Artificial Neural Networks and the Cross-Section of Equity Returns: Identifying Nonlinear Opportunities on the Johannesburg Stock Exchange

By

Joshua Reed

RDXJOS001

Master of Commerce specialising in Finance in the field of Investment Management

Faculty of Commerce

University of Cape Town

Supervisor: Professor Paul van Rensburg

Department of Finance and Tax

University of Cape Town

The copyright of this thesis vests in the author. No quotation from it or information derived from it is to be published without full acknowledgement of the source. The thesis is to be used for private study or non-commercial research purposes only.

Published by the University of Cape Town (UCT) in terms of the non-exclusive license granted to UCT by the author.

University of Cape Town

Declaration

I, *Joshua Matthew Reed*, hereby declare that the work on which this dissertation/thesis is based is my original work (except where acknowledgements indicate otherwise) and that neither the whole work nor any part of it has been, is being, or is to be submitted for another degree in this or any other university.

I empower the university to reproduce for the purpose of research either the whole or any portion of the contents in any manner whatsoever.

Signature: Joshua Matthew Reed

Date: 14 May 2024

Acknowledgements

I would like to thank my supervisor, Professor Paul van Rensburg, for his guidance and assistance. His valuable insights, feedback, and patience are much appreciated.

I would also like to thank my family, friends and colleagues for their continued motivation and support.

Abstract

This study investigates the potential of Artificial Neural Networks (ANNs) to forecast stock returns on the Johannesburg Stock Exchange (JSE) using fundamental and technical factors. The optimal neural network architecture is explored, considering varying model depths and node counts. The activation function, training algorithm, learning rate, number of epochs, batch size, and loss function are kept constant across architectures. The findings suggest that portfolios constructed from ANN forecasts have the potential to outperform an equal-weighted benchmark. Model performance depends on network architecture, with a three hidden layer model with 64 nodes in the first hidden layer yielding the best results. Addition of further hidden layers or nodes is found to reduce model generalization, mainly due to overfitting, while less complex models are found to underfit. Models with a reduced variables set outperformed, confirming the importance of feature selection. While ANNs are found to underperform a linear model, the top performing ANN outperforms on risk-adjusted metrics over the test period, suggesting benefits to non-linear return forecasting on the JSE. However, with no clear relationship between in-sample and test period performance across architectures, this superior performance could be data specific, highlighting challenges in selecting an optimal model ex-ante. On the other hand, limitations in feature selection and training likely constrained model performance, with potential to improve generalization. This study provides a foundation for further research into return forecasting with ANNs on the JSE, contributing to the growing field of artificial intelligence and machine learning in finance. Future research could further optimize model hyperparameters, improve feature selection, and account for the time-varying nature of features.

Table of Contents

Declaration ii
Acknowledgements iii
Abstract iv
Table of Contents v
List of Tables vii
List of Figures viii
1. Introduction 1
1.1 Research Background 1
1.2 Research Overview 2
1.3 Research Contributions 2
1.4 Outline of the Paper 3
2. Theoretical Overview 4
2.1 Introduction 4
2.2 Efficient Markets and the Cross-Section of Returns 4
2.3 The Case for Non-Linear Machine Learning Models 6
2.4 Artificial Neural Networks 7
2.5 Conclusion 18
3. Literature Review 19
3.1 Introduction 19
3.2 Cross-Section of Returns: Linear Studies 19
3.3 Artificial Neural Networks in Returns Research 20
3.4 Anomalies of Interest 36
3.5 Conclusion 37
4. Data 38
4.1 Introduction 38
4.2 Data Description 38
4.3 Adjustments to the Dataset 40
4.4 Potential Biases 41
4.5 Conclusion 41
5. Methodology 42
5.1 Introduction 42
5.2 Splitting the Dataset 42
5.3 Linear Model 42
5.4 Neural Network Structure and Regularization 45

5.5 Performance Evaluation 48
5.6 Conclusion 50
6. Research Findings and Discussion 51
6.1 Introduction 51
6.2 Univariate Linear Models 51
6.3 Multivariate Linear Models 58
6.4 Artificial Neural Networks 65
6.5 Discussion 90
6.6 Limitations and Future Research 96
6.7 Conclusion 96
7. Conclusion 97
References 98
Appendix 102
Appendix A: 102
Appendix B: 119

List of Tables

Table 3.1: Summary of Model Architecture and Parameters in ANN Return Forecasting: 35

Table 3.2: Input Features Used to Train ANN Models to Forecast Stock Returns: 36

Table 4.1: List of Fundamental and Technical Attributes 39

Table 4.2: Breakdown of Sectors by Firm Count 40

Table 5.1: Summary of Hidden Layer Structures Analysed 45

Table 5.2: Parameters Consistent Across All Model Structures 46

Table 6.1: Univariate Regression Results 54

Table 6.2: Pearson Correlation Matrix of Univariate Significant Factors 59

Table 6.3: Multivariate Regression Results – All Variables 60

Table 6.4: Multivariate Regression Results – Reduced Variables 61

Table 6.5: Linear Model Performance 63

Table 6.6: MLP Model Performance – One Layer 66

Table 6.7: MLP Model Performance – Two Layer 71

Table 6.8: MLP Model Performance – Three Layer 76

Table 6.9: MLP Model Performance – Four Layer 81

Table 6.10: MLP Model Performance – Five Layer 86

Table 6.11: Ranking of the Top Performing Neural Network Models 91

Table 6.12: Model Performance Comparison – Linear Models versus Top ANN Models 93

Table 6.13: Ranking of the Linear Benchmark and Top Neural Network Models 95

Table B.1: Ranking of Neural Network Model Performance within Hidden Layer Counts 119

Table B.2 – Rankings of Best Models for each Hidden Layer Count 124

List of Figures

Figure 2.1: An Artificial Neuron.....8

Figure 2.2: A Fully Connected 3-2-1 Multilayer Perceptron 10

Figure 6.1: Univariate Regression – Cumulative Slope Coefficient.....55

Figure 6.2: Cumulative Marginal Slope Coefficient – All Variables.....60

Figure 6.3: Cumulative Marginal Slope Coefficient – Reduced Variables.....62

Figure 6.4: Multivariate Regression – Cumulative Information Coefficient 64

Figure 6.5: Multivariate Regression – Cumulative Portfolio Returns 64

Figure 6.6: MLP – One Layer – Cumulative Information Coefficient 68

Figure 6.7: MLP – One Layer – Cumulative Portfolio Returns 69

Figure 6.8: MLP – Two Layer – Cumulative Information Coefficient 73

Figure 6.9: MLP – Two Layer – Cumulative Portfolio Returns 74

Figure 6.10: MLP – Three Layer – Cumulative Information Coefficient.....78

Figure 6.11: MLP – Three Layer – Cumulative Portfolio Returns.....79

Figure 6.12: MLP – Four Layer – Cumulative Information Coefficient.....83

Figure 6.13: MLP – Four Layer – Cumulative Portfolio Returns.....84

Figure 6.14: MLP – Five Layer – Cumulative Information Coefficient 88

Figure 6.15: MLP – Five Layer – Cumulative Portfolio Returns.....89

A.1 Data Preprocessing.....102

A.2 OLS Model Code 105

A.3 Artificial Neural Network Code.....108

A.4 Performance Metrics.....112

1. Introduction

1.1 Research Background

Superior insight into the drivers of equity returns offers active investors an edge in generating profitable investment strategies. Since the introduction of the efficient market hypothesis (Fama, 1970), a large body of research dedicated to testing this theory has emerged. Typically, research aims to uncover factors that explain differences in future returns between stocks. Factors that were once profitable have become less reliable over time, so investors are seeking new ways to identify exploitable inefficiencies (Rasekhschaffe and Jones, 2019).

Traditional approaches include the use of characteristic sorted portfolios and linear models to identify relationships between factors and returns. While these models are easy to interpret, they suffer from several shortfalls. Linear models impose a structural form *a priori*, which is not always suitable (Chen, Shih and Wu, 2006). Financial data is time-varying with high noise-to-signal levels and is not necessarily linearly related to return outcomes (Karolyi and Van Nieuwerburgh, 2020).

Research into uncovering nonlinear relationships in stock returns has been a hot topic since its inception, with ongoing development of methods to generate more accurate forecasts (Cao, Leggio and Schniederjans, 2004). This includes the use of machine learning (ML) methods that can better handle nonlinear relationships in data that is noisy and suffers from multicollinearity (Rasekhschaffe and Jones, 2019). Artificial Neural Networks (ANNs) are a ML method that simulates the learning process of biological neurons. The technique can effectively capture linear and nonlinear relationships between variables, proving to be successful at generating accurate forecasts in a range of tasks such as image classification and autonomous driving (Aggarwal, 2018).

Predicting stock returns has proven to be challenging due to a low signal-to-noise ratio and evolving anomaly strength over time. Still, neural networks have the potential to uncover complex relationships between predictors and subsequent stock returns, and have been found to outperform linear models and other ML methods (see for example Rasekhschaffe and Jones (2019) and Gu, Kelly and Xiu (2020)).

Appropriate network structure and sufficient training data is required for ANNs to successfully approximate relationships between inputs and outputs. Small data sets tend to result in overfitting when model structures are complex, while less complex models can lead to underfitting (Aggarwal, 2018; Scarpino, 2018). The JSE presents unique modelling challenges due to its limited stock universe, high sector concentration, and exposure to macroeconomic dynamics like currency fluctuations and commodity prices. However, it is likely an inefficient

market (Van Heerden and Van Rensburg, 2015), providing opportunities for profitable trading strategies to be established from firm-specific financial data. This study aims to build on existing research on the performance of ANNs in forecasting returns using firm-specific fundamental and technical factors, with a specific focus on identifying optimal network structure in the South African market.

1.2 Research Overview

In the spirit of identifying exploitable market inefficiencies, this research investigates the value of using ANNs to explain the cross-section of stock returns listed on the JSE using various fundamental and technical factors over the period January 2005 to August 2020. More specifically, the out-of-sample performance of a multilayer perceptron (MLP) neural network is compared to the performance of a benchmark linear model. The research is extended to a comparison of portfolios generated from ranked return forecasts to remove the noise inherent in return prediction of specific shares. To find the optimal ANN model, various model structures are considered with varying network depths and node counts.

1.3 Research Contributions

This study aims to answer the following key questions:

What is the optimal neural network structure in terms of model depth and node count for stock return prediction on the JSE?

The structure of ANNs can impact performance, and finding the optimal model requires experimentation. If the model is too simple, it may fail to capture relationships. If the model is too complex, it may overfit the training sample and fail to generalise (Aggarwal, 2018). Several network structures with varying numbers of hidden layers and nodes are explored to offer insights into enhanced model specification in a South African context.

Considering the optimal ANN model, the following question is examined:

Can ANNs trained using fundamental and technical factors predict stock returns on the JSE, and do they outperform multivariate linear models?

This question relates to both the point estimates of returns and the performance of a portfolio based off ranked predictions. Two sets of input features are considered to investigate the impact of feature selection – a larger feature set and a subset thereof. Strong out-of-sample prediction and portfolio performance will identify the model as useful for identifying market inefficiencies. However, ANNs are generally more computationally intensive, thus their use needs to be justified. Therefore, the performance of the ANN models is compared to traditional linear methods. Hodnett (2010) follows Haugen and Baker (1996) to construct a multivariate

model using a stepwise regression approach. In the test period, the best ANN model outperforms in early forecast years but underperforms in later years and overall.

1.4 Outline of the Paper

This research paper consists of 7 chapters. The first chapter introduced the background to the research and identified the main questions this paper seeks to answer. Chapter 2 provides a theoretical background to relevant topics in this study. This includes a brief overview of the efficient market hypothesis and the search for factor anomalies, the need for non-linear models in forecasting returns, and concepts relating to neural networks. Chapter 3 discusses existing literature relevant to this study. There is a brief reflection on efficient markets and identification of anomalies, followed by an in-depth overview of the use of neural networks in forecasting stock returns. Potentially important anomalies are briefly discussed. Chapter 4 introduces the data set used for this study, including the process of arriving at the finalized dataset. Chapter 5 provides an overview of the methodology and models used in the analysis. This includes an overview of the linear models, neural network structure, and the performance evaluation metrics. The key research findings are analysed and discussed in Chapter 6, noting limitations and areas of interest for future research. Chapter 7 concludes.

2. Theoretical Overview

2.1 Introduction

A large body of research investigating the efficient market hypothesis (EMH) postulated by Fama (1970) has emerged. Investors seek a variety of methods to identify exploitable pricing inefficiencies capable of generating consistent excess returns. With a large pool of potentially correlated explanatory factors, methods that deal with multicollinearity, reduce the dimension of inputs, and identify the most important factors over time are required (Karolyi and Van Nieuwerburgh, 2020). One such method is artificial neural networks (ANNs), with growing interest as improving data availability and computational power increase their practicality (Aggarwal, 2018). With enough training data, ANN techniques can theoretically learn any mathematical function (Hornik, Stinchcombe and White, 1989), and therefore have strong potential to uncover complex relationships between explanatory factors and returns.

This chapter provides a theoretical background on relevant topics to this paper. A basic overview of efficient markets and cross-sectional studies is considered in Section 2.2. Section 2.3 provides an argument in support of non-linear models for returns forecasting in a South African context. An overview of relevant concepts pertaining to ANNs is presented in Section 2.4. The chapter is briefly concluded in Section 2.5.

2.2 Efficient Markets and the Cross-Section of Returns

The EMH assumes that in a frictionless market with no transaction costs, full information, and consensus between investors, equity prices ‘fully reflect’ all available information (Fama, 1970). This is of course not true in the real world. However, violations of these assumptions do not imply market inefficiency. For example, if investors disagree about the information that is available, markets are only inefficient if an investor can consistently beat the market based on that information (Fama, 1970). Efficiency is defined as weak form, semi-strong, and strong form, based on the information set from which future returns can be forecast. Weak form tests consider past prices. Semi-strong tests relate to the speed of price adjustment to publicly available information. Strong form tests consider private information (Fama, 1970).

In contrast to this theory, a large body of research has emerged to identify an appropriate information set that can be used to forecast returns. Markets are considered inefficient if a predetermined trading strategy based on available information can earn excess risk-adjusted returns (Van Heerden and Van Rensburg, 2015). Predictability in returns is thought to be a result of mispricing (McLean and Pontiff, 2016), which goes against the EMH.

Research to identify inefficiencies generally follows two paths – cross-sectional and time-series analysis. Cross-sectional analysis aims to identify differences in returns between stocks or portfolios at a point in time, such as the CAPM of Sharpe (1964), the cross-sectional regressions by Fama and Macbeth (1973), and the characteristic sorted portfolios by Fama and French (1992). Time-series analysis considers the returns of a stock or portfolio over time. This is often performed to assess the impact of time-varying macroeconomic variables on returns (Drobetz and Otto, 2021). Cochrane (2011) suggests that these methods are essentially different ways of doing the same thing – understand a panel-data set.

While hundreds of potential anomalies have been published over time, there is concern that several are false discoveries due to poor research techniques (Harvey, Liu and Zhu, 2016; McLean and Pontiff, 2016). Harvey, Liu and Zhu (2016) consider 296 published factors that attempt to explain the cross-section of expected returns, finding that at least 80 are false discoveries. They argue that theory-based factors should require lower significance levels than empirically identified factors. Hou, Xue and Zhang (2020) consider a pool of 452 anomalies. They find that excluding microcap stocks, considering value-weighted rather than equal-weighted portfolios, and increasing the required significance level greatly reduces the number of significant factors. This is further reduced when restrictions on liquidity and other trading frictions are included. However, findings support replication of value, momentum, investment, and profitability anomalies (Hou, Xue and Zhang, 2020).

In addition to false discoveries, anomalies are unstable over time and are found to disappear post-publication (McLean and Pontiff, 2016). Characteristic portfolios of identified anomalies experience increased trading volume when they are published, thereby pricing out any excess profitability or arbitrage opportunities (McLean and Pontiff, 2016). However, this is dependent on market frictions. Characteristic portfolios consisting of stocks with higher arbitrage costs are found to maintain mispricing post-publication, suggesting anomalies may exist due to market frictions (McLean and Pontiff, 2016).

Overall, numerous factors have been identified that contribute to forecasting returns and constructing market beating portfolios. However, several anomalies are inconsistent over time and do not contribute significantly to return forecasts when accounting for market frictions. Methods that are non-linear in parameters have the potential to account for the time-varying nature of anomaly strength and better identify relationships between factors and excess returns (Karolyi and Van Nieuwerburgh, 2020). Section 2.3 discusses the case for non-linear machine learning models.

2.3 The Case for Non-Linear Machine Learning Models

Prediction models are commonly based on conventional linear statistical methods such as multivariate analysis. However, several restrictions are imposed on models such as assumptions regarding linearity, normality, and the distribution of errors (Chen, Shih and Wu, 2006). Machine Learning (ML) techniques are less restrictive and have demonstrated promising prediction potential compared to traditional linear methods (Chen, Shih and Wu, 2006). Gu, Kelly and Xiu (2020) note the potential of ML methods, including neural networks, to contribute to solving issues in finance such as market timing, portfolio optimization, and risk management. They describe ML as:

"(a) a diverse collection of high-dimensional models for statistical prediction, combined with (b) so-called "regularization" methods for model selection and mitigation of overfit, and (c) efficient algorithms for searching among a vast number of potential model specifications" (Gu, Kelly and Xiu, 2020).

ML methods have several benefits over linear models and other techniques that have traditionally been used to identify pricing inefficiencies. First, ML techniques can better capture nonlinearities between variables (Karolyi and Van Nieuwerburgh, 2020). Relationships between returns and factors are not necessarily linear, and studies that incorporate nonlinear relationships into models tend to identify improved forecasting performance (Karolyi and Van Nieuwerburgh, 2020). In a South African context, Phiri (2015) finds evidence against weak form efficiency only when nonlinearities are considered.

Second, these models can incorporate a larger number of predictive signals, even in the presence of multicollinearity. The models can reduce the vast number of anomalies into a smaller set of predictors and incorporate the most important features into generating signals (Karolyi and Van Nieuwerburgh, 2020).

Third, the models are structured to predict outcomes rather than identify causal relationships between variables. Models are optimized to have the best out-of-sample performance, opposed to identifying the association of variables. In particular, the models have the benefit of flexibility since no functional form is imposed prior to the analysis. The most appropriate model can be determined during training rather than preselecting a model that supports the hypothesis (Karolyi and Van Nieuwerburgh, 2020).

Additionally, the high-dimensional nature allows for complex relationships between returns and predictors to be captured (Gu, Kelly and Xiu, 2020). Further, nonlinear models have potential to capture time-varying relationships. For example, Rasekhschaffe and Jones (2019)

identify varying correlation over time between ANN return forecasts and commonly used linear factors, including those from the Fama and French (1992) 3-factor model.

However, these techniques are prone to drawbacks. For example, the models can be subject to poor generalisation if they simply learn the training sample and capture noise. Therefore, regularisation and sufficient training data is important. Additionally, the techniques can be computationally intensive, causing a trade-off between accuracy and efficiency (Gu, Kelly and Xiu, 2020). Further challenges are discussed in Section 2.4.5.

This study considers neural networks. Hornik, Stinchcombe and White (1989) show that ANNs with at least one hidden layer are universal approximators that can closely learn the relationship between input and output variables. ANNs have shown potential in several applications including forecasting returns. Further analysis is needed to understand their suitability and the optimal network structure for forecasting the cross-section of returns using firm specific fundamental factors in South Africa. Section 2.4 gives an overview of ANNs.

2.4 Artificial Neural Networks

This section provides an overview of important concepts relevant to this study. Aggarwal (2018) provides a comprehensive overview of neural networks and deep learning concepts.

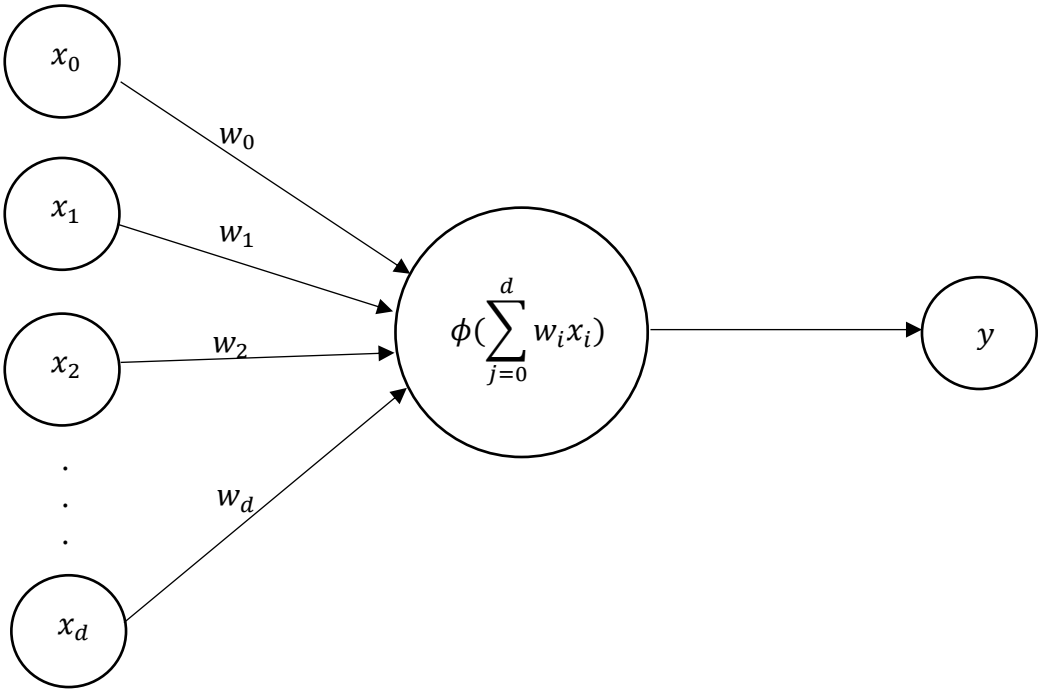
2.4.1 Overview

The neural network structure is based on the human nervous system. Biological neurons refer to the interconnected cells within the nervous system. Neurons receive signals originating from external stimuli in the form of electrical stimulation through dendrites. The electricity is stored within the cell body of the neuron, with voltage increasing as incoming signals grow stronger. When the voltage surpasses a threshold level, the neuron passes on an electrical signal through an axon. This signal is passed to another neuron through a synapse, which connects the axon of the first neuron to the dendrites of the second neuron. The synaptic connections vary in strength based on the response to external stimuli (Aggarwal, 2018). Put simply, a neuron produces a single output signal from the signals it has received. The output signal becomes the input signal of another neuron. Each neuron only produces an output signal when the threshold level has been exceeded (Scarpino, 2018). Learning depends on the strength of the synaptic connections, otherwise known as weights (Aggarwal, 2018).

ANNs attempt to simulate this process. In a single artificial neuron, numerical inputs are passed on to a neuron similar to how dendrites provide input to biological neurons. Each input is scaled by a synaptic weight that adjusts as the model learns the relationship between input and output variables. At each node, a nonlinear activation function determines whether to activate the node based on the weighted sum of the input variables, producing a single output.

The goal is for the model to generalize out-of-sample so that it can correctly forecast the output of unseen input variables (Aggarwal, 2018). A constant term, known as the bias, is often added to the weighted sum of input features within the activation function to capture any invariant part of the prediction (Aggarwal, 2018). It is added into the model construction as an input and usually takes the value of 1 as it is scaled by a varying weight variable like the other inputs (Scarpino, 2018). The weight of the bias feature then represents an intercept. Figure 2.1 depicts the structure of an artificial neuron.

Figure 2.1: An Artificial Neuron



Mathematically, the relationship can be expressed as (adapted from Aggarwal, 2018):

$$\hat{y} = \phi\left(\sum_{j=0}^{d} w_i x_i\right) \quad (1)$$

Where:

$\phi(.)$ represents the activation function

w_i are the weights of each input, where w_0 is the bias weight

x_i are the input features, where x_0 is the bias input

d is the number of input features

$\hat{y}$ is the predicted output variable

The ANN models considered in this study take the form of supervised relationships, in which each input into the model has an associated output.

2.4.2 Architecture

Individual neurons identified above are combined to create more sophisticated neural networks (Scarpino, 2018). The basic structure of a neuron in Figure 2.1 represents a single-layer feed-forward network with one output, known as a perceptron. The perceptron was first introduced by Rosenblatt (1957) and is based on the idea of neural networks as a modelling technique first introduced by McCulloch and Pitts (1943). The model has evolved over time to incorporate training and weights. This simple construction represents a variation of a linear model, with inputs directly mapped to the output. In the case that the identity activation function is used, this represents a typical linear OLS model (Aggarwal, 2018).

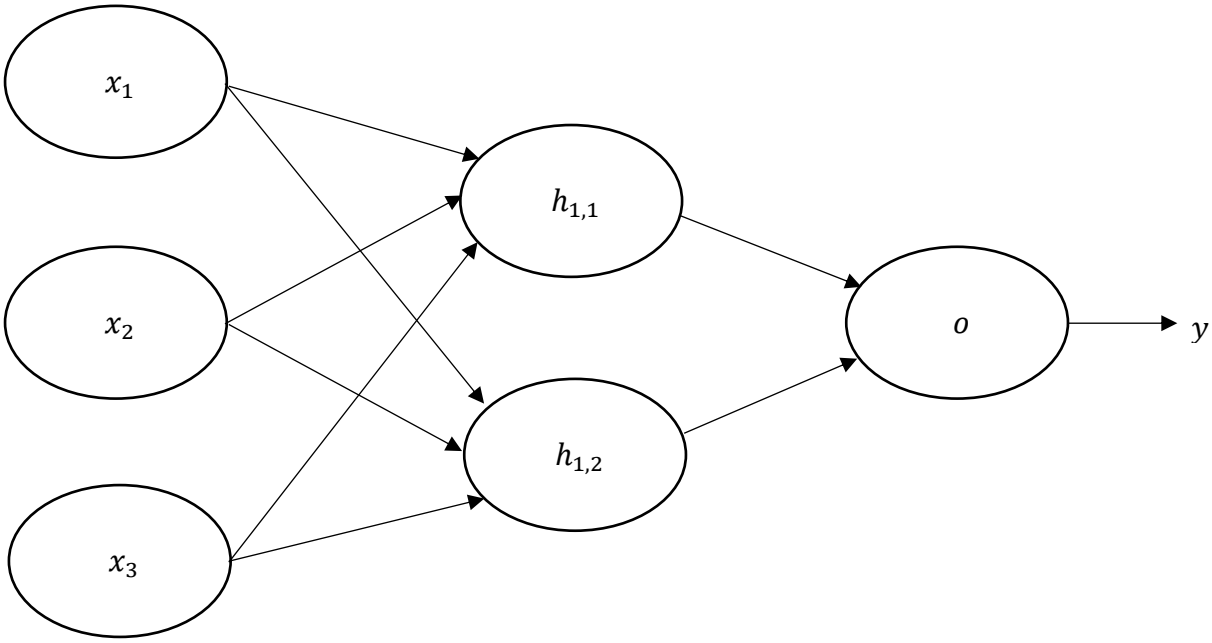
Single-layer networks may contain multiple outputs with several input neurons directly linked to each output. In single-layer models, computations only take place within the output neurons. These models are typically easy to interpret as data is transmitted directly from the input layer to the output layer with observable computations (Aggarwal, 2018). The model is referred to as feedforward because each layer transmits a signal in the direction of input to output variables. Models are usually fully-connected, with each input node connected to every node in the following layer (Aggarwal, 2018).

A more complex adaption of this is a multilayer feedforward network, known as a multilayer perceptron (MLP). In this case, the input and output are separated by additional hidden layers of computation neurons. The name arises because the computations within the hidden layer are not observable (Aggarwal, 2018). Figure 2.2 shows the structure of an MLP. The structure is usually written in the form $d - h_1 - h_2 - \cdots - h_k - o$, where there are *d* inputs into the model, *k* hidden layers with h_i hidden neurons in each layer, and *o* output variables. Figure 2.2 is an example of a 3-2-1 architecture. Three inputs (x_1, x_2, x_3) are connected to a single hidden layer (h_1) consisting of two nodes ($h_{1,1}, h_{1,2}$), which transform the weighted inputs based on the activation function, and transmit a signal to the output layer (*o)*. The output layer transforms the input from the hidden layer, usually with the identity function, to an output value *y*. A bias neuron can be added to each of the layers in the network.

Hornik, Stinchcombe and White (1989) suggest that a model with a single hidden layer and output neuron can compute almost any function. However, many hidden neurons are required, limiting the use of these models when limited data is available since the number of parameters to be learned increases. Deeper networks reduce the required number of hidden units in each layer to train models, thereby reducing the number of parameters to be estimated (Aggarwal,

2018). Additional hidden layers increase model complexity which can increase training time and the risk of overfitting (Scarpino, 2018).

Figure 2.2: A Fully Connected 3-2-1 Multilayer Perceptron



2.4.3 Activation Functions

The choice of activation function is important as it introduces nonlinearity into the model and can impact training efficiency and model generalisation. The identity function, known as the linear activation function, is the simplest activation function $\phi(.)$, and is typically used in the output node (Aggarwal, 2018). Scarpino (2018) divides the remaining common activation functions into two categories based on the nature of the output variable – classifier functions for categorical outputs, and rectifier functions for continuous outputs.

Rectifier functions receive their name due to similarities with rectifiers in electrical circuits, with the rectified linear unit function (ReLU) most similar. ReLU has become popular in recent years due to its ability to reduce the dimension of the input features and its efficient computational speed (Scarpino, 2018). It has the form:

$$\phi(v) = \max\{0, v\} \tag{2}$$

The ReLU activation function is used in this study. Other classifier functions include an updated version of ReLU in which the output signal is limited to a maximum of 6 to avoid overreactions to large inputs, and the Exponential Linear Unit function which is continuous at v = 0 (Scarpino, 2018).

Classifier functions tend to be ‘s-shaped’ and are useful for forecasting output categories. Examples include the sigmoid, tanh, hard tanh, and softsign functions (Aggarwal, 2018; Scarpino, 2018).

2.4.4 Training

The general objective of most ML algorithms is to identify relationships between variables so that it can accurately forecast outcomes of unseen input data. Training involves updating the model to better represent the experimental data (Scarpino, 2018). In a supervised neural network, the weights w_i are adjusted to minimize the error in prediction of all training instances of a dataset D. This is represented by a loss function L (Aggarwal, 2018):

$$Minimise_{\bar{W}} L = \sum_{(\bar{X},y)\epsilon D} (y - \hat{y})^2 \quad (3)$$

Where:

$\bar{W}$ is the vector of weights w_i

$\bar{X}$ is the vector of input features x_i

$\hat{y}$ is the predicted output variable defined previously

y is the actual observed output

D is the subset of training variables from the full dataset

The process of minimizing loss through updating the model’s variables is called optimization. This generally requires several iterations in a loop to find the minimum loss (Scarpino, 2018). Various optimization algorithms have been developed over time. The simplest and most common is known as the gradient descent algorithm. A function’s gradient refers to a vector of the partial derivatives of the function, which indicates the direction of steepest ascent. The opposite vector is in the direction of steepest descent (Scarpino, 2018). The gradient descent algorithm calculates the gradient of the loss function $J(.)$ and updates the set of trainable variables θ until the gradient of the loss function is zero. Mathematically (Scarpino, 2018):

$$\theta_t = \theta_{t-1} - \eta \nabla J(\theta) \quad (4)$$

Where:

θ_t is the value of the set of trainable variables at step t

$\nabla J(\theta)$ is the gradient of the loss function.

η is the learning rate

The gradient of the loss function should approach zero as training continues, thus the trainable variables will be largely unchanged when the function converges. The learning rate is usually determined prior to training the model and impacts the change in trainable variables in each step. If it is too small, training will be precise but take long and potentially converge at a local minimum rather than a global minimum. If it is set too large, training will be faster but the algorithm may oscillate around the minimum and fail to converge. It is common practice to select the learning rate through trial and error, by initially setting a larger value and reducing it until there is successful convergence. Typical learning rates are between 0,1 and 0,0001 (Scarpino, 2018).

