A Comparative Analysis of Machine Learning Models for Forecasting JSE Stock Returns

by

Cameron James Muir

MRXCAM001

SUBMITTED TO THE UNIVERSITY OF CAPE TOWN
In (partial if the degree was by coursework AND dissertation) fulfilment of the requirements for the degree

MCom in Finance in the field of Investment Management

Faculty of Commerce

UNIVERSITY OF CAPE TOWN

Supervisor: Paul Van Rensburg

Department of Finance & Tax

UNIVERSITY OF CAPE TOWN
FACULTY OF COMMERCE
Igniting Knowledge and Opportunity

Plagiarism Declaration

COMPULSORY DECLARATION:

1. This dissertation has been submitted to Turnitin (or equivalent similarity and originality checking software) and I confirm that my supervisor has seen my report and any concerns revealed by such have been resolved with my supervisor.
2. I certify that I have received Ethics approval (if applicable) from the Commerce Ethics Committee.
3. This work has not been previously submitted in whole, or in part, for the award of any degree in this or any other university. It is my own work. Each significant contribution to, and quotation in, this dissertation from the work, or works of other people has been attributed, and has been cited and referenced.

Student number	MRXCAM001
Student name	Cameron James Muir
Signature of Student	Signed by candidate
Date:	06/06/2025

Abstract

This study examines the application of machine learning models to predict the cross-section of Johannesburg Stock Exchange (JSE)- listed share returns. Four models are developed and compared using monthly data from 2005 to 2021: neural networks, random forest, long short-term memory (LSTM) networks, and conventional linear regression. The explanatory variables comprise nine firm-specific financial metrics, motivated by prior research. The sample is divided into a training period (2005–2016) and a testing period (2016–2021), further split into 1-year, 3-year, and 5-year testing intervals. The results show that the LSTM model performs best across most evaluation metrics and investment scenarios, with the random forest model close behind, offering slightly better risk-adjusted returns.

Table of Contents

List of Figures

List of Tables

1 Introduction

In recent years, the field of artificial intelligence (AI) has surged in popularity due to its integration of advanced computational methods, enabling computers to autonomously execute tasks that historically relied on human intelligence (Collins et al. 2021:6). This transformative trend has had a profound impact across diverse sectors of society, prompting questions about its future implications for the financial industry. AI techniques such as artificial neural networks (ANN), a biological nervous system-inspired information processing model (Basu, Bhattacharyya & Kim, 2010:25), utilise advanced algorithms to identify patterns and make accurate predictions from historical data. Such AI methods have revolutionised the financial sector, where investment decision-making depends on accurate forecasting and risk assessment.

However, the concept of accurate forecasting of future trends is challenged by the Efficient Market Hypothesis (EMH), which specifies that stock forecasting has limited viability due to the belief that stock prices follow a random pattern, as suggested by the Random Walk Theory. Although many critique the EMH's inherent limitations, it only serves as a theoretical foundation for market dynamics. This ongoing debate has prompted many researchers to contest EMH by delving into the field of machine learning in an attempt to augment traditional forecasting methods. In general, stock market forecasting is widely acknowledged as a highly relevant but exceptionally challenging task in finance (Chen & Hao, 2017:342). This shift towards adopting more innovative ideas has led many to explore machine learning and its capabilities on the many stock exchanges.

The Johannesburg Stock Exchange (JSE), the largest in Africa with a market capitalisation of USD 1.36 trillion (Chikwira & Mohammed, 2023:2), presents a unique opportunity to investigate the applicability of machine-learning techniques within a developing market. While numerous studies have examined machine learning in developed economies such as the United States of America (USA) and Taiwan (Chu et al. 2009), there is limited research on developing markets such as the JSE. Although some studies have explored the application of machine learning techniques in emerging markets such as Turkey (Kara et al. 2011) and Thailand (Inthachot et al. 2015), there remains a need for further research in other developing markets, such as the JSE. This research gap promotes the exploration of the unique dynamics and characteristics of machine learning within the JSE-specific emerging market context.

The All-Share Index (JALSH) can be analysed to address this research gap within the South African market. The JALSH represents approximately 99% of the full market capitalisation of all equities listed on the JSE and serves as a reliable proxy to assess the performance of the exchange (Johannesburg Stock Exchange, 2024). Leveraging machine learning methods in conjunction with panel data techniques can offer valuable insights into the unique dynamics of the JSE. The use of panel data enables a comprehensive examination of both cross-sectional and time-series data within the context of machine learning. Thus, promoting the intricate relationship between various factors influencing stock returns on the JSE and enhancing the understanding of stock prediction in this developing market landscape.

To better achieve this unionisation of machine learning and panel data, the focus will be placed on three advanced machine learning methods: neural networks, long short-term memory (LSTM), and random forests. Additionally, a basic linear regression model will be included for comparison purposes. These primary methods have demonstrated exceptional effectiveness in handling substantial and intricate datasets (Rumelhart, Hinton, & Williams, 1986; Hochreiter & Schmidhuber, 1997; Breiman, 2001). Neural networks are known for their ability to model complex non-linear relationships in data, making them highly effective for financial forecasting tasks. They can adapt to new patterns and provide robust predictions, particularly in the context of large and complex datasets. LSTM networks, a specialised type of neural network, excel at handling complex tasks that involve long periods and effectively managing input data that might be noisy or difficult to compress. By surpassing the limitations of traditional neural networks, LSTM has evolved into a powerful tool with a wide range of applications across real-world problems (Hochreiter & Schmidhuber, 1997:1735). Random forests represent a versatile machine learning method that excels at combining multiple decision trees to yield robust and accurate predictions. This method is particularly beneficial in handling high-dimensional datasets with numerous features (Breiman, 2001:5) and is perfectly suited for financial panel data. Basic linear regression, while not a machine learning method, offers a traditional statistical approach to modelling relationships between variables. It serves as a valuable benchmark due to its simplicity and interpretability despite its limitations in capturing the complex, non-linear relationships often present in financial markets. By integrating these methods, this research aims to leverage the strengths of each to improve the accuracy and robustness of stock market forecasting.

1.1 Research question

The primary research question for this study is:

- Which advanced machine learning model, specifically linear regression, Neural Network, random forest or LSTM, demonstrates the highest predictive accuracy for stock returns within the JSE?

1.2 Motivation

The integration of machine learning into stock market forecasting has transformed financial analysis, enabling more sophisticated predictive capabilities. However, its application to the South African market, particularly the JSE, remains limited, despite the JSE's status as Africa's largest exchange and its distinctive characteristics shaped by South Africa's resource-driven economy, socio-political dynamics, and global investment relevance. Existing studies, such as Marwala (2010) and Balusik et al. (2021), have applied models like neural networks, support vector machines, and LSTMs to predict aggregate indices such as the JSE Top 40, relying mainly on time-series data. These efforts are constrained by narrow model comparisons and a lack of focus on firm-specific factors that capture cross-sectional variation in this emerging market. This thesis aims to address this gap by forecasting cross-sectional stock returns for individual JSE-listed firms using a broader set of machine learning models and firm-level data. The goal is to develop a more robust, context-specific framework to support improved investment decision-making across the JSE and similar emerging markets.

This research thesis breakdown follows a structured sequence: initiating a Literature Review including an exploration of Machine Learning and its subsets – artificial neural networks, recurrent neural networks, LSTM, and random forests – and then extending into the domain of Panel Data and looking at linear regression. Subsequent sections unfold the Methodology, delineating the approach to sampling and data collection and culminating in a concise description of the overall research design. Followed by the Hyperparameter Optimisation and Results. The conclusion summarises all the findings of the study and includes a List of References, showcasing the range of influential sources that have shaped the study's foundation.

2 Literature Review

2.1 Machine Learning

Machine learning is centred around the pivotal challenge of creating computer programmes capable of autonomous learning and improvement based on experience (Mitchell, 1997:3). This has led researchers and data scientists to explore its expansive potential across diverse contexts. Gupta and Pandya (2022:1023) underscore the consensus among data scientists that a single universal algorithm cannot effectively address all challenges due to the intricate and unique nature of real-world problems. This realisation has given rise to a profusion of expertise and a range of machine learning subfields, as depicted in Figure 1 below.

Within this framework:

- Supervised Learning: training models using labelled data to learn a function that maps inputs to outputs, enabling predictions or classifications (Mahesh, 2020:381).
- Unsupervised Learning: the autonomous analysis of unlabelled data to uncover hidden patterns and predict outcomes without guidance (Gupta and Pandya, 2022:1023).
- Semi-supervised learning: combines labelled and unlabelled examples for output generation.
- Reinforcement Learning: learning strategies that incorporate indirect or delayed feedback as training information (Mitchell, 1997:17).

It's important to note that while neural networks are presented as their own section in Figure 1, they can function within both Supervised and Unsupervised learning paradigms depending on their objectives. A more comprehensive understanding of neural networks and their diverse applications will be further interpreted in the subsequent sections.

Figure 1: Machine Learning Breakdown

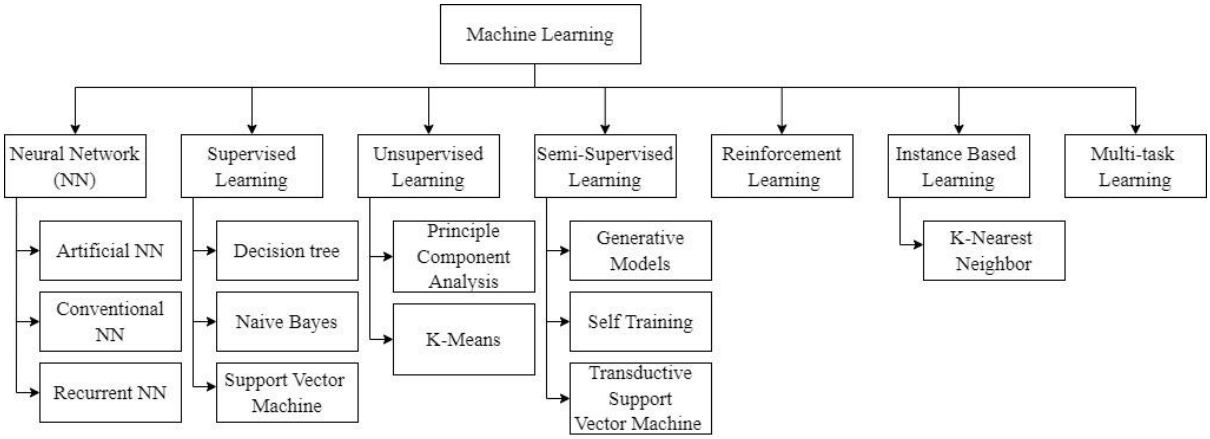
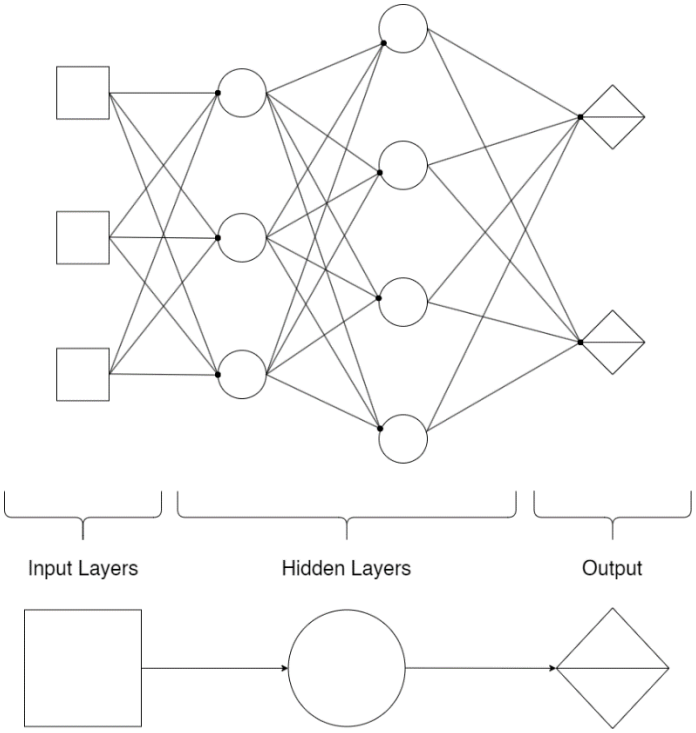


Figure 1 is a representation of various machine learning subsections adapted from Mahesh (2020). Each subsection represents distinct approaches and applications within machine learning. It should be noted that this diagram provides a summarised overview of the main approaches in the field without encompassing the entirety of machine learning.

2.2 Artificial Neural Networks

Artificial neural networks (ANNs), as discussed in a paper by Grossi and Buscema (2007:1046), are human brain-inspired systems that can adapt their internal structure in relation to a function objective. ANNs follow a concept that combines two key processes: feedforward propagation and backpropagation. Feedforward propagation entails transmitting input data through a set number of network layers to produce an output. Feedforward networks can be classified into two categories based on the number of layers they contain: single layer or multilayer (Sazli, 2006:12). Figure 2 illustrates a multilayer feedforward network with a 3-3-4-2 architecture.

Figure 2: Multilayer feedforward neural network



The multilayer feedforward neural network, adapted from Dongare, Kharde, and Kachare (2012:191), features an additional visual abstraction in the bottom half of the figure. This abstraction represents the input layer, two hidden layers, and the output layer in a simplified and compact form.

