Portfolios"*, Applied Stochastic Models in Business and Industry, 2016.

[30] M. Kraus and S. Feuerriegel, *"Decision Support from Financial Disclosures with Deep Neural Networks and Transfer Learning"*, Elsevier: Decision Support Systems, 2017.

[31] M. R. Vargas, B.S.L.P. de Lima and A.G. Evuskoff, *"Deep Learning for Stock Market Prediction from Financial News Articles"*, IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA), 2017.

[32] M. R. Hassan, B. Nath and M. Kirley, *"A Fusion Model of HMM, ANN and GA for Stock Market Forecasting"*, Elsevier: Expert Systems with Applications, 2007.

[33] K.J. Kim and I. Han, *"Genetic Algorithms Approach to Feature Discretization in Artificial Neural Networks for the Prediction of Stock Price Index"*, Elsevier: Expert Systems with Applications, 2000.

[34] X. Ding, Y. Zhang, T. Liu and J. Duan, *"Deep Learning for Event-Driven Stock Prediction"*, Twenty-fourth International Joint Conference on Artificial Intelligence, 2015.

[35] X. Ding, Y. Zhang, T. Liu and J. Duan, *"Knowledge-Driven Event Embedding for Stock Prediction"*, COLING 2016 The 26th International Conference on Computational Linguistics , 2016.

[36] Y. Liu, Q. Zeng, H. Yang and A. Carrio, *"Stock Price Movement Prediction from Financial News with Deep Learning and Knowledge Graph Embedding"*, Pacific Rim Knowledge Acquisition Workshop, 2018.

[37] J. Liu, Z. Lu and W. Du, *"Combining Enterprise Knowledge Graph and News Sentiment Analysis for Stock Price Volatility Prediction"*, Proceedings of the 52nd Hawaii International Conference on System Sciences, 2019.

[38] J. Long, Z. Chen, W. He, T. Wu and J. Ren, *"An Integrated Framework of Deep Learning and Knowledge Graph for Prediction of Stock Price Trend: An Application in Chinese Stock Exchange Market"*, Elsevier: Applied Soft Computing Journal, 2020.

[39] Hyperparameters Optimization, July 2020,
URL: https://towardsdatascience.com/hyperparameters-optimization-526348bb8e2d

[40] Python Implementation of Stratified Cross Validation, July 2020,
URL : https://scikit-learn.org/stable/modules/generated/sklearn.modelselection.StratifiedKFold.html

[41] Operating Income Definition, August 2020,
URL: https://www.investopedia.com/terms/o/operatingincome.asp:∼:text=Operating%20income%20is%20an%20accounting%20figure%20that%20measures,wages%2C%20depreciation%20and%20cost%20of%20goods%20sold%20%28COGS%29

[42] Normalized Earnings Definition, August 2020,
URL: https://www.investopedia.com/terms/n/normalizedearnings.asp:∼:text=Normalized%20earnings%20per%20share%20can%20be%20used%20to,or%20benefited%20from%20a%20number%20of%20one-off%20events

[43] Dividend per Share Definition, August 2020,
URL: https://www.investopedia.com/terms/d/dividend-per-share.asp

[44] Total Equity Definition, August 2020,
https://www.investopedia.com/terms/e/equity.asp

[45] Net Debt Definition, August 2020,
URL: https://www.investopedia.com/terms/n/netdebt.asp

[46] Free Cash Flow Definition, August 2020,
URL: https://www.investopedia.com/terms/f/freecashflow.asp

[47] Python Implementation of Min Max Normalisation, June 2020,
URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

[48] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.

[49] X. Glorot, A. Bordes and Yoshua, *Deep Sparse Rectifier Neural Networks*, Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS) 2011, Fort Lauderdale, FL, USA

[50] D.P. Kingma, J. Ba, *"Adam: A method for stochastic optimization"*, ICLR, 2015

[51] M. Abadi et al., "TensorFlow: A system for large-scale machine learning", OSDI, 2016.

[52] Seeking Alpha, June 2020,
URL: www.seekingalpha.com/

[53] Securities and Exchange Commission, June 2020,
URL: www.sec.gov/

[54] Network X Python library, August 2020, URL: https://pypi.org/project/networkx/

[55] Word2Vec Tutorial - The Skip-Gram Model, August 2020,
URL: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

[56] node2vec: Embeddings for Graph Data, August 2020,
URL : https://towardsdatascience.com/node2vec-embeddings-for-graph-data-32a866340fef

[57] L Maaten, G Hinton, *"Visualizing Data using t-SNE"*, Journal of Machine Learning research, 2008

Appendix

The tables below display the results of hyper-parameter testing for each of the algorithms used in the analysis. The top 5 hyper-parameter configurations for each algorithm are returned. Optimal hyper-parameters are obtained via a grid search, with these hyper-parameters then used to assess model performance over 10 folds of cross validation.