Due to potential problems around non-convergence, variations of the gradient descent algorithm have been developed which allow learning rates to vary between iterations. The momentum optimiser is considered first as it is most like gradient descent. However, it has reduced likelihood of oscillating and converges faster. An accumulation variable v_t which accounts for the gradient of loss, the learning rate, and a scaled value of the preceding accumulation is added to the previous trainable variables to update them. Mathematically (Scarpino, 2018):

$$v_t = \alpha v_{t-1} - \eta J(\theta) \tag{5}$$

$$\theta_t = \theta_{t-1} + v_t \tag{6}$$

Where:

θ_t, $\nabla J(\theta)$, and η are defined as before

v_t is the accumulation variable at step t

α is momentum

Momentum is a constant between 0 and 1 that determines how much the current step is influenced by the prior step. It is usually set to 0,9 (Scarpino, 2018). The accumulation variable helps the model reach convergence. If the optimizer is oscillating between two variables, αv_{t-1} will reduce the change in trainable variables at each step. On the other hand, it will be large when the optimizer is already quickly moving toward a minimum value, and lead to faster optimization (Scarpino, 2018).

The Adagrad algorithm is considered next. In contrast to the prior two algorithms, the learning rate is allowed to vary between steps and variables, which accounts for differences in the rate that different variables may converge. Subgradients, which apply to differentiable and non-differentiable functions, are calculated for each iteration rather than gradients (Scarpino, 2018). Duchi, Hazan and Singer (2011) developed the algorithm to improve gradient-based learning processes by dynamically including information of data observed in earlier iterations.

In simple notation, the variable learning rate for each variable and the optimization function can be defined as (Scarpino, 2018):

$$\eta_{t,i} = \frac{\eta}{\sqrt{G_{t,ii}}} \tag{7}$$

$$\theta_{t,i} = \theta_{t-1,i} - \eta_{t,i} g_t \tag{8}$$

Where:

η is the initial learning rate

$\eta_{t,i}$ is the learning rate of variable *i* at step t

$\theta_{t,i}$ is the value of trainable variable *i* at step t

$G_{t,ii}$ is the i^{th} element of the diagonal of matrix G at step t

g_t is the subgradient vector of the loss function at step t

The matrix G represents the outer product of the subgradient g_t with itself (Scarpino, 2018). In other words, $G_t = g_t g_t^T$ at each step. The Adagrad method is useful, however the learning rate decreases at every step and eventually reaches 0, stopping training.

The last optimization algorithm considered is the Adam Optimizer, which has similarities to both the Adagrad and momentum algorithms. The trainable parameters are updated by scaled forms of the gradient vector m_t and v_t. Mathematically (Scarpino, 2018):

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\nabla J(\theta) \tag{9}$$

$$v_t = \beta_1 v_{t-1} + (1 - \beta_2)[\nabla J(\theta)]^2 \tag{10}$$

$$\eta_t = \frac{\eta\sqrt{1 - \beta_2^t}}{1 - \beta_1^t} \tag{11}$$

$$\theta_t = \theta_{t-1} - \frac{\eta_t m_t}{\sqrt{v_t} + \varepsilon} \tag{12}$$

Where:

m_t is the first moment vector at step t

v_t is the second moment vector at step t

η_t is the learning rate at step t

θ_t is the value of the trainable parameters at step t as above

β_1, β_2 are hyperparameter inputs that scale the gradient of the loss function

ε is a hyperparameter included in the model to avoid the denominator from reaching zero. It is usually set as a small value.

This study considers implementation of the Adam Optimizer gradient-descent method through backpropagation. In single-layer structures, loss is a direct function of the weights in the model so there are limited issues in calculating partial derivatives. When additional layers are added, model complexity increases, and loss becomes a composition function of weights included in earlier layers (Aggarwal, 2018). Backpropagation is a popular learning method for optimizing the weights of more complex models due to its simplicity and speed (Scarpino, 2018). It was introduced by Werbos (1974) as a method for calculating the gradient of loss in ANNs. Backpropagation attempts to minimize the loss function by calculating the loss for a given set of inputs and network weights, finding the derivative of loss with respect to each weight, and updating the weights based on the calculated partial derivatives. Weights are updated using an optimization function discussed previously. The process is repeated until the gradient approaches zero or is acceptably small (Scarpino, 2018). The backpropagation learning algorithm makes use of the chain rule to compute the gradient of the loss function with respect to each weight in the model in two phases: the forward phase and the backward phase.

The forward phase consists of computing a predicted output $\hat{y}$ for a set of inputs x_i and the initial weights w_i of the network for a particular training instance. The predicted output $\hat{y}$ is compared to the output of the training instance to generate a loss. The backward phase makes use of the chain rule to calculate the partial derivative of the loss with respect to each weight in the model. The weights are adjusted based on these gradients. The gradients are calculated starting from the output node to the input node, hence the name (Aggerwal, 2018).

The loss of an ANN is the sum of the loss at each training point, therefore gradient descent should be performed simultaneously for all training points. This is computationally intensive for complex models. Therefore, the mini-batch stochastic gradient descent (SGD) method is typically utilised. Mini-batch SGD makes use of a random sample of training points at each iteration to calculate the gradient, and generally provides accurate estimates while reducing the computation time (Aggarwal, 2018). The mini-batch method is used in this study.

2.4.5 Potential Problems

While neural networks seem like a promising technique for approximating relationships between inputs and outputs, the model needs to be appropriately trained to reach the desired performance. This includes issues with under- or overfitting, vanishing and exploding gradients, local optima, and computational challenges (Aggarwal, 2018).

First, overfitting occurs when the model performs well in-sample but does not generalise well out-of-sample. This usually arises in complex models with small data sets (Aggarwal, 2018). With more parameters in the model, a greater number of data points are required to ensure better out-of-sample generalisation. There should be at least 2 to 3 times more training data points than parameters in the model (Aggarwal, 2018). In contrast, underfitting occurs when the model contains too few nodes and fails to accurately classify data (Scarpino, 2018). Section 2.4.6 provides methods for dealing with these issues to improve model generalisation.

Second, the issue of vanishing and exploding gradients generally occurs when additional hidden layers are added to the model. While additional layers can reduce the number of parameters to be trained, updates through backpropagation may be unstable, with negligible or increasingly large updates in layers further from the output. This is because the overall backpropagated derivative is the product of each local derivate. Assuming the local derivatives are normally distributed, with an expected value less than 1, the gradient will vanish as the path length gets longer. In contrast, if the expected value is greater than 1, the gradient will explode. Even if the expected value is equal to 1, the gradient is likely to be unstable (Aggarwal, 2018). The use of an appropriate activation function and optimization algorithms can limit the impact of this problem. For example, ReLU activation helps avoid vanishing and exploding gradients, while a sigmoid activation encourages this. Adaptive learning rates help in gradient descent optimization (Aggarwal, 2018). A recently developed technique known as batch normalisation is discussed in Section 2.4.6.

Third, issues with local optima refers to convergence at a local minimum rather than the global minimum of the loss function. In other words, the model could be improved. This issue is overcome by improving the choice of model initialization points by methods such as pretraining. Through pretraining, irrelevant parts of the parameter space are excluded and improved initialisation points are provided (Aggarwal, 2018).

Lastly, neural networks are computationally intensive and often require a long time to train. This is, however, becoming less of an issue as computers become more advanced and new data sources increase the potential of these models (Aggarwal, 2018).

2.4.6 Tuning the Neural Network

This section provides a basic overview of good practice in training neural networks, including techniques to avoid overfitting and improve generalisation.

Input Standardisation refers to the transformation of inputs so that the training and test data have the same mean and standard deviation to ensure consistency between data and avoid confusion in the model. Typically, the data is transformed to have a mean of 0 and standard

deviation of 1 (Scarpino, 2018). The input variables in this study are standardized cross-sectionally each month to have a mean of zero and standard deviation of one.

Batch Normalisation is like input standardisation but applies to each layer in the network (Scarpino, 2018). It was introduced by Ioffe and Szegedy (2015) to address the varying distribution of inputs to each layer in the network. Since normalising the inputs to a layer can reduce the layer's flexibility, the technique normalizes the value of each batch independently. During training, the mean and standard deviation of each batch are used. During testing, an estimate of the population mean and variance from the batches is used (Ioffe and Szegedy, 2015). When implemented, there is less concern about weight initialisation which helps avoid issues with vanishing and exploding gradients (Aggarwal, 2018). In addition, it acts as a regularisation technique that helps ensure a better fit and reduces the need for other regularisation techniques such as Dropout (Ioffe and Szegedy, 2015). In this study, Batch Normalisation is implemented to standardise the inputs into each hidden layer.

Weight Initialisation refers to the value to which weights are set prior to training. Often, they are just set to small, random values. However, techniques have been developed that provide random initialisation values based on the number of nodes in the layer receiving and providing the weights (Scarpino, 2018). For example, LeCun et al. (1998) propose that weights should be drawn randomly for a distribution, such as the uniform distribution, with a mean of 0 and standard deviation equal to $\frac{1}{\sqrt{m}}$ where m is the number of connections entering a node. Appropriate weight initialisation is especially important in deeper networks to ensure that the gradient is stable across layers (Aggarwal, 2018). In this study, the HeNormal initializer (He et al., 2015) is used to set the initial network weights.

Regularisation refers to finding the right structure for the neural network to ensure that the model neither under- nor overfits the data (Scarpino, 2018). Models are trained with the intention to generalise out-of-sample. This is not the case if there is overfitting, as the training set is simply memorised by the model (Aggarwal, 2018). There is generally a trade-off between bias and variance. Bias is the error due to not capturing relationships and leads to underfitting, which is consistent between training samples. Variance is the error due to overfitting the training sample, and therefore varies dependent on the training set (Aggarwal, 2018). Variance tends to be more of an issue as model complexity increases. There is often an optimal level of complexity dependent on the bias-variance trade-off (Aggarwal, 2018). Techniques to balance over- and underfitting to improve model generalisation follows.

Parameter Tuning and Hyperparameter Choice is an important consideration in ensuring strong out-of-sample performance. Neural networks consist of several hyperparameters and model designs that influence performance, such as the learning rate and choice of activation

function. Different model structures can be tuned using a validation dataset which is separate from the training and testing data. From this, the most appropriate combination of parameters can be determined based on the model performance on the validation data (Aggarwal, 2018). Thus out-of-sample performance is independent of the training and tuning phase. This simple split of data into training, validation, and testing subsets is known as the hold-out method. A more complex k-fold cross-validation method divides the full dataset into k equal subsets. The model is trained k-times, with a subset k used as the testing sample, and the remaining (k-1) subsets making up the training sample. While cross-validation is generally superior to the hold-out method, it is less efficient (Aggarwal, 2018).

Penalty-Based Regularisation is the most common method for reducing overfitting. Since overfitting tends to increase as the number of input features increases, the method improves generalisation by imposing a soft penalty on the value of parameters (Aggarwal, 2018). The penalty increases the value of the loss function by a value that depends on the weights and a constant λ (Scarpino, 2018). The L1 regularisation method increases loss by the absolute value of the network weight scaled by λ. L2 regularisation increases loss by the square of the weight scaled by λ (Scarpino, 2018). Weights are adjusted during the gradient descent process that ensure only repeated updates influence the magnitude of weights (Aggarwal, 2018). For both methods, a decrease in weights results in lower loss, while higher weights increase loss. Thus, the model is simplified by reducing non-essential weights (Scarpino, 2018). L2 regularisation tends to outperform L1 regularisation, especially when the number of inputs is low. However, L1 regularisation has the benefit of generating sparse solutions when most weights reduce to zero. In this sense, the L1 method acts as a feature selector allowing unnecessary inputs to be dropped (Aggarwal, 2018). As the ReLU activation function is used and the set of input features is not large, L1 and L2 regularisation is not implemented in this study, as there is little need to further reduce the dimension of the input data set.

Ensemble Methods combine several models together to generate a single improved model by improving either the bias or variance without negatively impacting the other component. Boosting methods are used to reduce bias, while bagging methods are used to reduce variance. Since more complex models tend to reduce bias at the expense of variance, most ensemble methods aim to reduce the latter. Bagging is a technique of subsampling with replacement, drawing on the reduction in variance attributable to repeated sampling for predicting the same test instance. With an infinite data set, the variance would be reduced to 0. In finite data sets, this is not possible but there is still a reduction in variance from combining predictions of a test instance based on varying training sets (Aggarwal, 2018). While these methods are not applied in this study, an ensemble approach is utilized.

Dropout is an ensemble method in which nodes are randomly removed from the network, including the connections and weights of each removed node (Aggarwal, 2018). A probability *p* is selected beforehand which determines the number of non-zero outputs generated by the model (Scarpino, 2018). Several neural networks are sampled in this method, with a small training set used to update the weights. The individual models are not powerful but have diverse predictions, thus an average of the predictions is generally accurate. The method is efficient but reduces the network complexity. Therefore, larger model structures should be constructed when dropout is used (Aggarwal, 2018). In this study, Dropout is used between the first and second hidden layers with the dropout rate set to 0.5 for models with more than one hidden layer.

Early Stopping is a method implemented to reduce variance. As discussed in Section 2.4.4, gradient descent methods optimize the model by reducing the gradient until loss converges at a minimum. This optimizes the model on the training data set, which may not generalize out-of-sample, since the final iterations may overfit the specifics of the training set. Early stopping attempts to limit overfitting by using a portion of the training data as a validation set. While the model is trained, the forecast error on the validation set is calculated. As training progresses, the in-sample error will continue reducing, but at some point the validation error will begin to rise. This rise may be due to noise, so training continues until the validation error rises for a predetermined number of iterations, indicating overfit. The termination point is selected in hindsight at the point overfitting begins (Aggarwal, 2018). In this study, early stopping is implemented during training with patience set to 10

2.5 Conclusion

This chapter provides an overview of concepts and methodologies relevant to this paper. An ongoing search for pricing inefficiencies in markets has resulted in a large pool of potential anomalies that can be utilised to reap market beating returns. However, these anomalies are inconsistent over time and have varying use in the real-world due to market frictions. ANNs potentially offer an additional tool to identify relevant features and forecast returns over time. However, appropriate tuning and regularisation techniques are essential to enhancing model generalization.

3. Literature Review

3.1 Introduction

This chapter builds on the theoretical background of Chapter 2. Section 3.2 provides an overview of studies with a linear focus. Section 3.3 considers applications of artificial neural networks (ANNs) in returns prediction. Section 3.4 summarises anomalies of interest from the literature. Section 3.5 concludes the chapter.

3.2 Cross-Section of Returns: Linear Studies

The introduction of the CAPM (Sharpe, 1964) provided investors with a simplified understanding of returns, by suggesting that they are simply a linear function of risk relative to market returns. Early studies support this positive relationship between risk and return (Fama and Macbeth, 1973), however it was subsequently disproved for later periods of the 20th century (Fama and French, 1992). Further studies have investigated factors beyond market returns that could explain returns in the cross-section. These studies seek to find information advantages in contrast to the efficient market hypothesis (Fama, 1970). Cross-sectional anomalies tend to relate to firm-specific fundamental and trading factors.

Fundamental factors include information relating to financial statements and corresponding ratios. Van Heerden and Van Rensburg (2015) note that these are generally broken down into value, growth, and size-based factors. Basu (1977, 1983) identifies earnings yield as a value measure of future returns, with high earnings yield associated with higher risk-adjusted returns. The value effect is supported by Rosenberg, Reid and Lanstein (1985) who identify book-to-market as a potential predictor of returns. Banz (1981) identifies a size effect, noting that small firms offer higher risk-adjusted returns. The relationship is nonlinear as the effect is pronounced for small firms, with little variation between large and medium sized firms (Banz, 1981). Basu (1983) concludes that value and size are separate factors, despite initial disagreement by Banz (1981). Fama and French (1992) support this, concluding that two factors – size and book-to-market equity – account for the cross-sectional variation in returns. This forms the basis of the Fama and French (1993) 3-factor model, since extended to include profitability and investment factors (Fama and French, 2015).

Technical trading rules relate to trends and patterns in trading information such as price volume, and trading activities (Tsai et al., 2011). Price considers the impact of past returns on future returns, generally tested for by serial correlation. Positive serial correlation indicates price momentum, while negative serial correlation indicates price reversal and a mean-reverting effect (Van Heerden and Van Rensburg, 2015). For example, De Bondt and Thaler

(1985) find evidence of a reversal effect in returns. Portfolios consisting of past losers from prior years are found to outperform past winners. Jegadeesh and Titman (1993) study shorter-term momentum effects, finding evidence in support of momentum for 6 to 12-month periods.

In a South African context, Van Heerden and Van Rensburg (2015) provide an updated study of the cross-section of returns. Technical and fundamental factors are considered over the period 1994 to 2011 to identify factors that best explain cross-sectional returns and test for market efficiency. The study supports previous findings that the CAPM does not hold in South Africa, and further notes that the JSE is likely an inefficient market. Value and momentum factors are identified to best explain returns.

3.3 Artificial Neural Networks in Returns Research

Extensive research on machine learning (ML) methods looking to improve predictions and capture nonlinearities in returns forecasting has been conducted. The literature differs in several aspects, such as model construction, regularization methods, location, time-period, and feature selection. Further, certain studies seek to classify shares into groups based on relative returns, while others determine point estimates. Lastly, several studies seek to explain the cross-section of returns, while others identify time-series variations. Section 3.3.1 provides an overview of cross-sectional studies that use ANNs to forecast equity returns, considering findings across developed and emerging markets. Section 3.3.2 briefly considers time-series studies. Section 3.3.3 summarizes differences in model specification, with a key focus on network structure.

3.3.1 Artificial Neural Networks in Returns – Cross-Section

In one of the earliest studies in the field, Kryzanowski, Galler and Wright (1993) investigate the annual classification of 120 Canadian stocks into positive or negative future returns using neural networks from 1987 to 1989, with each forecast based on four years of industry and company specific financial data alongside seven macroeconomic variables. The specific ANN used in this case is the Boltzmann Machine, a pattern-recognition model that attempts to identify the relationship between the input and output variables, increasing the weights of inputs that regularly occur with specific outputs. The model is found to be appropriate, correctly classifying stocks 72% of the time. Kryzanowski, Galler and Wright (1993) note that this may be due to smaller companies contained in the sample and the short period of the study. However, it is noted that model performance could improve if a longer training period is used.

Olson and Mossman (2003) later confirm the appropriateness of ANNs to classify and forecast returns in the Canadian stock market over the period 1983 to 1993. With models trained using 61 accounting ratios from the rolling prior six years, it is found that regression ANNs

outperform stepwise OLS models at forecasting point estimates of returns, and classification ANNs outperform logistic regression models at categorising firms based on returns. Classification models outperform regression models, with four and eight category models outperforming two and sixteen category models. The authors conclude that ANN models add value, with potential for improved effectiveness through changes to input variables and model specifications (Olson and Mossman, 2003).

In the US, Eakins and Stansell (2003) examine the returns forecasting ability of ANNs in the context of value-based firm specific features. The dataset consists of US firms from 1975 to 1996 with inflation-adjusted market capitalization above US$150 million. Return forecasts begin in 1977 based off a rolling model trained over the prior two years. The study finds that a fifty-stock portfolio constructed of stocks with the highest forecast returns is skewed to value stocks and tends to outperform benchmark indices and other value-based trading strategies on a risk-adjusted basis, supporting the notion of investing in value stocks as well as using ANNs to improve stock selection (Eakins and Stansell, 2003).

Additional support for the suitability of ANNs has been found in Asia. Tsai et al. (2011) consider the classification of quarterly returns of the Taiwanese electronic industry between 2002 and 2006 using 19 firm specific and 11 macroeconomic variables. A homogenous ensemble of MLP predictions, and a heterogenous ensemble blend of MLP, classification and regression tree, and logistic regression predictions are compared to the results of individual models. While the logistic regression model is found to be the most accurate individual model and the heterogenous model most accurate overall, MLP models are found to be more accurate at forecasting positive returns. Due to this, the MLP models were found to produce higher portfolio returns, with the homogenous MLP ensemble performing best (Tsai et al., 2011).

In the Japanese market, Abe and Nakayama (2018) examine the performance of deep and shallow MLPs to forecast monthly returns of the constituents of the MSCI Japan Index. 25 input features are used to forecast returns from 2002 to 2016, making use of a 120-month training set rolled forward monthly. The number of explanatory features is increased to 125 by including the four previous points of each input feature in three-month intervals. Deep neural networks are found to outperform shallow networks, as well as support vector regression and random forest models. Further, a heterogenous ensemble model outperforms each of the individual models, supporting Tsai et al. (2011). While Abe and Nakayama (2018) do not conclude superiority of deep networks, their potential is highlighted.

In a global context, Rasekhschaffe and Jones (2019) compare ML model, including ANNs, monthly return forecasts to an OLS model and a naïve factor-based model from 1994 to 2016. On average, 5907 stocks are included each month from 22 developed markets across the US,

Europe, and Asia. 194 fundamental and technical input features are considered. Different training samples are used to separately train models, including the past 12-months' data, and data from the same calendar month over the prior 10 years. Based on long-short portfolios generated from percentile ranked forecasts, ML methods are found to outperform the linear benchmarks. Rasekhschaffe and Jones (2019) note that the outperformance is likely due to factor timing, with time-varying cross-sectional correlations between ML forecasts and factors. Like Tsai et al. (2011) and Abe and Nakayama (2018), ensemble models outperform.

Gu, Kelly and Xiu (2020) provide a comprehensive overview of ML methods to forecast equity returns in the US, including ANNs. An average of 6200 stocks are included each month in the analysis from 1957 to 2016, utilizing 94 firm specific, 74 industry dummy, 8 macroeconomic, and additional interaction variables. The 60-year dataset is split into an initial 18-year training sample, a 12-year validation sample, and a 30-year test period. The model is trained once per year to make monthly forecasts, with the training sample expanding annually to include the most recent 12 months. ML methods are found to outperform traditional forecasting models, with ANNs performing best. All methods select similar factors relating to momentum, liquidity, and volatility as predictors, potentially pointing to a preference for smaller firms across models. Model performance improves when considered in a portfolio context, likely due noise at the individual stock level and erratic behaviour of some smaller stocks (Gu, Kelly and Xiu, 2020).

Drobetz and Otto (2021) extend this research to the European stock market between 1990 and 2019. The study considers 22 firm specific characteristics, along with interaction terms and second- and third-order polynomials, and country and industry dummy variables. Only firms with a market capitalization above EUR 25 million with no missing information are included, resulting in an average of 835 firm observations each month. The model is trained with rolling 8-year training and 2-year validation data for each year of monthly forecasts. ML models are found to generate higher risk-adjusted returns than the OLS benchmark, even when accounting for transaction costs, with ANNs performing best. Equal weighted portfolios outperform value weighted portfolios, indicating the importance of small firms. Value and momentum factors are found to be informative across models (Drobetz and Otto, 2021).

Tobek and Hronec (2021) follow a similar methodology, extending the research to global developed markets, including the US and 22 countries across Europe and Asia. A maximum of 153 firm specific fundamental and technical factors are used to forecast monthly returns between 1995 and 2016, with variables only included post-publication. An initial 21-year training and 11-year validation sample is used, with the training sample size expanding each year. ANNs are found to outperform linear and other ML models. The results are robust when accounting for size and liquidity constraints, as well as transaction costs. Returns are mostly attributed to the long component, thus investors constrained on the short side can still benefit.

Variable importance is found to vary across regions and methods, with momentum and value factors notably informative on average.

While there is extensive evidence to support the use of ANNs to forecast and classify stock returns in Developed Markets, fewer studies have been conducted in Emerging Markets. Cao, Leggio and Schniederjans (2005) investigate the effectiveness of ANNs to forecast the daily returns of 376 companies listed on the Shanghai Stock Exchange between 1999 and 2002 relative to linear models. Univariate models using market returns for linear models and market beta for ANNs, as well as multivariate models which further consider Book-to-Market and Size per the Fama and French (1993) 3-Factor model are constructed. Surprisingly, the multivariate model is found to have lower forecast accuracy measures for both the linear and nonlinear models, suggesting that model performance reduces when a size and value variable are included. Consistent with Developed Market studies, ANN models are found to outperform.

Cao, Parry and Leggio (2011) extend the study on the Shanghai Stock Exchange. The models are updated to include lagged returns, and a univariate time-series model is considered in addition to dynamically updated versions of the models used by Cao, Leggio and Schniederjans (2005). The data is split between 1999-2002 with 367 firms and 2003-2008 with 1179 firms to account for the vast growth and changes in the Chinese market. Again, little difference in forecasting accuracy is found between the univariate and multivariate models, and the ANN models each outperform their linear counterparts (Cao, Parry and Leggio, 2011).

In a South African context, Hodnett (2010) investigates the performance of linear and non-linear models on the JSE. The study considers 38 firm-specific fundamental and technical attributes from 1997-2007. Firms with insufficient trading turnover and those that experience a major corporate event are excluded. Monthly forecasts are based on rolling 60-months training data, updated annually. A genetic algorithm determines the optimal subset of features to be included each year. The ANN models initially outperform in the test period, but underperform the linear model in the final years and overall. Like international findings such as Gu, Kelly and Xiu (2020), ensemble forecasts are found to outperform. Further, models fail to accurately identify losers and therefore make gains on the long side.

3.3.2 Artificial Neural Networks in Returns – Time-Series

Kanas and Yannopoulos (2001) compare the forecast accuracy of an MLP and a multivariate OLS model on both the Dow Jones index in the US and Financial Times index in the UK, considering three features relating to trading volume, dividends, and lagged index returns. The models are trained from 1980 to 1993, with a two-year validation period from 1994 to 1995, and five-year out-of-sample testing period from 1996 to 2000. It is found that for both indices, out-of-sample ANN forecasts are more accurate than linear forecasts. Additionally, errors of

linear models in both indices are explained by ANN forecasts, while the linear model fails to explain ANN forecast errors – highlighting the importance of non-linear terms for out-of-sample return forecasting (Kanas and Yannopoulos, 2001).

Enke and Thawornwong (2005) find ANNs outperform linear models at forecasting the level and direction of the S&P 500 using mainly macroeconomic factors. The dataset consists of 31 variables, reduced to a subset of 15 by an information gain analysis. The study considers various ANN models, including MLPs, with a training period spanning 1976 to 1992 and an out-of-sample testing period from 1992 to 1999. Both classification and point estimate ANN models are found to outperform linear models in terms of accuracy, with classification models performing best. From a trading perspective, several ANN models were found to outperform a buy-and-hold strategy. Interestingly, there was limited overlap between forecast accuracy and trading strategy performance, with MLP models showing relatively poor forecast accuracy relative to simulated trading results. Enke and Thawornwong (2005) do not discredit the EMH as transaction costs may limit the profitability of ANN strategies.

In Japan, Qiu, Song and Akagi (2016) examine the use of ANN models to forecast returns of the Nikkei 225. The study considers a set of 71 factors, with a training period from 1993 to 2007, and out-of-sample testing period from 2008 to 2013. Like Enke and Thawornwong (2005), a subset of input features is selected, reducing the factor set to 18 using a fuzzy surfaces analysis (Lin, Cunningham III and Coggeshall, 1996). A genetic algorithm and simulated annealing are used to optimize the weights and bias of the model and avoid convergence at local minima. The models are found to be capable of forecasting returns, with improved prediction accuracy following the inclusion of genetic algorithms and simulated annealing. The simulated annealing model required a much shorter runtime and improved average performance, while the genetic algorithm model required longer runtime, but greatly improved performance (Qiu, Song and Akagi, 2016).

In South Africa, recent studies primarily explore the use of ANNs in time-series forecasting. Ramuada (2018) compares the performance of an ANN model to several ARMA-type linear models to predict the direction of next-day stock moves for four JSE listed companies using historical stock prices as input features. Daily stock price data collected over a four year period from January 2013 to January 2017 is considered across two different data splits – one with a training period of 500-days, and one with a training period of 700-days. Both cases considered a 200-day test period. The ANN model is found to outperform the linear ARMA-type models for two companies in the 500-day training period, and three companies in the 700-day training period, and is therefore the best model in five of eight cases. The ANN model is correct 71% of the time in its worst case and correct 89% of the time in its best case, supporting the potential of ANNs to forecast the direction of returns.

Pyon (2021) considers the use of ANNs to forecast the closing share price for the next day, week and month of three South African banks. The analysis spans a 50-month period from April 2016, with 66% of the data set used for training, and 34% for testing, making use of a moving window approach. The key input feature considered is past prices, however, the impact of external factors, such as index prices and exchange rates, and a forward-looking business cycle indicator is considered. The ANN models outperform linear alternatives, with forecast errors declining for longer horizons. Interestingly, the inclusion of additional features led to a decline in model performance, indicating the importance of feature selection.

Nengovhela (2022) explores the forecasting ability of several ML models to forecast daily returns of stock market indices of BRICS countries, including South Africa, utilizing macroeconomic factors such as interest rates, exchange rates, global index prices, and commodity prices. Daily data is collected from January 2000 to May 2021, with 80% used for training and 20% for testing. While a Random Forest model is found to perform best in South Africa, the ANN model forms part of the model confidence set suggesting that it is equally good at the 95% confidence level. An ensemble containing K-Nearest Neighbour, Support Vector Regression and ANNs is found to improve forecasting ability relative to individual models. The VIX Index, exchange rates, US interest rates and commodity prices are found to be the most important input features for forecasting daily returns on the JSE.

3.3.3 Model Specification and Regularization

The flexibility of neural networks makes them a powerful tool. However, flexibility is also a drawback as experimentation is required to find a suitable model structure that may overfit a particular sample (Tobek and Hronec, 2021). Model performance is sensitive to several factors, such as hyperparameters, structure and regularization methods. This section considers model structure, training algorithms and activation functions used in prior research, summarized for the cross-sectional studies in Table 3.1 below.

Optimal model structure in existing literature varies, likely due to low signal-to-noise ratios in financial data, as well as divergence in feature selection, time periods, and the underlying market on which the studies are based. Earlier studies have typically favoured a single hidden layer (Kanas and Yannopoulos 2001; Eakins and Stansell, 2003; Olson and Mossman, 2003; Enke and Thawornwong, 2005; Cao, Leggio and Schniederjans, 2005; Tsai et al., 2011; Qiu, Song and Akagi, 2016), as Hornik, Stinchcombe and White (1989) show that neural networks with at least one hidden layer are universal approximators. However, many hidden neurons can be required, increasing the number of parameters to be estimated. Deeper networks require fewer nodes in each layer as composition functions

between layers reduce the number of parameters required to approximate relationships, favourable when limited data is available and positive for generalization (Aggarwal, 2018).

Considering single hidden layers, Tsai et al. (2011) finds the top 3 MLP models to include 5, 20 and 30 nodes. The 5-node model is best overall, particularly for positive returns, while the more complex 30 node model is best at predicting negative returns. Qiu, Song and Akagi (2016) also find a less complex 10-node model to perform best, as adding hidden nodes increases computation time with limited reduction in MSE. Certain parameter combinations with high node counts led to early termination likely due to finding a local minimum, highlighting the importance of parameter selection and regularization.

Moving to deep neural networks, recent studies offer conflicting findings on optimal network depth. Abe and Nakayama (2018) find that deeper networks tend to outperform shallow networks. While the results do not confirm superiority of deep networks to other ML methods, the authors conclude that ANNs with more layers could improve prediction accuracy of stock returns. Gu, Kelly and Xiu (2020) find that shallower networks outperform the deepest networks they consider. Performance improves from one hidden layer up to a peak at three hidden layers, after which performance deteriorates as additional layers are added. This is supported by Drobetz and Otto (2021), as the predictive slope of ANN models is found to deteriorate as more hidden layers are added, potentially due to overfitting. In contrast to the theory that deeper networks can be favourable when data is limited (Aggarwal, 2018), Drobetz and Otto (2021) suggest that deep networks may be too complex for small datasets with a limited predictor set. Tobek and Hronec (2021) find that the optimal network structure changes over time and varies by region. Considering two variants – narrow and wide – of one to three hidden layer models, those with two and three hidden layers were more commonly optimal. For single and two hidden layer models, only narrow networks were favoured.

The variation in optimal model structure may also be due to varying training algorithms, activation functions, hyperparameter estimates, and regularization methods used across studies. Training algorithms and activation functions are considered in Table 3.1 below.