The optimal architecture for these networks is a topic of ongoing debate in the literature. Uzair and Jamil (2020:1) conducted a review examining the impact of hidden layers and identified that using three hidden layers is the most effective configuration. However, Xu and Chen (2008:685) arrived at different findings during their review of various neural network mechanisms, suggesting that five hidden layers are the most optimal. In another study, Gupta and Raza (2020:2868) proposed a new methodology to estimate hidden layers and neurons, concluding that two hidden layers provide the best performance.

As observed above, the optimal number of hidden layers in a feedforward network can vary, depending on several factors, such as the size of the dataset and the desired timeframe for completion (Uzair & Jamil, 2020:6). For instance, while a feedforward network with five hidden layers might yield advantageous results for smaller to medium-sized datasets (Xu & Chen, 2008:685), using two hidden layers could be more effective for larger sample sizes (Gupta & Raza, 2020:2865).

To provide further insight into the structure of an ANN, particularly the architecture depicted in Figure 2, it can be described in its mathematical form below. The input layer transfers the data to the first hidden layer. For each neuron in the first hidden layer (indexed by j), the weighted sum (z_j) of the input is calculated as follows:

$$z_j = \sum_{i=1}^{n_{input}} w_{ij} x_i + b_j \quad (1)$$

where:

- n_{input} is the number of input neurons.
- w_{ij} is the weight between the connection of the i-th input neuron and the j-th neuron in the first hidden layer.
- x_i is the value of the i-th input neuron.
- b_j is the bias associated with the j-th neuron in the first hidden layer.

After calculating the weighted sum for each neuron, an activation function is applied to get the output (activation) of each neuron in the first hidden layer:

$$a_j = f(z_j) \quad (2)$$

where a_j is the output of the j-th neuron in the first hidden layer and $f()$ is an activation function.

After applying the activation function to the neurons in the first hidden layer, the process can be adapted and repeated for each subsequent hidden layer until the output layer is reached. In the example, with only two hidden layers, the weighted sum for each neuron in the second hidden layer is calculated the same way it was calculated in the first hidden layer, but instead of x_i from the input neuron, a_j from the output of the j-th neuron in the first hidden layer is used. Once this is repeated for the second hidden layer, the output of the neural network can be calculated:

$$z_l = \sum_{k=1}^{n_{hidden2}} w_{kl} a_k + b_l \quad (3)$$

where:

- $n_{hidden2}$ is the number of neurons in the second hidden layer.
- w_{kl} is the weight of the connection between the k-th neuron in the second hidden layer and the l-th neuron in the output layer.
- a_k is the output (activation) of the k-th neuron in the second hidden layer.
- b_l is the bias associated with the l-th neuron in the output layer.

Much like the first hidden layer, an activation function is applied to the weighted sum for each neuron to get the final output layer:

$$a_l = f(z_l) \quad (4)$$

Once the weighted sum is calculated and an activation function is applied for each neuron in the output layer, the feedforward process concludes with generating the network's output. However, this is a very linear approach that follows a fixed path from input to output, excluding any feedback or adjustments to errors.

The concept of backpropagation is implemented to overcome this limitation and enhance the network's learning capabilities. Backpropagation, the most prevalent algorithm for training feed-forward neural networks (Sazli, 2006:12), was popularised by Rumelhart, Hinton, and Williams (1986:533). They explored backpropagation and its numerous advantages, demonstrating its ability to learn complex tasks beyond the capabilities of simpler methods.

Backpropagation fundamentally computes the impact of changing synapse strengths on error terms using the chain rule of calculus. It recursively calculates error signals in a hierarchical manner backwards from the output layer to the input layer to adjust weights and improve predictions (Lillicrap et al. 2020:340). The core concept of backpropagation is to repetitively minimise the difference between the actual outputs and predicted outputs to achieve the smallest possible error during each ANN training loop labelled an epoch.

This is achieved by an Optimisation process called gradient descent. Gradient descent uses the weights and biases of a neural network to calculate the gradient of an error function. The gradient directs the weights and biases to reduce the error, and after each epoch, the weights and biases are gradually moved towards the optimal values to minimise the error.

By referring to the same feedforward network discussed above, the flow breakdown of both feedforward and backpropagation processes in Figure 3 can be observed below. Focusing on the backpropagation element of the figure, the algorithm enables the network to iteratively adjust its weights and biases, aiming to minimise the error between predicted outputs and actual target values. For each neuron in the network, including neurons j and k, the algorithm computes error signals denoted as δ.

Figure 3: Feedforward and Backpropagation Flow

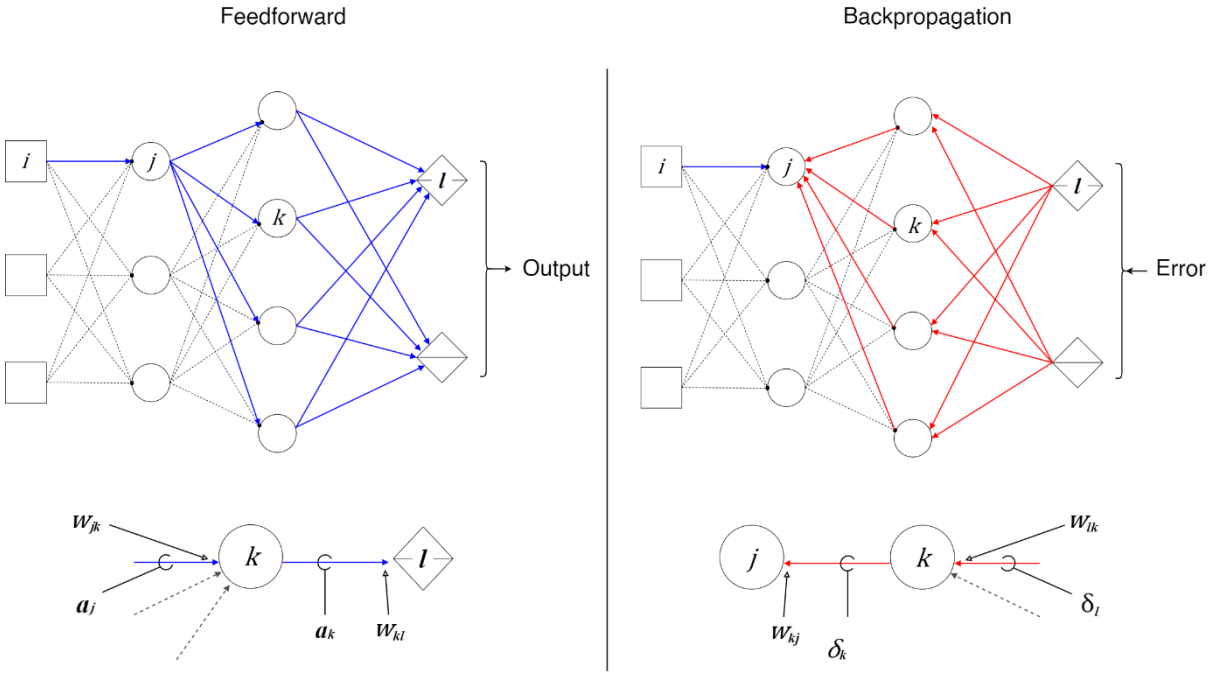


Figure 3 presents a graphical representation of the neural network described in Figure 2, depicting the feedforward and backpropagation processes. Adapted from Lillicrap et al. (2020:341), the diagram showcases the sequential steps of feedforward computation and the recursive nature of backpropagation. Additionally, the diagram includes two smaller images at the bottom, illustrating labelled connections for neurons *k* and *l*, and *j* and *k*, providing detailed insights into the interplay between adjacent layers during both feedforward and backpropagation phases.

Specifically, for neuron k, the error signal, δ*k*, is computed based on the difference between its desired output and its actual output, along with the derivative of its activation function concerning its total input. It follows a mathematical formula:

$$\delta_k = e_k f'(a_k) = (\sum_l \delta_l W_{kl}) f'(a_k) \quad (5)$$

where:

- e_k is the error between the target and the actual output for the *k*-th unit.
- $f'(a_k)$ is the derivative of the activation function from the neuron's output.
- $\sum_l \delta_l W_{kl}$ is the weighted sum of the subsequent layer errors (layer l).

Regarding the connection between the neurons k and j in Figure 3, dependencies in dataconsider the ΔW_{jk} . The change in weight between k and j represents the weight update rule for the neuron during the training process. The formula is provided:

$$\Delta W_{jk} = -\eta \frac{\partial E}{\partial W_{jk}} = -\eta a_j \delta_k \quad (6)$$

where:

- η is the learning rate.
- $\frac{\partial E}{\partial W_{jk}}$ is the partial derivative of the error function with respect to the weight.
- a_j being the output of neuron j

It is important to note that the hyperparameter learning rate (η) plays a crucial role in controlling the size of weight updates (Mitchell, 1997:88). A larger learning rate speeds up the learning process by allowing greater weight updates; however, this can risk missing optimal weights and may lead to algorithm instability. On the other hand, a smaller learning rate results in a slower learning process but offers more stability (Brownlee, 2019).

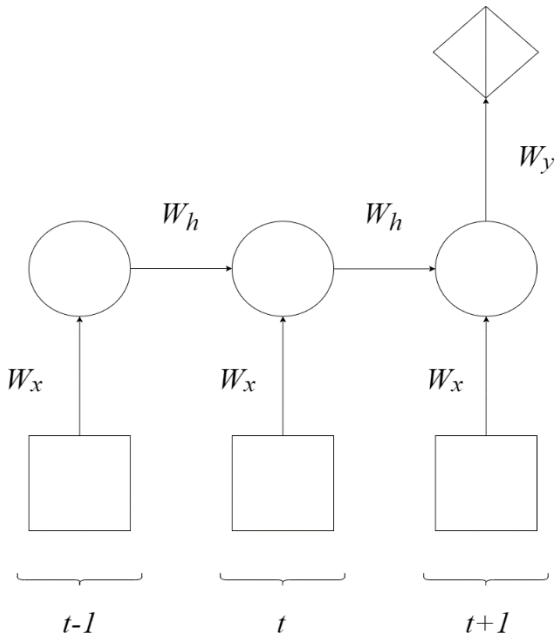
2.2.1 Recurrent Neural Networks

While traditional ANNs are powerful and widely used, they do face multiple challenges. One such challenge is dealing with time series data, such as stock market predictions or foreign exchange (forex) rate forecasting. This is because they cannot preserve the memory of past events over a rolling time period. In other words, they treat each input independently without

considering the sequential relationship between data points. Recurrent neural networks (RNNs) are utilised to address this limitation. RNNs are a type of network that uses the output of network units at time t as the input to other units at time t + 1 (Mitchell, 1997:119).

A paper by Schmidt (2019) explores RNNs in the context of language modelling and image captioning, expanding upon the most important concepts within RNNs and their various use cases. Figure 4 below provides a graphical representation of how an RNN works. The paper also addresses the problems with RNNs, specifically the "vanishing gradients" first identified by Hochreiter (1991). The vanishing gradient problem occurs during backpropagation when gradients become extremely small (usually less than 1), exponentially diminishing weight updates during each epoch. This issue hampers RNNs' ability to learn long-term dependencies in sequential data, making it challenging to effectively capture sequential patterns and relationships.

Figure 4: Many-to-One Recurrent Neural Network



Key	
W_y	Output weight
W_h	Hidden layer weight
W_x	Input weight
t	Time

The figure shows a many-to-one RNN design based off of Figure 2, which consists of three compact neural networks, each corresponding to a different time step: t-1, t, and t+1, with an output at time t+1.

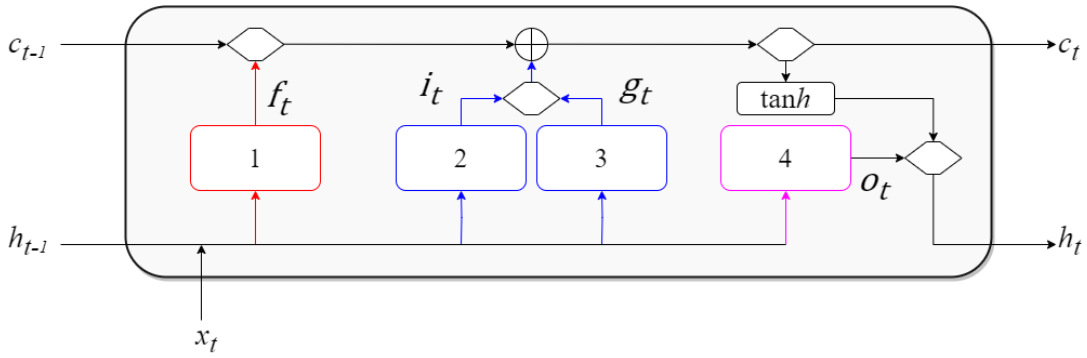
2.3 Long Short-Term Memory

LSTM is an RNN popularised as a means to solve the problem of vanishing gradients. LSTM can connect information separated by large, distinct time steps and use error signals to improve

its predictions. The internal structure of LSTM, with constant error carousels, enables it to remember important information from far back in the sequence, making it particularly powerful for tasks involving long-range dependencies in data (Hochreiter & Schmidhuber, 1997:1735).

The internal architecture of the LSTM can be outlined into three distinct sections: memory cells, gate mechanisms, and constant error flow. These sections are depicted in the memory cell (Figure 5), illustrating the fundamental concept behind LSTM: establishing a continuous error path connecting each time step (Landi et al. 2021:338). Memory cells are complex units that incorporate a central linear unit with a fixed self-connection, including the forget, input, and output gate mechanisms. These gating mechanisms control access to the memory contents stored within the cell and regulate whether information should be retained or discarded (Hochreiter & Schmidhuber, 1997:1741). As seen in Figure 5, the LSTM cell inputs are from a prior time step (t-1), and the outputs are from the next time step.