Six FCNN architectures are assessed for each task, with results averaged over 5 folds of cross validation. The best performing FCNN is compared with the performance of other Machine Learning algorithms averaged over 10 folds of cross validation.

Table 10: Section 4 - Hyperparameter Tuning Results 1

Statement	Algorithm	Hyper-Parameters	Average MAE	Std. Error
Balance Sheet	**Decision Tree**	**Max Tree Depth = 31, Splitter = random**	**35.63**	**3.73**
Balance Sheet	Decision Tree	Max Tree Depth = 24, Splitter = random	36.08	1.94
Balance Sheet	Decision Tree	Max Tree Depth = 43, Splitter = random	36.12	3.32
Balance Sheet	Decision Tree	Max Tree Depth = 34, Splitter = random	36.14	1.69
Balance Sheet	Decision Tree	Max Tree Depth = 24, Splitter = best	36.28	2.96
Balance Sheet	**KNN**	**K = 2, Weight = Distance, Metric = Euclidean**	**27.60**	**2.57**
Balance Sheet	KNN	K = 2, Weight = Distance, Metric = Manhattan	28.18	2.71
Balance Sheet	KNN	K = 1, Weight = Uniform, Metric = Euclidean	28.37	2.20
Balance Sheet	KNN	K = 1, Weight = Distance, Metric = Euclidean	28.37	2.20
Balance Sheet	KNN	K = 1, Weight = Distance, Metric = Manhattan	28.65	2.71
Balance Sheet	**Random Forest**	**Max Tree Depth = 14, No. Estimators = 26**	**30.93**	**1.45**
Balance Sheet	Random Forest	Max Tree Depth = 14, No. Estimators = 29	31.00	1.97
Balance Sheet	Random Forest	Max Tree Depth = 14, No. Estimators = 23	31.03	1.96
Balance Sheet	Random Forest	Max Tree Depth = 13, No. Estimators = 23	31.04	1.51
Balance Sheet	Random Forest	Max Tree Depth = 14, No. Estimators = 17	31.08	2.30
Balance Sheet	FCNN	No. Hidden Layers = 1, No. Nodes per Layer = 9	43.66	3.63
Balance Sheet	**FCNN**	**No. Hidden Layers = 1, No. Nodes per Layer = 81**	**39.96**	**2.54**
Balance Sheet	FCNN	No. Hidden Layers = 2, No. Nodes per Layer = 9, 9	40.91	3.94
Balance Sheet	FCNN	No. Hidden Layers = 2, No. Nodes per Layer = 9, 4	41.18	2.09
Balance Sheet	FCNN	No. Hidden Layers = 3, No. Nodes per Layer = 9, 9, 9	40.44	2.42
Balance Sheet	FCNN	No. Hidden Layers = 3, No. Nodes per Layer = 9, 4, 3	42.33	2.56
Cash Flow	**Decision Tree**	**Max Tree Depth = 6, Splitter = Best**	**36.01**	**0.98**
Cash Flow	Decision Tree	Max Tree Depth = 8, Splitter = Best	36.37	0.92
Cash Flow	Decision Tree	Max Tree Depth = 7, Splitter = Best	36.40	0.63
Cash Flow	Decision Tree	Max Tree Depth = 9, Splitter = Best	36.62	1.27
Cash Flow	Decision Tree	Max Tree Depth = 5, Splitter = Best	36.67	1.02
Cash Flow	**KNN**	**K = 37, Weight = Distance , Metric = Manhattan**	**48.32**	**1.83**
Cash Flow	KNN	K = 36, Weight = Distance , Metric = Manhattan	48.34	1.80
Cash Flow	KNN	K = 35, Weight = Distance , Metric = Manhattan	48.35	1.81
Cash Flow	KNN	K = 38, Weight = Distance , Metric = Manhattan	48.36	1.83
Cash Flow	KNN	K = 34, Weight = Distance , Metric = Manhattan	48.38	1.79
Cash Flow	**Random Forest**	**Max Tree Depth = 11, No. Estimators = 29**	**32.63**	**0.45**
Cash Flow	Random Forest	Max Tree Depth = 12, No. Estimators = 29	32.69	1.10
Cash Flow	Random Forest	Max Tree Depth = 10, No. Estimators = 16	32.73	0.60
Cash Flow	Random Forest	Max Tree Depth = 13, No. Estimators = 28	32.81	0.75
Cash Flow	Random Forest	Max Tree Depth = 14, No. Estimators = 29	32.88	1.16
Cash Flow	FCNN	No. Hidden Layers = 1, No. Nodes per Layer = 8	47.12	3.00
Cash Flow	**FCNN**	**No. Hidden Layers = 1, No. Nodes per Layer = 64**	**46.01**	**3.02**
Cash Flow	FCNN	No. Hidden Layers = 2, No. Nodes per Layer = 8, 8	47.09	2.91
Cash Flow	FCNN	No. Hidden Layers = 2, No. Nodes per Layer = 8, 4	52.28	10.57
Cash Flow	FCNN	No. Hidden Layers = 3, No. Nodes per Layer = 8, 8, 8	47.25	2.67
Cash Flow	FCNN	No. Hidden Layers = 3, No. Nodes per Layer = 8, 4, 2	52.55	12.85