Table 3.1: Summary of Model Architecture and Parameters in ANN Return Forecasting:

Study	Structure	Training Algorithm	Activation Function
Eakins and Stansell (2003)	6 – 3 – 1	Extended Delta Bar Delta Backpropagation	Tanh
Olson and Mossman (2003)	61 – 31 – 1[1]	Extended Delta Bar Delta Backpropagation	Tanh
Cao, Leggio and Schniederjans (2005)	Univariate: 1 – m – 1 for 4 ≤ m ≤ 10 Multivariate: 3 – n – 1 for 5 ≤ m ≤ 15	Online Backpropagation	Sigmoid
Hodnett (2010)	Genetic algorithm determines input features, cascade-correlation algorithm determines optimal number of hidden layers and nodes	Gradient Descent Backpropagation Extended Kalman Filter	Sigmoid
Tsai et al. (2011)	30 – n – 1 For n = 5, 10, 15, 20, 25, 30	Undisclosed	Undisclosed
Abe and Nakayama (2018)	125 – n – 1 Where, n represents 1, 1, 3 and 6 hidden layers with four node combinations each	Adam Optimization Backpropagation	Tanh
Rasekhschaffe and Jones (2019)	Two hidden layers including a bottleneck layer	Undisclosed	Tanh
Gu, Kelly and Xiu (2020)	920 – 32 – 1 920 – 32 – 16 – 1 920 – 32 – 16 – 8 – 1 920 – 32 – 16 – 8 – 4 – 1 920 – 32 – 16 – 8 – 4 – 2 – 1	Adam Optimization Backpropagation	ReLU
Drobetz and Otto (2021)	341 – 32 – 1 341 – 32 – 16 – 1 341 – 32 – 16 – 8 – 1 341 – 32 – 16 – 8 – 4 – 1 341 – 32 – 16 – 8 – 4 – 2 – 1	Adam Optimization Backpropagation	ReLU
Tobek and Hronec (2021)	150-150-1 150-32-1 150-150-150-1 150-32-16-1 150-150-150-100-1 150-32-16-8-1	Adam Optimization Backpropagation	ReLU

[1] Implied – with 61 input variables, a single hidden layer, and rules-based method to determine the number of hidden neurons equal to the sum of input and output variables divided by 2 (Olson and Mossman, 2003).

3.4 Anomalies of Interest

Despite their flexibility, ANNs require appropriate inputs to reach their potential. In theory, there is no need to reduce the input dimension, however, this can lead to convergence at local minima (Qiu, Song and Akagi, 2016). Feature selection and variable transformation is required to increase the signal-to-noise ratio prior to training or implementing parameter regularisation, helping to reduce overfit and improve generalisation (Rasekhschaffe and Jones, 2019).

Firstly, from linear models with South African relevance, Van Heerden and Van Rensburg (2015) consider 50 technical and fundamental factors. Findings support value and momentum factors in explaining the cross-section of returns, while the strength of the size premium is found to be decreasing. The Fama-French (2015) five-factor model suggests that along with a market factor, size, value, profitability, and investment are priced risk-factors.

From the literature on ANNs it is evident that relevant factors overlap with those identified by linear methods. Table 3.2 summarizes factors of interest across ANN studies.

Table 3.2: Input Features Used to Train ANN Models to Forecast Stock Returns:

Feature	Studies
Value	Eakins and Stansell, 2003 Cao, Leggio and Schniederjans, 2005 Hodnett, 2010 Rasekhschaffe and Jones, 2019 Drobetz and Otto, 2021 Tobek and Hronec, 2021
Size	Eakins and Stansell, 2003 Cao, Leggio and Schniederjans, 2005 Gu, Kelly and Xiu, 2020
Profitability	Kryzanowski, Galler and Wright, 1993 Rasekhschaffe and Jones, 2019
Growth	Rasekhschaffe and Jones, 2019 Hodnett, 2010
Dividends	Kanas and Yannopoulos, 2001 Eakins and Stansell, 2003
Leverage	Kryzanowski, Galler and Wright, 1993 Hodnett, 2010
Market Returns or Beta	Kanas and Yannopoulos, 2001 Cao, Leggio and Schniederjans, 2005
Market Liquidity and Trading Activity	Kryzanowski, Galler and Wright, 1993 Kanas and Yannopoulos, 2001 Gu, Kelly and Xiu, 2020 Tobek and Hronec, 2021
Volatility	Rasekhschaffe and Jones, 2019
Momentum	Hodnett, 2010 Gu, Kelly and Xiu, 2020 Drobetz and Otto, 2021 Tobek and Hronec, 2021

3.5 Conclusion

Within asset pricing research, studies undertake different strategies to identify priced risk factors and forecast returns. A range of anomalies have been identified that explain the cross-section of stock returns, however the strength of relationships tends to vary over time. Developments in computational efficiency and ML techniques offer new ways to uncover complex, non-linear relationships between anomalies and stock returns, including ANNs.

ANNs have shown strong potential in return forecasting, generally outperforming traditional linear models and other ML methods. The optimal model structure varies based on the information set and regularization techniques used, with conflicting findings relating to the benefit of using deeper network structures. Limited research exists in a South African context with respect to the relative effectiveness of deeper network structures to shallow networks and linear methods, with current research suggesting limited efficacy of ANNs relative to linear methods.

4. Data

4.1 Introduction

This section considers the data used in this study. Thank you to my supervisor, Professor Paul van Rensburg, and the Department of Finance and Tax for providing the data sourced from Bloomberg. The final dataset consists of 33 explanatory variables across 192 firms. Section 4.2 provides a description of the data and features. Section 4.3 discusses adjustments made to the data. Section 4.4 considers potential biases, and Section 4.5 concludes. Code used in preprocessing the data can be found in Appendix A Figure A.1.

4.2 Data Description

The initial monthly dataset considers 112 firm-specific variables obtained for 213 stocks over the period January 2005 to August 2020. The dataset is adjusted for corporate events, and returns are adjusted for dividend payments. Any firm with missing forward returns information during a month is removed from the dataset for that month. Following Hodnett (2010) and Van Rensburg and Robertson (2003), any share with turnover less than 0.01% in any given month is excluded for that month, ensuring stocks trade at least once.

Input variables missing more than 10% of observations are removed, resulting in 46 fewer input features. Two variables – c24mepsp and epsg24 – are reincluded despite missing more than 10% of observations as they were found to have relatively high Pearson correlations with forward returns in the training sample period (January 2005 to December 2015). From this new subset, similar input variables to those used by Hodnett (2010) and Van Heerden and Van Rensburg (2015) are selected for the final dataset, as these are South African studies.

Non-meaningful observations with negative market value and price are removed. Deviating from standard practice, firms with negative earnings yield and cashflow-to-price are excluded. While this limits the robustness of the study, comparisons of model performance relative to an equal weighted benchmark of included stocks, as well as between linear and ANN models, still provide valuable insights into model effectiveness and optimal network structure. Further, following Abe and Nakayama (2018), observations missing more than 50% of input variables in a month are removed for that month.

The final dataset consists of 32 explanatory factors across 192 firms, with an average of 126 firms each month. Table 4.1 below shows the final breakdown of independent variables into fundamental and technical factors. Fundamental factors relate to value, growth, liquidity and solvency, and operating performance. Technical factors relate to price momentum, size, and volatility. Table 4.2 provides a breakdown of firm count by sector.

Table 4.1: List of Fundamental and Technical Attributes

(a) Fundamental Factors

	Variable	Description
Value	bvtm	Log of Book Value to Market
	cftp	Cash-flow to Price
	ey	Earnings Yield
	salespt	Sales to Price
Growth	c12mcftp	12-month Change in Cash-Flow divided by Price
	c24mepsp	24-month change in Headline Earnings per Share divided by Price
	epsg12	12-month EPS Growth
	epsg12ml12	12-month EPS Growth less 12-month EPS Growth lagged by 12 months
	epsg24	24-month EPS Growth
	salesg12	12-month Sales Growth
	sd24mepsg12	Standard deviation of 12-month EPS Growth over 24 months
	tag12	12-month Growth in Total Assets
Liquidity and Solvency	de	Debt Equity Ratio
	cde12	12-month change in Debt Equity Ratio
	icbt	Inverse of Interest Cover before Tax
Operating Performance	atg12	Growth in Asset Turnover over 12 Months
	cfroa	Cashflow Return on Assets
	cfroe	Cashflow Return on Equity
	cfte	Cashflow to Earnings
	pm	Trailing 12 Month Profit Margin
	pmxat	Gross Profitability
	roa	Return on Assets
	roag12	Growth in Return on Assets over 12-months

(b) Technical Factors

	Variable	Description
Momentum	mom1	1 Month Prior Total Return
	mom3	3 Month Prior Total Return
	mom6	6 Month Prior Total Return
	mom12	12 Month Prior Total Return
	mom12minus1	1 Month Lagged 11 Month Return (mom12 – mom1)
Size	lmv	Log Market Capitalisation
	lnp	Log Share Price
Volatility	beta500d_zar_jalsh	Beta using 500 days daily data in ZAR
	retvar12	Variance of 12-month Returns

Table 4.2: Breakdown of Sectors by Firm Count

Sector	Number of Firms
Consumer Discretionary	28
Consumer Staples	25
Energy	2
Financials	32
Health Care	6
Industrials	20
Information Technology	4
Materials	30
Real Estate	42
Telecommunication Services	3

4.3 Adjustments to the Dataset

To limit issues when training the model, several adjustments are made to the dataset. Firstly, missing values are replaced by the cross-sectional median each month similarly to Gu, Kelly and Xiu (2020) and Drobetz and Otto (2021). Second, the input data is corrected for outliers by applying the Tukey 2.2 Rule cross-sectionally each month (Tukey, 1977; Hoaglin and Iglewitz, 1987), based on the following formula:

$$Lower\ Bound_{j,t} = Q1_{j,t} - C_{j,t} \tag{13}$$

$$Upper\ Bound_{j,t} = Q3_{j,t} + C_{j,t} \tag{14}$$

Where:

$Q1_{j,t} = First\ quartile\ of\ input\ feature\ j\ in\ month\ t$

$Q3_{j,t} = Third\ quartile\ of\ input\ feature\ j\ in\ month\ t$

$C_{j,t} = 2.2\ x\ (Q3_{j,t} - Q1_{j,t})$

Following van Rensburg and Robertson (2003), the input variables are standardized cross-sectionally each month to have a mean of zero and standard deviation of one. Mathematically:

$$Norm(x_{i,j,t}) = \frac{x_{i,j,t} - \mu(X_{j,t})}{\sigma(X_{j,t})} \tag{15}$$

For each observation i of input feature j in month t.

Monthly standardization maintains the relative relationships and characteristics of data within each month and allows the test dataset to be standardized independent of the training dataset. While the Min-Max Scaling function is common in research on ANNs to reduce the impact of

outliers, the z-score normalization method is used in this study to maintain normality assumptions for the linear models.

4.4 Potential Biases

This study avoids look-ahead and survivorship bias due to the structure of the data. Variables are lagged to ensure that forecasts are only made from available information, and the study is conducted cross-sectionally, so no firms are excluded for not being in the dataset for the full length of the study.

4.5 Conclusion

This chapter introduces the dataset and provides an overview of adjustments made to the data. The final dataset consists of 32 firm specific fundamental and technical input factors that have been identified in research to describe the cross-section of returns. There is an average of 126 firms each month from the JSE extending across a 188-month period from January 2005 to August 2020, largely consisting of Real Estate, Financial, Materials, Consumer Discretionary, Consumer Staples, and Industrial firms.

5. Methodology

5.1 Introduction

This chapter considers the methodology followed in this study. Section 5.2 describes the split of data into training and testing subsets. Section 5.3 introduces the structure of the linear models that are utilized in this study. A univariate analysis is used to identify variables to be included in both the multivariate linear and artificial neural network (ANN) models. Section 5.4 provides an overview of the ANN structure, including the regularization and tuning techniques implemented to improve model fit. Section 5.5 breaks down the performance evaluation methods, including the portfolio generating strategy. Section 5.6 concludes. Appendix A contains a subset of the code used to perform the analysis, including the linear analysis in Figure A.2, a 3 hidden layer ANN in Figure A.3, and performance evaluation in Figure A.4.

5.2 Splitting the Dataset

The dataset is split into two separate subsets – a training set and a test set. The training set spans an 11-year period from January 2005 to December 2015. The out-of-sample test set spans the rest of the period from January 2016 to August 2020 (56 months). The data maintains a temporal order ensuring the model is trained on data that would be available at the time. The test sample is separate from the training process and is used only to assess the model performance once training is complete.

5.3 Linear Model

A similar process to Van Heerden and Van Rensburg (2015) is followed to find an appropriate OLS model, making use of cross-sectional regressions like Fama and Macbeth (1973). Firstly, univariate cross-sectional regressions are conducted for each input feature to create a time-series of cross-sectional slopes for each factor. The process is applied to the training dataset. Mathematically (Van Heerden and Van Rensburg, 2015):

$$r_{i,t+1} = \gamma_{0,t+1} + \gamma_{1,t+1} A_{i,t} + \varepsilon_{i,t+1} \tag{16}$$

Where:

$r_{i,t+1}$ = realised return on share i for month *t* + 1

$\gamma_{0,t+1}$= intercept term

$\gamma_{1,t+1}$= cross-sectional slope coefficient

$A_{i,t}$ = value of the factor for stock *i* at the end of month *t*

$\varepsilon_{i,t+1}$= error term

The time-series mean of each cross-sectional slope coefficient is tested for statistical difference from zero using Student's t-test to determine potential factors to use in a multivariate model. Mathematically:

$$t - stat_j = \frac{\bar{\gamma}_j}{\frac{\sigma_{\bar{\gamma}_j}}{\sqrt{T}}} \tag{17}$$

Where:

$\bar{\gamma}_j =$ the time-series mean of cross-sectional slope coefficients of input feature j over the training period, T

$\sigma_{\bar{\gamma}_j}/\sqrt{T}$ = standard error of $\bar{\gamma}_j$

$\sigma_{\bar{\gamma}_j} =$ standard deviation of the time-series slope coefficients of input feature j over the training period, T

Following Hodnett (2010), the geometric cumulative cross-sectional slope coefficient is examined to assess the consistency of factors over the training sample period:

$$\text{Cumulative Slope Coefficient}_\text{j} = \prod_{t=0}^{T}\left(1 + \gamma_{j,t+1}\right) \tag{18}$$

Where:

$\gamma_{j,t+1}$ = cross-sectional slope coefficient for factor j in month t

T = the sample period over which the cumulative slope coefficient is calculated

Univariate significant factors at the 5% significance level are regressed in a multifactor model of similar form. Mathematically:

$$r_{i,t+1} = \gamma_{0,t+1} + \sum_{j=1}^{J} \gamma_{j,t+1} A_{j,i,t} + \varepsilon_{i,t+1} \tag{19}$$

Where:

$r_{i,t+1}$ = realised return on share i for month t + 1

$\gamma_{0,t+1}$= intercept term

$\gamma_{j,t+1}$= cross-sectional slope coefficient for factor J

$A_{j,i,t}$ = value of factor J for stock i at the end of month t

$\varepsilon_{i,t+1}$= error term

To avoid multicollinearity, input variables with the lowest t-statistic that have Pearson correlations above 0.7 with other inputs over the training period are excluded:

$$Pearson\ Correlation\ Coefficient = \rho_{X,Y} = \frac{\sigma_{X,Y}}{\sigma_X \sigma_Y} \tag{20}$$

Where:

$\sigma_{X,Y}$ = covariance of style attributes X and Y over the training period

σ_X, σ_Y = variance of style attributes X and Y, respectively, over the training period

Two multivariate linear models are considered. The first includes all univariate significant factors. From this, multivariate significant factors based on Student's t-test at the 5% level are used to create a reduced subset of variables for the second model.

Following Hodnett (2010), once a model has been estimated, an expected return factor model is constructed to forecast out-of-sample returns. The expected factor payoff is the time-series mean of the cross-sectional slope coefficients for each factor. The new specification for expected returns becomes:

$$E(r_{i,t+1}) = E(\gamma_{0,t+1}) + \sum_{j=1}^{J} E(\gamma_{j,t+1}) A_{j,i,t} \tag{21}$$

Where:

$E(r_{i,t+1})$ = expected return for share i in month *t* + 1

$E(\gamma_{0,t+1})$ = estimated intercept term

$E(\gamma_{j,t+1})$ = expected payoff to factor j

$A_{j,i,t}$ = value of factor j for stock *i* at the end of month *t*

Hodnett (2010) follows a dynamic moving-window approach in which factors are updated annually, ensuring relevant factors over the past 60 months are included in forecasting returns. A static approach is also tested in which factors are unchanged, with only the expected factor payoff, $E(\gamma_{j,t+1})$, updated annually. In both cases, the dynamic approach to expected factor payoffs better captures their time-varying nature.

For simplicity, this study does not employ a moving-window approach, but rather calculates the expected return factor model parameters once based on the full training sample period. While this will likely reduce model performance, a similar approach is used in the ANN model. Comparisons between models should still highlight differences in efficacy.

5.4 Neural Network Structure and Regularization

5.4.1 Model Structure

ANNs of various structural forms are investigated, from a single hidden layer up to five hidden layers. Common practice is followed such that each hidden layer has a number of neurons in the form 2^n, with the number of neurons halving in each successive layer. For each network depth, six hidden node variations are implemented. While this is unlikely to uncover the overall optimal model, it should provide insights into enhanced model specification in a South African context, and deepen understanding of the efficacy of using ANNs to forecast returns on the JSE. This should provide a reasonable lower bound of the models' potential (Gu, Kelly and Xiu, 2020). Table 5.1 summarizes the network structures which are tested.

Table 5.1: Summary of Hidden Layer Structures Analysed

Number of Hidden Layers	Number of Hidden Nodes
1	8 16 32 64 128 256
2	8 – 4 16 – 8 32 – 16 64 – 32 128 – 64 256 – 128
3	8 – 4 – 2 16 – 8 – 4 32 – 16 – 8 64 – 32 – 16 128 – 64 – 32 256 – 128 – 64
4	16 – 8 – 4 – 2 32 – 16 – 8 – 4 64 – 32 – 16 – 8 128 – 64 – 32 – 16 256 – 128 – 64 – 32 512 – 256 – 128 – 64
5	32 – 16 – 8 – 4 – 2 64 – 32 – 16 – 8 – 4 128 – 64 – 32 – 16 – 8 256 – 128 – 64 – 32 – 16 512 – 256 – 128 – 64 – 32 1028 - 512 – 256 – 128 – 64

Table 5.2 summarizes the model parameters that are unchanged across all models. Like Gu, Kelly and Xiu (2020), the ReLU activation function and Adam Optimizer are used, and the model is optimized with the objective of minimizing the mean square error. The Adam optimizer is implemented with the initial learning rate set to 0.00001, with the other hyperparameters kept at default in Keras (beta_1=0.9, beta_2=0.999, epsilon=1e-08, and decay=0.0). Batch Size was maintained at 64. The number of epochs was set to 500, with early stopping implemented with patience set to 10 to prevent overfitting. These choices were guided by preliminary experimentation of hyperparameter combinations, as they were found to offer a balance between computational efficiency and learning during training. Formal grid search and cross-validation hyperparameter tuning was not conducted for each model structure due to computational constraints. Additionally, keeping the hyperparameters consistent across model architectures allows for performance comparisons to be more easily attributed to model depth and node counts rather than tuning effects.

Table 5.2: Parameters Consistent Across All Model Structures

Parameter	**Setting**
Activation Function	Rectified Linear Unit Function (ReLU)
Training Algorithm	Adam
Learning Rate	0.00001
Epochs	500
Batch Size	64
Loss	Mean Square Error

Adopted from Cao, Leggio and Schniederjans (2005), a single hidden layer neural network with no direct link between input nodes and output nodes can be defined as:

$$\hat{y}(\theta) = g(\theta, x) = \sum_{i=1}^{nh} w_{2_i} \phi \left(\sum_{j=1}^{n} w_{1_{i,j}} x_j + b_{1_{j,i}} \right) + b_2 \qquad (22)$$

Where:

x_j = inputs

$\hat{y}$ = output

θ = set of parameters $\{w_{1_{i,j}}, b_{1_{j,i}}, w_{2_i}, b_2\}$

$w_{1_{i,j}}$ = weights applied to the input variables to hidden neurons

w_{2_i} = weights applied to the outputs of the hidden neurons to the output layer

$b_{1_{j,i}}$ = bias term to the hidden layer

b_2 = bias term to the output layer

$\phi(.)$ = activation function

n = number of inputs

nh = number of neurons in the hidden layer

Like the linear models, two sets of input variables are considered. As this study is looking to compare the efficacy of ANNs relative to linear models, the same set of input features are used. Separate use of all univariate significant variables and the reduced variable subset should provide insights into the importance of feature selection in a South African context.

5.4.2 Regularization and Tuning

To improve model performance, various techniques considered in Section 2.4.6 are utilized. These methods help improve generalization by reducing the impact of over- and underfitting. As discussed in Section 4.3, the inputs are standardised. Following Gu, Kelly and Xiu (2020), Batch Normalisation is implemented to standardise the inputs into each hidden layer. Early stopping is implemented during training with patience set to 10. In the deep network structures (those with more than one hidden layer), Dropout is used between the first and second hidden layers with the dropout rate set to 0.5.

An ensemble approach is implemented to limit overfitting and reduce convergence at local minima. Each model is separately trained for three random seeds. For each seed, 5-fold cross-validation is used to generate separate forecasts, resulting in an ensemble of 15 iterations. While cross-validation is typically used to tune hyperparameters, for simplicity the parameters in Table 5.2 are kept constant. This approach reduces the likelihood of finding the optimal configuration for each model, but creates a direct comparison across structures.

As the ReLU activation function is used and the set of input features is not large, L1 and L2 penalisation of the weight parameters is not implemented, as there is little need to further reduce the dimension of the input data set. As noted in Section 2.4.3, both the ReLU activation function and penalty-based regularization techniques act as feature selectors and reduce the dimension of the input feature set (Aggarwal, 2018; Scarpino, 2018). Early testing of model structures and regularization methods making use of these methods led to little variation in predictions, typically centred near the mean return of the training sample.

Lastly, the HeNormal initializer (He et al., 2015) is used to set the initial network weights. While the method was developed for use in extremely deep models, it was specifically derived for the ReLU activation function (He et al., 2015). Weights in each layer l are initialized randomly from a normal distribution with a mean of zero and standard deviation of $\sqrt{\frac{2}{n_l}}$, where n_l represents the number of connections between layers l and $l-1$.

5.5 Performance Evaluation

Several performance metrics are considered to evaluate models. While the metrics are focused on testing out-of-sample performance, they are also considered for the training sample to gain further insights into over- and underfitting.

Firstly, following Gu, Kelly and Xiu (2020), the performance of each model is assessed via an out-of-sample R^2 metric, defined as:

$$R^2_{oos} = 1 - \frac{\sum_{(i,t)\in T}\left(r_{i,t+1} - \hat{r}_{i,t+1}\right)^2}{\sum_{(i,t)\in T} {r_{i,t+1}}^2} \tag{23}$$

Where:

$r_{i,t+1}$ = realised return on share i for month $t + 1$

$\hat{r}_{i,t+1}$ = predicted return for share i in month $t + 1$

T = testing subsample

Gu, Kelly and Xiu (2020) do not demean the denominator as predicting future returns using mean forecasts is generally found to underperform forecasts of 0. At the stock level, mean returns contain too much noise to include in the model.

In addition to the demeaned R^2 metric, the slope coefficient and associated t-statistic of the linear regression of forecast and actual returns is considered to assess if the slope coefficient is significantly different from zero over the whole out-of-sample period. Regression:

$$r_{i,t+1} = \gamma_0 + \gamma_1 \hat{r}_{i,t+1} + \varepsilon_{i,t+1} \tag{24}$$

Where:

$r_{i,t+1}$ = realised return on share i for month $t + 1$

$\hat{r}_{i,t+1}$ = predicted return for share i in month $t + 1$

γ_0= intercept term

γ_1= slope coefficient

$\varepsilon_{i,t+1}$= error term

T-statistic:

$$t - stat = \frac{\gamma_1}{\sigma_{\gamma_1}} \tag{25}$$

Where:

$\sigma_{\gamma_1} =$ standard deviation of the slope coefficients

Following Hodnett (2010), the average information coefficient (IC) and Qian and Hua (2004) information ratio (IR) are considered. Additionally, the cumulative IC is calculated. These metrics provide insights into the average and cumulative model performance based on monthly correlations between forecast and realized returns, as well as the variation in performance over time. Average IC:

$$\overline{\mathrm{IC}} = \frac{\sum_{t=1}^{T} \rho_t\left(r_{i,t+1}, \hat{r}_{i,t+1}\right)}{T} \tag{26}$$

Where:

$r_{i,t+1}$ = realised return on share i for month $t + 1$

$\hat{r}_{i,t+1}$ = predicted return for share i in month $t + 1$

$\rho_t(r_{i,t+1}, \hat{r}_{i,t+1})$ = cross-sectional correlation coefficient between $r_{i,t+1}$ and $\hat{r}_{i,t+1}$ in month t

T = number of months in the test period

Cumulative IC:

$$\text{Cumulative IC}_\mathrm{T} = \prod_{t=1}^{T}(1 + IC_t) \tag{27}$$

Where:

IC_t = cross-sectional information coefficient for month t

Qian and Hua IR:

$$Quan\ and\ Hua\ IR = \frac{\overline{IC}}{\sigma_{IC}} \tag{28}$$

Where:

$\overline{IC}$ = average IC

σ_{IC} = standard deviation of the monthly IC

In addition, the slope t-statistic of monthly regressions of forecast and realized returns is considered in a similar manner to the univariate analysis (Section 5.3). Hodnett (2010) notes that the mean slope coefficient is an alternative to the average IC, while the t-statistic accounts for the volatility of forecasting accuracy through the standard error of the slope coefficient:

$$slope\ t-stat\ =\ \frac{\overline{\gamma_1}}{\frac{\sigma_{\overline{\gamma_1}}}{\sqrt{T}}} \tag{29}$$

Where:

$\overline{\gamma_1}$= average slope coefficient of the monthly regressions of forecast and realized returns

$\sigma_{\overline{\gamma_1}}/\sqrt{T}$ = standard error of $\overline{\gamma_1}$

The final performance criteria apply a portfolio approach to eliminate the noise inherent at the individual stock level. Forecast returns are ranked each month to form separate equal weighted portfolios for the top and bottom 25% of forecast returns, rebalanced monthly. To assess model performance, the average monthly returns, cumulative payoff, annualized returns, and risk-adjusted returns (annualized return/standard deviation of monthly returns) are considered for the portfolios relative to a benchmark of an equally weighted portfolio of all stocks rebalanced monthly. Trading frictions are ignored, however this should not impact the purpose of comparing performance across models.

5.6 Conclusion

This chapter provides an overview of the methodology followed in this study. The cleaned dataset is split into two groups for training and testing the various models. A univariate linear regression analysis is performed to identify factors of interest. From this, two benchmark linear OLS models are constructed – one with all univariate significant factors, and a second with a reduced factor set. For both datasets, various MLP models are constructed with varying model depths and node counts. Several regularization methods are used to improve out-of-sample performance.

Once all models are trained, various performance evaluation metrics are considered to compare the out-of-sample performance of the linear and ANN models. To reduce noise at the stock level, portfolios are constructed for the top and bottom ranked stocks in terms of forecast returns to further assess model performance.

6. Research Findings and Discussion

6.1 Introduction

This chapter presents the results and provides insights into the key research findings. Section 6.2 provides an overview of the univariate analysis and factor performance during the training period. Section 6.3 considers the construction and performance of the multivariate factor models, while the artificial neural network (ANN) models are considered in Section 6.4. Key findings and insights across the models are discussed in Section 6.5. Section 6.6 discusses limitations of the study and considers future research areas. Section 6.7 concludes.

6.2 Univariate Linear Models

Table 6.1 shows the output of the univariate regressions of each factor over the training period. Factors with mean slope coefficients significantly different from zero at the 5% level are shown in bold. The cumulative slope coefficient of each factor is shown in Figure 6.1 Panel A to Panel G to assess consistency, direction, and magnitude over time.

Table 6.1 indicates that 11 of the 32 considered fundamental and technical factors have mean slope coefficients significantly different from zero, and are therefore considered as candidate factors for the multivariate and ANN models. The factor styles are further discussed in Sections 6.2.1 to 6.2.7.

6.2.1 Value

Table 6.1 indicates that ey has a mean slope coefficient significantly different from zero. In line with previous studies, value is positively related to stock performance (see for example Van Rensburg and Robertson (2003) and Van Heerden and Van Rensburg (2015) in South Africa). A similar positive relationship is evident for cftp, however it is not significant. Both bvtm and salespt surprisingly have a negative relationship with forward returns.

Hodnett (2010) and Van Heerden and Van Rensburg (2015) both identified cftp and bvtm to be the most significant value factors on the JSE, with the strength of bvtm increasing and cftp decreasing over time. Figure 6.1 Panel A shows that the cumulative slope coefficient of value factors is inconsistent through time. At the start of the training period from 2005, cftp and bvtm have the highest cumulative payoff, in line with Hodnett (2010) and Van Rensburg and Van Heerden (2015), however both are volatile over the full period. The cumulative slope coefficient is highest for ey, with increased reliability post-2009.

Inconsistencies may be due to reduced profitability as investor interest in cftp and bvtm grew (Mclean and Pontiff, 2016). Further, changes in market dynamics following the Global

Financial Crisis (GFC) may have contributed to changing factor payoffs. For the purposes of this study, ey is considered a candidate factor.

6.2.2 Growth

In Table 6.1, five of the eight growth metrics considered are significant, with growth in cashflow, earnings, and sales associated with outperformance. In contrast to Hodnett (2010) and Van Heerden and Van Rensburg (2015), this supports the presence of a growth effect on the JSE. Hodnett (2010) identified significant growth factors in the later subsample, but volatility in factor payoffs made them unsuitable candidates for forecasting returns. Van Heerden and Van Rensburg (2015) found no significant growth factors over the whole sample period, but like Hodnett (2010) found increasing significance in the later subsample.

Figure 6.1 Panel B highlights consistent positive slope coefficients for Growth factors over the training sample period other than during 2008. Favourable cumulative slope coefficients for c12mcftp, c24mepsp, epsg12, epsg24 and salesg12 further support their potential as candidate factors in the multivariate and ANN models. Like the value factors, cftp and bvtm, the change in factor significance relative to previous findings may relate to changing market dynamics following the GFC. For epsg12ml12, tag12 and sd24mepsg12, volatile cumulative slope coefficients suggest that they are unsuitable candidate factors, in line with Table 6.1.

6.2.3 Liquidity and Solvency

Per Table 6.1, no liquidity and solvency factors considered have mean slope coefficients significantly different from zero. Negative mean coefficients for de and cde12 indicate that firms with high or increasing debt tend to have lower returns. In contrast to this, a positive coefficient for icbt indicates that firms with higher interest expense as a percentage of earnings before interest and tax tend to outperform. Figure 6.1 Panel C shows that cumulative factor payoffs are low and inconsistent over time. These findings support Hodnett (2010) and Van Heerden and Van Rensburg (2015), with limited evidence of liquidity and solvency effects on the JSE. Based on this, these factors are not considered as candidates for the multivariate linear and ANN models.

6.2.4 Operating Performance

Of the eight operating performance metrics considered in Table 6.1, roag12 is found to be significant. A positive mean coefficient suggests that improving ROA is related to outperformance. Positive coefficients for atg12, cfroa, cfroe, pm, pmxat and roa further suggest that fundamentally superior companies are associated with higher returns. Figure 6.1 Panel D indicates volatile slope coefficients for operating performance factors, with roag12 offering the highest cumulative slope coefficient over the training period despite poor

performance from 2008 to 2010. The inconsistent performance of these factors is in line with Hodnett (2010) and Van Heerden and Van Rensburg (2015), who generally found no significant operating performance factors. Despite this, roag12 is considered as a potential candidate factor based on its significant average slope coefficient in the sample period.

6.2.5 Momentum

Table 6.1 shows that four of the five momentum factors are significant. The mean coefficients of mom6, mom12 and mom12minus1 suggest positive momentum effects, while the negative mean coefficient of mom1 suggests a short-term reversal effect. Figure 6.1 Panel E shows moderately consistent positive slope coefficients for mom6, mom12 and mom12minus1, while the short-term reversal factor, mom1, experiences consistent negative slope coefficients. These findings are in line with Hodnett (2010) and Van Heerden and Van Rensburg (2015), in which positive momentum over 6- and 12-months, and short-term reversal effects over 1-month were identified on the JSE. Based on this, mom6, mom12, mom12minus1 and mom1 are considered as candidate factors.