Figure 5: Long Short-Term Memory cell



Key		
1	Forget Gate	$w_{fh}w_{fx}b_f$
2	Input Gate	$w_{ih}w_{ix}b_i$
3	Input Gate (Input Node)	$w_{gh}w_{gx}b_g$
4	Output Gate	$w_{oh}w_{ox}b_o$

The figure, adapted from Schmidt (2019), shows a Long-Short-Term Memory cell with four distinct labelled gates. The gates are represented by their own number and colour: Red 1, Blue 2 & 3, and Purple 4. The arrows within the cell represent the flow of information from timestamp t-1 to timestamp t.

2.3.1 LSTM Cell Breakdown

Gate 1, also known as the forget gate (ft), is a critical component responsible for controlling the flow of information within the LSTM network. The forget gate was not included in the original work by Hochreiter and Schmidhuber (1997) but was later added by Gers et al. (2000:2451) as a means to stop the memory cells from growing indefinitely, causing the network to break down. It utilises the sigmoid activation function (σ) to ensure that the gate's output remains within a range of 0 to 1. To compute ft, the concatenation of the previous hidden state (h_{t-1}) and the current input (x_t) is taken, represented by the vector [h_{t-1}, x_t]. This vector is then multiplied by weight matrices (w_{fx} and w_{fh}) and added to a bias term (b_f) in the formula for ft:

$$f_t = \sigma(w_{fx}x_t + w_{fh}h_{t-1} + b_f) \quad (7)$$

The updated value of ft is then combined with the previous cell state (c_{t-1}) with the same dimensionality to eliminate any irrelevant information and retain essential information in the updated cell state $c_t = ft \times c_{t-1}$.

Similar processes are repeated for both input gates 2, 3 and output gate 4, except the sigmoid function is not used within gate 3, but the tanh function is used :

$$i_t = \sigma(w_{ix}x_t + w_{ih}h_{t-1} + b_i) \quad (8)$$

$$g_t = tanh(w_{gx}x_t + w_{gh}h_{t-1} + b_g) \quad (9)$$

$$o_t = \sigma(w_{ox}x_t + w_{oh}h_{t-1} + b_o) \quad (10)$$

The tanh function scales the values in the cell state to be within the range of -1 to 1. This wider range of values as opposed to the sigmoid functions 0 to 1, enables the network to capture more nuanced information, making it suitable for learning from complex sequential data. The outcome of multiplying Gates 2 and 3 values is added to Gate 1's c_t finding, creating the formula:

$$c_t = (ft \times c_{t-1}) + (i_t \times g_t) \quad (11)$$

This c_t value is thus transferred into the following LSTM cell, but before leaving the network, it is augmented by tanh and multiplied by the finding of gate 4 (o_t) creating the new hidden states value (h_t) :

$$h_t = o_t \times tanh(c_t) \quad (12)$$

2.3.2 Prior LSTM Research

LSTM networks have been widely used in various fields, including natural language processing, sentiment analysis, and time series prediction, with their application expanding to the financial industry. Sen, Dutta and Mehtab (2021) developed a deep learning-based regression model that uses an LSTM network to forecast future stock prices. Their model automatically scrapes the Yahoo finance API for historical stock prices of 75 significant stocks from 15 critical sectors of the Indian stock market. The LSTM network is then trained on this data to predict future stock prices accurately. The study's results demonstrated the effectiveness and efficiency of the LSTM network while also providing valuable insights into the most profitable sectors, with the fast-moving consumer goods sector standing out as particularly promising.

A paper by Siami-Namini, Tavakoli, and Namin (2018) also tested the effectiveness of an LSTM network using financial data while comparing it to traditional techniques like Autoregressive Integrated Moving Average (ARIMA). LSTM's ability to handle complex temporal patterns and long-term dependencies resulted in an average reduction in error rates of 84% to 87% compared to ARIMA. It was also found that the LSTM epoch had no significant impact on its performance.

However, in contradiction to these two papers, Selvin et al. (2017) found that in the particular case of minute-wise stock prediction of National Stock Exchange of India-listed companies, a convolutional neural network (CNN) could capture changes in stock price trends better than the LSTM network and RNN. Specifically, the authors applied a sliding window approach to capture the data's short-term dependencies and evaluated the models' performance using percentage error.

Lastly, Ghosh, Neufeld, and Sahoo (2022) investigated the effectiveness of random forests and LSTM networks using multi-feature settings to forecast directional movements of stock prices

while physically completing intraday trading using the S&P 500. The results showed that LSTM models outperformed random forests regarding daily return, with 0.64% for LSTM and 0.54% for random forests prior to transaction costs. These values were higher compared to the single-feature approaches employed by other authors.

2.4 Random Forests

Random forests, developed by Breiman (2001), is a machine learning algorithm designed to mitigate the overfitting tendency of individual decision trees. The fundamental concept of random forests involves constructing a collection of predictors by creating multiple decision trees, which are grown using randomly selected subsets of data (Biau, 2012:1063). Breiman's work was influenced by countless studies, such as Ho's (1998) random subspace method, Amit and German's (1997) work on geometric feature selection, and the random split selection approach by Dietterich (2000). These studies, coupled with Breiman's prior work on a learning technique called bagging, influenced Breiman to develop random forests.

2.4.1 Random Forests Core Concepts

When discussing random forests, it is imperative to understand the meaning of classification trees and the methods used to mitigate overfitting. Classification trees, as depicted in Figure 6 below, constitute the foundation of random forests. They utilise binary decisions based on whether a feature $x_i \leq k$, where k serves as a threshold. The parent node encompasses all data points denoted by $D = \{(x_i, y_i)\}_{i=1}^{n}$. This dataset is subsequently distributed among the children of each node, following the principles of the classification function. This facilitates data division into distinct classes, ensuring diversity among child nodes. This process, often referred to as gini impurity, contributes to overfitting mitigation and enhanced generalisation (Khaidem, Saha & Dey, 2016:7).

Figure 6: Classification Tree and Data

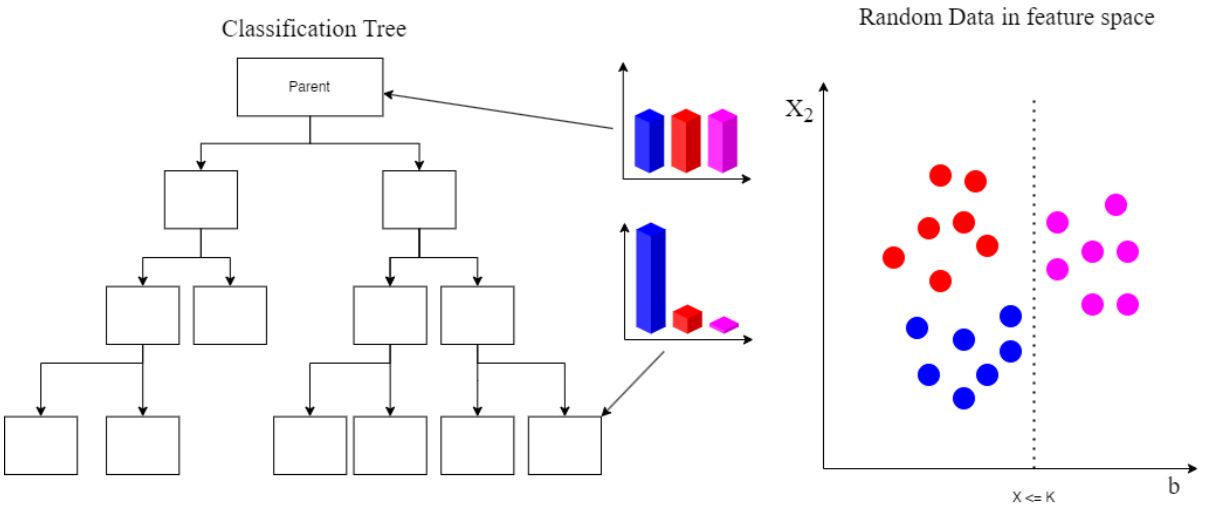


Figure 6 illustrates a classification tree adapted from Lee, Ullah & Wang, 2020:389, alongside randomised data points grouped into three clusters: Red, Blue, and Purple. The tree's initial data distribution is evenly divided at the parent node. Notably, one child node holds a more significant concentration within the blue cluster. This visual demonstrates data partitioning within classification trees.

Regarding random vectors and data representation within a random forest, consider a random dataset represented in Figure 6 containing random vectors X = $(X_1, \dots X_d)$. The joint distribution linked with each random vector depicts the probability distribution of various values. Imagine the joint distribution as a mechanism that assigns probabilities to discrete regions within the dataset's 'space'. Each region corresponds to a collection of conceivable values for the random vector. To aid visualisation, these distinct regions are visually represented using different colours, each signifying a unique segment of the 'space'. Understanding this joint distribution provides insights into how the variables within the vector will interact and change in different situations.

As discussed above, classification is an important concept within random forests. If each tree within the forest is unique with random variables, then each tree may be associated with a classifier function. The classifier function can be defined as $h_k(x) = h(x|\theta_k)$ where $h(x|\theta_k)$ is the decision tree k with randomly chosen vector parameters θ_k using data x. Thus, a random forest can be seen as a group of classifiers $h(\theta_1), \dots, h(x|\theta_k)$ where each classifier contributes to the final aggregate outcome.

2.4.2 Bagging and Ensemble Learning

A core concept of random forests is bootstrap aggregating (bagging). This technique is utilised during the creation of random forests and is employed to enhance the robustness of forecasts (Lee, Ullah & Wang, 2020:389). By training multiple classifiers using bootstrapped datasets (samples drawn randomly with replacement), bagging effectively reduces classification variance (Gislason, Benediktsson & Sveinsson, 2006:294). These trained datasets, known as base learners, collectively form an ensemble of predictors. Variance reduction is achieved through aggregating outputs from individual base learners, thus reinforcing the overall estimator's robustness (Lee, Ullah & Wang, 2020:413).

2.4.3 Prior Random Forest Research

The unique capability of random forests to aggregate decisions or forecasts from a substantial subset of data has led researchers to explore their applications across diverse industries. This approach has been a subject of considerable interest due to its potential to enhance predictions in various contexts. Notably, Abraham et al. (2022) set out to combine genetic algorithms and random forest techniques to predict stock returns using four international stock indices (S&P 500, NIKKEI 225, CAC40, and DAX). The authors found that their approach outperformed several other classification algorithms regarding accuracy and precision. Specifically, they reported an average accuracy of 67% for their model.

In another practical application, Khaidem, Saha and Dey (2016) undertook a distinct avenue of exploration within the field of random forests. Their study employed random forest classifiers to craft predictive models for forecasting stock direction movement, leveraging past authors' financial datasets. The authors achieved notable success in their test, with accuracy rates ranging from 85% to 95% in their long-term forecasts. This predictive accuracy emphasised the potential of machine learning models, particularly random forest classifiers, to forecast stock trends and enable the development of novel trading strategies and adaptive stock portfolio management. The implications of their findings extend beyond financial forecasting, beckoning further examination of random forests' applicability across a diverse array of industries and domains.

2.5 Linear Regression

Linear regression is one of the most fundamental and widely used statistical techniques in predictive modelling. It involves modelling the relationship between a dependent variable and one or more independent variables using a linear equation (Uyanık & Güler, 2013:234). Linear regression can be categorised into simple and multiple linear regression. While simple linear regression models the relationship with a single independent variable, multiple linear regression involves two or more. In predictive contexts such as financial forecasting, it is common to use lagged values of the independent variables to estimate future values of the dependent variable. The basic form of a multiple linear regression model with time-lagged predictors is:

$$Y = \beta_0 + \beta_1 X_{1,t-1} + \beta_2 X_{2,t-1} + \ldots + \beta_n X_{n,t-1} + \epsilon_t \quad (13)$$

where:

- Y_t is the dependent variable at time t.
- $X_{n,t-1}$the independent variables measured at time t-1.
- β_0 the intercept.
- β_n are the coefficients.
- ϵ_t is the error term at time t.

Linear regression is a fundamental technique in financial forecasting, valued for its ease of use and interpretability. It is commonly applied in risk assessment, portfolio optimisation, and stock price prediction. However, the intrinsic complexity of financial data, including outliers, volatility, and non-linear patterns, presents difficulties for linear regression models. Adrian, Crump, and Moench (2013) expand upon this thought process and demonstrate that while linear regression remains a widely used tool, the unique characteristics of financial data necessitate a more rigorous approach to model selection and validation to overcome the limitations of linear methods. Their study shows that a comprehensive assessment of model assumptions is crucial to ensure the validity and reliability of the regression analysis when working with complex financial data.

2.6 Panel Data and Forecasting

Panel data analysis, also known as cross-sectional time series analysis, is widely employed in economics, social sciences and, more recently, financial research. It involves analysing datasets encompassing independent and dependent observations across numerous entities over multiple periods. The distinct advantage of panel data lies in its ability to capture individual entity-specific effects and temporal dynamics, allowing researchers to explore relationships that might otherwise remain hidden in purely traditional cross-sectional or time-series analyses. In this context, the fusion of panel data analysis with advanced machine learning methods such as Neural networks, LSTM and random forests presents a promising avenue for unveiling hidden patterns and insights within these rich datasets.