Table 11: Section 4 - Hyperparameter Tuning Results 2

Statement	Algorithm	Hyper-Parameters	Average MAE	Std. Error
Income Statement	**Decision Tree**	**Max Tree Depth = 8, Splitter = Best**	**30.75**	**4.45**
Income Statement	Decision Tree	Max Tree Depth = 6, Splitter = Best	30.76	4.56
Income Statement	Decision Tree	Max Tree Depth = 7, Splitter = Best	30.92	4.97
Income Statement	Decision Tree	Max Tree Depth = 10, Splitter = Best	30.93	5.00
Income Statement	Decision Tree	Max Tree Depth = 9, Splitter = Best	31.28	4.97
Income Statement	**KNN**	**K = 3, Weight = Distance, Metric = Manhattan**	**37.43**	**6.63**
Income Statement	KNN	K = 2, Weight = Distance, Metric = Manhattan	37.48	6.34
Income Statement	KNN	K = 4, Weight = Distance, Metric = Manhattan	37.92	6.26
Income Statement	KNN	K = 5, Weight = Distance, Metric = Manhattan	38.04	6.39
Income Statement	KNN	K = 2, Weight = Uniform, Metric = Manhattan	38.16	6.28
Income Statement	**Random Forest**	**Max Tree Depth = 14, No. Estimators = 22**	**26.48**	**3.33**
Income Statement	Random Forest	Max Tree Depth = 14, No. Estimators = 27	26.554	3.47
Income Statement	Random Forest	Max Tree Depth = 14, No. Estimators = 26	26.557	2.67
Income Statement	Random Forest	Max Tree Depth = 13, No. Estimators = 19	26.59	3.46
Income Statement	Random Forest	Max Tree Depth = 14, No. Estimators = 29	26.62	3.50
Income Statement	FCNN	No. Hidden Layers = 1, No. Nodes per Layer = 10	48.20	12.18
Income Statement	**FCNN**	**No. Hidden Layers = 1, No. Nodes per Layer = 100**	**38.37**	**6.64**
Income Statement	FCNN	No. Hidden Layers = 2, No. Nodes per Layer = 10, 10	41.18	7.41
Income Statement	FCNN	No. Hidden Layers = 2, No. Nodes per Layer = 10, 5	48.68	19.6
Income Statement	FCNN	No. Hidden Layers = 3, No. Nodes per Layer = 10, 10, 10	41.41	7.29
Income Statement	FCNN	No. Hidden Layers = 3, No. Nodes per Layer = 10, 5, 3	41.84	7.52
Merged Statements	**Decision Tree**	**Max Tree Depth = 17, Splitter = Random**	**29.54**	**1.43**
Merged Statements	Decision Tree	Max Tree Depth = 20, Splitter = Random	29.79	0.86
Merged Statements	Decision Tree	Max Tree Depth = 18, Splitter = Random	29.84	1.63
Merged Statements	Decision Tree	Max Tree Depth = 19, Splitter = Random	29.93	2.14
Merged Statements	Decision Tree	Max Tree Depth = 22, Splitter = Random	30.26	2.18
Merged Statements	**KNN**	**K = 2, Weight = Distance , Metric = Manhattan**	**25.88**	**1.47**
Merged Statements	KNN	K = 2, Weight = Uniform , Metric = Manhattan	26.63	1.36
Merged Statements	KNN	K = 1, Weight = Uniform , Metric = Manhattan	26.94	1.74
Merged Statements	KNN	K = 1, Weight = Distance , Metric = Manhattan	26.94	1.74
Merged Statements	KNN	K = 3, Weight = Distance , Metric = Manhattan	27.17	1.42
Merged Statements	**Random Forest**	**Max Tree Depth = 15, No. Estimators = 28**	**24.34**	**1.23**
Merged Statements	Random Forest	Max Tree Depth = 15, No. Estimators = 19	24.70	1.18
Merged Statements	Random Forest	Max Tree Depth = 14, No. Estimators = 27	24.74	0.98
Merged Statements	Random Forest	Max Tree Depth = 13, No. Estimators = 24	24.75	1.31
Merged Statements	Random Forest	Max Tree Depth = 14, No. Estimators = 24	24.76	1.58
Merged Statements	FCNN	No. Hidden Layers = 1, No. Nodes per Layer = 26	34.89	1.67
Merged Statements	**FCNN**	**No. Hidden Layers = 1, No. Nodes per Layer = 676**	**32.26**	**2.82**
Merged Statements	FCNN	No. Hidden Layers = 2, No. Nodes per Layer = 26, 26	35.62	8.1
Merged Statements	FCNN	No. Hidden Layers = 2, No. Nodes per Layer = 26, 13	38.21	8.72
Merged Statements	FCNN	No. Hidden Layers = 3, No. Nodes per Layer = 26, 26, 26	41.09	22.48
Merged Statements	FCNN	No. Hidden Layers = 3, No. Nodes per Layer = 26, 13, 6	41.13	17.85