6.2.6 Size

In Table 6.1, neither size factor considered is significant. While the mean coefficient is negative in line with the small-firm effect, a lack of significance contrasts existing research. Hodnett (2010) found both lmv and lnp to be negatively and significantly related to forward returns on the JSE, with the strength of the negative relationship increasing over time. A small-firm effect on the JSE is supported by Van Heerden and Van Rensburg (2015), however the effect is found to be decreasing over time. Figure 6.1 Panel F supports a decreasing size premium on the JSE from 2005 to mid-2009. However, from 2009 to 2016 there appears to be a consistent negative payoff to both lmv and lnp, suggesting a return of the small-firm effect on the JSE. Despite this, lmv and lnp are not considered as candidate factors due to the lack of significance over the training period.

6.2.7 Volatility

Table 6.1 shows that neither volatility factor considered is significant. Both factors have a negative mean coefficient, suggesting that riskier stocks underperformed over the training period. Figure 6.1 Panel G highlights the inconsistencies of these factors, particularly retvar12. From 2005 to 2008, more volatile stocks tended to be rewarded with higher returns, with a large reversal in 2008, followed by inconsistent factor payoffs over time. It is evident that high beta stocks tended to underperform following the GFC. Van Heerden and Van Rensburg (2015) similarly found inconsistent results for retvar12 and a beta factor. Due to the lack of significance, neither factor is considered as a candidate.

Table 6.1: Univariate Regression Results

This table shows the time-series mean slope coefficients over the training period of the univariate regressions discussed in Section 5.3, along with the associated standard errors and t-statistics. Significant factors at the 5% significance level are in bold. The final column shows the rank of the absolute value of the t-statistics.

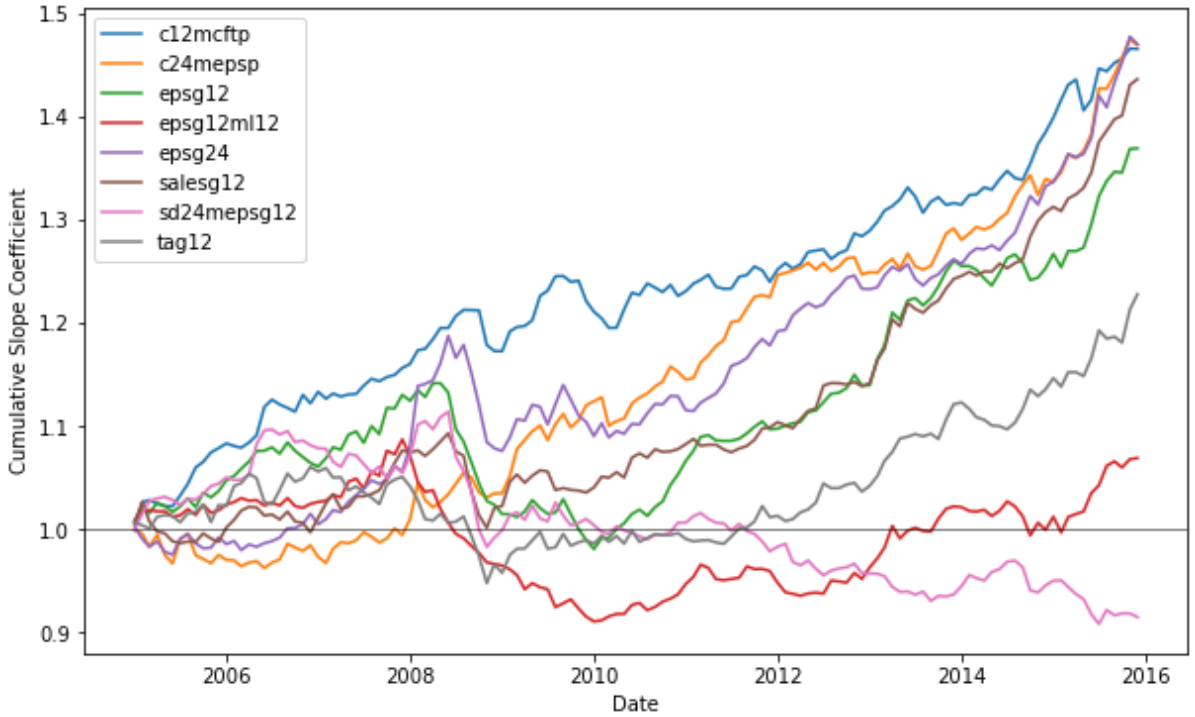
	Factor	Average Coefficient	Standard Error	t-statistic	Rank
Value	bvtm	-0.000	0.001	-0.346	30
	cftp	0.001	0.001	0.943	23
	ey	**0.002**	**0.001**	**2.190**	**9**
	salespt	-0.001	0.001	-0.548	29
Growth	**c12mcftp**	**0.003**	**0.001**	**4.293**	**1**
	c24mepsp	**0.003**	**0.001**	**3.601**	**3**
	epsg12	**0.002**	**0.001**	**2.964**	**7**
	epsg12ml12	0.001	0.001	0.736	26
	epsg24	**0.003**	**0.001**	**3.211**	**5**
	salesg12	**0.003**	**0.001**	**3.567**	**4**
	sd24mepsg12	-0.001	0.001	-0.750	25
	tag12	0.002	0.001	1.935	12
Liquidity and Solvency	de	-0.000	0.001	-0.621	28
	cde12	-0.000	0.001	-0.154	31
	icbt	0.001	0.001	0.667	27
Operating Performance	atg12	0.001	0.001	1.284	20
	cfroa	0.002	0.001	1.838	14
	cfroe	0.001	0.001	1.237	21
	cfte	-0.001	0.001	-1.181	22
	pm	0.001	0.001	1.516	17
	pmxat	0.001	0.001	1.489	18
	roa	0.001	0.001	1.471	19
	roag12	**0.002**	**0.001**	**2.291**	**8**
Momentum	**mom1**	**-0.002**	**0.001**	**-2.137**	**10**
	mom3	0.001	0.001	0.874	24
	mom6	**0.003**	**0.001**	**2.089**	**11**
	mom12	**0.004**	**0.001**	**3.169**	**6**
	mom12minus1	**0.005**	**0.001**	**3.874**	**2**
Size	lmv	-0.002	0.001	-1.642	16
	lnp	-0.002	0.001	-1.745	15
Volatility	beta500d_zar_jalsh	-0.003	0.002	-1.848	13
	retvar12	-0.000	0.001	-0.119	32

Figure 6.1: Univariate Regression – Cumulative Slope Coefficient

Panel A to Panel G provide a visual representation of the cumulative slope coefficients over the training period from the univariate regressions, discussed in Section 5.3.

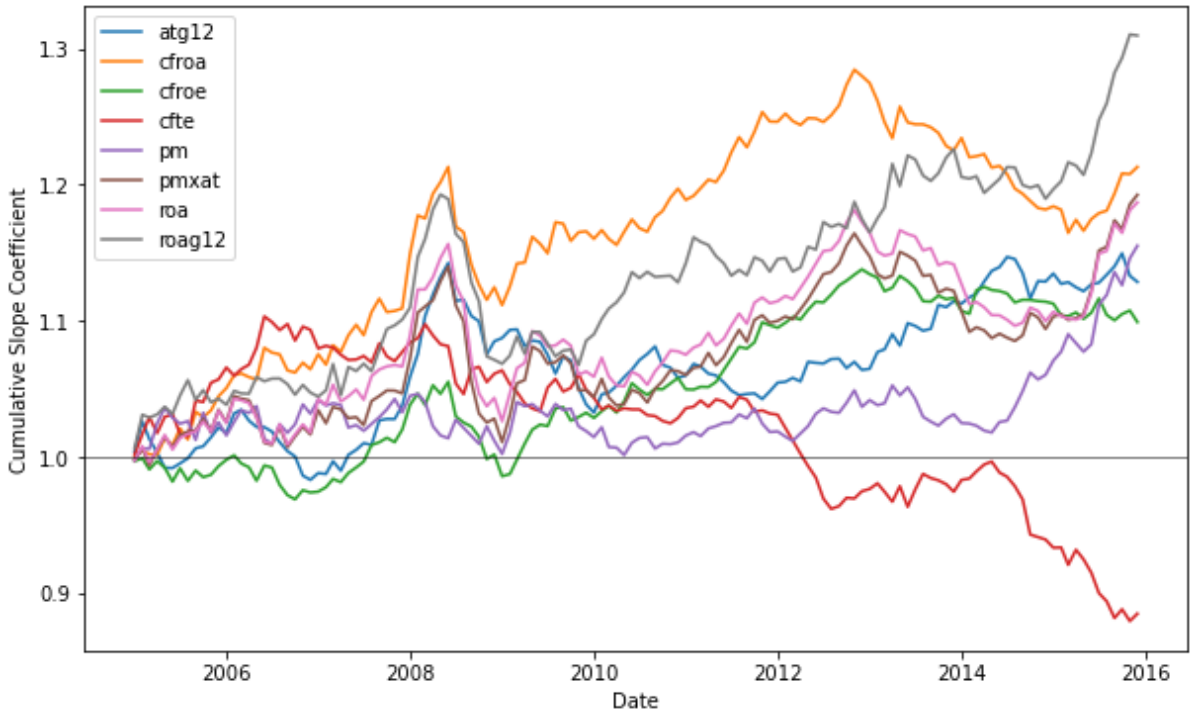
Panel A: Value

Panel B: Growth

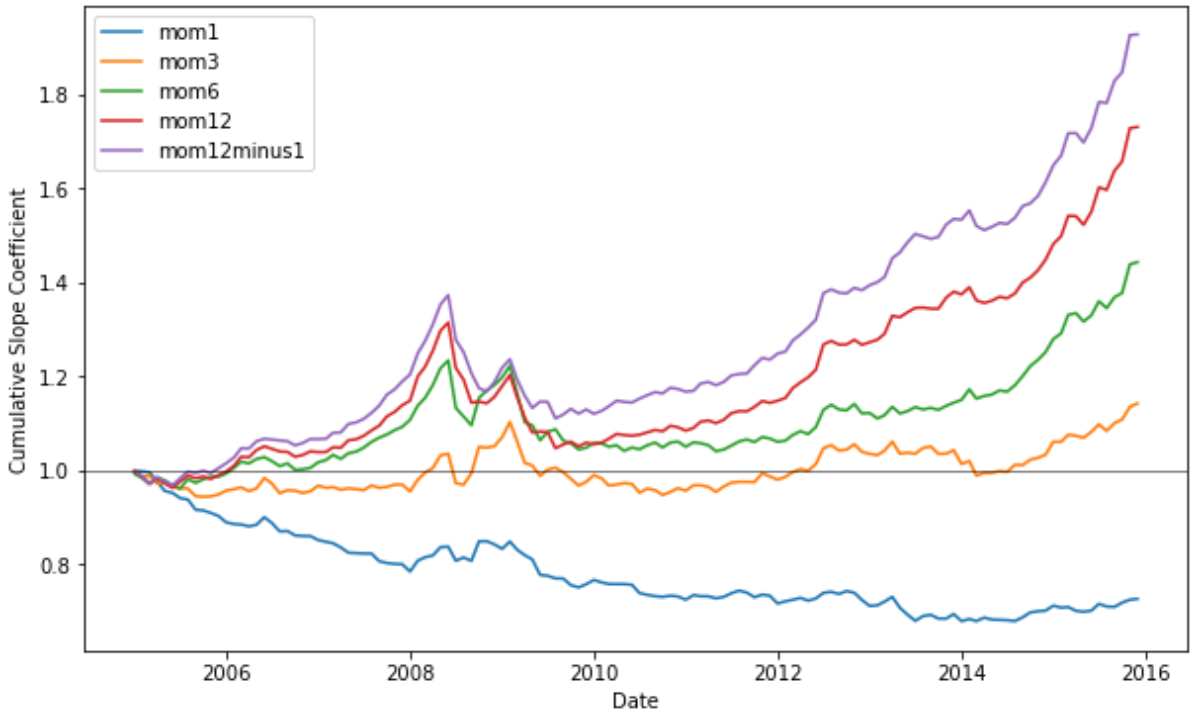


Panel C: Liquidity and Solvency

Panel D: Operating Performance



Panel E: Momentum



Panel F: Size

Panel G: Volatility

6.3 Multivariate Linear Models

Section 6.3.1 discusses the construction of the benchmark multivariate linear models used in this analysis. Section 6.3.2 considers the performance of these models.

6.3.1 Model Construction

From the univariate significant factors discussed in Section 6.2, a multivariate model is constructed. While this approach may limit model performance by excluding variables that are only significant when combined with other variables, it still provides a reasonable base to compare the performance of linear and ANN models at forecasting returns. Hodnett (2010) notes that a univariate factor model is successful at filtering out less important variables for constructing multifactor expected return models, such as those of Van Rensburg and Robertson (2003).

To avoid potential issues with multicollinearity, Table 6.2 shows the Pearson correlations between factors over the full training period, with correlations above 0.7 in bold. From this, mom12 is excluded from the analysis as it has a correlation above 0.7 with both mom6 and mom12minus1 and ranked below mom12minus1 in the univariate analysis based on its lower t-statistic.

Table 6.2: Pearson Correlation Matrix of Univariate Significant Factors

	c12mcftp	c24mepsp	epsg12	epsg24	salesg12	roag12	mom1	mom6	mom12	mom12minus1
ey	0.04	0.42	0.01	0.28	-0.02	0.06	-0.07	-0.13	-0.13	-0.11
c12mcftp		0.17	0.10	0.15	0.11	0.13	0.05	0.12	0.13	0.13
c24mepsp			0.15	0.64	0.05	0.21	0.03	0.10	0.18	0.18
epsg12				0.14	0.21	0.46	0.05	0.16	0.21	0.22
epsg24					0.07	0.16	0.06	0.15	0.24	0.24
salesg12						0.13	0.05	0.14	0.21	0.21
roag12							0.04	0.12	0.17	0.17
mom1								0.41	0.31	0.10
mom6									**0.72**	0.66
mom12										**0.97**

Table 6.3 summarizes the output of the multivariate regression of all univariate significant variables (excluding mom12 – referred to as “All Variables”). Significant factors at the 5% level are shown in bold. Figure 6.2 shows the cumulative marginal slope coefficient for the factors in the multivariate model. The same calculation is followed as in the univariate analysis, but it is referred to as ‘marginal’ as the slope coefficient represents the impact of a change in the specific factor while holding the other factors constant.

As seen in Table 6.3, only four of the ten univariate significant variables are significant when controlling for other factors in the model. The multivariate model suggests significant value (ey), growth (c12mcftp), momentum (mom12minus1) and short-term Reversal (mom1) effects on the JSE. This is supported by Figure 6.2, with consistent increases in the cumulative marginal slope coefficient for ey, mom12minus1, and c12mcftp, and a consistent decline for mom1.

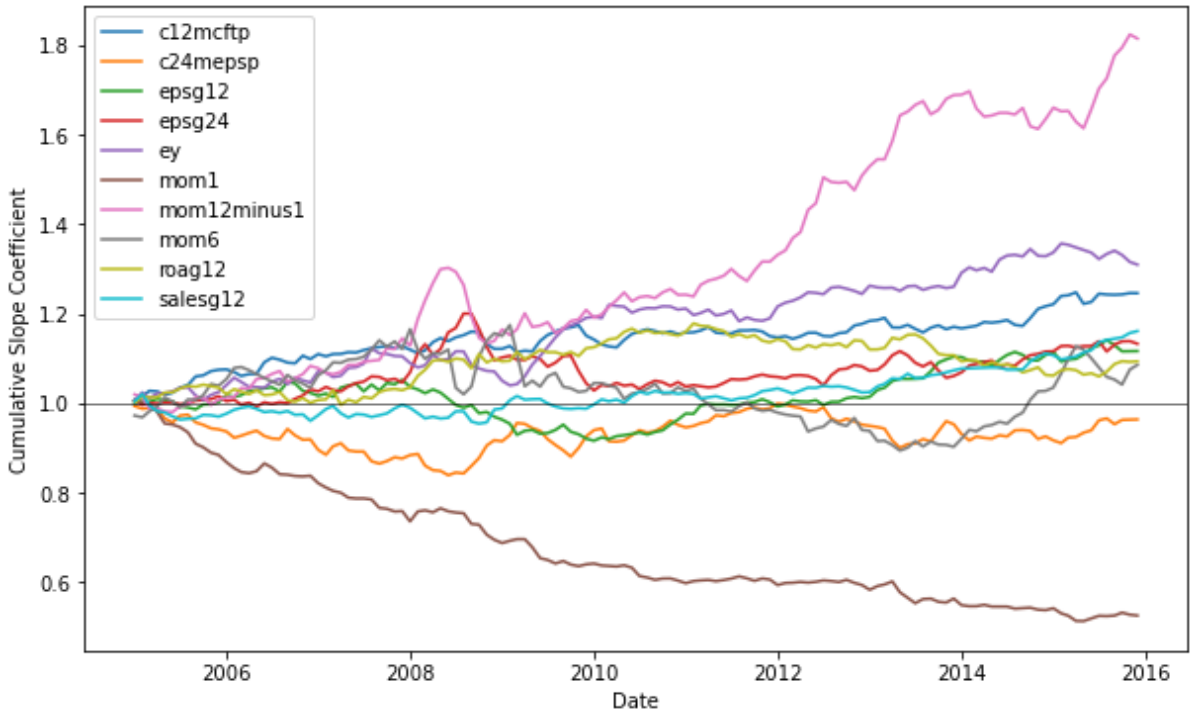
Table 6.3: Multivariate Regression Results – All Variables

This table shows the time-series mean slope coefficients over the training period of the multivariate regressions of all univariate significant variables (excluding mom12), discussed in Section 5.3, along with the associated standard errors and t-statistics. Significant factors at the 5% significance level are in bold. The final column shows the rank of the absolute value of the t-statistics.

	Factor	Average Coefficient	Standard Error	t-statistic	Rank
Value	**ey**	**0.002**	**0.001**	**2.326**	**4**
Growth	**c12mcftp**	**0.002**	**0.001**	**2.453**	**3**
	c24mepsp	-0.000	0.001	-0.179	10
	epsg12	0.001	0.001	1.115	6
	epsg24	0.001	0.001	1.023	7
	salesg12	0.001	0.001	1.672	5
Operating Performance	roag12	0.001	0.001	0.864	8
Momentum	**mom1**	**-0.005**	**0.001**	**-4.662**	**1**
	mom6	0.001	0.002	0.488	9
	mom12minus1	**0.005**	**0.001**	**3.393**	**2**

Figure 6.2: Cumulative Marginal Slope Coefficient – All Variables

A visual representation of the cumulative marginal slope coefficients over the training period from the multivariate regression of all univariate significant factors, discussed in Section 5.3.



The significant variables (referred to as “Reduced Variables”) from the Table 6.3 – ey, c12mcftp, mom1 and mom12minus1 – are used to create a second multivariate model to forecast returns. Feature selection in ANNs is important for improving model performance by reducing the noise-to-signal ratio prior to training (Rasekhschaffe and Jones, 2019), thus a smaller feature set may improve model performance. As a key focus of this study is to compare return forecasts between linear models and ANNs, the performance of the linear model using the smaller feature set is also considered. Table 6.4 summarizes the output of the multivariate regression using the Reduced Variables subset. Figure 6.3 shows the cumulative marginal slope coefficient for the factors in the Reduced Variables model.

Per Table 6.4, all variables are significant at the 5% level, supporting the view of a value, growth, momentum, and short-term reversal effect on the JSE over the training period. Figure 6.3 highlights positive cumulative marginal slope coefficients for ey, c12mcftp and mom12minus1, and a negative cumulative marginal slope coefficient for mom1, in line with the univariate analysis and All Variable linear model.

The inclusion of value and momentum effects is supported by Van Heerden and Van Rensburg (2015), with both found to be robust across different sample periods. As discussed in Section 6.2, the inclusion of a growth metric differs from the findings of Hodnett (2010) and Van Heerden and Van Rensburg (2015) but is supported by the univariate and multivariate analysis in this study. Lastly, the exclusion of a size factor supports the disappearance of its effect post-2003 (Van Heerden and Van Rensburg, 2015).

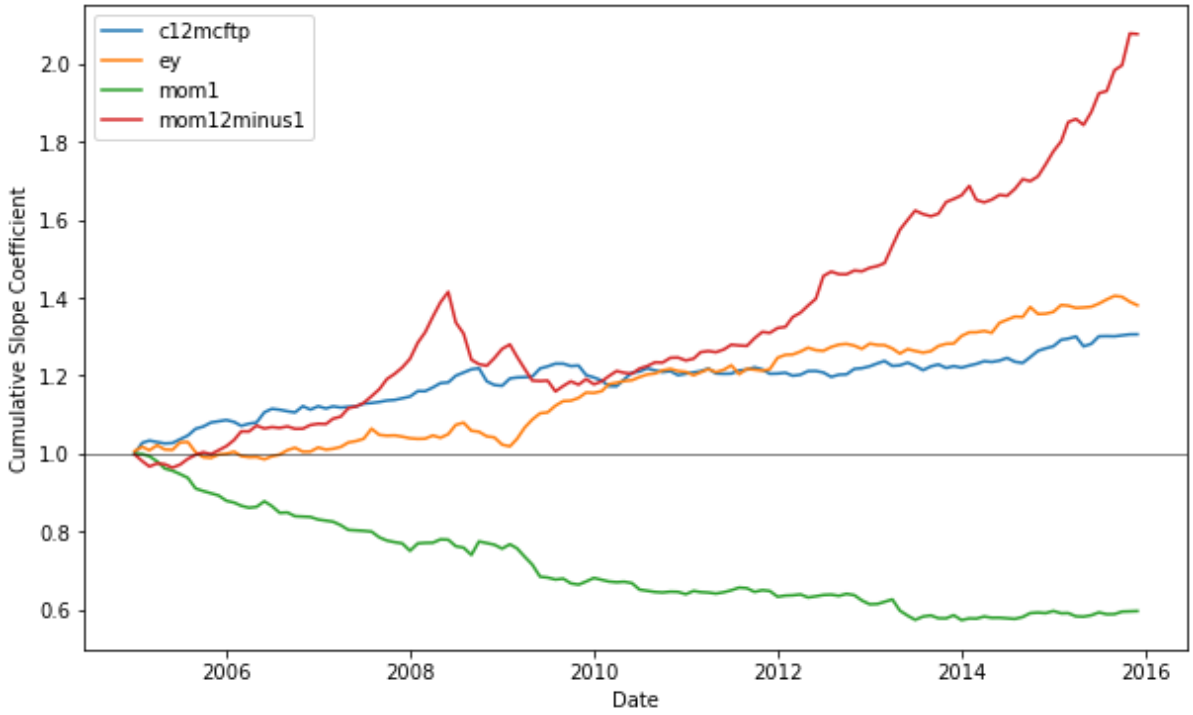
Table 6.4: Multivariate Regression Results – Reduced Variables

This table shows the time-series mean slope coefficients over the training period of the multivariate regressions of the Reduced Variables subset, discussed in Section 5.3, along with the associated standard errors and t-statistics. Significant factors at the 5% significance level are in bold. The final column shows the rank of the absolute value of the t-statistics.

	Factor	Average Coefficient	Standard Error	t-statistic	Rank
Value	**ey**	**0.002**	**0.001**	**3.122**	**3**
Growth	**c12mcftp**	**0.002**	**0.001**	**3.007**	**4**
Momentum	**mom1**	**-0.004**	**0.001**	**-3.545**	**2**
	mom12minus1	**0.006**	**0.001**	**4.511**	**1**

Figure 6.3: Cumulative Marginal Slope Coefficient – Reduced Variables

A visual representation of the cumulative marginal slope coefficients over the training period from the multivariate regression of the Reduced Variables subset, discussed in Section 5.3.



6.3.2 Model Performance

The out-of-sample performance of the two linear models is considered in this section. Table 6.5 summarizes the performance evaluation metrics discussed in Section 5.5. Figures 6.4 and 6.5 show the cumulative information coefficient (IC) and model portfolio performance of both portfolios over the testing period, offering insights into consistency.

The regression metrics confirm a positive significant relationship between forecast and actual returns for both models. A slope coefficient close to 0.5 suggests regression to the mean, with extreme forecasts likely to have actual returns closer to average. The All Variables model outperforms across all metrics, and better identifies winners and losers in a portfolio context. This may suggest that important variables are omitted from the Reduced Variables model, and the All Variables model may, on average, better capture time varying factor importance.

Performance for both models was notably worse in the test period, likely due to overfit and changing factor payoffs. Per Table 6.5, benchmark performance over the periods varied, with negative returns and higher volatility during the test period potentially contributing to changing payoffs and weak out-of-sample performance. However, Figures 6.4 and 6.5 indicate that both models performed well from 2019, a period of weak benchmark returns.

Figures 6.4 and 6.5 highlight poor and volatile predictive power between 2016 and 2019, with models incorrectly identifying winners and losers over this period. This is likely due to a reversal of at least one of the value, growth, momentum, or short-term reversal factors. From 2019 to the end of the test period, model performance notably improved, particularly following the COVID-19 related market sell-off.

Both models perform worse than the linear models of Hodnett (2010), likely due to keeping variables and expected factor payoffs constant. Hodnett (2010) saw a decline in out-of-sample performance when input features were kept constant. While underperformance is undesirable, model weights and features are also held constant in the ANN models. Despite poor initial performance, the linear models offer an adequate baseline to compare to ANN models. Both models outperform the benchmark on the upside and downside, successfully differentiating winners from losers over the test period.

Table 6.5: Linear Model Performance

This table shows the performance of both linear models based on the metrics discussed in Section 5.5 across the training and test periods, along with an equal weighted benchmark portfolio.

		All Variables		Reduced Variables		Benchmark	
		Train Period	Test Period	Train Period	Test Period	Train Period	Test Period
Regression Metrics	Demeaned R2	4.399%	0.148%	4.338%	0.135%		
	Slope Coefficient	0.888	0.521	0.931	0.538		
	T-stat	**11.077**	**3.385**	**10.609**	**3.229**		
Information Coefficient Metrics	Average IC	0.108	0.044	0.102	0.041		
	Cumulative IC	207550	4.17	107276	3.67		
	Standard Deviation of IC	0.147	0.189	0.146	0.185		
	Qian and Hua IR	0.732	0.232	0.699	0.221		
Average Monthly Regression Metrics	Average Slope Coefficient	0.939	0.585	0.971	0.582		
	Slope T-stat	**7.606**	1.853	**7.309**	1.754		
Portfolio Metrics – Top Quartile	Average Monthly Returns	2.501%	0.926%	2.442%	0.711%	1.661%	-0.016%
	Cumulative Payoff	22.850	1.536	21.097	1.362	8.040	0.931
	Annualized Return	32.903%	9.638%	31.942%	6.852%	20.863%	-1.525%
	Volatility	15.652%	19.901%	15.866%	19.723%	12.920%	16.156%
	Returns/Risk	2.102	0.484	2.013	0.347	1.615	-0.094
Portfolio Metrics – Bottom Quartile	Average Monthly Returns	0.687%	-0.534%	0.753%	-0.277%	1.661%	-0.016%
	Cumulative Payoff	2.219	0.673	2.415	0.784	8.040	0.931
	Annualized Return	7.517%	-8.123%	8.347%	-5.086%	20.863%	-1.525%
	Volatility	13.972%	19.929%	14.046%	19.218%	12.920%	16.156%
	Returns/Risk	0.538	-0.408	0.594	-0.265	1.615	-0.094

Figure 6.4: Multivariate Regression – Cumulative Information Coefficient

A visual representation of the cumulative information coefficient of both linear models over the test period, discussed in Section 5.5.

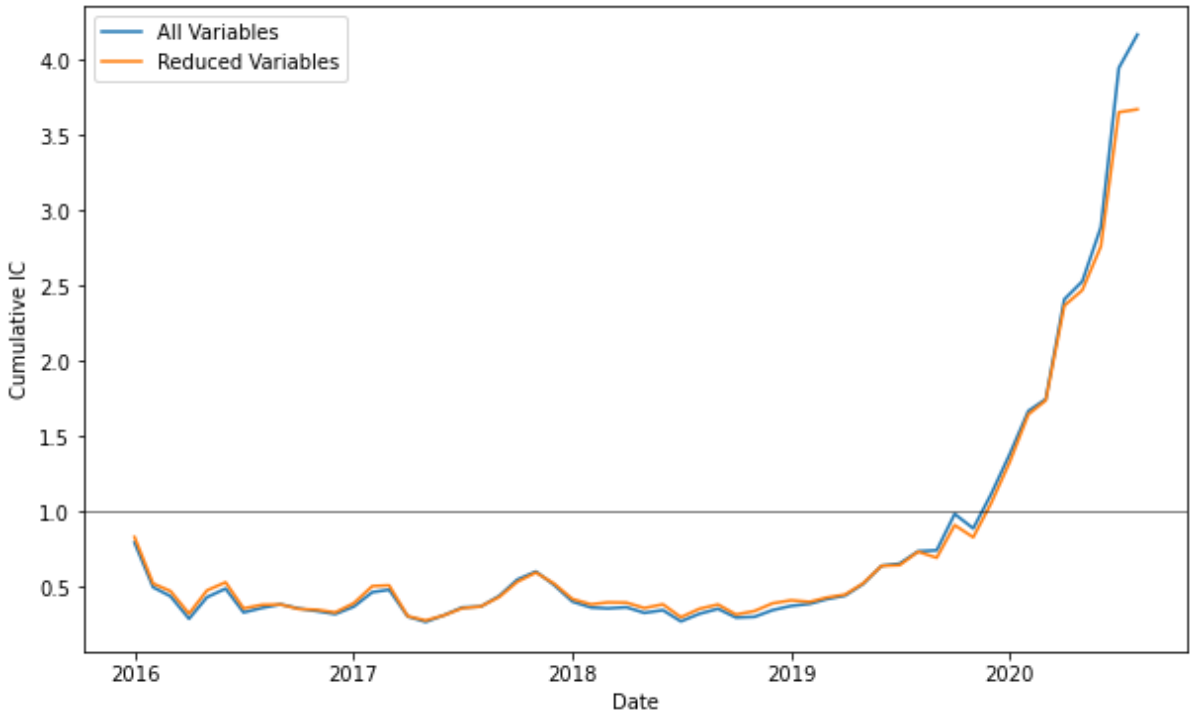
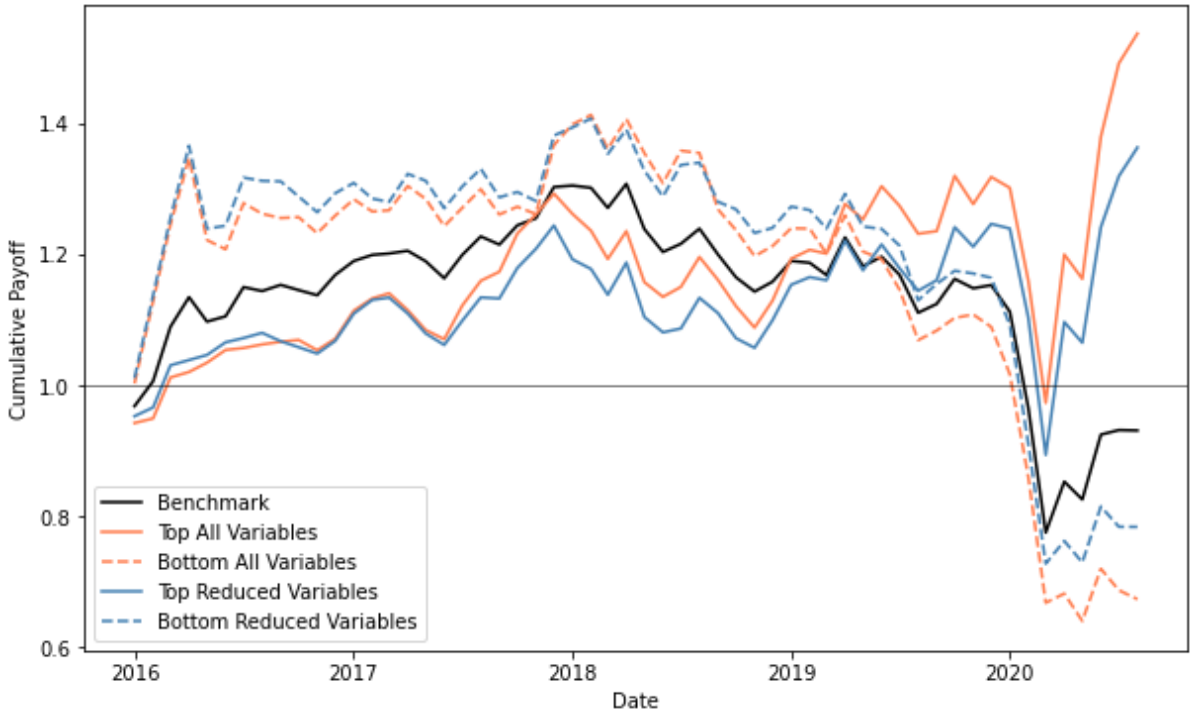


Figure 6.5: Multivariate Regression – Cumulative Portfolio Returns

A visual representation of the cumulative portfolio performance of both linear models over the test period for the top and bottom quartile of ranked returns, discussed in Section 5.5. The benchmark represents an equal weighted portfolio rebalanced monthly of all stocks in the dataset each month.