Combining panel data and machine learning is not new, but its prominence and use have grown in popularity. This is due to significant advancements in both fields. The amalgamation of these two models, specifically LSTM and panel data, has been explored in a paper by Sarkar and De Bruyn (2021). Utilising a dataset extracted from a marketing campaign, the authors ran two empirical applications using the LSTM neural network. The first method was to test the LSTM in a more straightforward direct marketing setting with a smaller panel dataset, tracking only two variables over time. The second application was more complex and utilised a larger panel dataset. The authors then compared the LSTM model to 271 traditional hand-crafted feature engineering models. It was found that the LSTM network utilising panel data is effective in its predictions and can outperform conventional benchmark models regarding average fit and performance.

Furthermore, the exploration of panel data's combination with distinct machine learning methods is exemplified in the study conducted by Chen (2021). The paper focuses on applying panel data within a random forest algorithm. By utilising data from the Boston housing panel dataset, the study contrasts the predictive efficacy of the random forest algorithm against traditional linear regression methods. The paper identifies that machine learning methodologies, particularly random forests, exhibit superior predictive accuracy compared to linear regression methods. However, the paper notes that while random forests excel in predictive power, their interpretability might not uniformly match the depth offered by traditional linear regression methods.

In addition to advanced machine learning models, traditional linear regression remains a vital tool in panel data analysis. Linear regression, with its straightforward interpretability and

robust statistical foundations, has been extensively used for forecasting and analysing financial data. A study by Ur Rehman and Gul (2017) uses panel regression to predict firm stock returns in the Pakistani equity market, demonstrating that linear regression models can provide valuable insights into the relationships between firm-level variables and stock performance. Furthermore, the paper by Hjalmarsson (2010) discusses how panel data estimators can improve the predictive power of linear regression models for forecasting international stock returns compared to single-country time series approaches. These studies show that while linear regression may not always match the predictive accuracy of more complex models, it remains an essential baseline technique for panel data analysis, offering transparency and interpretability in forecasting tasks.

The integration of panel data with both traditional and advanced machine learning techniques thus provides a comprehensive approach to financial forecasting. While LSTM and random forests offer superior predictive capabilities, linear regression contributes essential interpretability and is a crucial benchmark for evaluating model performance.

2.7 Prior Machine Learning Findings on Stock Returns in Emerging Markets

When it comes to better understanding machine learning and stock market forecasting, specifically in emerging markets like the JSE, few studies have been conducted to expand this field further. One such study by Balusik, de Magalhaes, and Mbuvha (2021) explores using LSTM networks to forecast the JSE Top 40 index. Their research demonstrates that LSTM networks outperform traditional time series models, such as the Seasonal Autoregressive Integrated Moving Average (SARIMA) model, in predicting intraday directional movements and closing prices of the index. In a similar line of research, Cao, Leggio, and Schniederjans (2005) investigated the effectiveness of neural networks compared to linear models for predicting stock price movements in the emerging Chinese market, specifically on the Shanghai Stock Exchange. They discovered that neural networks significantly outperformed linear models in forecasting price returns. This conclusion was consistent across their sample firms, highlighting the potential of neural networks as a powerful tool for stock price prediction in emerging markets like China. In addition to the studies mentioned, another noteworthy study was conducted by Marwala (2010) on the JSE All Share Index. The study employed artificial intelligence techniques, including neural networks, support vector machines, and neuro-fuzzy systems, alongside a random walk method to predict the future price of the stock market index.

While the AI techniques outperformed the linear ARMA model, the random walk method was the most accurate for forecasting.

2.8 Theoretical Foundations of Stock Return Predictability

The Efficient Market Hypothesis (EMH), formalised by Fama (1970), posits that stock prices fully reflect available information, rendering consistent outperformance through forecasting challenging. The EMH comprises three forms: the weak form, where prices incorporate historical data such as past prices; the semi-strong form, encompassing public information like earnings and dividends; and the strong form, including all information, both public and private. The weak and semi-strong forms are central to stock return predictability, as deviations suggest market inefficiencies, particularly in emerging markets like the JSE, where information dissemination may be less efficient (Cakici et al., 2023).

Stock return predictability hinges on two key information types: momentum and fundamental factors. Momentum indicators, such as past returns, align with weak-form EMH violations by exploiting historical price trends, while fundamental factors, including dividend yields and earnings, support semi-strong EMH and enable longer-term forecasts. Foundational research highlights their potential predictive power. Fama and French (1988) found that dividend yields could predict long-horizon returns in the US market, suggesting that valuation ratios are informative signals. Ang and Bekaert (2007) tested a present value model incorporating macroeconomic and valuation variables, such as interest rates, dividend yields, and earnings yields, across major economies. They found robust short-run predictability from interest rates but challenged the reliability of dividend and earnings yields once sample biases were corrected.

In emerging markets like the JSE, informational frictions and liquidity constraints can amplify return predictability. Cakici et al. (2023) provide global evidence that machine learning models, leveraging technical indicators like momentum and reversal alongside fundamental indicators such as earnings-to-price and book-to-market, achieve superior cross-sectional return predictability. These models excel at capturing non-linear relationships, making them particularly effective in volatile, data-constrained environments. However, their performance depends heavily on market depth and idiosyncratic risk, both of which are particularly relevant in South Africa's emerging market context.

3 Research and Methodology

The research and methodology sections are designed to effectively answer the study's main research questions. The section can be split into three sections: 1.) Data Collection and Cleaning, 2.) Model Creation, and 3.) Results and Analysis. Each section has its subsections and processes that will be expanded upon below.

3.1 Data Collection and Cleaning

This study utilises index and panel data derived from the JALSH (FTSE/JSE All Share Index), which tracks the performance of South African-listed companies and represents approximately 99% of the full market capitalisation. The JALSH serves as a comprehensive benchmark for the broader South African equity market, encompassing firms across various sectors and industries.

Monthly data was extracted from Bloomberg, covering the period from February 2005 to February 2023. However, for the purposes of model development and testing, a fixed sample window from 1 February 2005 to 1 February 2021 is used. This ensures a consistent training-testing framework aligned with best practice, as detailed in Section 3.2.

The panel dataset uses forward monthly return as the dependent variable, serving as a proxy for future performance, alongside independent variables representing valuation, profitability, risk, momentum, and past performance. These, outlined in Table 1, are based on established literature relevant to the South African equity market.

In line with Van Rensburg and Robertson (2003), market value and EPS are included to capture firm size and earnings power, with EPS serving as a practical proxy for valuation despite the original use of the P/E ratio. Risk is represented by 12-month return variance (RetVar12), reflecting historical volatility and the JSE's heteroscedastic return patterns (Szczygielski & Chipeta, 2023). Momentum and reversal factors draw from Cakici et al. (2023), who demonstrate the predictive power of short-term trends in global and emerging markets.

Additional variables include cash flow per share (CFP), stock price, and book value per share (BVPS), selected to capture core fundamentals. Studies by Alexandroi (2019) and Foerster et al. (2017) highlight the superior predictive value of cash flows over earnings. Charteris and Strydom (2016) reinforce the importance of price-based valuation in South Africa, while BVPS, central to the book-to-market ratio (Fama & French, 1993), is also included as a standalone metric due to its local relevance.

Table 1: Independent Variable Breakdown

Metric	Factor	Abbreviation	Definition
Valuation	Market Value	MV	Reflects the total value of a company's outstanding shares.
	Book Value per Share	BVPS	Indicates the equity value on a per-share basis.
	Stock Price	Price	Represents the current trading value of a share.
Profitability	Earnings per Share	EPS	Measures a company's profitability per share.
	Cash Flow per Share	CFP	Shows the cash flow generated per share of the company.
Risk	12-month Return Variance	Retvar12	Captures the variability in returns over the past year.
Momentum	1-month Momentum	Mom1	Measures the price change over the past month.
	12-month Momentum	Mom12	Measures the price change over the past year.
Performance	Returns	Returns	Denotes the gains or losses from an investment over a period.

3.2 Methodology

The creation of the models follows a three-step process to achieve the desired outcomes. This process systematically guides the development of each model, ensuring accuracy and efficacy in predicting the JALSH index. The following subsections outline each step in detail:

3.2.1 Step One: Data Splitting

The dataset is divided into training and testing subsets to support robust model development and evaluation. A ten-year training period, spanning from February 2005 to February 2016, was selected to expose the models to a wide range of market conditions, including periods of extreme volatility (e.g., the 2008 Global Financial Crisis), relative stability (2012–2015), and cyclical recovery. This broad exposure enhances the models' ability to learn structural relationships and adapt to diverse financial regimes.

The initial dataset comprises 210 unique South African listed equities drawn from the JSE All Share Index (JALSH), covering the period from 2005 to 2021. All firms with sufficient accounting and return data were retained, including those that delisted during the sample period. Firms were only dropped if critical variables were entirely missing or unusable. No industry or sector filters were applied.

Importantly, firms were not required to be continuously listed from 2005 to 2021. Instead, the inclusion criterion was based on whether a company had valid data during any portion of the sample window. This approach allows the model to learn from both surviving and non-surviving firms, reducing survivorship bias and enhancing the representativeness of the training dataset.

For instance, 116 unique companies appear in the dataset in 2005, while 167 are observed in 2021. Approximately 43 companies appear in 2005 but not in 2021, indicating likely delisting’s or mergers. Including these firms ensures the model reflects realistic investment risks, including the possibility of failure or market exit, a crucial consideration in forward-looking return prediction.

Overall, the cleaned dataset contains 34,033 firm-month observations, with 21,007 observations covering the training period (2005–2016) and 10,039 observations in the testing period (2017–2021). This comprehensive data coverage across time and firms supports robust model development and validation.

The subsequent five-year testing window, from February 2016 to February 2021, was designed to assess model generalisability under new and contrasting conditions, ranging from the relative calm of 2016–2017 to the severe market dislocation brought about by COVID-19 in 2020 and the ensuing recovery. This structured out-of-sample evaluation period enables a clear measurement of forecasting accuracy across varying environments.

To further evaluate model robustness, the testing period is segmented into three forecast horizons: one year, three years, and five years. This allows performance to be assessed at multiple temporal scales, offering insights into both short-term responsiveness and long-term predictive capacity. While more recent data extending to 2023 was available, this study focuses on a fixed 16-year window to preserve the integrity of a clean training-testing split. This approach is consistent with best practice in machine learning research, where a well-separated and temporally distant test set is essential for evaluating true out-of-sample performance.

3.2.2 Step Two: Model Creation

The machine learning models are developed using TensorFlow, a versatile framework renowned for building deep learning models in Python. Libraries such as NumPy, Pandas, and Scikit-learn aid in data manipulation and processing to ensure optimal model training using the selected training data range.

The neural network model creation process employs TensorFlow's Keras API to construct and train a sequential model for predicting forward returns. Initial data preprocessing standardises feature data with Standard Scaler from scikit-learn, ensuring uniform scaling. Hyperparameters like layer units, learning rate, and layer count are chosen, with the model architecture featuring densely connected layers using ReLU activation and L2 regularisation for overfitting prevention. Model training utilises Adam optimiser and Mean Squared Error (MSE) loss for effective convergence and performance evaluation. The trained neural network model is then used to make predictions for each month's data in the test set, following a sequential process of extracting features, predicting target variables, and compiling predictions with actual values for evaluation.

The LSTM model captures temporal dependencies and long-term memory within JALSH index data using TensorFlow's Keras Sequential model. The implementation focuses on preprocessing the input data, transforming it into a format suitable for LSTM, organised as

(samples, time steps, and features), where each sample corresponds to a time step. By organising the data this way, the LSTM enhances its capability to model dynamic relationships within the data, leading to improved predictive accuracy and generalisation. The architecture includes LSTM layers, Dropout for regularisation, and a Dense layer for output, normalised for consistent scaling. Hyperparameters such as the number of LSTM units, learning rate, and regularisation parameters like dropout rates are carefully selected to optimise performance. Training involves the MSE loss function, Adam optimiser, and early stopping to curb overfitting, with trained models saved for future use. Predictions are made for each month's data in the test set, logging hyperparameters and test loss values for evaluation.

The random forest model leverages the random forest regressor from Scikit-learn's ensemble module, amalgamating multiple decision trees to enhance predictive accuracy. Key hyperparameters are utilised for model performance, including estimators, depth, split, leaf, and features. Predictions mirror the other model processes, applying the model to each month's data for forecasted returns. This iterative process extracts relevant features, predicts target variables, and compiles predictions with actual values into a comprehensive data frame for detailed performance assessment.

The linear regression model, created using Scikit-learn's linear regression module, serves as a straightforward benchmark. This model is developed by defining the features and target variable and fitting the model to the training data. Despite its simplicity, it follows the same monthly prediction methodology as the more complex models. The predictions and actual returns are compiled into a comprehensive data frame, allowing for the evaluation of the model's performance.

3.2.3 Step Three: Fine Tuning Hyperparameters

Hyperparameter optimisation is a crucial step in model creation, helping to mitigate the risk of overfitting or underfitting, both of which can lead to inaccurate predictions. By optimising hyperparameters, the model becomes better equipped to generalise from training data to unseen data, ensuring reliable performance. For this study, hyperparameter optimisation is conducted using Optuna, a framework developed by Akiba et al. (2019: 2626), which employs algorithms like the Tree-structured Parzen Estimator (TPE) to explore hyperparameter search spaces efficiently. For each model, neural network, LSTM, and random forest, Optuna's TPE Sampler systematically sampled a set range of hyperparameter configurations to optimise performance

based on MSE, conducting 100 trials per model. After each trial, the program systematically narrows its search, reducing the MSE until only minimal improvements are observed.