Table 12: Section 5(i) - Hyperparameter Tuning Results 1

Statement	Algorithm	Hyper-Parameters	Mean Accuracy	Std. Error
Balance Sheet	**SVM**	**C = 0.1, Kernel = linear**	**0.675**	**0.0002**
Balance Sheet	SVM	C = 10, Kernel = linear	0.675	0.0002
Balance Sheet	SVM	C = 10, Kernel = linear	0.675	0.0002
Balance Sheet	SVM	C = 10, Kernel = polynomial degree 2	0.675	0.0002
Balance Sheet	SVM	C = 10, Kernel = rbf	0.675	0.0002
Balance Sheet	**Decision Tree**	**Max Tree Depth = 10, Splitter = random, Criterion = Entropy**	**0.681**	**0.008**
Balance Sheet	Decision Tree	Max Tree Depth = 4, Splitter = best, Criterion = Entropy	0.681	0.010
Balance Sheet	Decision Tree	Max Tree Depth = 6, Splitter = best, Criterion = Gini Index	0.679	0.011
Balance Sheet	Decision Tree	Max Tree Depth = 4, Splitter = best, Criterion = Gini Index	0.678	0.011
Balance Sheet	Decision Tree	Max Tree Depth = 11, Splitter = random, Criterion = Entropy	0.677	0.011
Balance Sheet	**KNN**	**K = 61, Weight = Distance, Metric = Manhattan**	**0.692**	**0.006**
Balance Sheet	KNN	K = 62, Weight = Distance, Metric = Manhattan	0.6913	0.004
Balance Sheet	KNN	K = 65, Weight = Distance, Metric = Manhattan	0.6913	0.004
Balance Sheet	KNN	K = 66, Weight = Distance, Metric = Manhattan	0.6913	0.004
Balance Sheet	KNN	K = 56, Weight = Distance, Metric = Manhattan	0.691	0.006
Balance Sheet	**Random Forest**	**Max Tree Depth = 9, No. Estimators = 25**	**0.713**	**0.003**
Balance Sheet	Random Forest	Max Tree Depth = 7, No. Estimators = 29	0.710	0.006
Balance Sheet	Random Forest	Max Tree Depth = 7, No. Estimators = 26	0.709	0.003
Balance Sheet	Random Forest	Max Tree Depth = 6, No. Estimators = 21	0.709	0.006
Balance Sheet	Random Forest	Max Tree Depth = 5, No. Estimators = 21	0.709	0.006
Balance Sheet	FCNN	No. Hidden Layers = 1, No. Nodes per Layer = 9	0.674	0.001
Balance Sheet	**FCNN**	**No. Hidden Layers = 1, No. Nodes per Layer = 81**	**0.677**	**0.004**
Balance Sheet	FCNN	No. Hidden Layers = 2, No. Nodes per Layer = 9, 9	0.675	0.001
Balance Sheet	FCNN	No. Hidden Layers = 2, No. Nodes per Layer = 9, 4	0.675	0.0003
Balance Sheet	FCNN	No. Hidden Layers = 3, No. Nodes per Layer = 9, 9, 9	0.675	0.0003