6.4 Artificial Neural Networks

This section considers the performance of the ANN models. Like the linear models, each model is trained once on the training data to forecast returns in the test period. Separate models are trained for all univariate significant variables ("All Variables") and the reduced subset of significant variables from the multivariate analysis ("Reduced Variables"). Network structure and regularization methods employed are discussed in Section 5.4. Section 6.4.1 to Section 6.4.5 consider the single hidden layer models up to the five hidden layer models.

6.4.1 Single Hidden Layer

The out-of-sample performance of the single hidden layer ANN models is considered in this section. Table 6.6 summarizes the performance evaluation metrics discussed in Section 5.5. Figures 6.6 and 6.7 show the cumulative IC and model portfolio performance over the testing period. In the table and figures, performance is separately shown for the All Variables and Reduced Variables models in Panel A and B, respectively.

It is evident that performance is poor across all model structures, with negative test sample slope coefficients and average IC metrics, as well as poor portfolio performance relative to the benchmark. Considering Table 6.6 Panel A and Panel B, weak in-sample performance relative to the linear models may indicate underfitting, with poor out-of-sample performance supporting limited learning. In contrast, weak out-of-sample relative to in-sample performance may suggest that the models are capturing noise when training, leading to poor generalization.

In both Panels, in-sample performance tends to increase with more nodes, however there is no direct relationship with performance during the training and test periods, likely due to noise in the data. In contrast to Tsai et al. (2011) and Qiu, Song and Akagi (2016), the more complex 256 node model is found to perform best in Panel A, supporting the view that many nodes may be needed in shallow models (Aggarwal, 2018). In Panel B, the less complex 32 node model performs best, potentially due to the reduced variable set. Performance appears to deteriorate between the 128 and 256 node models, likely due to the high number of parameters needed to be estimated with limited data (Aggarwal, 2018).

Figures 6.6 and 6.7 emphasize the weak and volatile performance of the models across the test period. The top performing 256 node model in Panel A appeared to perform well from 2018 to 2020, but was weak overall. In Panel B, the top performing 32 node model was similarly weak and highly volatile. Both models appear better at identifying outperformers relative to underperformers. Overall, test period performance was weak across all single layer models relative to the linear benchmarks. The 32 node Reduced Variables model performed best in the test period.

Table 6.6: MLP Model Performance – One Layer

This table shows the performance of the All Variables ANN model in Panel A and Reduced Variables ANN Model in Panel B based on the metrics discussed in Section 5.5 across the training and test periods, along with an equal weighted benchmark portfolio.

Panel A: All Variables

Number of Nodes:		8		16		32		64		128		256	
		Train Period	Test Period	Train Period	Test Period	Train Period	Test Period	Train Period	Test Period	Train Period	Test Period	Train Period	Test Period
Regression Metrics	Demeaned R2	3.782%	0.025%	3.753%	0.013%	3.811%	0.030%	3.849%	0.000%	4.257%	0.002%	4.842%	0.016%
	Slope Coefficient	0.355	-0.179	0.192	-0.091	0.170	-0.089	0.167	-0.002	0.276	-0.018	0.364	-0.049
	T-stat	4.427	-1.372	3.837	-0.981	4.940	-1.504	5.537	-0.036	9.946	-0.370	14.057	-1.104
Information Coefficient Metrics	Average IC	0.024	-0.008	0.024	-0.014	0.027	-0.009	0.029	-0.001	0.052	-0.007	0.089	0.006
	Cumulative IC	12.399	0.549	13.530	0.369	18.668	0.478	27.899	0.767	513.521	0.565	54476	1.152
	Standard Deviation of IC	0.093	0.074	0.096	0.080	0.099	0.092	0.090	0.090	0.082	0.081	0.083	0.083
	Qian and Hua IR	0.252	-0.106	0.255	-0.181	0.274	-0.096	0.328	-0.008	0.630	-0.085	1.079	0.072
Average Monthly Regression Metrics	Average Slope Coefficient	0.290	-0.055	0.157	-0.139	0.105	-0.031	0.107	0.022	0.174	-0.022	0.278	-0.004
	Slope T-stat	3.224	-0.493	3.104	-1.446	3.242	-0.565	3.968	0.440	7.122	-0.497	10.705	-0.090
Portfolio Metrics – Top Quartile	Average Monthly Returns	1.851%	0.055%	1.869%	-0.287%	1.977%	-0.091%	1.818%	0.043%	2.168%	-0.185%	2.450%	0.208%
	Cumulative Returns	10.363	0.953	10.643	0.774	12.183	0.869	10.077	0.944	15.951	0.834	22.757	1.022
	Annualized Return	23.685%	-1.016%	23.985%	-5.333%	25.518%	-2.974%	23.371%	-1.237%	28.631%	-3.825%	32.854%	0.474%
	Volatility	13.433%	17.966%	13.148%	19.234%	13.371%	19.333%	12.383%	18.305%	12.769%	18.010%	12.738%	19.699%
	Returns/Risk	1.763	-0.057	1.824	-0.277	1.908	-0.154	1.887	-0.068	2.242	-0.212	2.579	0.024
Portfolio Metrics – Bottom Quartile	Average Monthly Returns	1.398%	0.154%	1.327%	0.128%	1.330%	0.161%	1.288%	-0.159%	1.069%	0.183%	0.830%	0.004%
	Cumulative Returns	5.793	1.025	5.284	1.016	5.299	1.020	5.001	0.855	3.830	1.050	2.793	0.944
	Annualized Return	17.316%	0.529%	16.338%	0.336%	16.368%	0.422%	15.757%	-3.298%	12.985%	1.041%	9.786%	-1.232%
	Volatility	13.534%	16.173%	14.677%	15.523%	13.626%	16.994%	14.660%	16.564%	14.170%	15.094%	14.125%	16.215%
	Returns/Risk	1.279	0.033	1.113	0.022	1.201	0.025	1.075	-0.199	0.916	0.069	0.693	-0.076
Benchmark	Average Monthly Returns	1.661%	-0.016%										
	Cumulative Returns	8.040	0.931										
	Annualized Return	20.863%	-1.525%										
	Volatility	12.920%	16.156%										
	Returns/Risk	1.615	-0.094										

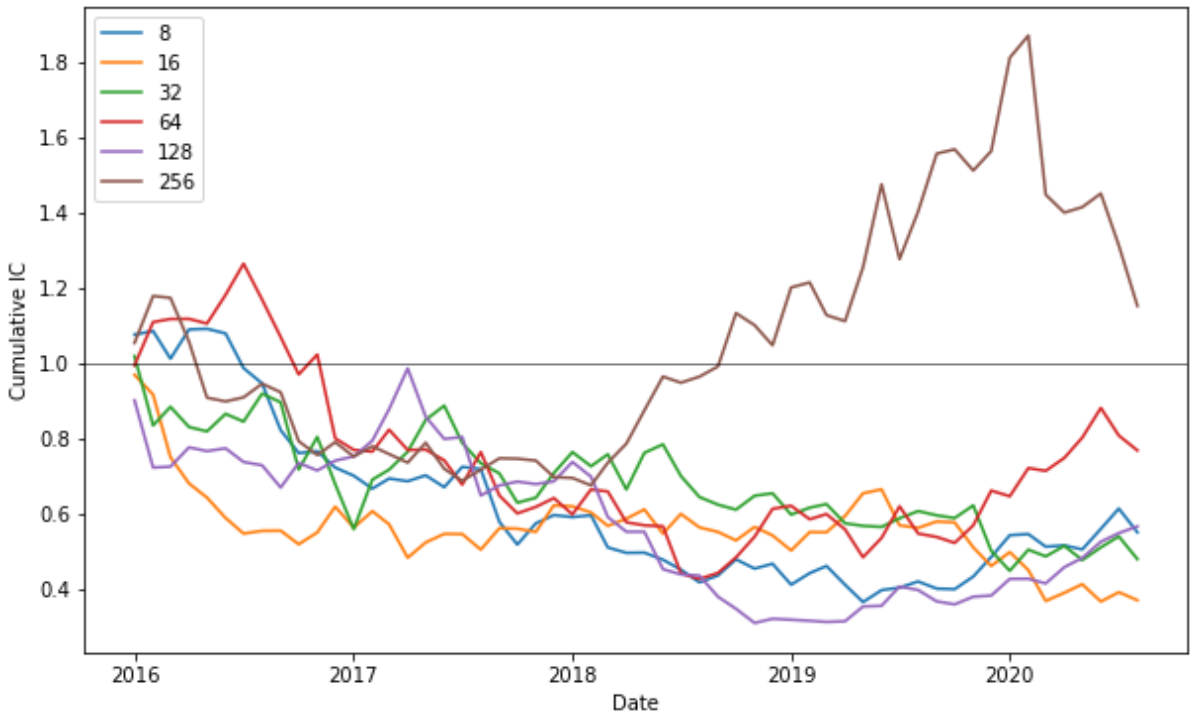
Panel B: Reduced Variables

Number of Nodes:		8		16		32		64		128		256	
		Train Period	Test Period	Train Period	Test Period	Train Period	Test Period	Train Period	Test Period	Train Period	Test Period	Train Period	Test Period
Regression Metrics	Demeaned R2	3.704%	0.006%	3.740%	0.019%	3.732%	0.014%	3.944%	0.069%	4.030%	0.001%	3.990%	0.015%
	Slope Coefficient	0.287	-0.144	0.237	-0.157	0.150	0.092	0.308	-0.200	0.338	0.016	0.315	-0.090
	T-stat	2.560	-0.655	3.558	-1.194	3.357	1.041	6.820	-2.310	7.801	0.195	7.358	-1.073
Information Coefficient Metrics	Average IC	0.019	-0.006	0.022	-0.022	0.018	0.010	0.045	-0.015	0.054	0.007	0.054	-0.009
	Cumulative IC	5.648	0.596	8.955	0.235	5.767	1.443	201.191	0.360	644.552	1.187	634.331	0.497
	Standard Deviation of IC	0.105	0.077	0.099	0.080	0.093	0.087	0.091	0.082	0.087	0.088	0.090	0.079
	Qian and Hua IR	0.179	-0.082	0.219	-0.276	0.190	0.119	0.493	-0.179	0.617	0.078	0.599	-0.120
Average Monthly Regression Metrics	Average Slope Coefficient	0.296	-0.109	0.180	-0.252	0.100	0.095	0.248	-0.134	0.261	0.022	0.265	-0.092
	Slope T-stat	2.390	-0.633	2.620	-1.758	2.444	1.124	5.766	-1.401	6.928	0.286	6.794	-1.148
Portfolio Metrics – Top Quartile	Average Monthly Returns	1.840%	0.027%	1.874%	-0.381%	1.815%	0.282%	2.053%	-0.484%	2.068%	0.037%	2.037%	-0.389%
	Cumulative Returns	10.083	0.932	10.551	0.729	9.827	1.105	13.141	0.698	13.506	0.944	13.086	0.761
	Annualized Return	23.378%	-1.497%	23.887%	-6.536%	23.089%	2.157%	26.385%	-7.416%	26.699%	-1.226%	26.337%	-5.683%
	Volatility	13.396%	18.694%	13.346%	20.034%	12.974%	15.725%	14.150%	18.676%	13.641%	18.049%	13.055%	15.182%
	Returns/Risk	1.745	-0.080	1.790	-0.326	1.780	0.137	1.865	-0.397	1.957	-0.068	2.017	-0.374
Portfolio Metrics – Bottom Quartile	Average Monthly Returns	1.438%	0.099%	1.238%	0.167%	1.450%	-0.068%	1.203%	0.365%	1.256%	-0.091%	1.144%	0.434%
	Cumulative Returns	5.945	0.982	4.574	1.044	6.035	0.895	4.434	1.150	4.682	0.881	4.044	1.146
	Annualized Return	17.592%	-0.396%	14.823%	0.930%	17.753%	-2.354%	14.497%	3.045%	15.067%	-2.690%	13.545%	2.963%
	Volatility	13.715%	17.272%	13.777%	14.593%	13.819%	17.405%	12.788%	16.575%	13.773%	17.626%	13.812%	20.941%
	Returns/Risk	1.283	-0.023	1.076	0.064	1.285	-0.135	1.134	0.184	1.094	-0.153	0.981	0.141
Benchmark	Average Monthly Returns	1.661%	-0.016%										
	Cumulative Returns	8.040	0.931										
	Annualized Return	20.863%	-1.525%										
	Volatility	12.920%	16.156%										
	Returns/Risk	1.615	-0.094										

Figure 6.6: MLP – One Layer – Cumulative Information Coefficient

A visual representation of the cumulative information coefficient of the ANN models over the test period, as discussed in Section 5.5. Panel A considers the All Variables models, while Panel B considers the Reduced Variables models. The numbers in the legend represent the number of nodes in the hidden layer.

Panel A: All Variables



Panel B: Reduced Variables

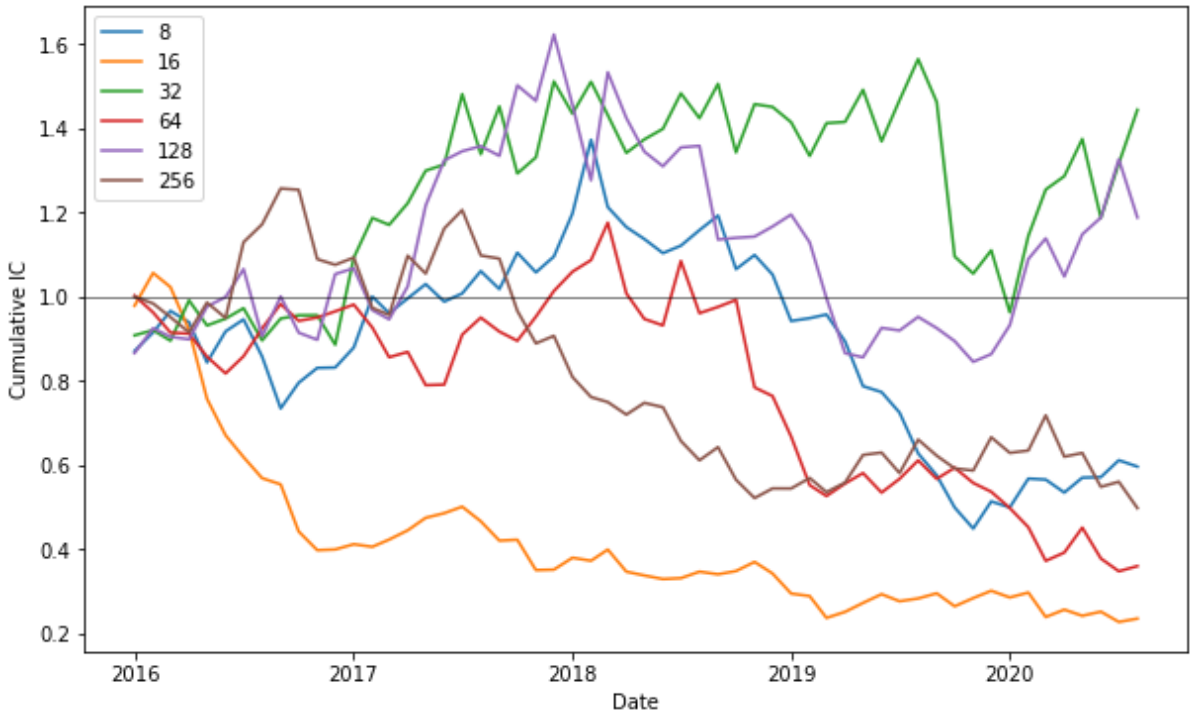
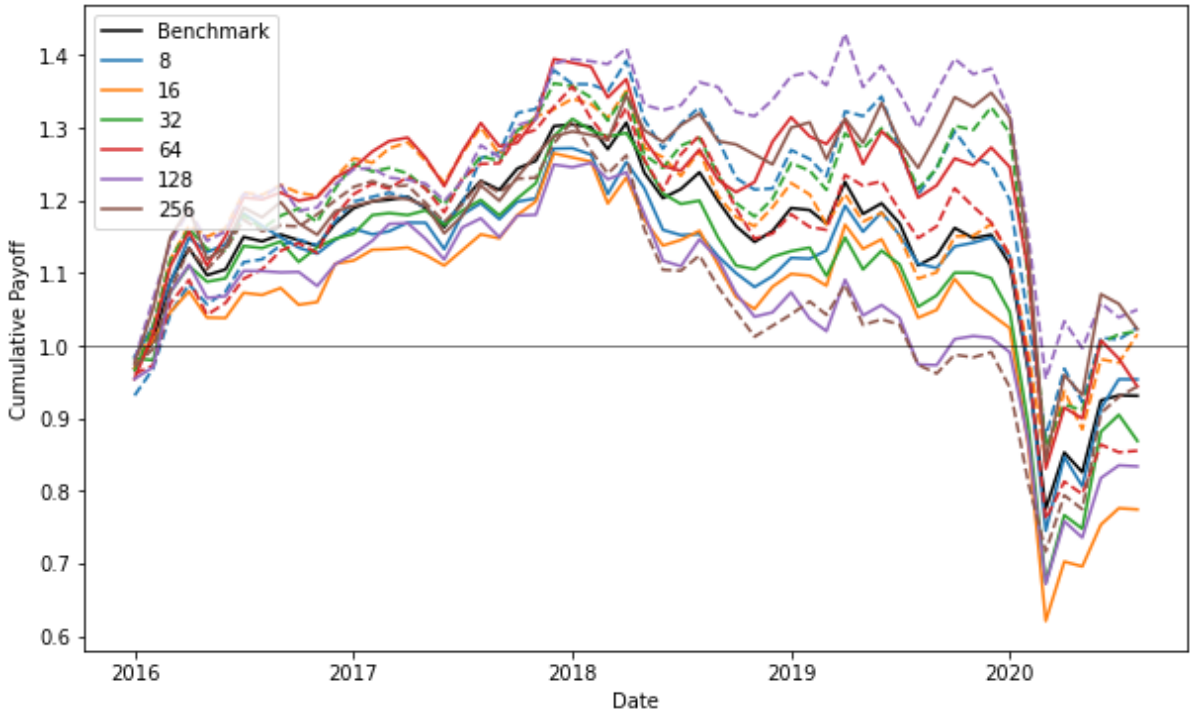


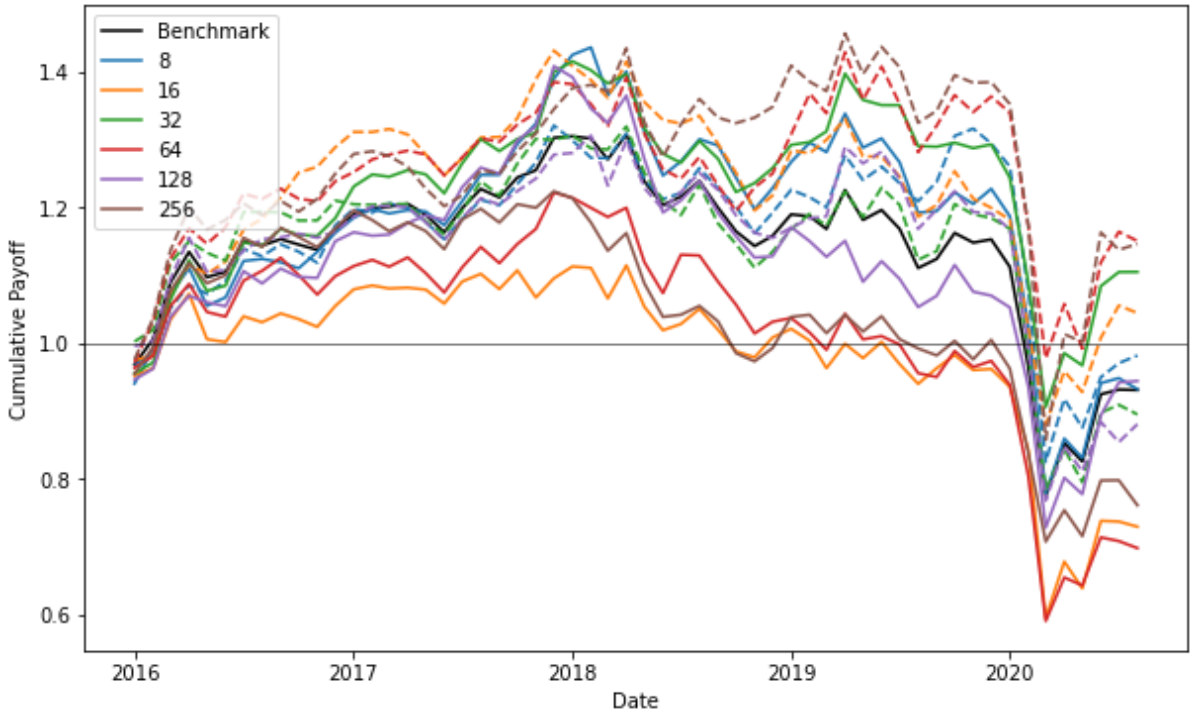
Figure 6.7: MLP – One Layer – Cumulative Portfolio Returns

A visual representation of the cumulative portfolio performance of the ANN models over the test period for the top and bottom quartile of ranked returns, as discussed in Section 5.5. The benchmark represents an equal weighted monthly rebalanced portfolio of all stocks in the dataset each month. Panel A considers the All Variables models, while Panel B considers the Reduced Variables models. The numbers in the legend represent the number of nodes in the hidden layer. Solid lines represent the Top Quartile portfolio, dashed lines represent the Bottom Quartile portfolio.

Panel A: All Variables



Panel B: Reduced Variables



6.4.2 Two Hidden Layers

The out-of-sample performance of the two hidden layer ANN models is considered in this section. Table 6.7 summarizes the performance evaluation metrics discussed in Section 5.5. Figures 6.8 and 6.9 show the cumulative IC and model portfolio performance over the testing period. In the table and figures, performance is separately shown for the All Variables and Reduced Variables models in Panel A and B, respectively.

Like the single layer models, performance is weak and below the linear models. However, the best performing model outperforms the top one layer model, suggesting benefits to deeper structures. Considering Table 6.7, in-sample performance tends to improve as more nodes are added to the hidden layers. Weak in-sample performance in the 8 and 16 node models is likely due to the inclusion of Dropout, which tends to require a larger model structure (Aggarwal, 2018). In Panel A, weak in-sample performance for the 128 node model may be due to vanishing or exploding gradients, or convergence at local optima (Aggarwal, 2018), while poor relative test period performance for the 256 node model is likely due to overfitting. In Panel B, deteriorating performance for more complex structures is likely due to, as before, the need to estimate more parameters with a limited dataset (Aggarwal, 2018).

In both Panels, there is no relationship between training and test period performance. The 16 node models in both cases perform best out-of-sample despite poor in-sample performance, suggesting that this is likely by chance. In Panel A, no model appears to generalize well and performance lags the single layer alternatives. In Panel B, the 64 node model performs best, outperforming the top single layer model in terms of IC and Portfolio Metrics, with better ability to identify both winners and losers. Improving performance supports the use of deeper, more complex models with appropriate regularization methods to forecast returns on the JSE. Outperformance of the Reduced Variables model, as with the single layer models, supports the need for appropriate feature selection to reduce noise and improve generalization (Rasekhschaffe and Jones, 2019).

Figures 6.8 and 6.9 highlight weak and volatile performance across models. Excluding the 16 node model, all models in Panel A failed to forecast returns or differentiate out- and underperformers. In Panel B, the top performing 64 node model appears to perform well initially until mid-2017, followed by a period of weak and volatile performance. From a portfolio perspective, the model successfully identified both winners and losers over the full test period. While this model underperformed the linear benchmark overall, it appears to be more consistent over time, particularly pre-2019. Still, test period performance was generally weak across all two layer models relative to the linear benchmarks.

Table 6.7: MLP Model Performance – Two Layer

This table shows the performance of the All Variables ANN model in Panel A and Reduced Variables ANN Model in Panel B based on the metrics discussed in Section 5.5 across the training and test periods, along with an equal weighted benchmark portfolio.

Panel A: All

Number of Nodes:		8-4		16-8		32-16		64-32		128-64		256-128	
		Train Period	Test Period	Train Period	Test Period	Train Period	Test Period	Train Period	Test Period	Train Period	Test Period	Train Period	Test Period
Regression Metrics	Demeaned R2	3.673%	0.023%	3.672%	0.011%	3.696%	0.003%	3.828%	0.001%	3.665%	0.003%	5.126%	0.062%
	Slope Coefficient	0.483	-1.413	-0.156	0.237	0.282	-0.107	0.459	0.026	0.002	-0.004	1.373	-0.370
	T-stat	1.229	-1.329	-1.143	0.928	2.294	-0.456	5.217	0.154	0.407	-0.446	15.687	-2.176
Information Coefficient Metrics	Average IC	0.007	-0.013	-0.011	0.009	0.010	-0.004	0.033	-0.006	0.008	-0.008	0.088	-0.016
	Cumulative IC	1.669	0.365	0.129	1.391	2.017	0.674	41.151	0.573	1.667	0.524	45416	0.317
	Standard Deviation of IC	0.079	0.092	0.093	0.085	0.097	0.078	0.097	0.085	0.092	0.082	0.090	0.092
	Qian and Hua IR	0.089	-0.145	-0.119	0.111	0.103	-0.051	0.342	-0.075	0.088	-0.099	0.978	-0.174
Average Monthly Regression Metrics	Average Slope Coefficient	0.718	-1.483	-0.231	0.246	0.117	-0.033	0.341	-0.029	0.006	-0.004	0.915	-0.177
	Slope T-stat	0.993	-0.898	-1.522	0.958	0.985	-0.185	4.060	-0.193	1.383	-0.548	10.156	-1.080
Portfolio Metrics – Top Quartile	Average Monthly Returns	1.624%	-0.018%	1.603%	0.088%	1.700%	0.197%	1.793%	-0.037%	1.836%	-0.064%	2.441%	-0.183%
	Cumulative Returns	7.599	0.914	7.385	0.988	8.308	1.057	9.492	0.907	10.008	0.903	21.943	0.831
	Annualized Return	20.246%	-1.916%	19.933%	-0.255%	21.224%	1.194%	22.702%	-2.062%	23.294%	-2.167%	32.415%	-3.888%
	Volatility	13.498%	18.076%	13.682%	16.136%	14.206%	15.279%	13.391%	17.602%	13.524%	16.949%	13.444%	18.487%
	Returns/Risk	1.500	-0.106	1.457	-0.016	1.494	0.078	1.695	-0.117	1.722	-0.128	2.411	-0.210
Portfolio Metrics – Bottom Quartile	Average Monthly Returns	1.625%	0.186%	1.626%	-0.053%	1.544%	0.087%	1.349%	0.179%	1.426%	0.142%	0.743%	0.175%
	Cumulative Returns	7.544	1.032	7.577	0.905	6.807	0.976	5.268	1.019	5.839	1.018	2.377	1.029
	Annualized Return	20.167%	0.676%	20.214%	-2.111%	19.048%	-0.514%	16.307%	0.413%	17.400%	0.389%	8.191%	0.621%
	Volatility	14.043%	17.393%	13.879%	17.074%	13.904%	17.426%	14.066%	18.129%	13.941%	15.936%	14.218%	17.083%
	Returns/Risk	1.436	0.039	1.456	-0.124	1.370	-0.029	1.159	0.023	1.248	0.024	0.576	0.036
Benchmark	Average Monthly Returns	1.661%	-0.016%										
	Cumulative Returns	8.040	0.931										
	Annualized Return	20.863%	-1.525%										
	Volatility	12.920%	16.156%										
	Returns/Risk	1.615	-0.094										

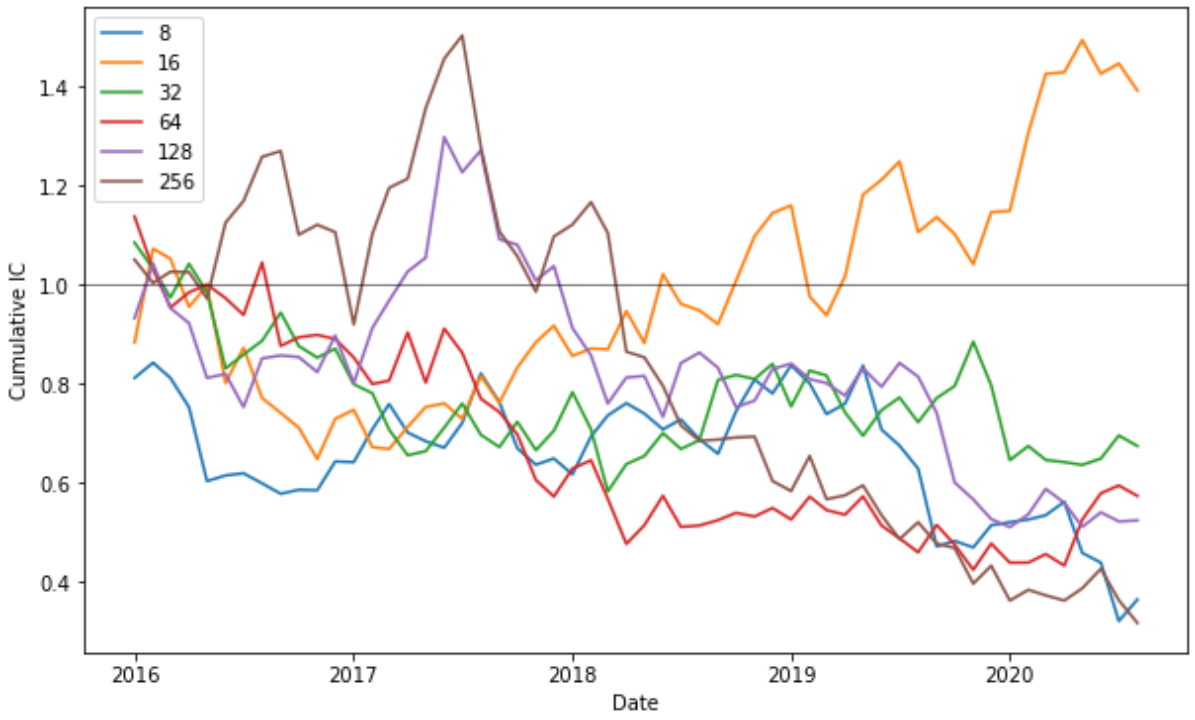
Panel B: Reduced

Number of Nodes:		8-4		16-8		32-16		64-32		128-64		256-128	
		Train Period	Test Period	Train Period	Test Period	Train Period	Test Period	Train Period	Test Period	Train Period	Test Period	Train Period	Test Period
Regression Metrics	Demeaned R2	3.667%	0.001%	3.665%	0.013%	3.697%	0.001%	3.710%	0.000%	3.665%	0.001%	3.670%	0.006%
	Slope Coefficient	0.012	0.005	-0.043	0.250	0.284	0.049	0.257	-0.005	0.002	0.002	0.005	-0.007
	T-stat	0.680	0.123	-0.334	0.975	2.353	0.227	2.760	-0.025	0.314	0.160	1.009	-0.661
Information Coefficient Metrics	Average IC	0.001	-0.013	-0.001	0.015	0.025	-0.002	0.020	0.014	0.006	-0.005	0.008	0.001
	Cumulative IC	0.642	0.360	0.494	1.938	15.105	0.756	7.828	1.766	1.292	0.634	2	0.893
	Standard Deviation of IC	0.097	0.096	0.098	0.082	0.094	0.076	0.093	0.082	0.091	0.079	0.097	0.084
	Qian and Hua IR	0.014	-0.138	-0.006	0.184	0.268	-0.028	0.215	0.164	0.067	-0.064	0.083	0.016
Average Monthly Regression Metrics	Average Slope Coefficient	0.001	-0.048	-0.033	0.298	0.383	0.002	0.214	0.149	0.004	-0.003	0.004	0.002
	Slope T-stat	0.065	-1.249	-0.236	1.480	3.306	0.009	2.446	1.066	0.889	-0.345	0.776	0.181
Portfolio Metrics – Top Quartile	Average Monthly Returns	1.592%	0.223%	1.584%	0.068%	1.853%	-0.120%	1.951%	0.340%	1.660%	-0.075%	1.752%	0.017%
	Cumulative Returns	7.225	1.082	7.224	0.971	10.148	0.874	11.621	1.131	8.013	0.898	8.973	0.942
	Annualized Return	19.695%	1.695%	19.693%	-0.620%	23.449%	-2.848%	24.980%	2.681%	20.827%	-2.290%	22.076%	-1.273%
	Volatility	14.099%	14.172%	13.470%	16.775%	14.133%	16.722%	13.520%	16.668%	13.046%	16.595%	13.593%	16.964%
	Returns/Risk	1.397	0.120	1.462	-0.037	1.659	-0.170	1.848	0.161	1.596	-0.138	1.624	-0.075
Portfolio Metrics – Bottom Quartile	Average Monthly Returns	1.627%	0.121%	1.702%	0.048%	1.348%	-0.081%	1.456%	-0.313%	1.571%	0.112%	1.812%	-0.010%
	Cumulative Returns	7.627	0.978	8.433	0.959	5.315	0.888	6.075	0.781	7.077	0.996	9.838	0.920
	Annualized Return	20.286%	-0.479%	21.389%	-0.890%	16.400%	-2.512%	17.823%	-5.161%	19.470%	-0.076%	23.101%	-1.778%
	Volatility	13.555%	19.080%	13.437%	16.973%	13.301%	17.317%	13.889%	17.090%	13.697%	16.676%	12.636%	18.265%
	Returns/Risk	1.497	-0.025	1.592	-0.052	1.233	-0.145	1.283	-0.302	1.421	-0.005	1.828	-0.097
Benchmark	Average Monthly Returns	1.661%	-0.016%										
	Cumulative Returns	8.040	0.931										
	Annualized Return	20.863%	-1.525%										
	Volatility	12.920%	16.156%										
	Returns/Risk	1.615	-0.094										

Figure 6.8: MLP – Two Layer – Cumulative Information Coefficient

A visual representation of the cumulative information coefficient of the ANN models over the test period, as discussed in Section 5.5. Panel A considers the All Variables models, while Panel B considers the Reduced Variables models. The numbers in the legend represent the number of nodes in the first hidden layer.