The neural network hyperparameters included optimising layer units, dropout, and learning rates. The LSTM model focused on Layer units, dropout rates, learning rates, and L2 regularisation. In contrast, the random forest model explored and optimised hyperparameters like the number of estimators, maximum tree depth, and minimum samples per leaf. Optuna's pruning mechanism was used to terminate unpromising trials early, enhancing efficiency. Early stopping criteria were also implemented across all models to prevent overfitting and ensure generalisation, guiding the search towards configurations yielding accurate predictions. This systematic approach minimised the MSE between predicted and actual values, improving overall model performance.

3.3 Model Performance Analysis

Key evaluation metrics will be used to compare the performance of the models during the out-of-sample testing period. These metrics are the MSE, the Mean Absolute Error (MAE), the Root Mean Square Error (RMSE), The Directional Accuracy (DA), The Information Coefficient (IC) and the Spearman's IC. These valuation methods have been used in similar machine learning index forecasting papers such as Donghwan et al. (2020:33) and Chaigusin, Chirathamjaree and Clayden, (2008:671). MSE calculates the average of the squares of the errors between predicted and actual values, providing a measure of the overall prediction accuracy by penalising larger errors more. MAE measures the average magnitude of errors between predictions and actual values, disregarding their direction. RMSE quantifies the average error size after squaring the errors, providing an overall assessment of prediction accuracy. DA is the percentage of instances where the predicted and actual returns have the same direction. IC measures the correlation between predicted and actual values, offering insights into the model's predictive power. Spearman's IC assesses the rank correlation between predicted and actual values, indicating how well the model preserves the order of the data. These metrics will help evaluate how well the models' forecasts align with market behaviour.

$$\text{MSE} = \frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2 \qquad (14)$$

$$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n} | y_t - \hat{y}_t | \tag{15}$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2} \tag{16}$$

where:

- n = The number of data points.
- y_t = The actual value at time t.
- $\hat{y}_t$ = The predicted value at time t.

$$DA = \left(\frac{Ncorrect}{Ntotal}\right) \times 100 \tag{17}$$

where:

- Ncorrect is the number of instances where the predicted and actual returns have the same direction (positive, negative, or zero).
- Ntotal is the total number of instances.

$$IC = \frac{Cov(P_{it}, A_{it})}{\sigma P_{it} \sigma A_{it}} \tag{18}$$

where:

- P_{it}= Predicted return for asset i at time t.
- A_{it} = Actual return for asset i at time t.
- Cov(P_{it}, A_{it}) = Covariance between predicted and actual returns across assets at time t.
- σP_{it} = Cross-sectional standard deviation of predicted returns at time t.
- σA_{it} = Cross-sectional standard deviation of actual returns at time t.

$$Spearman's\ IC\ =\ \frac{Cov(Rank(P_{it}), Rank(A_{it}))}{\sigma_{Rank(P_{it})}\sigma_{Rank(A_{it})}} \quad (19)$$

where:

- Rank(P_{it}) = Rank of the predicted return for asset i at time t.
- Rank(A_{it}) = Rank of the actual return for asset i at time t.
- Cov(Rank(P_{it}),Rank(A_{it})) = Covariance between the ranks of predicted and actual returns at time t.
- σRank(P_{it}) = Cross-sectional standard deviation of the ranks of predicted returns at time t.
- σRank(A_{it}) = Cross-sectional standard deviation of the ranks of actual returns at time.

3.4 Cross-sectional Investment and Portfolio Creation

While traditional mathematical metrics offer valuable insights into the statistical accuracy of machine learning models, they may not fully capture the practical implications of such predictions in real-world investment scenarios. These quantitative results, while useful, do not directly translate into actionable strategies for portfolio management. To bridge this gap, this study applies the model's outputs to portfolio construction and evaluates performance in a manner reflective of actual investment decision-making.

Following a methodology adapted from Nakagawa, Abe, and Komiyama (2020:376), the investment evaluation framework is designed to account for the dynamic nature of financial markets. The analysis is segmented into 1-, 3-, and 5-year intervals to examine performance over short-, medium-, and long-term horizons. This segmentation ensures that both stable and volatile market conditions are captured, thereby providing a more comprehensive understanding of the model's robustness and practical utility.

The portfolio creation methodology involves a dynamic investment cycle with monthly investments ignoring all transaction costs. At each time step, t, corresponding to the end of a month, the stock universe Ut represents all the benchmark stocks at that time. In the context of the benchmark JSE All Share Index, i $\in Ut$ comprises a varying number of stocks, reflecting economic activities across various sectors.

Each stock i within Ut has an associated return Rit, representing the unit return of the stock between $t-1$ and t. Additionally, each stock is characterised by the nine listed factors as $xit \in R^9$. The investment strategies employed include:

1. **Long Portfolio Strategy:** This strategy buys the top 20% quintile (i.e. one-fifth) of the stocks with equal weight, aiming to outperform the average return of all stocks. The long portfolio $Lt \subset Ut$ is defined as $|Lt| = \frac{1}{5}|Ut|$ .The return from the long portfolio is the average return of Lt, given by:

$$R_t^L = \frac{1}{|L_t|}\sum_{i \in L_t} R_{it} \tag{20}$$

The long portfolio strategy is straightforward to implement, making it a popular choice among investors. By focusing on the top-performing stocks, it aims to capture the upside potential of the best-performing assets. However, this approach may be susceptible to market downturns as it does not include a hedging mechanism against market declines.

2. **Long-Short Portfolio Strategy:** This strategy not only buys the top 20% quintile of the stocks but also sells the bottom 20% quintile. The short portfolio $St \subset Ut$ is defined as $|St| = \frac{1}{5}|Ut|$. The return from the short portfolio is:

$$R_t^S = \frac{1}{|S_t|}\sum_{i \in S_t} R_{it} \tag{21}$$

The Long-Short portfolio return is thus:

$$R_t^{LS} = R_t^L - R_t^S \tag{22}$$

Due to its market-neutral stance, the long-short portfolio demonstrates resilience during significant market downturns. Balancing long and short positions mitigates overall portfolio risk, fostering stability across diverse market scenarios.

3.5 Portfolio Performance Metrics

Active management testing metrics play a pivotal role in the comprehensive evaluation of investment portfolios, offering insights into their capacity to outperform market benchmarks while managing risks effectively. Critical financial metrics, including Alpha, Tracking Error, and Information Ratio, are fundamental tools for evaluating portfolio performance and informing investment strategies.

These metrics and their formulas, outlined below, are fundamental in evaluating both the Long Portfolio and the Long-Short Portfolio. For the Long Portfolio, the annualised return compares the excess return (Alpha) to the average return of all stocks within the JSE All Share Index. This Alpha is calculated as the difference between the portfolio's actual return and the return expected from the overall market index, providing insight into the portfolio's ability to outperform the market. The risk assessment for the Long Portfolio, measured by Tracking Error, represents the standard deviation of Alpha, offering a gauge of the portfolio's consistency in generating excess returns. The risk/return metric, known as the Information Ratio, is derived from Alpha divided by Tracking Error, indicating how effectively the portfolio balances risk and return.

$$\text{Alpha} = \prod_{t=1}^{T}(1 + at)^{\frac{12}{T}} - 1 \qquad (23)$$

where :

- $at = R_t^L - \frac{1}{|U_t|}\sum_{i \in U_t} R_{it}$
- T = Monthly Periods.
  - 1 year = 12
  - 3 years = 36
  - 5 years = 60

$$Tracking\ Error = \sqrt{\frac{12}{T-1} \times (at - \mu a)^2} \qquad (24)$$

where :

- $\mu a = (\frac{1}{T}) \sum_{t=1}^{T} at$

$$Information\ Ratio = \frac{\text{Alpha}}{\text{Tracking Error}} \tag{25}$$

Similarly, the Long-Short Portfolio strategy enhances risk management by buying the top 20% of stocks and shorting the bottom 20% quintile. This strategy's performance metrics include Annualised Return, which measures the compounded return over time, and Risk, calculated as the standard deviation of returns. The Risk/Return ratio assesses how efficiently the strategy utilises risk to achieve returns, offering a comparative measure against the Long Portfolio's performance metrics.

$$\text{Annualised Return} = \prod_{t=1}^{T} (1 + R_t^{LS})^{\frac{12}{T}} - 1 \tag{26}$$

$$Risk = \sqrt{\frac{12}{T-1} \times (R_t^{LS} - \mu^{LS})^2} \tag{27}$$

where :

- $\mu^{LS} = (\frac{1}{T}) \sum_{t=1}^{T} R_t^{LS}$

$$\text{Risk/Return} = \frac{\text{Annualised Return}}{\text{Risk}} \tag{28}$$

To further explore each portfolio's robustness, Maximum Drawdown (MaxDD) is applied to provide insight into worst-case scenarios during adverse market conditions. It measures the largest potential loss from peak to trough, thus assessing the portfolio's risk profile.

$$MaxDD = min_{k\in[1,T]}\left(0, \frac{W_k^{Port}}{max_{j\in[1,k]}W_j^{Port}} - 1\right) \quad (29)$$

where:

- $W_k^{Port} = \prod_{i=1}^{k}(1 + R_i^{Port})$
- $R_i^{Port} = R_t^L \; or \; R_t^{LS}$

4 Model Optimising and Results

This section expands upon the optimisation process of the machine learning models, with a focus on fine-tuning hyperparameters using the Optuna framework. Following the Optimisation, a comprehensive analysis of the model predictions is presented, assessing their performance in predicting returns for JSE All Share constituents.

4.1 Neural Network Optimisation

The hyperparameter optimisation process using Optuna unveiled crucial insights into the relative importance of key hyperparameters, guiding the selection of optimal configurations for the neural network model. The hyperparameter importance values in Figure 7 below shed light on each parameter's varying influence on model performance.

The first hidden layer (Units 2) emerged as the most influential factor, with an importance value of 0.68, indicating that variations in the number of units in the hidden layer significantly impact the model's ability to capture complex patterns and enhance predictive accuracy. This finding underscores the importance of fine-tuning the architecture at a granular level to achieve optimal performance. The number of layers (N layers) played a substantial role, albeit with a lesser importance value of 0.27, suggesting that the depth of the neural network architecture contributes significantly to model Optimisation but to a slightly lesser extent compared to the specific configuration of individual layers. Surprisingly, the learning rate (Learning Rate) demonstrated a comparatively lower importance value of 0.06, signifying that while crucial for training stability and convergence, its impact on overall model Optimisation was relatively minor.

Figure 7: Neural Network Hyperparameter Importance

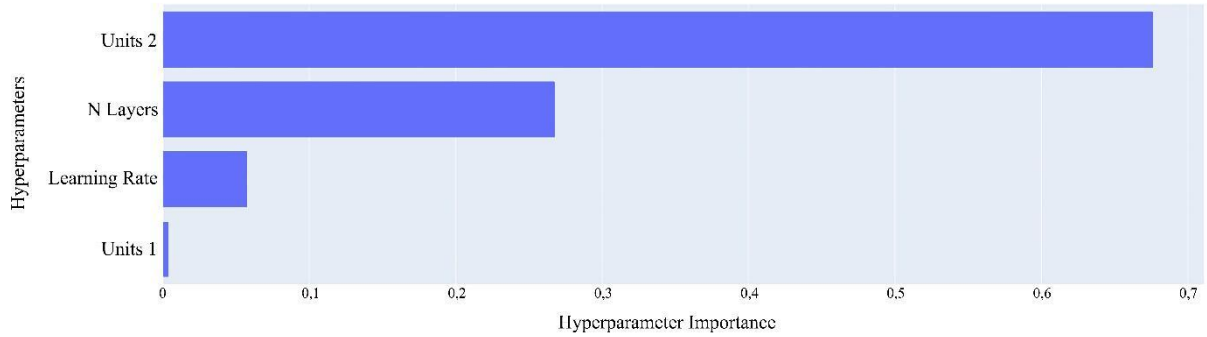


Figure 7 illustrates the importance of different hyperparameters in a neural network, as determined by the Optuna tests. The bars represent the significance of each hyperparameter in reducing the mean squared error, with larger bars indicating greater importance. According to the graph, "Units 2" has the highest importance, followed by "N Layers," "Learning Rate," and "Units 1," (first layer) which has the lowest importance.

Understanding the nuanced relationships between hyperparameters and model performance allowed for the strategic prioritisation of optimising the number of hidden layers and specific layers. This led to selecting hyperparameter configurations that maximise predictive accuracy and generalisation performance for financial forecasting tasks.

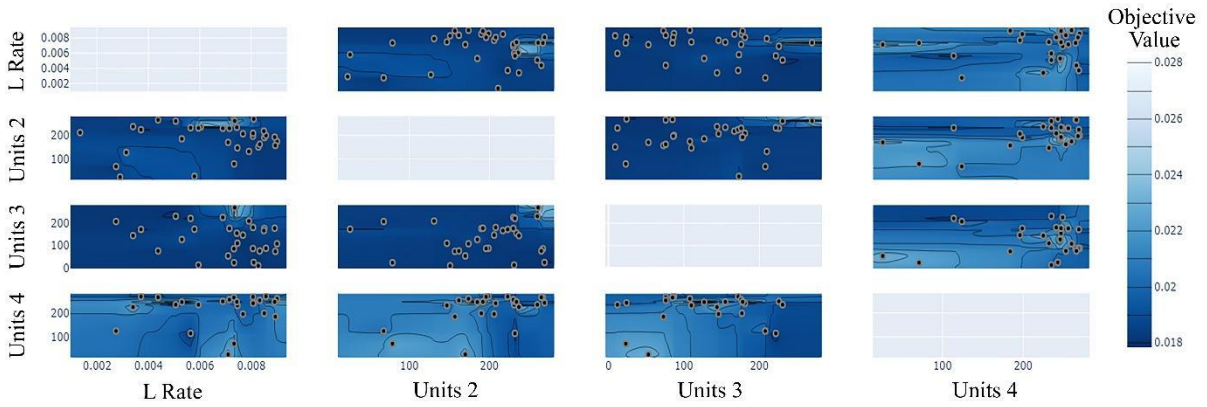
An in-depth analysis of the contour plots seen below in Figure 8 further elucidates the influence of hyperparameters on objective values concerning the learning rate (L Rate). A notable feature in the first hidden layer is the wide flattening of contour lines, indicating a stable region of hyperparameter configurations that yield low validation losses. This stability extends across a range of the layer's values, with a subtle increase in gradient observed between 230 and 260 units, particularly within a learning rate range of 0.006 to 0.007.