Balance Sheet	FCNN	No. Hidden Layers = 3, No. Nodes per Layer = 9, 4, 3	0.676	0.0008
Cash Flow	**SVM**	**C = 10, Kernel = polynomial degree 3**	**0.681**	**0.002**
Cash Flow	SVM	C = 10000, Kernel = polynomial degree 3	0.680	0.003
Cash Flow	SVM	C = 0.1, Kernel = polynomial degree 3	0.678	0.004
Cash Flow	SVM	C = 10000, Kernel = polynomial degree 2	0.678	0.004
Cash Flow	SVM	C = 1000, Kernel = polynomial degree 3	0.678	0.002
Cash Flow	**Decision Tree**	**Max Tree Depth = 5, Splitter = Random, Criterion = Entropy**	**0.678**	**0.003**
Cash Flow	Decision Tree	Max Tree Depth = 2, Splitter = Best, Criterion = Gini Index	0.677	0.006
Cash Flow	Decision Tree	Max Tree Depth = 2, Splitter = Best, Criterion = Entropy	0.677	0.006
Cash Flow	Decision Tree	Max Tree Depth = 9, Splitter = Random, Criterion = Entropy	0.676	0.002
Cash Flow	Decision Tree	Max Tree Depth = 8, Splitter = Random, Criterion = Entropy	0.676	0.007
Cash Flow	**KNN**	**K = 54, Weight = Distance , Metric = Manhattan**	**0.685**	**0.005**
Cash Flow	KNN	K = 52, Weight = Distance , Metric = Manhattan	0.684	0.006
Cash Flow	KNN	K = 52, Weight = Uniform , Metric = Manhattan	0.684	0.007
Cash Flow	KNN	K = 46, Weight = Distance , Metric = Manhattan	0.683	0.003
Cash Flow	KNN	K = 50, Weight = Distance , Metric = Manhattan	0.683	0.005
Cash Flow	**Random Forest**	**Max Tree Depth = 7, No. Estimators = 21**	**0.692**	**0.004**
Cash Flow	Random Forest	Max Tree Depth = 6, No. Estimators = 29	0.688	0.005
Cash Flow	Random Forest	Max Tree Depth = 8, No. Estimators = 25	0.688	0.005
Cash Flow	Random Forest	Max Tree Depth = 11, No. Estimators = 28	0.688	0.008
Cash Flow	Random Forest	Max Tree Depth = 9, No. Estimators = 20	0.688	0.007
Cash Flow	**FCNN**	**No. Hidden Layers = 1, No. Nodes per Layer = 8**	**0.677**	**0.005**
Cash Flow	FCNN	No. Hidden Layers = 1, No. Nodes per Layer = 64	0.676	0.004
Cash Flow	FCNN	No. Hidden Layers = 2, No. Nodes per Layer = 8, 8	0.676	0.002
Cash Flow	FCNN	No. Hidden Layers = 2, No. Nodes per Layer = 8, 4	0.675	0.001
Cash Flow	FCNN	No. Hidden Layers = 3, No. Nodes per Layer = 8, 8, 8	0.675	0.0007
Cash Flow	FCNN	No. Hidden Layers = 3, No. Nodes per Layer = 8, 4, 2	0.675	0.0003