Panel A: All Variables



Panel B: Reduced Variables

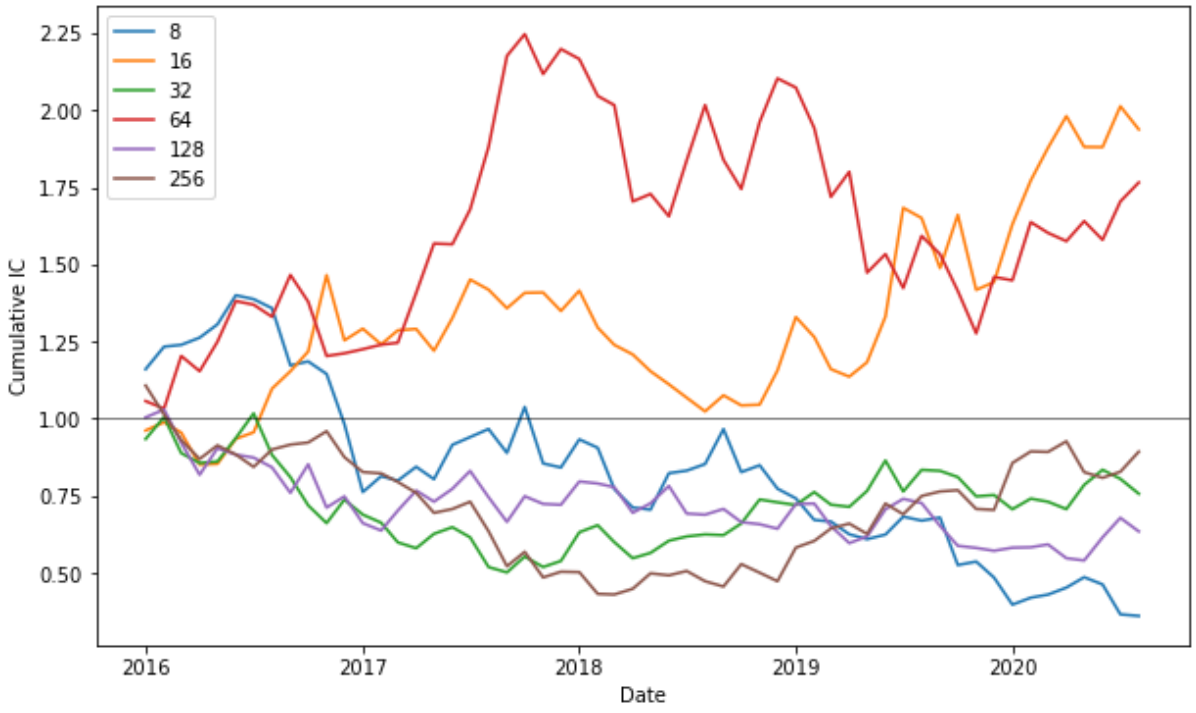
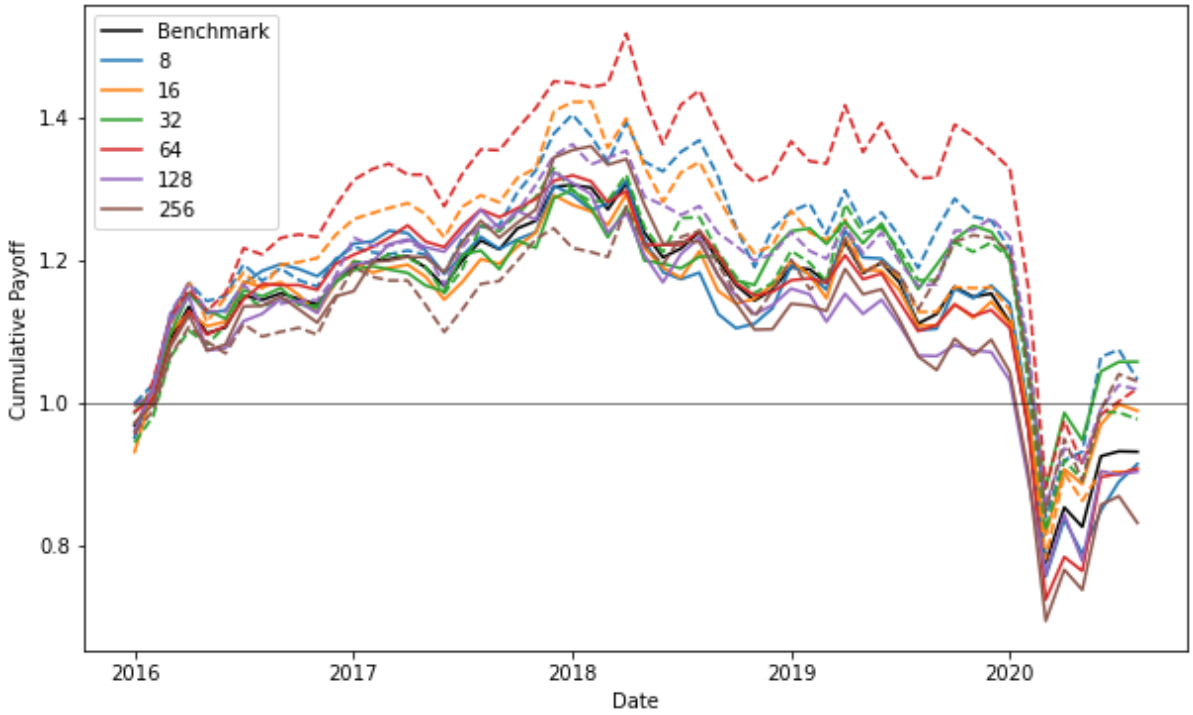


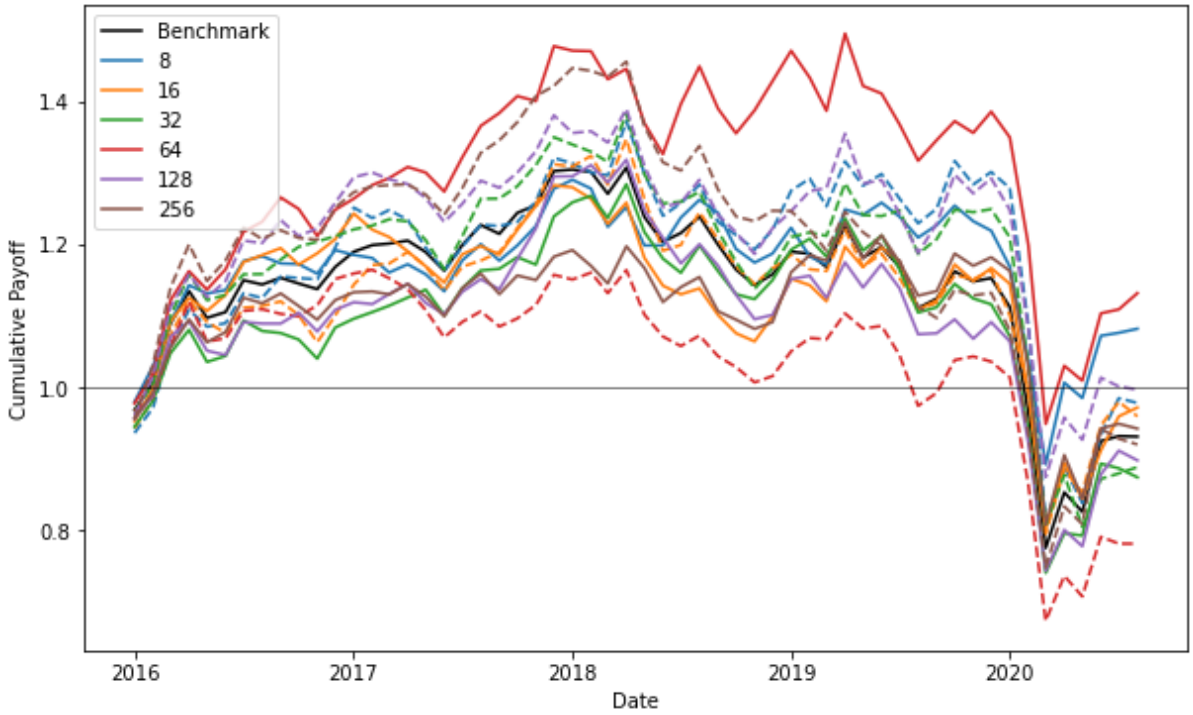
Figure 6.9: MLP – Two Layer – Cumulative Portfolio Returns

A visual representation of the cumulative portfolio performance of the ANN models over the test period for the top and bottom quartile of ranked returns, as discussed in Section 5.5. The benchmark represents an equal weighted monthly rebalanced portfolio of all stocks in the dataset each month. Panel A considers the All Variables models, while Panel B considers the Reduced Variables models. The numbers in the legend represent the number of nodes in the first hidden layer. Solid lines represent the Top Quartile portfolio, dashed lines represent the Bottom Quartile portfolio.

Panel A: All Variables



Panel B: Reduced Variables



6.4.3 Three Hidden Layers

The out-of-sample performance of the three hidden layer ANN models is considered in this section. Table 6.8 summarizes the performance evaluation metrics discussed in Section 5.5. Figures 6.10 and 6.11 show the cumulative IC and model portfolio performance over the testing period. In the table and figures, performance is separately shown for the All Variables and Reduced Variables models in Panel A and B, respectively.

While overall performance tends to lag the linear benchmarks, the top performing model outperforms the top two layer model, further supporting deeper structures. Similar patterns to previous models are identified in Table 6.8. First, in-sample performance tends to improve with the addition of nodes. Second, less complex structures – 8, 16 and 32 node models in this case – suffer from weak in sample performance, likely due to the inclusion of Dropout. Third, there is evidence of overfitting for complex All Variables models. Fourth, the in-sample performance of the Reduced Variables model deteriorates with complex structures. Fifth, the best Reduced Variables model outperforms the best All Variables model. Lastly, there appears to be no relationship between training and test period performance. As before, certain models perform better in the test period than training period, likely due to chance.

Considering Panel A, there appears to be an improvement in out-of-sample performance relative to the one and two layer models. Of the models with decent in sample performance, the 64 node model performed best, with evidence of overfitting with the addition of further nodes. In Panel B, the 64 node model performs best, outperforming the top two layer model across all metrics. Notably, the t-statistic is significant for this model out-of-sample, with a slope coefficient close to 0.5 like the linear models. Improving model performance again supports the use of deeper, more complex models on the JSE.

Figures 6.10 and 6.11 highlight largely weak and volatile performance. In Panel A, excluding the 16 and 32 node models, the top performing 64 node model appears to perform poorly for most of the test period, with improved performance from 2019. Like the linear models, initial portfolio performance was poor, but improvement in both the Top and Bottom Quartile over time resulted in overall adequate performance relative to the benchmark. In Panel B, excluding the 256 node model due to poor in sample performance, the top performing 64 node model appears to have relatively consistent performance across the test period. While it experienced periods of strong and weak performance, the model appears to appropriately distinguish winners from losers over the period, with particular strength at forecasting returns for underperformers. Still, three layer models appear to underperform the linear models over the full test period, but show notable improvement over the one and two layer models.

Table 6.8: MLP Model Performance – Three Layer

This table shows the performance of the All Variables ANN model in Panel A and Reduced Variables ANN Model in Panel B based on the metrics discussed in Section 5.5 across the training and test periods, along with an equal weighted benchmark portfolio.

Panel A: All

Number of Nodes:		8-4-2		16-8-4		32-16-8		64-32-16		128-64-32		256-128-64	
		Train Period	Test Period	Train Period	Test Period	Train Period	Test Period	Train Period	Test Period	Train Period	Test Period	Train Period	Test Period
Regression Metrics	Demeaned R2	3.673%	0.026%	3.666%	0.064%	3.664%	0.008%	3.769%	0.007%	3.986%	0.002%	5.235%	0.025%
	Slope Coefficient	0.012	-0.029	0.506	4.319	-0.019	0.223	0.517	0.166	0.643	0.060	1.198	-0.196
	T-stat	1.190	-1.421	0.491	2.226	-0.129	0.758	4.163	0.720	7.317	0.342	16.267	-1.380
Information Coefficient Metrics	Average IC	0.012	-0.013	0.000	0.027	0.000	0.015	0.028	0.009	0.050	0.007	0.103	-0.007
	Cumulative IC	2.989	0.359	0.597	3.641	0.604	1.920	27.092	1.440	391.360	1.226	254400	0.574
	Standard Deviation of IC	0.091	0.098	0.091	0.091	0.090	0.076	0.076	0.073	0.089	0.088	0.097	0.077
	Qian and Hua IR	0.137	-0.134	0.005	0.302	0.002	0.192	0.370	0.126	0.563	0.084	1.062	-0.090
Average Monthly Regression Metrics	Average Slope Coefficient	0.020	-0.030	-0.243	3.671	-0.018	0.356	0.391	0.238	0.496	0.198	0.891	-0.084
	Slope T-stat	2.007	-1.391	-0.242	2.139	-0.130	1.439	3.985	1.349	5.929	1.099	10.860	-0.659
Portfolio Metrics – Top Quartile	Average Monthly Returns	1.906%	-0.114%	1.705%	0.280%	1.634%	-0.145%	1.822%	0.325%	2.169%	0.120%	2.518%	-0.141%
	Cumulative Returns	11.057	0.868	8.373	1.081	7.762	0.857	9.824	1.118	15.362	0.992	24.255	0.846
	Annualized Return	24.416%	-2.981%	21.310%	1.687%	20.477%	-3.249%	23.085%	2.425%	28.192%	-0.171%	33.626%	-3.529%
	Volatility	12.919%	17.954%	14.085%	18.074%	13.032%	17.120%	13.543%	17.203%	13.766%	18.140%	13.391%	18.761%
	Returns/Risk	1.890	-0.166	1.513	0.093	1.571	-0.190	1.705	0.141	2.048	-0.009	2.511	-0.188
Portfolio Metrics – Bottom Quartile	Average Monthly Returns	1.577%	0.335%	1.663%	0.012%	1.692%	0.220%	1.396%	-0.013%	1.233%	-0.020%	0.696%	0.128%
	Cumulative Returns	7.146	1.132	8.017	0.959	8.201	1.049	5.600	0.927	4.535	0.925	2.247	1.009
	Annualized Return	19.575%	2.700%	20.833%	-0.894%	21.082%	1.026%	16.954%	-1.613%	14.734%	-1.647%	7.639%	0.189%
	Volatility	13.560%	16.467%	13.380%	14.513%	14.371%	17.541%	14.104%	16.964%	13.929%	16.573%	13.878%	16.395%
	Returns/Risk	1.444	0.164	1.557	-0.062	1.467	0.058	1.202	-0.095	1.058	-0.099	0.550	0.012
Benchmark	Average Monthly Returns	1.661%	-0.016%										
	Cumulative Returns	8.040	0.931										
	Annualized Return	20.863%	-1.525%										
	Volatility	12.920%	16.156%										
	Returns/Risk	1.615	-0.094										

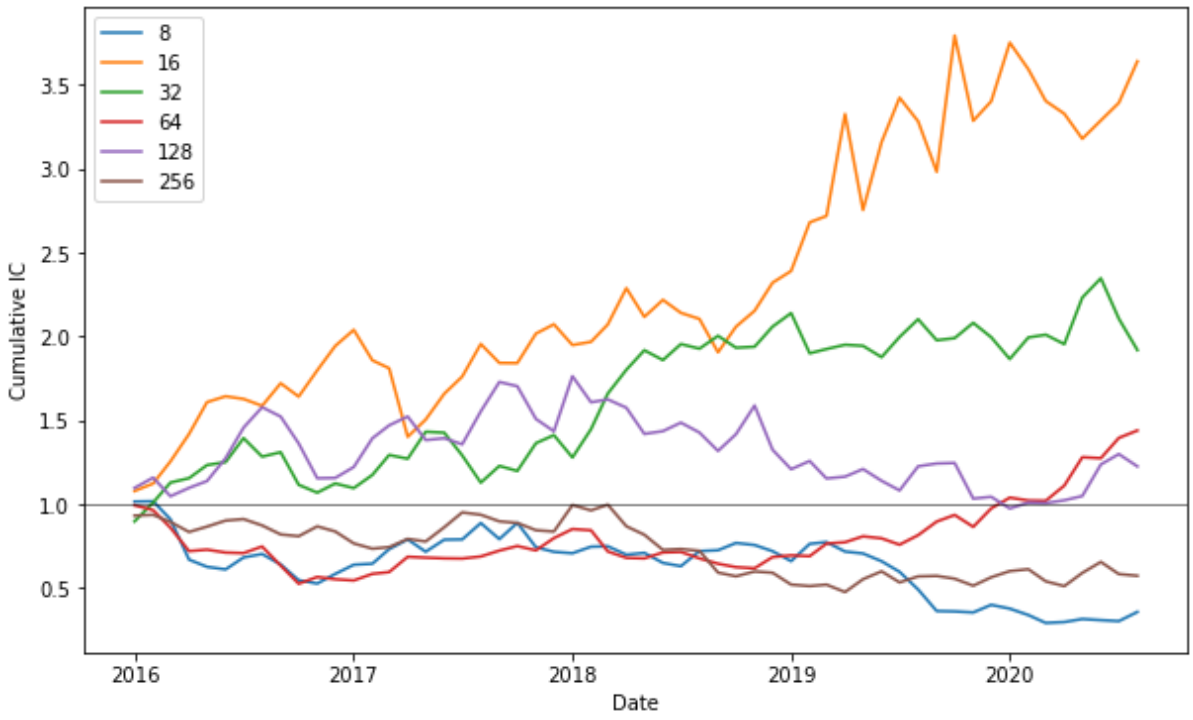
Panel B: Reduced

Number of Nodes:		8-4-2		16-8-4		32-16-8		64-32-16		128-64-32		256-128-64	
		Train Period	Test Period	Train Period	Test Period	Train Period	Test Period	Train Period	Test Period	Train Period	Test Period	Train Period	Test Period
Regression Metrics	Demeaned R2	3.702%	0.000%	3.687%	0.001%	3.666%	0.010%	3.756%	0.065%	3.763%	0.001%	3.666%	0.009%
	Slope Coefficient	-0.012	-0.001	1.526	0.348	0.118	0.374	0.465	0.510	0.325	-0.048	-0.002	0.007
	T-stat	-2.515	-0.067	1.968	0.212	0.552	0.886	3.910	2.229	4.053	-0.300	-0.523	0.839
Information Coefficient Metrics	Average IC	-0.016	-0.002	0.013	0.009	-0.001	-0.004	0.021	0.024	0.026	-0.005	-0.009	0.022
	Cumulative IC	0.062	0.732	3.059	1.408	0.553	0.641	7.682	3.013	15.936	0.611	0.146	2.751
	Standard Deviation of IC	0.099	0.085	0.099	0.073	0.089	0.089	0.103	0.094	0.100	0.087	0.100	0.090
	Qian and Hua IR	-0.161	-0.024	0.134	0.120	-0.006	-0.043	0.202	0.257	0.261	-0.057	-0.092	0.246
Average Monthly Regression Metrics	Average Slope Coefficient	-0.008	-0.001	1.377	0.807	0.011	0.151	0.252	0.464	0.269	-0.083	-0.004	0.015
	Slope T-stat	-1.631	-0.139	1.734	0.586	0.056	0.413	2.004	1.942	3.209	-0.588	-0.823	1.758
Portfolio Metrics – Top Quartile	Average Monthly Returns	1.617%	0.149%	1.751%	0.144%	1.734%	-0.094%	1.849%	0.364%	1.992%	0.078%	1.377%	0.514%
	Cumulative Returns	7.468	1.031	8.976	1.010	8.788	0.892	10.090	1.154	12.247	0.954	5.441	1.242
	Annualized Return	20.055%	0.650%	22.079%	0.205%	21.845%	-2.410%	23.386%	3.110%	25.578%	-0.999%	16.648%	4.746%
	Volatility	14.080%	15.135%	13.500%	17.096%	13.425%	16.197%	14.145%	15.914%	13.661%	19.029%	14.317%	17.266%
	Returns/Risk	1.424	0.043	1.636	0.012	1.627	-0.149	1.653	0.195	1.872	-0.052	1.163	0.275
Portfolio Metrics – Bottom Quartile	Average Monthly Returns	1.837%	0.053%	1.566%	-0.175%	1.566%	-0.288%	1.432%	-0.495%	1.307%	0.063%	1.688%	-0.240%
	Cumulative Returns	10.004	0.960	7.014	0.852	6.997	0.794	5.879	0.714	5.019	0.970	8.178	0.812
	Annualized Return	23.290%	-0.862%	19.373%	-3.366%	19.346%	-4.823%	17.472%	-6.959%	15.795%	-0.657%	21.051%	-4.356%
	Volatility	13.660%	16.967%	13.828%	16.114%	13.956%	16.825%	14.029%	15.624%	13.647%	16.577%	14.232%	17.486%
	Returns/Risk	1.705	-0.051	1.401	-0.209	1.386	-0.287	1.245	-0.445	1.157	-0.040	1.479	-0.249
Benchmark	Average Monthly Returns	1.661%	-0.016%										
	Cumulative Returns	8.040	0.931										
	Annualized Return	20.863%	-1.525%										
	Volatility	12.920%	16.156%										
	Returns/Risk	1.615	-0.094										

Figure 6.10: MLP – Three Layer – Cumulative Information Coefficient

A visual representation of the cumulative information coefficient of the ANN models over the test period, as discussed in Section 5.5. Panel A considers the All Variables models, while Panel B considers the Reduced Variables models. The numbers in the legend represent the number of nodes in the first hidden layer.

Panel A: All Variables



Panel B: Reduced Variables

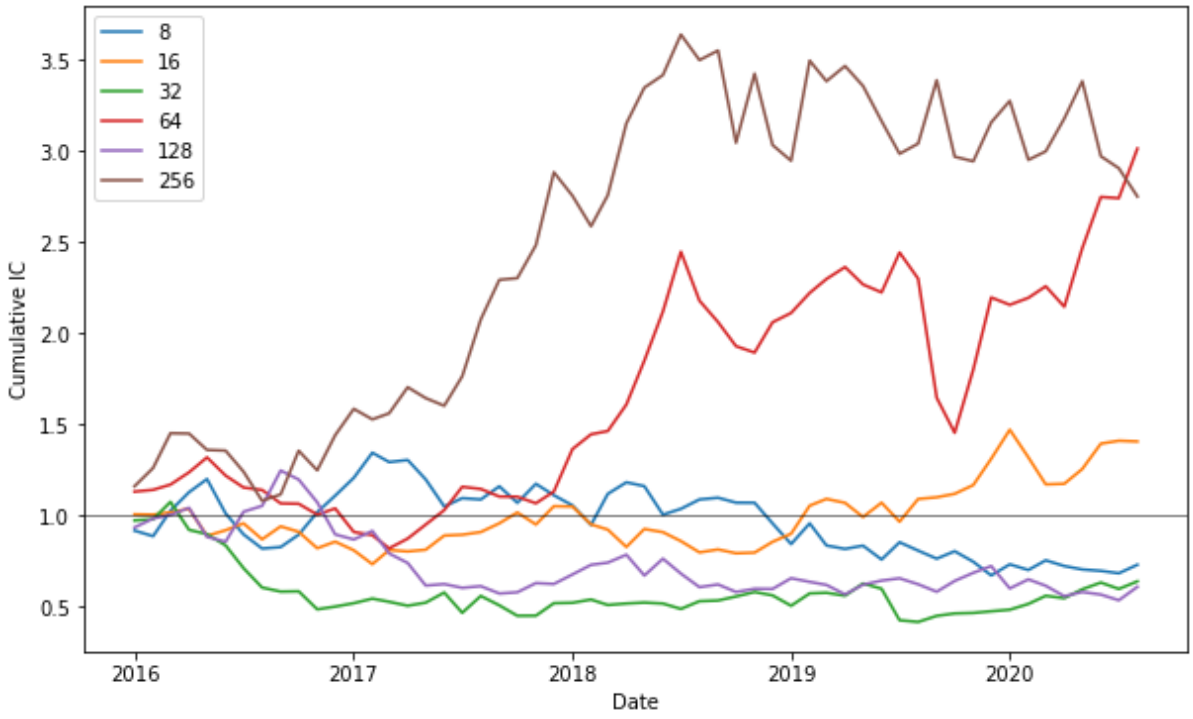
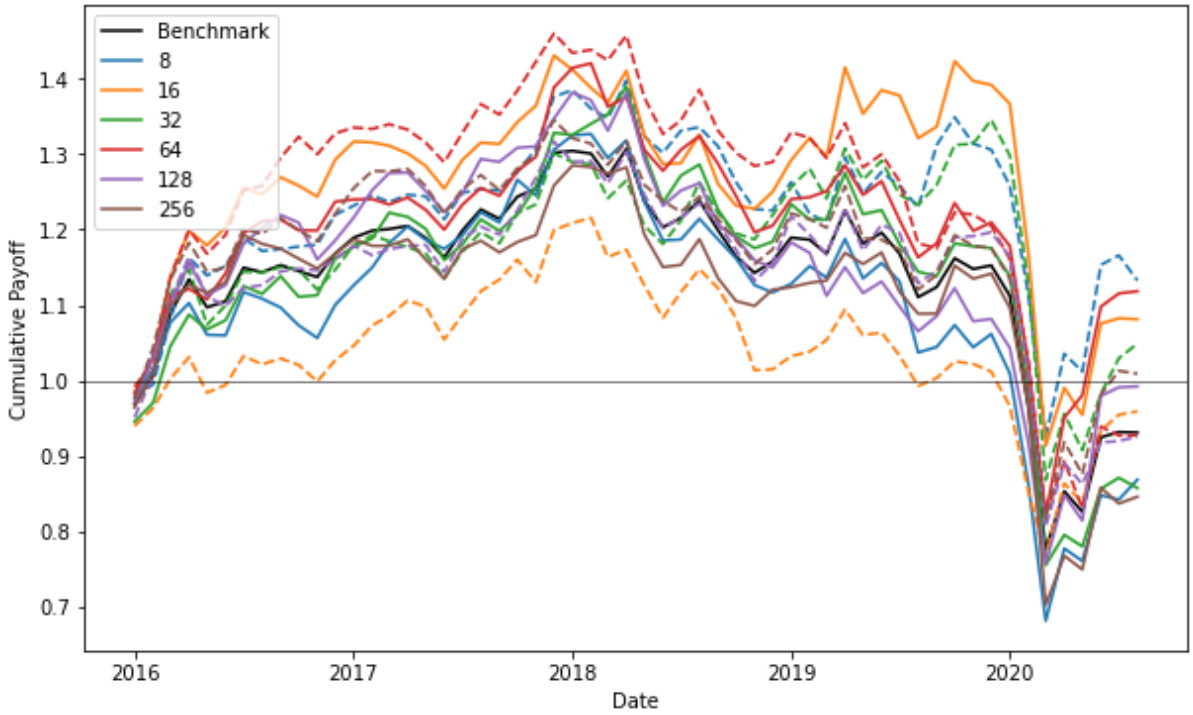


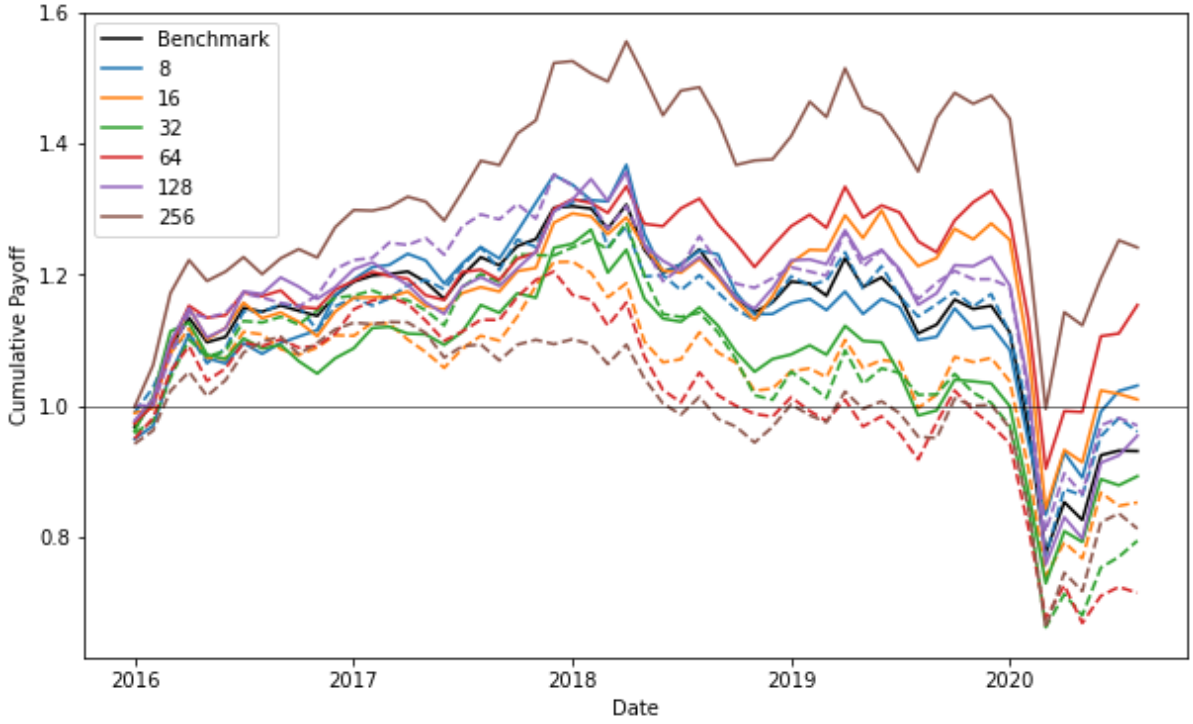
Figure 6.11: MLP – Three Layer – Cumulative Portfolio Returns

A visual representation of the cumulative portfolio performance of the ANN models over the test period for the top and bottom quartile of ranked returns, as discussed in Section 5.5. The benchmark represents an equal weighted monthly rebalanced portfolio of all stocks in the dataset each month. Panel A considers the All Variables models, while Panel B considers the Reduced Variables models. The numbers in the legend represent the number of nodes in the first hidden layer. Solid lines represent the Top Quartile portfolio, dashed lines represent the Bottom Quartile portfolio.