Transitioning to the second hidden layer (Units 3), the contour plots illustrate an even flatter gradient, emphasising the model's robustness to changes in this layer's configuration. This observation suggests that variations in the second hidden layer have minimal impact on model performance, showcasing the model's resilience to changes in this aspect of architecture.

Contrastingly, the third hidden layer exhibits significant gradient changes concerning the learning rate, highlighting the importance of fine-tuning hyperparameters in deeper layers for optimal model convergence and predictive accuracy. Specific combinations of the third hidden layer units and learning rate led to optimal model performance, with a notable optimum observed at 114 units.

Leveraging insights from the contour plots, strategic navigation of the hyperparameter space identifies key regions of stability and sensitivity, guiding the selection of optimal configurations for the neural network model in financial forecasting tasks.

Figure 8: Neural Network Contour Plot Matrix



The contour plot matrix above illustrates the varying trials conducted during the Optuna test. The goal of the plot is to visualise how different combinations of the hyperparameters influence the objective function's value, helping to identify the darkest region where the best performance (lowest objective value) can be achieved.

The Optimisation History plot below tracks the hyperparameter Optimisation process, revealing insights into model performance refinement. Initially marked by trial 1 with a higher objective value of 0.019, the model undergoes rapid improvements from trial 1 to trial 7, indicating exploration of promising hyperparameter combinations. Subsequent trials show a flattening gradient and smaller decreases in objective value, especially around trial 36, signifying finer adjustments leading to improved model performance. Trial 48 represents the culmination of this process, showcasing optimal values for validation loss achieved through strategic hyperparameter tuning efforts.

The most optimal hyperparameter configuration for the neural network, identified at trial 48, includes 232 units in the first hidden layer, 222 units in the second hidden layer, and 114 units in the third hidden layer, with a learning rate of 0.0056. This configuration reflects the model's peak performance, capturing underlying patterns and enhancing predictive accuracy for financial forecasting tasks.

Figure 9: Neural Network Optimisation History Plot

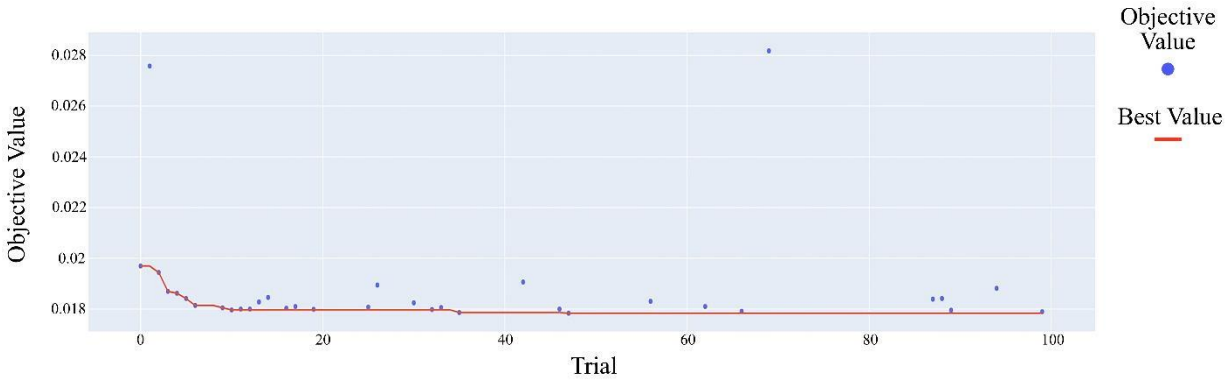


Figure 9 illustrates a line graph in which the red line tracks the best objective value achieved, showing how it improves over successive trials, while the blue dots represent the objective values of individual trials, highlighting the optimisation process.

4.2 Random Forest Optimisation

Building upon the structured hyperparameter exploration for the random forest, the Hyperparameter Importance test revealed that the minimum number of samples required to be at a leaf node in a decision tree significantly influenced the performance of the random forest model, with an importance of 97%, followed by the maximum depth of a decision tree at 2%. These insights were further clarified through the contour plot analysis below, which elucidated the impact of these hyperparameters on the model's performance. The Y-axis represents Min Samples Leaf, and the X-axis depicts Max Depth, revealing a colour gradient where darker shades signify closer proximity to the objective value (MSE) and lighter shades indicate greater deviation. Notably, the contour plot highlights a region of significantly darker colour encompassing a maximum depth range of 1 to 10 and a minimum sample leaf range of 3 to 5. This alignment between hyperparameter importance and the contour plot underscores the critical role of 'Min Samples leaf' and 'Max Depth' in optimising the random forest model's accuracy.

Figure 10: Min Sample Leaf and Max Depth Contour Plot

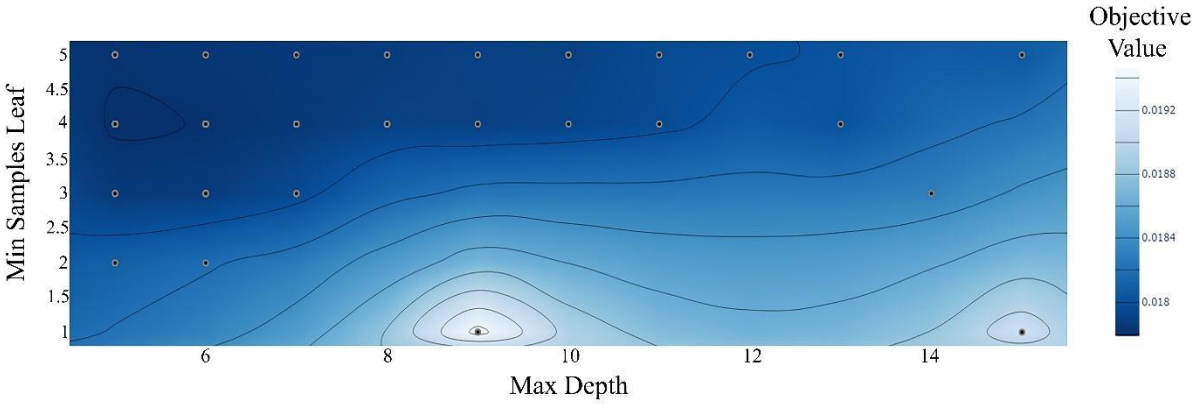


Figure 10 depicts a contour plot where the colour gradient indicates the objective value across different combinations of 'Max Depth' and 'Min Samples Leaf'. The plot highlights the regions where optimal values are achieved, with darker areas indicating lower objective values, suggesting better model performance.

The Parallel Coordinate Plot provides a comprehensive view of how various hyperparameters influence the performance of the random forest model. A notable finding is the cluster of dark lines indicating favourable objective values, particularly within a maximum depth range of 5 to 7. This suggests moderate depths are optimal, balancing model complexity and overfitting. Lines also gravitate towards higher maximum feature values (0.8 to 0.99), emphasising the importance of considering diverse features during training. The clustering around a minimum sample leaf setting of 4 also underscores its significance for optimal model accuracy. The plot also shows that having 120 to 145 estimators positively impacts performance. In summary, the optimal hyperparameters identified are 135 estimators, a maximum depth of 5, minimum samples split at 4, minimum samples leaf set at 4, and maximum features at 0.95. This configuration maximises performance and mitigates overfitting, showcasing a balanced and robust approach to random forest model tuning.

Figure 11: Random Forest Parallel Coordinate Plot

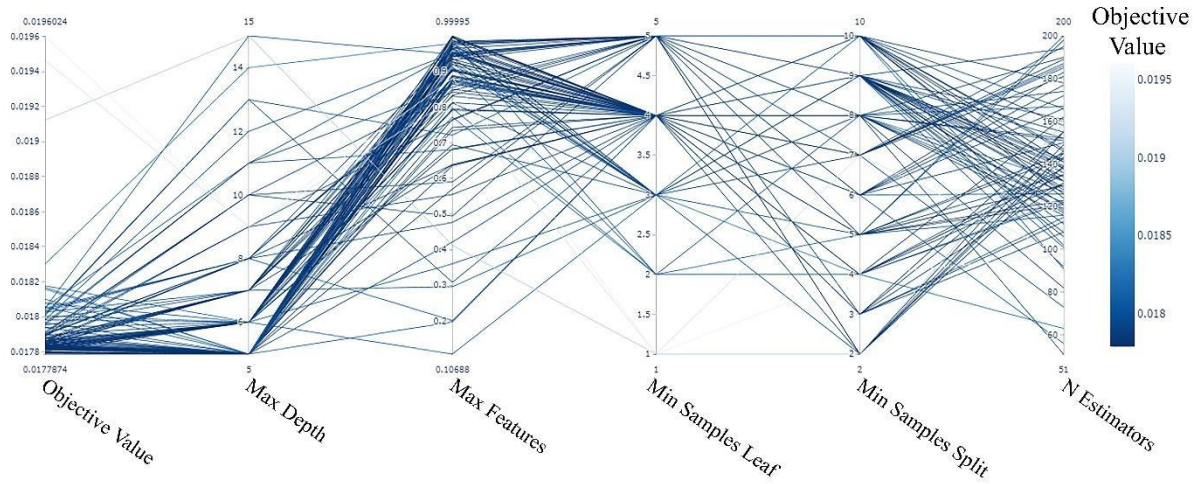


Figure 11 illustrates a parallel coordinate plot visualising the relationships between multiple hyperparameters and the objective value across different trials. The lighter-coloured lines represent less successful trials with higher mean squared errors, while the darker lines indicate better-performing trials. This plot helps identify patterns and trade-offs, making it easier to understand how different hyperparameter combinations impact model performance.

4.3 LSTM Optimisation

In the Optuna study conducted for the LSTM model, the Hyperparameter Importance graph below unveiled pivotal insights crucial for model optimisation. L2 Regularization emerged as the most influential hyperparameter, boasting an importance value of 0.53. This underscores its role in mitigating overfitting and enhancing model generalisation. The learning rate exhibited an impact of 0.19, showcasing its crucial role in fine-tuning the model's training dynamics. Moreover, the dropout rate at 0.15 and the number of units at 0.14 demonstrated their respective contributions to enhancing model robustness and managing model complexity.

Figure 12: LSTM Hyperparameter Importance

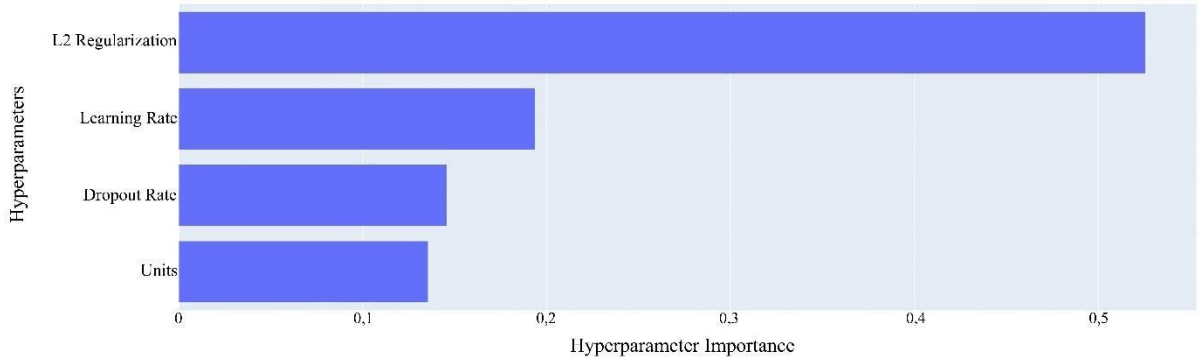


Figure 12 illustrates the importance of different hyperparameters in a LSTM, as determined by the Optuna tests. The bars represent the significance of each hyperparameter in reducing the mean squared error, with larger bars indicating greater importance. According to the graph, "L2 Regularization" has the highest importance, followed by "Learning Rate," "Dropout Rate," and "Units," which has the lowest importance.

The Optimisation History Plot below shows a notable trend where most optimal hyperparameters are identified within the first 20 trials. This trend is characterised by a sharp drop-off in Objective Value from trial 1 to trial 12, indicating rapid progress in finding promising configurations early in the optimisation process. Subsequently, the performance metrics stabilise within a narrow range of 0.0177 to 0.0178 until trial 51, where the most optimal hyperparameters are discovered. These optimal hyperparameters include units set at 102.0, dropout rate at 0.1046, learning rate at 0.0053, and L2 regularisation at 4.7185e-06. This specific configuration represents the best-performing setup for the LSTM model based on the Optimisation process.