Table 13: Section 5(i) - Hyperparameter Tuning Results 2

Statement	Algorithm	Hyper-Parameters	Mean Accuracy	Std. Error
Income Statement	**SVM**	**C = 10000, Kernel = polynomial degree 3**	**0.682**	**0.004**
Income Statement	SVM	C = 10000, Kernel = polynomial degree 2	0.680	0.003
Income Statement	SVM	C = 1000, Kernel = rbf	0.679	0.003
Income Statement	SVM	C = 1000, Kernel = polynomial degree 3	0.679	0.003
Income Statement	SVM	C = 100, Kernel = rbf	0.679	0.0002
Income Statement	**Decision Tree**	**Max Tree Depth = 4, Splitter = Best, Criterion = Gini Index**	**0.692**	**0.008**
Income Statement	Decision Tree	Max Tree Depth = 5, Splitter = Best, Criterion = Entropy	0.691	0.007
Income Statement	Decision Tree	Max Tree Depth = 4, Splitter = Best, Criterion = Entropy	0.690	0.008
Income Statement	Decision Tree	Max Tree Depth = 5, Splitter = Best, Criterion = Gini Index	0.688	0.010
Income Statement	Decision Tree	Max Tree Depth = 3, Splitter = Best, Criterion = Entropy	0.688	0.009
Income Statement	**KNN**	**K = 90, Weight = Distance, Metric = Euclidean**	**0.702**	**0.007**
Income Statement	KNN	K = 94, Weight = Distance, Metric = Euclidean	0.702	0.008
Income Statement	KNN	K = 95, Weight = Distance, Metric = Euclidean	0.702	0.010
Income Statement	KNN	K = 91, Weight = Distance, Metric = Euclidean	0.701	0.007
Income Statement	KNN	K = 84, Weight = Distance, Metric = Euclidean	0.701	0.010
Income Statement	**Random Forest**	**Max Tree Depth = 6, No. Estimators = 28**	**0.705**	**0.005**
Income Statement	Random Forest	Max Tree Depth = 6, No. Estimators = 20	0.704	0.007
Income Statement	Random Forest	Max Tree Depth = 4, No. Estimators = 19	0.703	0.008
Income Statement	Random Forest	Max Tree Depth = 6, No. Estimators = 27	0.703	0.007
Income Statement	Random Forest	Max Tree Depth = 5, No. Estimators = 21	0.703	0.005
Income Statement	**FCNN**	**No. Hidden Layers = 1, No. Nodes per Layer = 10**	**0.6824**	**0.004**
Income Statement	FCNN	No. Hidden Layers = 1, No. Nodes per Layer = 100	0.6818	0.006
Income Statement	FCNN	No. Hidden Layers = 2, No. Nodes per Layer = 10, 10	0.681	0.002
Income Statement	FCNN	No. Hidden Layers = 2, No. Nodes per Layer = 10, 5	0.681	0.004
Income Statement	FCNN	No. Hidden Layers = 3, No. Nodes per Layer = 10, 10, 10	0.6820	0.004
Income Statement	FCNN	No. Hidden Layers = 3, No. Nodes per Layer = 10, 5, 3	0.680	0.002
Merged Statements	**SVM**	**C = 10000, Kernel = polynomial degree 3**	**0.685**	**0.004**
Merged Statements	SVM	C = 1000, Kernel = polynomial degree 3	0.680	0.002
Merged Statements	SVM	C = 10000, Kernel = polynomial degree 2	0.678	0.003
Merged Statements	SVM	C = 50, Kernel = rbf	0.678	0.002
Merged Statements	SVM	C = 100, Kernel = rbf	0.678	0.001
Merged Statements	**Decision Tree**	**Max Tree Depth = 2, Splitter = Best, Criterion = Gini Index**	**0.700**	**0.003**
Merged Statements	Decision Tree	Max Tree Depth = 2, Splitter = Best, Criterion = Entropy	0.700	0.003
Merged Statements	Decision Tree	Max Tree Depth = 6, Splitter = Best, Criterion = Entropy	0.699	0.012
Merged Statements	Decision Tree	Max Tree Depth = 3, Splitter = Best, Criterion = Gini Index	0.696	0.008
Merged Statements	Decision Tree	Max Tree Depth = 3, Splitter = Best, Criterion = Entropy	0.696	0.005
Merged Statements	**KNN**	**K = 26, Weight = Uniform , Metric = Manhattan**	**0.711**	**0.012**
Merged Statements	KNN	K = 39, Weight = Distance , Metric = Manhattan	0.709	0.010
Merged Statements	KNN	K = 38, Weight = Uniform , Metric = Manhattan	0.709	0.011
Merged Statements	KNN	K = 25, Weight = Uniform , Metric = Manhattan	0.709	0.012
Merged Statements	KNN	K = 24, Weight = Uniform , Metric = Manhattan	0.708	0.012
Merged Statements	**Random Forest**	**Max Tree Depth = 4, No. Estimators = 14**	**0.714**	**0.011**
Merged Statements	Random Forest	Max Tree Depth = 5, No. Estimators = 11	0.714	0.006
Merged Statements	Random Forest	Max Tree Depth = 6, No. Estimators = 25	0.714	0.007
Merged Statements	Random Forest	Max Tree Depth = 5, No. Estimators = 27	0.713	0.010
Merged Statements	Random Forest	Max Tree Depth = 4, No. Estimators = 22	0.713	0.014
Merged Statements	FCNN	No. Hidden Layers = 1, No. Nodes per Layer = 26	0.6822	0.004
Merged Statements	**FCNN**	**No. Hidden Layers = 1, No. Nodes per Layer = 676**	**0.688**	**0.006**
Merged Statements	FCNN	No. Hidden Layers = 2, No. Nodes per Layer = 26, 26	0.685	0.003
Merged Statements	FCNN	No. Hidden Layers = 2, No. Nodes per Layer = 26, 13	0.684	0.005
Merged Statements	FCNN	No. Hidden Layers = 3, No. Nodes per Layer = 26, 26, 26	0.683	0.008
Merged Statements	FCNN	No. Hidden Layers = 3, No. Nodes per Layer = 26, 13, 6	0.675	0.0003