Panel A: All Variables



Panel B: Reduced Variables



6.4.4 Four Hidden Layers

The out-of-sample performance of the four hidden layer ANN models is considered in this section. Table 6.9 summarizes the performance evaluation metrics discussed in Section 5.5. Figures 6.12 and 6.13 show the cumulative IC and model portfolio performance over the testing period. In the table and figures, performance is separately shown for the All Variables and Reduced Variables models in Panel A and B, respectively.

Performance of the top performing four layer models deteriorates from the best three layer models, with increased evidence of overfitting with increased model complexity. Considering Table 6.9, similar patterns to those discussed in Section 6.4.3 are evident, such as improving in sample performance with node count growth and subsequent overfitting, the negative impact of Dropout on less complex models, deteriorating performance of the Reduced Variables models with high node counts, relative outperformance of the Reduced Variables model, and a lack of relationship between training and test sample performance.

In Panel A, test period performance peaks for the 64 node model, after which there is evidence of overfitting. Relative to the top three layer All Variables model, the model underperforms. In Panel B, the 128 node model performs best, but also underperforms the top three layer model. The addition of a hidden layer appeared to reduce model performance and generalization, with greater evidence of overfitting. This suggests that three hidden layers may be the optimal network depth for model generalization on the JSE, with increased complexity increasing computational needs without enhancing out-of-sample performance.

Figures 6.12 and 6.13 further highlight the decline in performance relative to the three layer models. In Panel A, all models show relatively weak performance over the whole period, with a moderate uptick in performance for the top 64 node model since 2019, showing some ability to differentiate between winners and losers over the period. In Panel B, the top performing 128 node model showed initial mixed performance, followed by relatively steady positive performance from 2018. The model is moderately successful at differentiating winners from losers, with greater success at forecasting the Top Quartile.

Table 6.9: MLP Model Performance – Four Layer

This table shows the performance of the All Variables ANN model in Panel A and Reduced Variables ANN Model in Panel B based on the metrics discussed in Section 5.5 across the training and test periods, along with an equal weighted benchmark portfolio.

Panel A: All

Number of Nodes:		16-8-4-2		32-16-8-4		64-32-16-8		128-64-32-16		256-128-64-32		512-256-128-64	
		Train Period	Test Period	Train Period	Test Period	Train Period	Test Period	Train Period	Test Period	Train Period	Test Period	Train Period	Test Period
redRegression Metrics	Demeaned R2	3.673%	0.013%	3.670%	0.010%	3.704%	0.014%	3.902%	0.008%	4.659%	0.010%	6.263%	0.025%
	Slope Coefficient	-0.005	-0.007	0.638	-0.587	0.363	-0.290	0.707	-0.164	1.123	-0.140	1.393	-0.177
	T-stat	-1.246	-1.000	0.971	-0.872	2.569	-1.013	6.292	-0.756	12.908	-0.858	21.040	-1.381
Information Coefficient Metrics	Average IC	-0.007	-0.021	0.004	-0.010	0.015	0.004	0.038	-0.016	0.083	-0.011	0.134	-0.009
	Cumulative IC	0.208	0.248	1.083	0.461	4.085	1.058	76.468	0.327	23164	0.435	10142659	0.463
	Standard Deviation of IC	0.096	0.080	0.085	0.087	0.094	0.079	0.092	0.086	0.092	0.087	0.098	0.099
	Qian and Hua IR	-0.074	-0.266	0.049	-0.114	0.160	0.052	0.408	-0.187	0.899	-0.126	1.364	-0.088
Average Monthly Regression Metrics	Average Slope Coefficient	-0.004	-0.008	0.318	-0.361	0.325	0.129	0.530	-0.260	0.805	-0.089	1.051	-0.079
	Slope T-stat	-0.996	-1.324	0.302	-0.154	2.295	0.546	4.974	-1.378	9.865	-0.567	13.766	-0.613
Portfolio Metrics – Top Quartile	Average Monthly Returns	1.632%	-0.163%	1.757%	-0.038%	1.806%	0.193%	1.860%	-0.131%	2.372%	-0.139%	2.966%	-0.283%
	Cumulative Returns	7.647	0.848	9.014	0.917	9.748	1.023	10.407	0.844	20.044	0.866	43.414	0.782
	Annualized Return	20.315%	-3.481%	22.126%	-1.845%	22.999%	0.495%	23.733%	-3.571%	31.329%	-3.037%	40.889%	-5.133%
	Volatility	13.819%	17.471%	13.644%	16.728%	12.623%	19.087%	13.015%	19.829%	13.509%	16.652%	12.990%	18.432%
	Returns/Risk	1.470	-0.199	1.622	-0.110	1.822	0.026	1.824	-0.180	2.319	-0.182	3.148	-0.278
Portfolio Metrics – Bottom Quartile	Average Monthly Returns	1.787%	0.126%	1.476%	0.196%	1.469%	-0.072%	1.302%	0.326%	0.874%	-0.015%	0.390%	0.056%
	Cumulative Returns	9.501	0.993	6.311	1.035	6.200	0.891	4.919	1.110	2.855	0.931	1.487	0.975
	Annualized Return	22.712%	-0.160%	18.233%	0.744%	18.041%	-2.448%	15.584%	2.270%	10.005%	-1.523%	3.672%	-0.542%
	Volatility	12.795%	17.773%	13.075%	17.711%	13.608%	17.689%	14.466%	18.074%	13.555%	16.217%	14.466%	15.383%
	Returns/Risk	1.775	-0.009	1.395	0.042	1.326	-0.138	1.077	0.126	0.738	-0.094	0.254	-0.035
Benchmark	Average Monthly Returns	1.661%	-0.016%										
	Cumulative Returns	8.040	0.931										
	Annualized Return	20.863%	-1.525%										
	Volatility	12.920%	16.156%										
	Returns/Risk	1.615	-0.094										

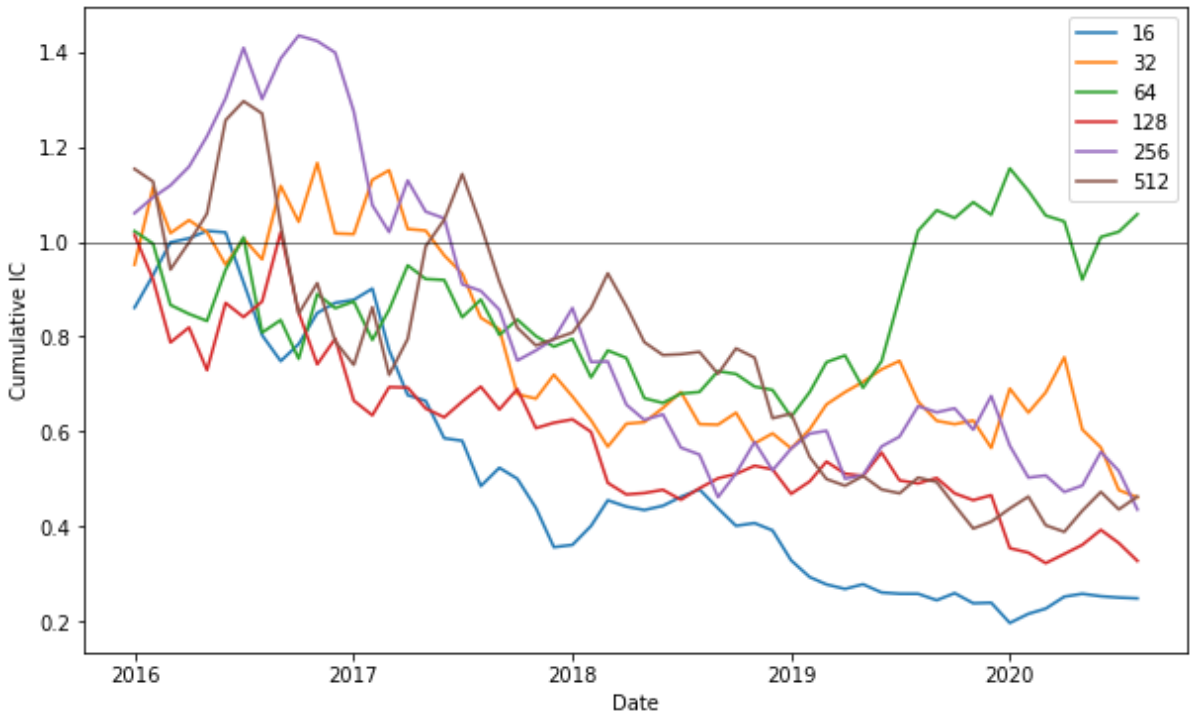
Panel B: Reduced

Number of Nodes:		16-8-4-2		32-16-8-4		64-32-16-8		128-64-32-16		256-128-64-32		512-256-128-64	
		Train Period	Test Period	Train Period	Test Period	Train Period	Test Period	Train Period	Test Period	Train Period	Test Period	Train Period	Test Period
Regression Metrics	Demeaned R2	3.664%	0.035%	3.666%	0.021%	3.689%	0.001%	3.813%	0.026%	4.033%	0.011%	3.668%	0.001%
	Slope Coefficient	0.001	-0.012	-0.148	0.589	0.258	-0.057	0.467	0.265	0.636	-0.145	-0.004	0.002
	T-stat	0.225	-1.647	-0.607	1.278	2.021	-0.228	4.971	1.415	7.839	-0.904	-0.838	0.255
Information Coefficient Metrics	Average IC	-0.004	-0.025	-0.011	0.009	0.014	-0.007	0.032	0.014	0.055	-0.018	-0.012	-0.002
	Cumulative IC	0.353	0.205	0.131	1.410	3.798	0.517	41.392	1.832	822.369	0.292	0.124	0.736
	Standard Deviation of IC	0.093	0.076	0.095	0.081	0.092	0.092	0.084	0.084	0.083	0.082	0.090	0.085
	Qian and Hua IR	-0.039	-0.328	-0.112	0.116	0.155	-0.079	0.381	0.170	0.670	-0.222	-0.128	-0.022
Average Monthly Regression Metrics	Average Slope Coefficient	-0.001	-0.014	-0.379	0.509	0.249	-0.196	0.340	0.235	0.527	-0.194	-0.005	0.001
	Slope T-stat	-0.195	-2.500	-1.446	1.286	2.046	-0.771	4.479	1.458	7.283	-1.304	-1.396	0.087
Portfolio Metrics – Top Quartile	Average Monthly Returns	1.605%	-0.395%	1.646%	0.162%	1.822%	0.033%	1.956%	0.161%	2.017%	-0.289%	1.601%	0.137%
	Cumulative Returns	7.401	0.749	7.824	1.028	9.845	0.935	11.743	1.016	12.589	0.795	7.364	1.012
	Annualized Return	19.958%	-6.004%	20.565%	0.584%	23.109%	-1.420%	25.099%	0.344%	25.892%	-4.788%	19.902%	0.260%
	Volatility	13.624%	16.652%	13.481%	16.380%	13.422%	18.842%	13.302%	17.624%	13.937%	16.643%	13.599%	16.670%
	Returns/Risk	1.465	-0.361	1.525	0.036	1.722	-0.075	1.887	0.020	1.858	-0.288	1.464	0.016
Portfolio Metrics – Bottom Quartile	Average Monthly Returns	1.755%	-0.020%	1.607%	-0.118%	1.572%	0.333%	1.342%	-0.019%	1.039%	0.109%	1.774%	0.068%
	Cumulative Returns	9.038	0.917	7.455	0.863	7.094	1.126	5.269	0.916	3.534	0.997	9.253	0.967
	Annualized Return	22.157%	-1.832%	20.036%	-3.120%	19.497%	2.575%	16.309%	-1.869%	12.160%	-0.067%	22.418%	-0.715%
	Volatility	13.339%	17.635%	13.387%	18.125%	13.616%	16.933%	13.455%	18.121%	13.670%	16.399%	13.439%	17.123%
	Returns/Risk	1.661	-0.104	1.497	-0.172	1.432	0.152	1.212	-0.103	0.890	-0.004	1.668	-0.042
Benchmark	Average Monthly Returns	1.661%	-0.016%										
	Cumulative Returns	8.040	0.931										
	Annualized Return	20.863%	-1.525%										
	Volatility	12.920%	16.156%										
	Returns/Risk	1.615	-0.094										

Figure 6.12: MLP – Four Layer – Cumulative Information Coefficient

A visual representation of the cumulative information coefficient of the ANN models over the test period, as discussed in Section 5.5. Panel A considers the All Variables models, while Panel B considers the Reduced Variables models. The numbers in the legend represent the number of nodes in the first hidden layer.

Panel A: All Variables



Panel B: Reduced Variables

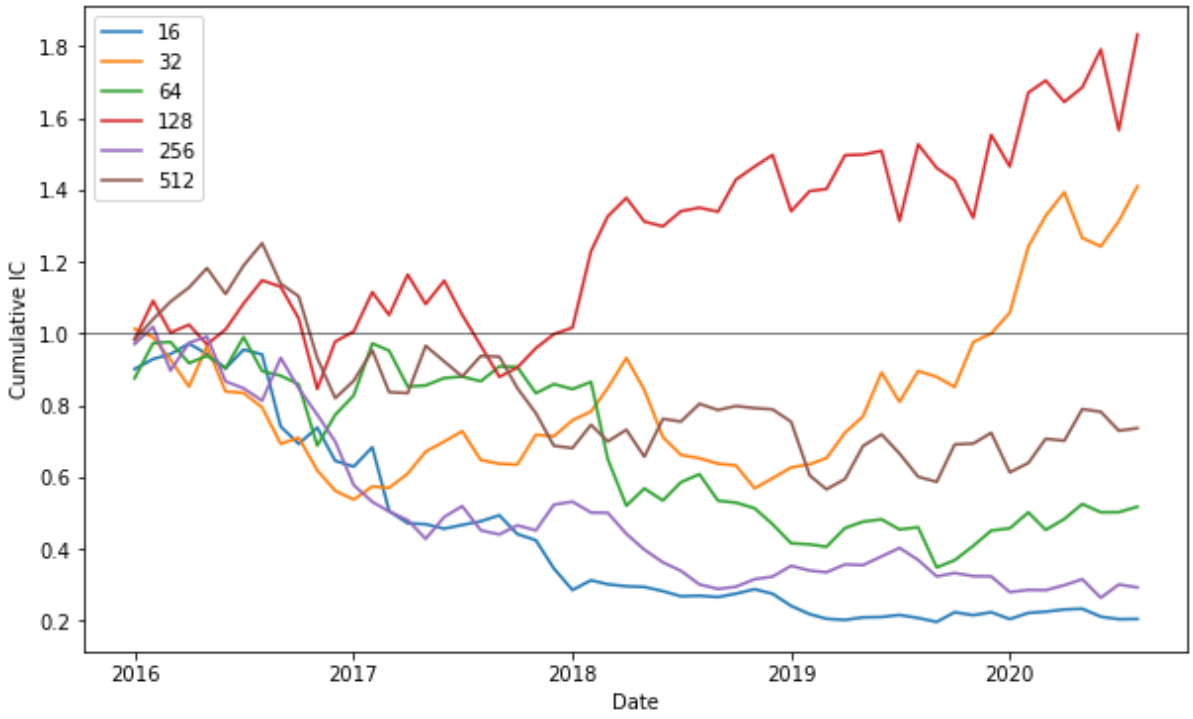
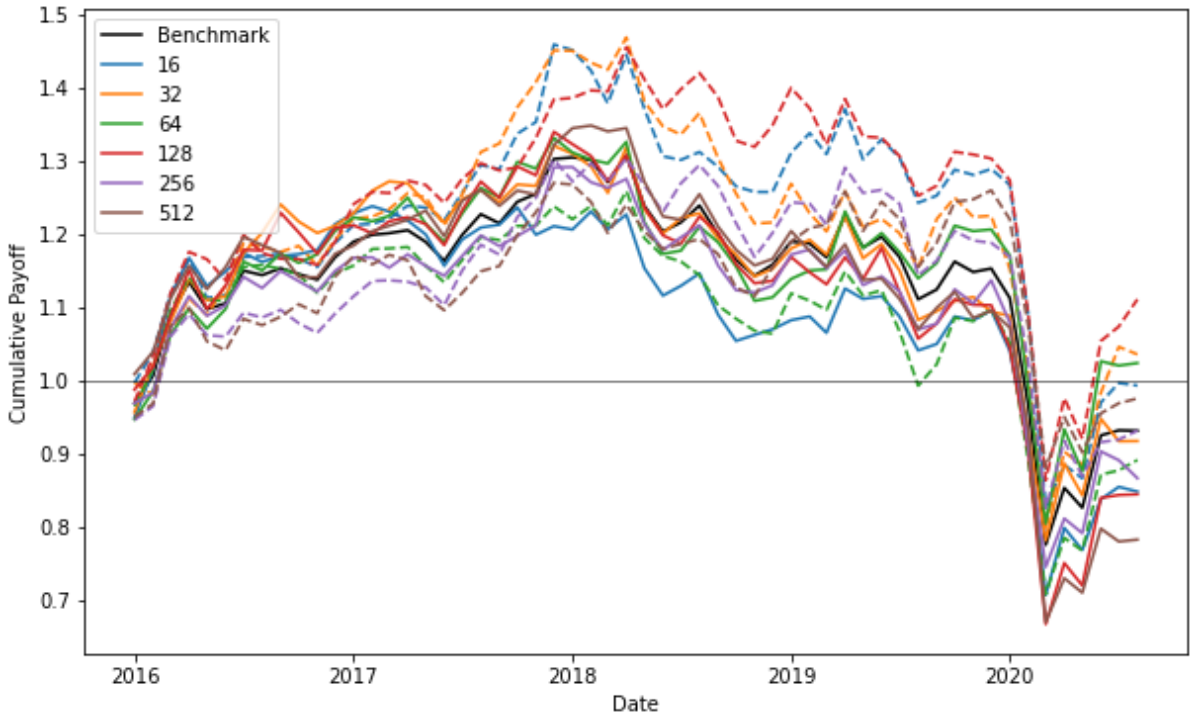


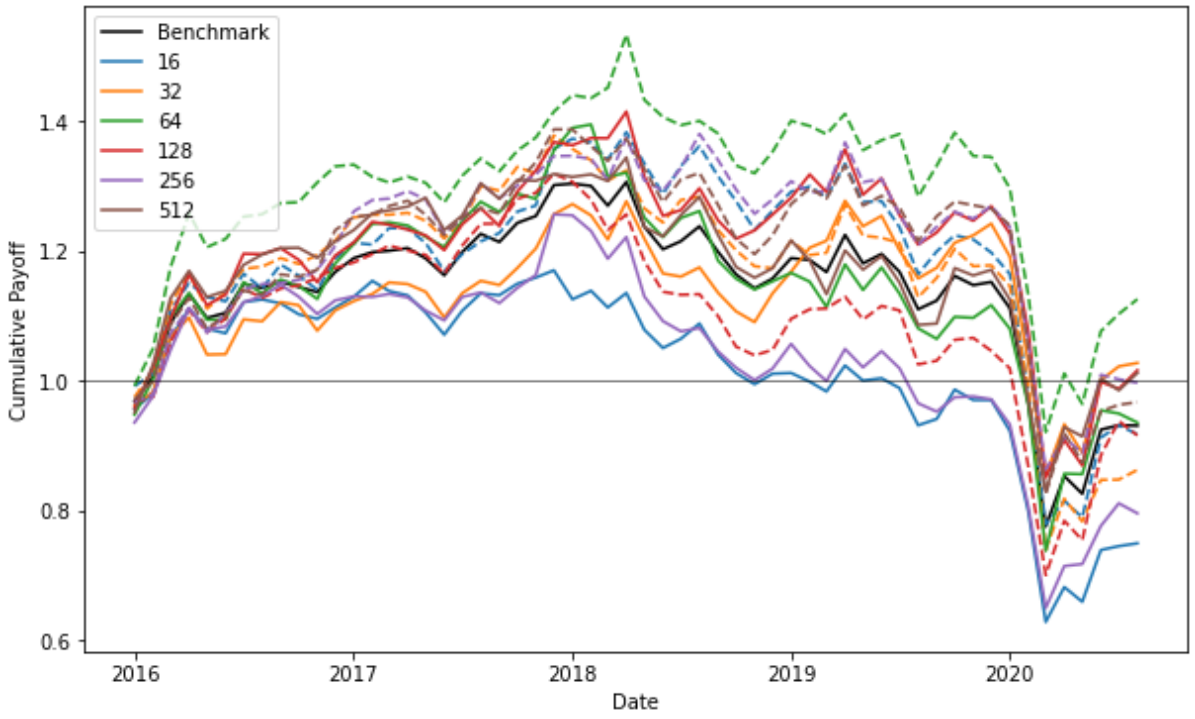
Figure 6.13: MLP – Four Layer – Cumulative Portfolio Returns

A visual representation of the cumulative portfolio performance of the ANN models over the test period for the top and bottom quartile of ranked returns, as discussed in Section 5.5. The benchmark represents an equal weighted monthly rebalanced portfolio of all stocks in the dataset each month. Panel A considers the All Variables models, while Panel B considers the Reduced Variables models. The numbers in the legend represent the number of nodes in the first hidden layer. Solid lines represent the Top Quartile portfolio, dashed lines represent the Bottom Quartile portfolio.

Panel A: All Variables



Panel B: Reduced Variables



6.4.5 Five Hidden Layers

The out-of-sample performance of the five hidden layer ANN models is considered in this section. Table 6.10 summarizes the performance evaluation metrics discussed in Section 5.5. Figures 6.14 and 6.15 show the cumulative IC and model portfolio performance over the testing period. In the table and figures, performance is separately shown for the All Variables and Reduced Variables models in Panel A and B, respectively.

It is evident that the addition of another hidden layer leads to further deterioration in model performance, with evidence of overfitting as model complexity increases, and underfitting in models with fewer nodes. Considering Table 6.10, similar patterns to the previous models are evident, however, the best performing All Variable model outperforms the best Reduced Variables model. Additionally, there is no evidence of deteriorating in sample performance for the Reduced Variables model with high node counts.

In Panel A, of the models with acceptable in-sample performance, the 128 node model performs best in terms of Demeaned R2, T-Stat, IC metrics, and Bottom Quartile portfolio metrics. The 256 node model performs best in terms of Slope T-Stat and Top Quartile portfolio metrics. Both offer similar performance to the top four layer model. In Panel B, none of the models perform particularly well, with the 512 node model performing best on regression and portfolio metrics. The deterioration of performance relative to the three layer models supports the view that three hidden layers may be the optimal network depth for model generalization on the JSE, with the addition of hidden layers increasing computational needs with no clear enhancement to out-of-sample performance.

Figures 6.14 and 6.15 show volatile and weak performance, particularly for the Reduced Variables models. In Panel A, the 128 node model performed well initially, but declined substantially from 2018, with poor portfolio performance overall. The 256 node model, despite weak IC metrics, showed relatively consistent ability to distinguish winners from losers. In Panel B, excluding the 32 node model due to weak in sample performance, all models performed poorly. Overall, the addition of a fifth hidden layer appears to detract from performance, particularly for the Reduced Variable model.

Table 6.10: MLP Model Performance – Five Layer

This table shows the performance of the All Variables ANN model in Panel A and Reduced Variables ANN Model in Panel B based on the metrics discussed in Section 5.5 across the training and test periods, along with an equal weighted benchmark portfolio.

Panel A: All

Number of Nodes:		32-16-8-4-2		64-32-16-8-4		128-64-32-16-8		256-128-64-32-16		512-256-128-64-32		1024-512-256-128-64	
		Train Period	Test Period	Train Period	Test Period	Train Period	Test Period	Train Period	Test Period	Train Period	Test Period	Train Period	Test Period
Regression Metrics	Demeaned R2	3.667%	0.016%	3.670%	0.005%	3.684%	0.002%	4.403%	0.001%	5.572%	0.011%	6.999%	0.046%
	Slope Coefficient	0.004	0.012	-0.335	-0.421	0.206	0.084	1.271	0.042	1.567	-0.151	1.544	-0.239
	T-stat	0.645	1.105	-0.971	-0.630	1.810	0.381	11.109	0.185	17.959	-0.899	23.927	-1.889
Information Coefficient Metrics	Average IC	-0.004	0.010	-0.003	-0.011	0.012	0.005	0.067	0.002	0.115	-0.008	0.148	-0.015
	Cumulative IC	0.387	1.393	0.402	0.451	2.550	1.058	2941	0.958	1164735	0.513	50880969	0.347
	Standard Deviation of IC	0.084	0.086	0.092	0.082	0.095	0.095	0.097	0.075	0.088	0.091	0.097	0.090
	Qian and Hua IR	-0.043	0.112	-0.029	-0.132	0.122	0.058	0.691	0.026	1.312	-0.086	1.533	-0.165
Average Monthly Regression Metrics	Average Slope Coefficient	-0.001	0.009	-0.224	-0.351	0.166	0.052	0.928	0.117	1.179	-0.085	1.142	-0.103
	Slope T-stat	-0.281	0.853	-0.648	-0.628	1.573	0.210	7.729	0.523	13.989	-0.572	14.629	-0.808
Portfolio Metrics – Top Quartile	Average Monthly Returns	1.610%	0.312%	1.509%	0.032%	1.798%	-0.212%	2.419%	0.272%	2.717%	-0.066%	2.980%	-0.128%
	Cumulative Returns	7.542	1.135	6.467	0.952	9.582	0.836	21.631	1.086	31.386	0.899	43.768	0.865
	Annualized Return	20.163%	2.743%	18.496%	-1.055%	22.808%	-3.758%	32.243%	1.782%	36.794%	-2.248%	40.992%	-3.067%
	Volatility	12.825%	14.303%	14.267%	16.708%	13.232%	15.804%	12.446%	17.633%	13.317%	17.059%	13.639%	17.447%
	Returns/Risk	1.572	0.192	1.296	-0.063	1.724	-0.238	2.591	0.101	2.763	-0.132	3.005	-0.176
Portfolio Metrics – Bottom Quartile	Average Monthly Returns	1.684%	-0.123%	1.621%	0.013%	1.586%	-0.241%	1.094%	-0.058%	0.636%	-0.105%	0.384%	0.094%
	Cumulative Returns	8.171	0.863	7.558	0.943	7.177	0.806	3.770	0.910	2.074	0.885	1.486	0.991
	Annualized Return	21.041%	-3.105%	20.186%	-1.260%	19.622%	-4.521%	12.823%	-2.004%	6.856%	-2.577%	3.667%	-0.203%
	Volatility	13.862%	18.111%	13.621%	16.626%	14.043%	18.205%	14.107%	16.099%	13.996%	16.090%	14.050%	16.060%
	Returns/Risk	1.518	-0.171	1.482	-0.076	1.397	-0.248	0.909	-0.124	0.490	-0.160	0.261	-0.013
Benchmark	Average Monthly Returns	1.661%	-0.016%										
	Cumulative Returns	8.040	0.931										
	Annualized Return	20.863%	-1.525%										
	Volatility	12.920%	16.156%										
	Returns/Risk	1.615	-0.094										

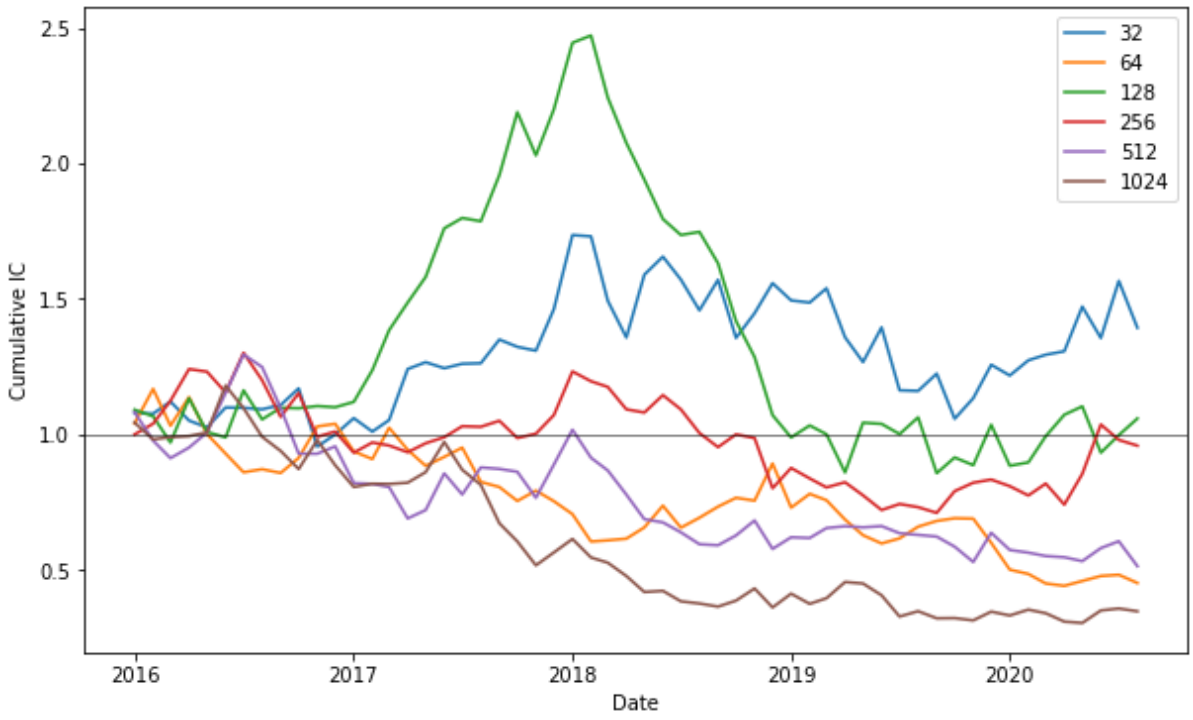
Panel B: Reduced

Number of Nodes:		32-16-8-4-2		64-32-16-8-4		128-64-32-16-8		256-128-64-32-16		512-256-128-64-32		1024-512-256-128-64	
		Train Period	Test Period	Train Period	Test Period	Train Period	Test Period	Train Period	Test Period	Train Period	Test Period	Train Period	Test Period
Regression Metrics	Demeaned R2	3.664%	0.071%	3.669%	0.017%	3.763%	0.003%	3.983%	0.003%	5.112%	0.001%	5.925%	0.001%
	Slope Coefficient	0.001	0.015	0.250	-0.604	0.487	-0.104	0.752	-0.096	1.171	0.022	1.203	-0.021
	T-stat	0.245	2.336	0.908	-1.130	4.055	-0.433	7.285	-0.481	15.606	0.148	19.590	-0.173
Information Coefficient Metrics	Average IC	-0.011	0.018	0.009	-0.016	0.029	-0.008	0.051	-0.010	0.097	-0.004	0.124	-0.005
	Cumulative IC	0.127	2.229	1.880	0.299	24.243	0.497	407.659	0.454	137864	0.626	2675806	0.633
	Standard Deviation of IC	0.094	0.083	0.087	0.102	0.094	0.089	0.098	0.092	0.083	0.091	0.105	0.076
	Qian and Hua IR	-0.117	0.213	0.097	-0.157	0.307	-0.094	0.523	-0.106	1.169	-0.046	1.181	-0.069
Average Monthly Regression Metrics	Average Slope Coefficient	-0.004	0.008	0.344	-1.023	0.476	-0.133	0.631	-0.148	0.859	0.009	0.920	-0.060
	Slope T-stat	-1.303	1.670	1.362	-1.911	4.238	-0.615	6.053	-0.807	11.901	0.067	12.503	-0.534
Portfolio Metrics – Top Quartile	Average Monthly Returns	1.520%	0.139%	1.826%	-0.016%	2.025%	0.105%	2.141%	-0.279%	2.619%	0.089%	2.819%	0.012%
	Cumulative Returns	6.674	1.011	9.796	0.930	12.793	0.987	14.791	0.796	27.443	0.977	35.637	0.937
	Annualized Return	18.835%	0.230%	23.053%	-1.551%	26.076%	-0.273%	27.751%	-4.770%	35.135%	-0.497%	38.383%	-1.382%
	Volatility	13.211%	16.708%	14.114%	16.280%	13.515%	17.294%	13.906%	16.972%	13.899%	17.674%	13.660%	16.960%
	Returns/Risk	1.426	0.014	1.633	-0.095	1.929	-0.016	1.996	-0.281	2.528	-0.028	2.810	-0.081
Portfolio Metrics – Bottom Quartile	Average Monthly Returns	1.826%	-0.360%	1.658%	0.377%	1.404%	0.337%	1.176%	-0.006%	0.700%	-0.001%	0.361%	0.183%
	Cumulative Returns	9.892	0.760	8.024	1.168	5.720	1.129	4.239	0.927	2.252	0.929	1.419	1.035
	Annualized Return	23.163%	-5.702%	20.842%	3.380%	17.180%	2.636%	14.031%	-1.615%	7.661%	-1.576%	3.236%	0.745%
	Volatility	13.450%	17.281%	12.827%	15.423%	13.380%	16.801%	13.512%	17.381%	14.053%	17.244%	14.956%	17.070%
	Returns/Risk	1.722	-0.330	1.625	0.219	1.284	0.157	1.038	-0.093	0.545	-0.091	0.216	0.044
Benchmark	Average Monthly Returns	1.661%	-0.016%										
	Cumulative Returns	8.040	0.931										
	Annualized Return	20.863%	-1.525%										
	Volatility	12.920%	16.156%										
	Returns/Risk	1.615	-0.094										

Figure 6.14: MLP – Five Layer – Cumulative Information Coefficient

A visual representation of the cumulative information coefficient of the ANN models over the test period, as discussed in Section 5.5. Panel A considers the All Variables models, while Panel B considers the Reduced Variables models. The numbers in the legend represent the number of nodes in the first hidden layer.