Figure 13: LSTM Optimisation History Plot

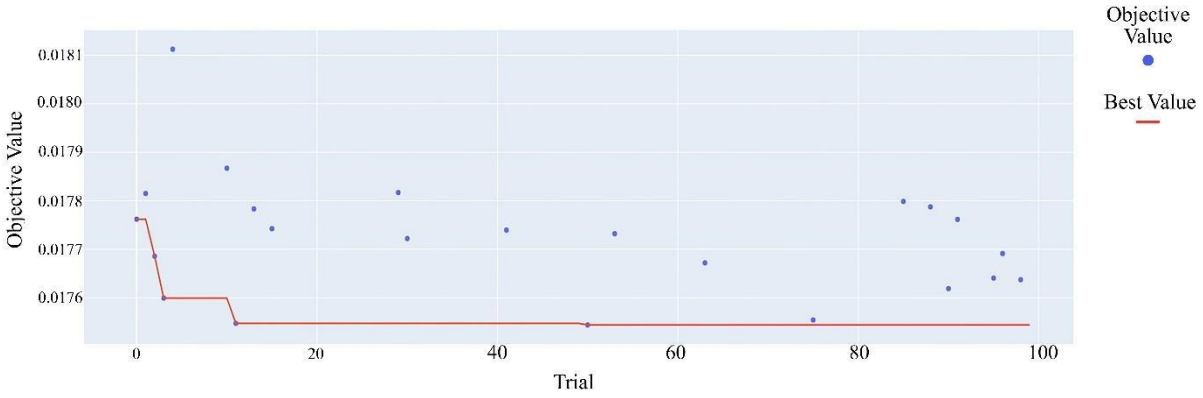


Figure 13 illustrates a line graph where the red line tracks the best objective value achieved, showing how it improves over successive trials, while blue dots represent the objective values of individual trials, highlighting the optimisation process.

4.4 Results

The conclusion of the hyperparameter optimisation phase marked a pivotal shift toward practical testing and analysis. This segment involved rigorously running and evaluating all models to assess their effectiveness in predictive analytics. The following section explores the findings and empirical results from these evaluations, examining both stringent statistical metrics and their practical applicability within the JSE All Share Index.

Table 2: Evaluation Metrics

	Linear regression	**Neural network**	**Random forest**	**LSTM**
Mean Squared Error	0.0185	0.0297	0.0178	0.0177
Root Mean Squared Error	0.1359	0.1722	0.1334	0.1329
Mean Absolute Error	0.0827	0.1006	0.0819	0.0810
Directional Accuracy	0.4974	0.4941	0.5062	0.5189
Information Coefficient	-0.0134	0.0423	0.1317	0.1337
Spearman's IC	0.0039	0.0130	0.0526	0.0708

Table 3 displays the out-of-sample prediction accuracy results for all six-performance metrics. Each row represents an overall prediction accuracy metric, while each column corresponds to a different machine-learning model.

Table 3 comprehensively evaluates the models for predicting returns within the JSE. The LSTM demonstrated the most accurate predictions among these models, with the lowest MSE of 0.0177 and RMSE of 0.1329. Additionally, its MAE of 0.0810 and Spearman's IC of 0.0708 showcased superior predictive capabilities. Its Directional Accuracy of 51.89% indicated a stronger ability to predict the direction of stock return movements than other models.

The random forest model demonstrated competitive performance with an MSE of 0.0178, RMSE of 0.1334, Directional Accuracy of 50.62%, IC of 0.1317, and Spearman's IC of 0.0526. In comparison, the linear regression and neural network models had higher errors, lower directional accuracy, and lower Spearman's IC values. The linear regression model showed an MSE of 0.0185, RMSE of 0.1359, Directional Accuracy of 49.74%, Information Coefficient of -0.0134, and Spearman's IC of 0.0039. Similarly, the neural network model had an MSE of 0.0297, RMSE of 0.1722, Directional Accuracy of 49.41%, IC of 0.0423, and Spearman's IC of 0.0130.

These results suggest that the LSTM and random forest models offer superior predictive capabilities for returns within the JSE, outperforming traditional linear regression and neural network approaches across multiple metrics. The combination of lower error rates, higher correlation with actual returns, and robustness to nonlinear relationships, as indicated by higher Information Coefficient and Spearman's IC values, positions the LSTM model as a promising choice for financial forecasting tasks in the JSE context. Additionally, the random forest model's strong performance across these metrics highlights its potential, especially in capturing complex patterns and reducing prediction errors.

Unexpectedly, the neural network model demonstrated the least effectiveness among the evaluated models in predicting returns within the JSE. This result could be attributed to several factors inherent to neural networks, such as their susceptibility to overfitting, complex architecture leading to increased computational requirements, and challenges in interpreting the model's decisions compared to other models and the simpler linear regression model. These complexities might have hindered the neural network's ability to generalise unseen data well, resulting in higher prediction errors and lower directional accuracy.

Although the quantitative results provide valuable insights into model performance, their direct translation into actionable strategies for active portfolio management remains limited. However, these results lay the foundation for further exploration and experimentation. The subsequent section focuses on constructing portfolios using these predictive models and

evaluating their performance to assess the model's practical applicability in investment decision-making within the JSE.

Table 3: Long Only Portfolio Breakdown

Long Only Portfolio				
Time (Years)	Linear regression	Neural network	Random forest	LSTM
Alpha				
1	-0.0876	0.1864	0.0021	0.0500
3	-0.0439	0.0465	0.0659	0.0963
5	0.0193	0.0624	0.1616	0.1861
Tracking Error				
1	0.0837	0.1794	0.0018	0.0478
3	0.0236	0.0248	0.0352	0.0517
5	0.0078	0.0256	0.0669	0.0772
Information Ratio				
1	-1.0465	1.0388	1.1665	1.0458
3	-1.8583	1.8785	1.8694	1.8623
5	2.4899	2.4364	2.4153	2.4098
Maximum Drawdown				
1	0.1096	0.0207	0.0592	0.0241
3	0.1599	0.1979	0.0779	0.0553
5	0.2132	0.1979	0.1888	0.1087

Table 4 provides a breakdown of the Long Only Portfolio performance over different time horizons (1 year, 3 years, and 5 years in column 1) for each model: linear regression, neural network, random forest, and LSTM. The metrics include Alpha, Tracking Error, Information Ratio, and Maximum Drawdown, each represented by its subsection within the table.

An analysis of the model performance for the Long Only Portfolio strategy reveals notable trends and characteristics, as detailed in Table 3. The LSTM model demonstrates strong performance across multiple metrics, particularly in generating superior excess returns and achieving notably lower Maximum Drawdowns. Its adeptness at adjusting to market dynamics is evident during periods of heightened volatility, such as the disruptions caused by the COVID-

19 pandemic in 2020, as well as subsequent challenges from the Russia-Ukraine war, geopolitical uncertainty, and the global energy crisis. These characteristics underscore the LSTM model's potential as a robust investment tool, particularly for investors with a long-term horizon prioritising absolute returns.

The random forest model also performs competitively, particularly in risk-adjusted metrics. It achieves a slightly higher Information Ratio than LSTM across all horizons, indicating better returns per unit of risk. While its Alpha is lower than LSTM's, its lower Tracking Error and stable risk management enhance its attractiveness for risk-averse investors. The random forest model's capacity to exploit market opportunities during volatile periods, such as those in 2020 and beyond, further supports its suitability for a Long Only Portfolio strategy.

The neural network model generates strong excess returns in the first year, but its performance diminishes over longer periods, particularly during increased market volatility. Its improving risk-adjusted performance and decreasing risk profile are notable, making it suitable for short-term strategies and adapting to market fluctuations.

In contrast, the linear regression model struggles to generate positive returns and maintain favorable risk-adjusted performance, limiting its suitability for a Long Only Portfolio strategy. Its challenges in managing risk during volatile periods render it less competitive compared to LSTM, random forest, and neural network models.

Given the period of stability up to 2020, followed by a structurally volatile market environment, the LSTM, random forest, and neural network models exhibit critical characteristics such as adaptability and robust risk management. While LSTM excels in absolute returns and drawdown management, the random forest's slightly superior information ratio makes it a compelling alternative for risk-adjusted strategies. These attributes position both models as suitable choices for a Long Only Portfolio strategy based on the JSE All Share Index across various market conditions.

Table 4: Long-Short Portfolio Breakdown

Long-Short Portfolio				
Time (Years)	Linear regression	Neural network	Random forest	LSTM
Annualised Return				
1	-0.2417	0.3194	-0.0791	0.0359
3	-0.1144	0.1113	0.1449	0.2092
5	-0.0149	0.1691	0.3188	0.3855
Risk				
1	0.2295	0.3086	0.0759	0.0340
3	0.0618	0.0596	0.0777	0.1127
5	0.0068	0.0698	0.1328	0.1609
Risk/Return				
1	-1.0532	1.0351	-1.0419	1.0545
3	-1.8518	1.8685	1.8647	1.8557
5	-2.1933	2.4232	2.4015	2.3958
Maximum Drawdown				
1	0.1595	0.0134	0.1012	0.0731
3	0.2890	0.2303	0.1150	0.0791
5	0.3253	0.3069	0.2660	0.3260

Table 5 shows the Long-Short Portfolio performance over different time horizons (1 year, 3 years, and 5 years in column 1) for each model: linear regression, neural network, random forest, and LSTM. The metrics include Annualised Return, Risk, Risk/Return, and Maximum Drawdown, each represented by its subsection within the table.

After analysing the performance of the models for a Long-Short Portfolio strategy, as detailed in Table 4, the LSTM model emerges as the most suitable choice, though it slightly trails the random forest model in risk-adjusted returns at the 3-year horizon. The LSTM model demonstrates a consistent upward trajectory in annualised returns, indicating its strong potential for capturing market trends effectively. This characteristic, combined with its competitive Risk/Return ratios, makes it a strong candidate for a Long-Short Portfolio strategy, particularly in navigating volatile market conditions post-2020, as observed in the broader market context.

The random forest model presents a competitive performance, particularly in risk-adjusted metrics, achieving a slightly higher Risk/Return ratio at the 3-year horizon comparable performance at 5 years, driven by lower risk values. While its annualised returns are generally lower than LSTM's, its stability in risk management during volatile periods enhances its attractiveness for risk-averse investors pursuing a Long-Short Portfolio strategy.

The neural network model exhibits strong initial performance in Annualised Returns but shows a decline over longer periods. Its improving risk profile and Risk/Return ratios make it suitable for short-term strategies in volatile market conditions.

In contrast, the linear regression model struggles to generate positive annualised returns and maintain favourable risk-adjusted performance. Its difficulties in managing risk during volatile periods render it less suitable compared to LSTM, random forest, and neural network models for a Long-Short Portfolio strategy.

Among the evaluated machine learning models for forecasting returns within the JSE using financial panel data, the LSTM model stands out as the optimal choice for a Long-Short Portfolio strategy. Its superior Annualised Returns, competitive risk-adjusted performance, and robust predictive accuracy set it apart. While Random Forest offers a slight edge in risk-adjusted performance at the 3-year horizon, LSTM's consistent outperformance in returns and adaptability to volatile market dynamics make it the preferred model for generating accurate forecasted returns and effective risk management within the dynamic environment of the JSE All Share Index.

5 Further Research and Improvements

Since the LSTM model achieved the highest predictive accuracy while exhibiting comparable risk-return performance to the random forest model, future research could investigate whether advanced deep learning architectures can enhance both predictive power and portfolio outcomes. Promising candidates include Bidirectional LSTM (Bi-LSTM), Convolutional Neural Network LSTM (CNN-LSTM), and Symbolic Genetic Programming LSTM (SGP-LSTM). Wang (2025) demonstrated that Bi-LSTM reduced RMSE by 15–20% compared to standard LSTM when forecasting Google stock data, while a CNN-LSTM model developed by Yang (2025) achieved significantly lower MSE than ARIMA and standalone LSTM in predicting 54 financial indicators for A-share companies. These architectures may offer improved stability and adaptability, particularly in volatile market environments like those of the JSE.

In parallel with model enhancements, future research could refine the factor set used in forecasting. While this study incorporates key variables such as EPS and CFP, additional metrics, including revenue growth, gross margin, operating margin, and net margin, may provide a more comprehensive view of firm-level efficiency and profitability. Furthermore, risk can be explored more thoroughly by complementing total risk measures such as retvar with indicators of systematic risk like Beta. This aligns with the findings of Van Rensburg and Robertson (2003), who highlight the relevance of Beta in the South African market. These refinements could enhance both the interpretability and robustness of predictive models.

Further improvements may also be achieved through more sophisticated portfolio construction techniques. While this study employed equally weighted portfolios as a baseline, alternative approaches, such as market capitalisation weighting, minimum variance, or momentum-based strategies, could better align with different investor risk preferences. Additionally, portfolio optimisation methods like the Subset Resampling Portfolio (Shen & Wang, 2017) may enhance risk-adjusted performance by reducing estimation error and maximising the utility of predictive signals across larger asset pools.

Collectively, these avenues offer valuable opportunities to strengthen both the theoretical foundation and practical applications of machine learning in emerging market equity forecasting and portfolio design.

6 Conclusion

This thesis evaluated machine learning models for predicting stock returns within the JSE All Share Index, focusing on linear regression, neural networks, random forests, and LSTM models. The analysis emphasised hyperparameter optimisation and model performance across multiple statistical metrics. The most optimal hyperparameters found for each model include:

- Neural network: 232 units in the first hidden layer, 222 units in the second hidden layer, and 114 units in the third hidden layer, with a learning rate of 0.0056.
- Random forest: 135 estimators, a maximum depth of 5, a minimum sample split of 4, a minimum sample leaf set of 4, and maximum features at 0.95.
- LSTM: 102 units, a dropout rate of 0.1046, a learning rate of 0.0053, and L2 regularisation at 4.7185e-06.