Table 14: Section 5(ii) - Hyperparameter Tuning Results 1

Statement	Algorithm	Hyper-Parameters	Average MAE	Std. Error
Balance Sheet	**Decision Tree**	**Max Tree Depth = 10, Splitter = Random**	**16.17**	**0.87**
Balance Sheet	Decision Tree	Max Tree Depth = 2, Splitter = Random	16.24	1.10
Balance Sheet	Decision Tree	Max Tree Depth = 1, Splitter = Random	16.36	1.03
Balance Sheet	Decision Tree	Max Tree Depth = 3, Splitter = Random	16.44	1.19
Balance Sheet	Decision Tree	Max Tree Depth = 6, Splitter = Best	16.63	1.17
Balance Sheet	**KNN**	**K = 99, Weight = Distance, Metric = Manhattan**	**15.9475**	**1.08**
Balance Sheet	KNN	K = 98, Weight = Distance, Metric = Manhattan	15.9478	1.09
Balance Sheet	KNN	K = 97, Weight = Distance, Metric = Manhattan	15.95	1.09
Balance Sheet	KNN	K = 96, Weight = Distance, Metric = Manhattan	15.96	1.10
Balance Sheet	KNN	K = 95, Weight = Distance, Metric = Manhattan	15.96	1.09
Balance Sheet	**Random Forest**	**Max Tree Depth = 5, No. Estimators = 27**	**15.85**	**0.97**
Balance Sheet	Random Forest	Max Tree Depth = 5, No. Estimators = 4	15.99	0.98
Balance Sheet	Random Forest	Max Tree Depth = 3, No. Estimators = 13	16.04	0.90
Balance Sheet	Random Forest	Max Tree Depth = 9, No. Estimators = 26	16.07	1.10
Balance Sheet	Random Forest	Max Tree Depth = 4, No. Estimators = 7	16.07	1.34
Balance Sheet	FCNN	No. Hidden Layers = 1, No. Nodes per Layer = 9	15.9046	1.07
Balance Sheet	**FCNN**	**No. Hidden Layers = 1, No. Nodes per Layer = 81**	**15.9048**	**1.07**
Balance Sheet	FCNN	No. Hidden Layers = 2, No. Nodes per Layer = 9, 9	15.9046	1.07
Balance Sheet	FCNN	No. Hidden Layers = 2, No. Nodes per Layer = 9, 4	15.9046	1.07
Balance Sheet	FCNN	No. Hidden Layers = 3, No. Nodes per Layer = 9, 9, 9	15.9046	1.07
Balance Sheet	FCNN	No. Hidden Layers = 3, No. Nodes per Layer = 9, 4, 3	15.9046	1.07
Cash Flow	**Decision Tree**	**Max Tree Depth = 1, Splitter = Random**	**16.42**	**1.37**
Cash Flow	Decision Tree	Max Tree Depth = 2, Splitter = Random	16.43	1.41
Cash Flow	Decision Tree	Max Tree Depth = 4, Splitter = Random	16.46	1.39
Cash Flow	Decision Tree	Max Tree Depth = 6, Splitter = Random	16.47	1.31
Cash Flow	Decision Tree	Max Tree Depth = 5, Splitter = Random	16.49	1.54
Cash Flow	**KNN**	**K = 99, Weight = Distance , Metric = Manhattan**	**16.21**	**1.39**
Cash Flow	KNN	K = 96, Weight = Distance , Metric = Manhattan	16.22	1.40
Cash Flow	KNN	K = 98, Weight = Distance , Metric = Manhattan	16.223	1.39
Cash Flow	KNN	K = 95, Weight = Distance , Metric = Manhattan	16.225	1.40
Cash Flow	KNN	K = 94, Weight = Distance , Metric = Manhattan	16.229	1.40
Cash Flow	**Random Forest**	**Max Tree Depth = 1, No. Estimators = 8**	**16.45**	**1.17**
Cash Flow	Random Forest	Max Tree Depth = 1, No. Estimators = 21	16.48	1.01
Cash Flow	Random Forest	Max Tree Depth = 1, No. Estimators = 14	16.50	1.02
Cash Flow	Random Forest	Max Tree Depth = 1, No. Estimators = 12	16.51	1.03
Cash Flow	Random Forest	Max Tree Depth = 1, No. Estimators = 25	16.53	1.02
Cash Flow	FCNN	No. Hidden Layers = 1, No. Nodes per Layer = 8	15.90	1.43
Cash Flow	FCNN	No. Hidden Layers = 1, No. Nodes per Layer = 64	15.90	1.43
Cash Flow	FCNN	No. Hidden Layers = 2, No. Nodes per Layer = 8, 8	15.90	1.43
Cash Flow	**FCNN**	**No. Hidden Layers = 2, No. Nodes per Layer = 8, 4**	**15.91**	**1.43**
Cash Flow	FCNN	No. Hidden Layers = 3, No. Nodes per Layer = 8, 8, 8	15.90	1.43
Cash Flow	FCNN	No. Hidden Layers = 3, No. Nodes per Layer = 8, 4, 2	15.90	1.43