The test findings reveal that the LSTM model consistently outperformed the other models in predictive accuracy, achieving the lowest MSE of 0.0177, RMSE of 0.1329, and MAE of 0.0810. It also recorded the highest Spearman's IC of 0.0708 and a Directional Accuracy of 51.89%, reflecting its effectiveness in capturing return correlations and directions. The random forest model performed competitively, with an MSE of 0.0178, MAE of 0.0819, Directional Accuracy of 50.62%, and a strong Information Coefficient of 0.1317, though it slightly trailed LSTM in overall accuracy.

These models' practical applicability was assessed by constructing equally weighted Long Only and Long-Short Portfolios. The LSTM model demonstrated strong performance, characterised by high Alpha and Annualised Returns, low Maximum Drawdowns, and competitive risk-adjusted metrics, particularly in volatile market conditions driven by the COVID-19 pandemic, Russia-Ukraine war, geopolitical uncertainty, and global energy crisis. The random forest model, while slightly superior in risk-adjusted metrics, achieving a higher Information Ratio in the Long Only Portfolio across all horizons and a marginally better Risk/Return ratio at the 3-year horizon in the Long-Short Portfolio, generally trailed LSTM in absolute returns. The neural network model showed promise for short-term strategies with strong initial returns and improving risk profiles, but was less consistent over longer horizons. The linear regression model underperformed, struggling with positive returns and risk-adjusted performance, especially during volatile periods.

These findings highlight the critical importance of model selection in financial forecasting. The LSTM model's superior accuracy, high returns, and adaptability to volatile market dynamics make it the preferred choice for investment strategies in emerging markets like the JSE. However, the random forest model's competitive risk-adjusted performance offers a compelling alternative for risk-averse investors. This study demonstrates that careful hyperparameter optimisation and model selection can significantly enhance predictive performance, providing valuable insights for practitioners and researchers alike.

7 Reference List

Abraham, R. Samad, M.E. Bakhach, A.M. El-Chaarani, H. Sardouk, A. Nemar, S.E. and Jaber, D. 2022. Forecasting a stock trend using genetic algorithm and random forest. *Journal of Risk and Financial Management*, *15*(5), p.188.

Adrian, T., Crump, R.K. and Moench, E., 2013. Pricing the term structure with linear regressions. *Journal of Financial Economics*, *110*(1), pp.110-138.

Akiba, T. Sano, S. Yanase, T. Ohta, T. and Koyama, M. 2019, July. Optuna: A next-generation hyperparameter Optimisation framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* pp. 2623-2631.

Alexandroi, S., 2019. Cash Flow as a Predictor of Share Returns: Evidence from the Johannesburg Stock Exchange.

Amit, Y. and Geman, D. 1997. Shape quantization and recognition with randomized trees. *Neural computation*, *9*(7), pp.1545-1588.

Ang, A. and Bekaert, G., 2007. Stock return predictability: Is it there?. *The Review of Financial Studies*, 20(3), pp.651-707.

Balusik, A. de Magalhaes, J. and Mbuvha, R. 2021. Forecasting The JSE Top 40 Using Long Short-Term Memory Networks. *arXiv preprint arXiv:2104.09855*.

Basu, J.K. Bhattacharyya, D. and Kim, T.H. 2010. Use of artificial neural network in pattern recognition. *International journal of software engineering and its applications*, *4*(2), pp. 1-34

Biau, G. 2012. Analysis of a random forests model. *The Journal of Machine Learning Research*, *13*, pp.1063-1095.

Breiman, L. 2001. Random forests. *Machine learning*, *45*, pp.5-32.

Brownlee, J. 2019. How to configure the learning rate when training deep learning neural networks. *Machine Learning Mastery*, *6*.

Cakici, N., Fieberg, C., Metko, D. and Zaremba, A., 2023. Machine learning goes global: Cross-sectional return predictability in international stock markets. *Journal of Economic Dynamics and Control*, *155*, p.104725.

Cao, Q. Leggio, K.B. and Schniederjans, M.J. 2005. A comparison between Fama and French's model and artificial neural networks in predicting the Chinese stock market. *Computers & Operations Research*, *32*(10), pp.2499-2512

Chaigusin, S. Chirathamjaree, C. and Clayden, J. 2008, December. The use of neural networks in the prediction of the stock exchange of Thailand (SET) Index. In *2008 International Conference on Computational Intelligence for Modelling Control & Automation* (pp. 670-673). IEEE.

Charteris, A. and Strydom, B., 2016. Stock return predictability in South Africa: An alternative approach. *Economic Research Southern Africa (ERSA) working paper*, *608*, pp.1-21.

Chen, J.M. 2021. An introduction to machine learning for panel data. *International Advances in Economic Research*, *27*(1), pp.1-16.

Chen, Y. and Hao, Y. 2017. A feature weighted support vector machine and K-nearest neighbor algorithm for stock market indices prediction. *Expert Systems with Applications*, *80*, pp.340-355.

Chikwira, C. and Mohammed, J.I. 2023. The Impact of the Stock Market on Liquidity and Economic Growth: Evidence of Volatile Market. *Economies*, *11*(6), p.155.

Chu, H.H. Chen, T.L. Cheng, C.H. and Huang, C.C. 2009. Fuzzy dual-factor time-series for stock index forecasting. *Expert systems with applications*, *36*(1), pp.165-171.

Collins, C. Dennehy, D. Conboy, K. and Mikalef, P. 2021. Artificial intelligence in information systems research: A systematic literature review and research agenda. *International Journal of Information Management*, *60*, p.102383.

Dietterich, T.G. 2000. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine learning*, *40*, pp.139-157.

Dongare, A.D. Kharde, R.R. and Kachare, A.D. 2012. Introduction to artificial neural network. *International Journal of Engineering and Innovative Technology (IJEIT)*, *2*(1), pp.189-194.

Donghwan, S. Busogi, M. Chung Baek, A.M. and Kim, N. 2020. Forecasting stock market index based on pattern driven long short-term memory. *Economic Computation & Economic Cybernetics Studies & Research*, *54*(3).

Fama, E.F., 1970. Efficient capital markets. *Journal of finance*, 25(2), pp.383-417.

Fama, E.F. and French, K.R., 1988. Dividend yields and expected stock returns. *Journal of financial economics*, 22(1), pp.3-25.

Fama, E.F. and French, K.R., 1993. Common risk factors in the returns on stocks and bonds. *Journal of financial economics*, *33*(1), pp.3-56.

Foerster, S., Tsagarelis, J. and Wang, G., 2017. Are cash flows better stock return predictors than profits?. *Financial Analysts Journal*, *73*(1), pp.73-99.

Gers, F.A. Schmidhuber, J. and Cummins, F. 2000. Learning to forget: Continual prediction with LSTM. *Neural computation*, *12*(10), pp.2451-2471.

Ghosh, P. Neufeld, A. and Sahoo, J.K. 2022. Forecasting directional movements of stock prices for intraday trading using LSTM and random forests. *Finance Research Letters*, *46*, p.102280.

Gislason, P.O. Benediktsson, J.A. and Sveinsson, J.R. 2006. Random forests for land cover classification. *Pattern recognition letters*, *27*(4), pp.294-300.

Grossi, E. and Buscema, M. 2007. Introduction to artificial neural networks. *European journal of gastroenterology & hepatology*, *19*(12), pp.1046-1054.

Gupta, M. and Pandya, S.D. 2022. A Comparative Study on Supervised Machine Learning Algorithm. *International Journal for Research in Applied Science and Engineering Technology (IJRASET)*, *10*(1), pp.1023-1028.

Gupta, T.K. and Raza, K. 2020. Optimizing deep feedforward neural network architecture: A tabu search based approach. *Neural Processing Letters*, *51*, pp.2855-2870.

Ho, T.K. 1998. The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, *20*(8), pp.832-844.

Hochreiter, S. and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*, *9*(8), pp.1735-1780.

Hochreiter, S. 1991. Untersuchungen zu dynamischen neuronalen Netzen. *Diploma, Technische Universität München*, *91*(1), p.31.

Hjalmarsson, E., 2010. Predicting global stock returns. *Journal of Financial and Quantitative Analysis*, *45*(1), pp.49-80.

Inthachot, M. Boonjing, V. and Intakosum, S. 2015. Predicting SET50 index trend using artificial neural network and support vector machine. In *Current Approaches in Applied Artificial Intelligence: 28th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2015, Seoul, South Korea, June 10-12, 2015,* pp. 404-414.

Johannesburg Stock Exchange , 2024. All Share (J203). Available at: https://www.jse.co.za/all-share-j203 (Accessed 08 March 2024).

Kara, Y. Boyacioglu, M.A. and Baykan, Ö.K. 2011. Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange. *Expert systems with Applications*, *38*(5), pp.5311-5319.

Khaidem, L. Saha, S. and Dey, S.R. 2016. Predicting the direction of stock market prices using random forest. *arXiv preprint arXiv:1605.00003*.

Landi, F. Baraldi, L. Cornia, M. and Cucchiara, R. 2021. Working memory connections for LSTM. *Neural Networks*, *144*, pp.334-341.

Lee, T.H. Ullah, A. and Wang, R. 2020. Bootstrap aggregating and random forest. *Macroeconomic forecasting in the era of big data: Theory and practice*, pp.389-429.

Lillicrap, T.P. Santoro, A. Marris, L. Akerman, C.J. and Hinton, G. 2020. Backpropagation and the brain. *Nature Reviews Neuroscience*, *21*(6), pp.335-346.

Mahesh, B. 2020. Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, *9*(1), pp.381-386.

Markowitz, H. 1952. Portfolio Selection. *The Journal of Finance*, *7*(1), pp.77–91.

Marwala, L.R. 2010. *Forecasting the stock market index using artificial intelligence and techniques* (Doctoral dissertation).

Mitchell, T.M. and Mitchell, T.M., 1997. *Machine learning* (Vol. 1, No. 9). New York: McGraw-hill.

Nakagawa, K., Abe, M. and Komiyama, J., 2020, October. Ric-nn: A robust transferable deep learning framework for cross-sectional investment strategy. In 2020 IEEE 7th *International Conference on Data Science and Advanced Analytics* (DSAA) (pp. 370-379). IEEE.

Rumelhart, D.E. Hinton, G.E. and Williams, R.J. 1986. Learning representations by back-propagating errors. *nature*, *323*(6088), pp.533-536.

Sarkar, M. and De Bruyn, A. 2021. LSTM response models for direct marketing analytics: Replacing feature engineering with deep learning. *Journal of Interactive Marketing*, *53*(1), pp.80-95.

Sazli, M.H. 2006. A brief review of feed-forward neural networks. *Communications Faculty of Sciences University of Ankara Series A2-A3 Physical Sciences and Engineering*, *50*(01).

Schmidt, R.M., 2019. *Recurrent neural networks (rnns): A gentle introduction and overview*. arXiv preprint arXiv:1912.05911.

Selvin, S. Vinayakumar, R. Gopalakrishnan, E.A. Menon, V.K. and Soman, K.P. 2017, September. Stock price prediction using LSTM, RNN and CNN-sliding window model. In *2017 international conference on advances in computing, communications and informatics (icacci)* (pp. 1643-1647). IEEE.

Sen, J. Dutta, A. and Mehtab, S. 2021, May. Profitability analysis in stock investment using an LSTM-based deep learning model. In *2021 2nd International Conference for Emerging Technology (INCET)* (pp. 1-9). IEEE.

Shen, W. and Wang, J. 2017, February. Portfolio selection via subset resampling. risk. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 31, No. 1).

Siami-Namini, S. Tavakoli, N. and Namin, A.S. 2018, December. A comparison of ARIMA and LSTM in forecasting time series. In *2018 17th IEEE international conference on machine learning and applications (ICMLA)* (pp. 1394-1401). IEEE.

Szczygielski, J.J. and Chipeta, C., 2023. Properties of returns and variance and the implications for time series modelling: Evidence from South Africa. *Modern Finance*, *1*(1), pp.35-55.

Ur Rehman, H. and Gul, F., 2017. Stock Return Predictability Using Panel Regression: Empirical Evidence from Pakistani Equity Market. *Journal of Managerial Sciences*, *11*(2).

Uyanık, G.K. and Güler, N. 2013. A study on multiple linear regression networks. analysis. *Procedia-Social and Behavioral Sciences*, *106*, pp.234-240.

Uzair, M. and Jamil, N. 2020, November. Effects of hidden layers on the efficiency of neural networks. In *2020 IEEE 23rd international multitopic conference (INMIC)* (pp. 1-6). IEEE.

Van Rensburg, P. and Robertson, M., 2003. Size, price-to-earnings and beta on the JSE Securities Exchange. *Investment Analysts Journal*, *32*(58), pp.7-16.

Wang, H., 2025. Enhancing Stock Price Forecasting Accuracy Using LSTM and Bi-LSTM Models. *In ITM Web of Conferences* (Vol. 70, p. 04008). EDP Sciences.

Xu, S. and Chen, L. 2008. A novel approach for determining the optimal number of hidden layer neurons for FNN's and its application in data mining.

Yang, A., 2025. Big data-driven corporate financial forecasting and decision support: a study of CNN-LSTM machine learning models. *Frontiers in Applied Mathematics and Statistics*, 11, p.1566078.