Table 15: Section 5(ii) - Hyperparameter Tuning Results 2

Statement	Algorithm	Hyper-Parameters	Average MAE	Std. Error
Income Statement	**Decision Tree**	**Max Tree Depth = 6, Splitter = Random**	**16.32**	**1.38**
Income Statement	Decision Tree	Max Tree Depth = 2, Splitter = Random	16.40	0.97
Income Statement	Decision Tree	Max Tree Depth = 1, Splitter = Random	16.42	0.99
Income Statement	Decision Tree	Max Tree Depth = 3, Splitter = Random	16.44	0.96
Income Statement	Decision Tree	Max Tree Depth = 5, Splitter = Random	16.58	0.88
Income Statement	**KNN**	**K = 99, Weight = Distance, Metric = Manhattan**	**15.542**	**1.04**
Income Statement	KNN	K = 98, Weight = Distance, Metric = Manhattan	15.543	1.04
Income Statement	KNN	K = 97, Weight = Distance, Metric = Manhattan	15.545	1.04
Income Statement	KNN	K = 96, Weight = Distance, Metric = Manhattan	15.546	1.03
Income Statement	KNN	K = 95, Weight = Uniform, Metric = Manhattan	15.547	1.04
Income Statement	**Random Forest**	**Max Tree Depth = 14, No. Estimators = 22**	**26.48**	**3.33**
Income Statement	Random Forest	Max Tree Depth = 14, No. Estimators = 27	26.554	3.47
Income Statement	Random Forest	Max Tree Depth = 14, No. Estimators = 26	26.557	2.67
Income Statement	Random Forest	Max Tree Depth = 13, No. Estimators = 19	26.59	3.46
Income Statement	Random Forest	Max Tree Depth = 14, No. Estimators = 29	26.62	3.50
Income Statement	**FCNN**	**No. Hidden Layers = 1, No. Nodes per Layer = 10**	**15.9046**	**1.026**
Income Statement	FCNN	No. Hidden Layers = 1, No. Nodes per Layer = 100	15.9014	1.026
Income Statement	FCNN	No. Hidden Layers = 2, No. Nodes per Layer = 10, 10	15.90464	1.026
Income Statement	FCNN	No. Hidden Layers = 2, No. Nodes per Layer = 10, 5	15.90464	1.026
Income Statement	FCNN	No. Hidden Layers = 3, No. Nodes per Layer = 10, 10, 10	15.90464	1.026
Income Statement	FCNN	No. Hidden Layers = 3, No. Nodes per Layer = 10, 5, 3	15.90464	1.026
Merged Statements	**Decision Tree**	**Max Tree Depth = 4, Splitter = Random**	**16.30**	**0.88**
Merged Statements	Decision Tree	Max Tree Depth = 2, Splitter = Random	16.36	1.33
Merged Statements	Decision Tree	Max Tree Depth = 1, Splitter = Random	16.37	1.26
Merged Statements	Decision Tree	Max Tree Depth = 4, Splitter = Best	16.67	1.77
Merged Statements	Decision Tree	Max Tree Depth = 3, Splitter = Random	16.75	1.98
Merged Statements	**KNN**	**K = 99, Weight = Distance , Metric = Manhattan**	**15.497**	**1.42**
Merged Statements	KNN	K = 98, Weight = Distance , Metric = Manhattan	15.498	1.41
Merged Statements	KNN	K = 97, Weight = Distance , Metric = Manhattan	15.5032	1.41
Merged Statements	KNN	K = 96, Weight = Distance , Metric = Manhattan	15.5033	1.41
Merged Statements	KNN	K = 98, Weight = Uniform , Metric = Manhattan	15.51	1.42
Merged Statements	**Random Forest**	**Max Tree Depth = 6, No. Estimators = 25**	**15.58**	**1.32**
Merged Statements	Random Forest	Max Tree Depth = 8, No. Estimators = 23	15.589	1.65
Merged Statements	Random Forest	Max Tree Depth = 9, No. Estimators = 28	15.595	1.38
Merged Statements	Random Forest	Max Tree Depth = 6, No. Estimators = 20	15.616	1.50
Merged Statements	Random Forest	Max Tree Depth = 7, No. Estimators = 27	15.619	1.58
Merged Statements	**FCNN**	**No. Hidden Layers = 1, No. Nodes per Layer = 26**	**15.90**	**1.60**
Merged Statements	FCNN	No. Hidden Layers = 1, No. Nodes per Layer = 676	15.90	1.60
Merged Statements	FCNN	No. Hidden Layers = 2, No. Nodes per Layer = 26, 26	15.90	1.60
Merged Statements	FCNN	No. Hidden Layers = 2, No. Nodes per Layer = 26, 13	15.90	1.60
Merged Statements	FCNN	No. Hidden Layers = 3, No. Nodes per Layer = 26, 26, 26	15.90	1.60
Merged Statements	FCNN	No. Hidden Layers = 3, No. Nodes per Layer = 26, 13, 6	15.90	1.60