Panel A: All Variables



Panel B: Reduced Variables

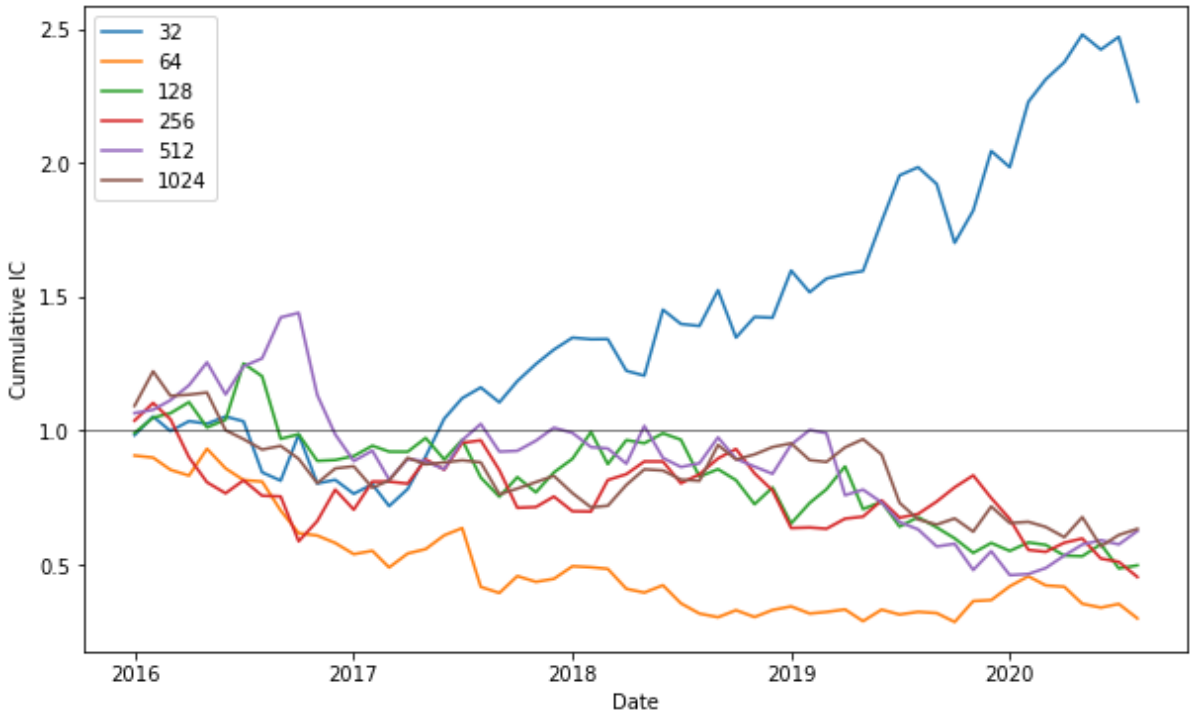
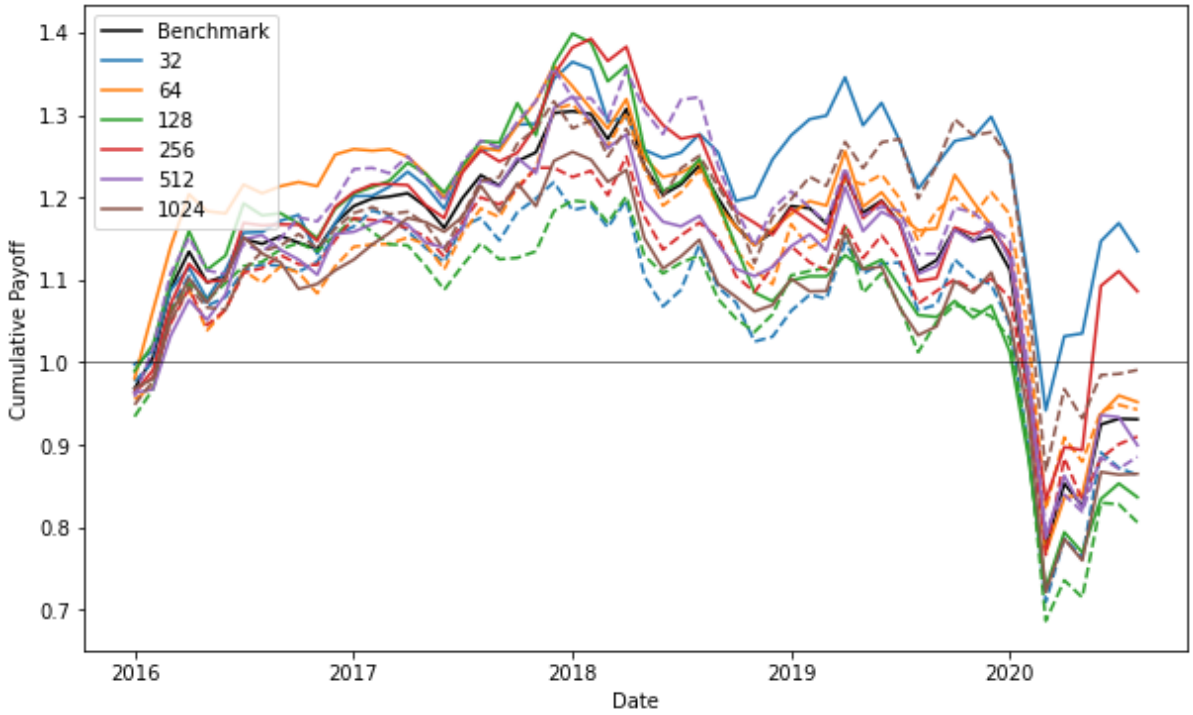


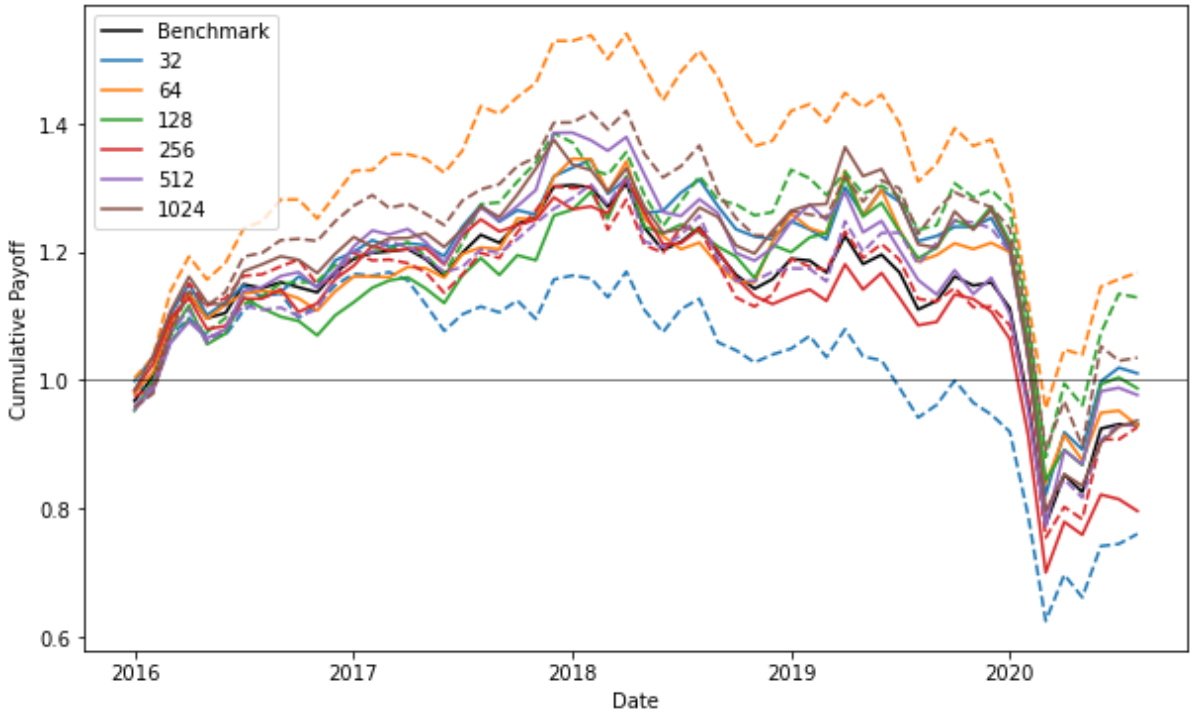
Figure 6.15: MLP – Five Layer – Cumulative Portfolio Returns

A visual representation of the cumulative portfolio performance of the ANN models over the test period for the top and bottom quartile of ranked returns, as discussed in Section 5.5. The benchmark represents an equal weighted monthly rebalanced portfolio of all stocks in the dataset each month. Panel A considers the All Variables models, while Panel B considers the Reduced Variables models. The numbers in the legend represent the number of nodes in the first hidden layer. Solid lines represent the Top Quartile portfolio, dashed lines represent the Bottom Quartile portfolio.

Panel A: All Variables



Panel B: Reduced Variables



6.5 Discussion

This section discusses the key findings from the research, with an emphasis on differences between models. Section 6.5.1 covers findings on optimizing ANN structure in a South African context. Section 6.5.2 compares the performance between the optimal ANNs and the linear benchmark models.

6.5.1 Optimal Neural Network Structure

This section highlights key insights with respect to node count, network depth, and variable selection from Sections 6.4.1 to 6.4.5. Table 6.11 summarizes the top models for each variable subset and hidden layer count. Refer to Appendix B Tables B.1 and B.2 for comprehensive rankings across performance metrics for all configurations. Models in which out-of-sample performance exceeds in sample performance are excluded.

Like Gu, Kelly and Xiu (2020), the optimal network depth appears to be three hidden layers, supporting the potential of deeper models to forecast returns on the JSE. The results reflect the bias-variance trade-off, with model performance initially increasing with model complexity before reaching a point beyond which performance deteriorates. The improved performance from one to three hidden layers supports the theoretical advantage of deeper networks when limited data is available, as composition functions between layers reduce the number of parameters required to approximate relationships (Aggarwal, 2018).

Waning performance with the addition of further hidden layers supports the theory that deep networks may be too complex for small data sets (Aggarwal, 2018). Deeper networks require estimating more parameters which increases the risk of overfitting, particularly for small data sets with low signal-to-noise ratios (Aggarwal, 2018; Scarpino, 2018; Drobetz and Otto, 2021). Evidence of overfitting in models with higher node counts further suggests that the data set is insufficient for more complex model structures.

For less complex models, there are likely too few nodes to accurately classify the data leading to underfitting (Aggarwal, 2018; Scarpino, 2018). This is evident in poor in-sample and out-of-sample performance for models with low node counts and fewer hidden layers, implying weak ability to capture underlying relationships. For the single hidden layer model, particularly the All Variables model, higher node counts led to improved performance, in line with the theory that shallow networks may require many hidden nodes to approximate complex functions (Hornik, Stinchcombe and White, 1989).

Performance of the Reduced Variables models outperformed with four of the top five models overall, emphasizing the importance of feature selection (Qiu, Song and Akagi, 2016; Rasekhschaffe and Jones, 2019). In South African studies, Hodnett (2010) and Pyon (2021)

similarly find models with smaller feature sets to outperform, likely avoiding overfitting noisy data in larger feature sets. Appropriate feature selection prior to training reduces the noise-to-signal ratio (Rasekhschaffe and Jones, 2019), particularly important for financial data. The inclusion of irrelevant features can lead to issues in the learning process, leading to overfitting or convergence issues.

While the choice of constant hyperparameters influenced the outcome and the specific optimal architecture found, these findings offer insights into optimal model specification for JSE returns forecasting. In line with the bias-variance trade-off, simpler models risk underfitting the data, while overly complex models risk overfitting the data. These findings provide useful insights for investors seeking new tools to forecast returns on the JSE.

Table 6.11: Ranking of the Top Performing Neural Network Models

This table shows the top performing models by node count (in the first hidden layer) for each network depth considered for both the All Variables and Reduced Variables models, as well as the ranking of each model within the variable subset and overall. The Reduced Variables three layer model with 64 nodes in the first hidden layer was best overall.

Variables	Hidden Layers	Best Model	Rank Within Subset	Overall Rank
All	**1**	256	4	8
	2	32-16	5	9
	3	64-32-16	1	5
	4	64-32-16-8	2	6
	5	128-64-32-16-8	3	7
Reduced	**1**	32	4	4
	2	64-32	3	2
	3	64-32-16	1	1
	4	128-64-32-16	2	3
	5	512-256-128-64-32	5	9

6.5.2 Performance of Neural Network Relative to Linear Benchmark

This section compares the out-of-sample performance of best performing ANN models to the benchmark linear models. Table 6.12 Panels A and B shows the performance over the whole test period (January 2016 to August 2020) and excluding COVID-19 volatility (January 2016 to January 2020). Table 6.13 Panels A and B summarizes ranked out-of-sample performance of the models in both periods.

Like Hodnett (2010), the top ANN model underperformed the linear benchmark overall. The regression and IC metrics in Table 6.12 show superior forecasting performance for the linear models, while the portfolio metrics highlight better ability to distinguish between top and bottom performers. This contradicts the theory that ANNs can effectively capture both linear and

nonlinear relationships between variables (Aggarwal, 2018), likely due to a limited data set with low signal-to-noise ratios, poor feature selection, and sub-optimal model tuning. For example, the low learning rate may have caused the models to converge at local mimimums. Still, the Reduced Variables ANN outperforms on certain risk adjusted metrics, including Qian and Hua IR and Slope T-stat, suggesting greater stability in performance over time. This aligns with the linear models' strong performance post-2019 evident in Figures 6.4 and 6.5, and the ANN's more consistent behaviour evident in Panel B of Figures 6.10 and 6.11.

Excluding COVID-19 (January 2016 to January 2020), the top ANN model outperforms the linear models on all metrics other than Average IC and Top Quartile Annualized Return, with greater ability to forecast Bottom Quartile returns, and greater risk-adjusted returns for the Top Quartile. This suggests that the ANN models may be more suited to exploiting market inefficiencies during 'normal' market conditions. While this is subject to bias in model selection and could be period and data specific, the findings support the potential of ANN models to forecast returns on the JSE.

The ability of the ANN to forecast returns that outperform an equal-weighted benchmark indicates some degree of predictability exists on the JSE based on fundamental factors. However, overall inconsistent performance of the models, including the linear models, does not provide sufficient evidence against market efficiency on the JSE. Further, with no clear relationship between training and test period performance and with high sensitivity to model structure, optimizing the ANN network architecture ex-ante remains a challenge.

.

Table 6.12: Model Performance Comparison – Linear Models versus Top ANN Models

This table shows the performance of the linear benchmark models and the top performing All Variables and Reduced Variables ANN Models (both three hidden layers with 64 nodes in the first layer). Panel A shows Test Period performance to August 2020, while Panel B shows Test Period performance to January 2020.

Panel A: As of August 2020

		Linear – All Variables		Linear – Reduced Variables		ANN – All Variables		ANN – Reduced Variables	
		Train Period	**Test Period**	**Train Period**	**Test Period**	**Train Period**	**Test Period**	**Train Period**	**Test Period**
Regression Metrics	Demeaned R2	4.399%	0.148%	4.338%	0.135%	3.769%	0.007%	3.756%	0.065%
	Slope Coefficient	0.888	0.521	0.931	0.538	0.517	0.166	0.465	0.510
	T-stat	11.077	3.385	10.609	3.229	4.163	0.720	3.910	2.229
Information Coefficient Metrics	Average IC	0.108	0.044	0.102	0.041	0.028	0.009	0.021	0.024
	Cumulative IC	207550	4.17	107276	3.67	27.092	1.440	7.682	3.013
	Standard Deviation of IC	0.147	0.189	0.146	0.185	0.076	0.073	0.103	0.094
	Qian and Hua IR	0.732	0.232	0.699	0.221	0.370	0.126	0.202	0.257
Average Monthly Regression Metrics	Average Slope Coefficient	0.939	0.585	0.971	0.582	0.391	0.238	0.252	0.464
	Slope T-stat	7.606	1.853	7.309	1.754	3.985	1.349	2.004	1.942
Portfolio Metrics – Top Quartile	Average Monthly Returns	2.501%	0.926%	2.442%	0.711%	1.822%	0.325%	1.849%	0.364%
	Cumulative Returns	22.850	1.536	21.097	1.362	9.824	1.118	10.090	1.154
	Annualized Return	32.903%	9.638%	31.942%	6.852%	23.085%	2.425%	23.386%	3.110%
	Volatility	15.652%	19.901%	15.866%	19.723%	13.543%	17.203%	14.145%	15.914%
	Returns/Risk	2.102	0.484	2.013	0.347	1.705	0.141	1.653	0.195
Portfolio Metrics – Bottom Quartile	Average Monthly Returns	0.687%	-0.534%	0.753%	-0.277%	1.396%	-0.013%	1.432%	-0.495%
	Cumulative Returns	2.219	0.673	2.415	0.784	5.600	0.927	5.879	0.714
	Annualized Return	7.517%	-8.123%	8.347%	-5.086%	16.954%	-1.613%	17.472%	-6.959%
	Volatility	13.972%	19.929%	14.046%	19.218%	14.104%	16.964%	14.029%	15.624%
	Returns/Risk	0.538	-0.408	0.594	-0.265	1.202	-0.095	1.245	-0.445
Benchmark	Average Monthly Returns	1.661%	-0.016%						
	Cumulative Returns	8.040	0.931						
	Annualized Return	20.863%	-1.525%						
	Volatility	12.920%	16.156%						
	Returns/Risk	1.615	-0.094						

Panel B: As of January 2020

		Linear – All Variables		Linear – Reduced Variables		MLP – All Variables		MLP – Reduced Variables	
		Train Period	**Test Period**	**Train Period**	**Test Period**	**Train Period**	**Test Period**	**Train Period**	**Test Period**
Regression Metrics	Demeaned R2	4.399%	0.088%	4.338%	0.092%	3.769%	0.080%	3.756%	0.104%
	Slope Coefficient	0.888	0.104	0.931	0.133	0.517	0.011	0.465	0.265
	T-stat	11.077	0.764	10.609	0.906	4.163	0.052	3.910	1.300
Information Coefficient Metrics	Average IC	0.108	0.025	0.102	0.023	0.028	0.003	0.021	0.020
	Cumulative IC	207550	1.382	107276	1.329	27.092	1.040	7.682	2.157
	Standard Deviation of IC	0.147	0.188	0.146	0.185	0.076	0.073	0.103	0.097
	Qian and Hua IR	0.732	0.131	0.699	0.126	0.370	0.048	0.202	0.210
Average Monthly Regression Metrics	Average Slope Coefficient	0.939	0.165	0.971	0.156	0.391	0.059	0.252	0.321
	Slope T-stat	7.606	0.596	7.309	0.525	3.985	0.360	2.004	1.326
Portfolio Metrics – Top Quartile	Average Monthly Returns	2.501%	0.59%	2.442%	0.49%	1.822%	0.376%	1.849%	0.548%
	Cumulative Returns	22.850	1.301	21.097	1.239	9.824	1.178	10.090	1.284
	Annualized Return	32.903%	6.656%	31.942%	5.386%	23.085%	4.101%	23.386%	6.313%
	Volatility	15.652%	11.137%	15.866%	10.865%	13.543%	9.989%	14.145%	9.486%
	Returns/Risk	2.102	0.598	2.013	0.496	1.705	0.411	1.653	0.665
Portfolio Metrics – Bottom Quartile	Average Monthly Returns	0.687%	0.12%	0.753%	0.26%	1.396%	0.353%	1.432%	-0.070%
	Cumulative Returns	2.219	1.017	2.415	1.092	5.600	1.161	5.879	0.943
	Annualized Return	7.517%	0.407%	8.347%	2.174%	16.954%	3.717%	17.472%	-1.438%
	Volatility	13.972%	14.827%	14.046%	14.338%	14.104%	10.924%	14.029%	11.111%
	Returns/Risk	0.538	0.027	0.594	0.152	1.202	0.340	1.245	-0.129
Benchmark	Average Monthly Returns	1.661%	0.255%						
	Cumulative Returns	8.040	1.112						
	Annualized Return	20.863%	2.638%						
	Volatility	12.920%	9.695%						
	Returns/Risk	1.615	0.272						

Table 6.13: Ranking of the Linear Benchmark and Top Neural Network Models

This table shows the ranking of the out-of-sample performance of the linear benchmark models and the top performing All Variables and Reduced Variables ANN Models (both three hidden layers with 64 nodes in the first layer). Panel A shows performance to August 2020, while Panel B shows performance to January 2020.

Panel A: As of August 2020

	Rank			
	Linear - All	**Linear - Reduced**	**ANN - All**	**ANN - Reduced**
Demeaned R2	1	2	4	3
T-stat	1	2	4	3
Average IC	1	2	4	3
Cumulative IC	1	2	4	3
Qian and Hua IR	2	3	4	1
Average Slope Coefficient	1	2	4	3
Slope T-stat	2	3	4	1
Annualized Return – Top Quartile	1	2	4	3
Returns/Risk – Top Quartile	1	2	4	3
Annualized Return – Bottom Quartile	1	3	4	2
Returns/Risk – Bottom Quartile	2	3	4	1
Average	**1.3**	**2.4**	**4.0**	**2.4**
Rank	**1**	**2**	**4**	**2**

Panel B: As of January 2020

	Rank			
	Linear - All	**Linear - Reduced**	**ANN - All**	**ANN - Reduced**
Demeaned R2	3	2	4	1
T-stat	3	2	4	1
Average IC	1	2	4	3
Cumulative IC	2	3	4	1
Qian and Hua IR	2	3	4	1
Average Slope Coefficient	2	3	4	1
Slope T-stat	2	3	4	1
Annualized Return – Top Quartile	1	3	4	2
Returns/Risk – Top Quartile	2	3	4	1
Annualized Return – Bottom Quartile	2	3	4	1
Returns/Risk – Bottom Quartile	2	3	4	1
Average	**2.0**	**2.7**	**4.0**	**1.3**
Rank	**2**	**3**	**4**	**1**

6.6 Limitations and Future Research

To simplify the analysis, steps are taken which likely limit model performance. Firstly, a linear model is used to select features, potentially excluding important variables. Second, the time-varying nature of anomalies is not accounted for. Third, the activation function, training algorithm, learning rate, number of epochs, batch size, and loss function are kept constant across architectures, resulting in suboptimal performance for some architectures.

Future research could improve on variable selection algorithms, use a moving-window approach to capture changing variable importance over time, and use techniques such as grid search to find the optimal hyperparameter set for different model structures. Additionally, the data period could be extended to examine the robustness of performance and assess whether optimal network structure is time-varying.

Further, the black-box nature of ANNs limits insight into the drivers of performance. While the Reduced Variable subset outperformed, this study does not provide insights into specific return drivers. Future research could uncover which variables drive returns in non-linear models, and examine variable importance relative to linear models.

Lastly, trading frictions are ignored in this study. Identified anomalies may not be exploitable, and thus the theoretical portfolio performance may not be achievable. Future studies could explore whether pricing inefficiencies persist when trading frictions are accounted for.

6.7 Conclusion

This section compares the performance of linear models to ANNs for forecasting returns on the JSE. Value, growth, operating performance, and momentum were identified as relevant factors. Considering two variable subsets, two benchmark multivariate models and various ANN models were trained.

The top performing ANN model for both variable subsets is found to have three hidden layers with a 64-32-16 node count in the hidden layers. Addition of further hidden layers or nodes was found to reduce model generalization, mainly due to overfitting, while less complex models were found to underfit. Models with the Reduced Variables subset outperformed, showing the importance of variable selection to improve model generalization.

Relative to the linear models, the ANNs underperformed. However, the top performing ANN outperformed on risk-adjusted Qian and Hua IR and Slope T-statistic metrics. Additionally, when considering a pre-COVID test period, the top ANN model outperforms on most metrics, producing greater risk-adjusted returns for both the Top and Bottom Quartile portfolios. Overall, these findings support the potential of deep ANNs to forecast returns on the JSE.

7. Conclusion

This study explores the potential of artificial neural networks (ANNs) to forecast returns on the Johannesburg Stock Exchange (JSE) through two key questions:

What is the optimal neural network structure for stock return prediction on the JSE?
Can ANNs trained using fundamental and technical factors predict stock returns on the JSE, and do they outperform multivariate linear models?

The optimal neural network architecture is explored through varying model depths and node counts, while keeping the activation function, training algorithm, learning rate, number of epochs, batch size, and loss function constant. Performance peaks at three hidden layers with 64 nodes in the first hidden layer. Addition of further hidden layers or nodes leads to reduced model generalization, mainly due to overfitting, while less complex models underfit. Highlighting the importance of feature selection, models trained on a reduced variable subset outperform, potentially due to low signal-to-noise ratios in financial data.

Portfolios constructed from forecasts of certain ANN models outperform an equal-weighted benchmark portfolio, suggesting that ANNs trained on fundamental and technical factors could be used to predict stock returns on the JSE. However, performance is highly sensitive to the model structure and feature selection. Relative to multivariate linear models, ANN models underperform, apart from the top performing ANN model on certain risk-adjusted metrics. Further, prior to COVID-19, the top ANN model outperforms, suggesting greater forecasting potential during less volatile periods, demonstrating their potential to outperform linear models. However, this comparison suffers from model selection bias, and the performance could be data specific. With no clear relationship between training and test period performance, selecting the optimal model architecture ex-ante remains a challenge.

Still, this study shows the use and potential advantages of ANNs to forecast returns on the JSE. Limitations in feature selection and training likely constrained model performance, with potential to improve generalization. Limitations include the feature selection process, a single training period, constant hyperparameters, a relatively short sample period, and a lack of insight into the drivers of returns in the ANN models. Future research could explore variable selection algorithms, moving-window approaches, hyperparameter optimization techniques, and drivers of returns within ANN models over a longer sample period.

With growing interest and possibilities in artificial intelligence and machine learning, this paper offers a lower-bound for the potential of neural networks and non-linear return forecasting on the JSE, important to investors seeking new investment strategies. Further research may unlock insights into the stability of features and optimal model structure over time.

References

Abe, M. & Nakayama, H. 2018. Deep Learning for Forecasting Stock Returns in the Cross-Section. *Proceedings of the 22nd Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining.* 3-6 June 2018. Australia: PAKDD. 273-284.

Aggarwal, C.C. 2018. *Neural Networks and Deep Learning.* Cham, Switzerland: Springer.

Banz, R.W. 1981. The Relationship Between Return and Market Value of Common Stocks. *Journal of Financial Economics.* 9(1):3-18.

Basu, S. 1977. Investment Performance of Common Stocks in Relation to Their Price-Earnings Ratios: A Test of the Efficient Market Hypothesis. *The Journal of Finance.* 32(3):663-682.

Basu, S. 1983. The Relationship Between Earnings' Yield, Market Value and Return for NYSE Common Stocks: Further Evidence. *Journal of Financial Economics.* 12(1):129-156.

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

Cao, Q., Parry, M.E. & Leggio, K.B. 2011. The three-factor model and artificial neural networks: predicting stock price movement in China. *Annals of Operations Research.* 185(1):25-44.

Cochrane, J.H. 2011. Presidential Address: Discount Rates. *The Journal of Finance.* 66(4):1047-1108.

De Bondt, W.F.M. & Thaler, R. 1985. Does the Stock Market Overreact? *The Journal of Finance.* 40(3):793-805.

Drobetz, W. & Otto, T. 2021. Empirical Asset Pricing via Machine Learning: Evidence from the European Stock Market. *Journal of Asset Management*. 22(7):507-538.

Duchi, J., Hazan, E. & Singer, Y. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research.* 12(7):2121-2159.

Eakins, S.G. & Stansell, S.R. 2003. Can value-based stock selection criteria yield superior risk-adjusted returns: an application of neural networks. *International Review of Financial Analysis.* 12(1):83-97.

Enke, D. & Thawornwong, S. 2005. The use of data mining and neural networks for forecasting stock market returns. *Expert Systems with Applications.* 29(4):927-940.

Fama, E.F. 1970. Efficient Capital Markets: A Review of Theory and Empirical Work. *The Journal of Finance.* 25(2):383-417.

Fama, E.F. & French, K.R. 1992. The Cross-Section of Expected Stock Returns. *The Journal of Finance.* 47(2):427-465.

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

Fama, E.F. & French, K.R. 2015. A five-factor asset pricing model. *Journal of Financial Economics.* 116(1)1-22.

Fama, E.R. & MacBeth, J.D. 1973. Risk, Return, and Equilibrium: Empirical Tests. *The Journal of Political Economy.* 81(3):607-636.

Gu, S., Kelly, B. & Xiu, D. 2020. Empirical Asset Pricing via Machine Learning. *The Review of Financial Studies.* 33(5):2223-2273.

Harvey, C.R., Liu, Y. & Zhu, H. 2016. …and the Cross-Section of Expected Returns. *The Review of Financial Studies.* 29(1):5-68.

Haugen, R.A. & Baker, N.L. 1996. Commonality in the determinants of expected stock returns. *Journal of Financial Economics.* 43(3):401-439.

He, K., Ren, S., Sun, J. & Zhang, S. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV).* 7-13 December 2015. Santiago, Chile: IEEE Computer Society. 1026-1034.

Hoaglin, D.C. & Iglewitz, B. 1987. Fine-Tuning Some Resistance Rules for Outlier Labelling. *Journal of the American Statistical Association*. 82(400):1147-1149.

Hodnett, K. 2010. *Analysis of the cross-section of equity returns on the JSE Securities Exchange based on linear and nonlinear modeling techniques*. Ph.D. Thesis. University of Cape Town. Available: http://hdl.handle.net/11427/10795 [2025, June 8].

Hornik, K., Stinchcombe, M. & White, H. 1989. Multilayer Feedforward Networks are Universal Approximators. *Neural Networks.* 2(5):359-366.

Hou, K., Xue, C. & Zhang, L. 2020. Replicating Anomalies. *The Review of Financial Studies.* 33(5):2019-2133.

Ioffe, S. & Szegedy, C. 2015. Batch normalization: accelerating deep network training by reducing internal covariate shift. *Proceedings of the 32nd International Conference on Machine Learning*. 7-9 July 2015. France:PMLR. 448-456.

Jegadeesh, N. & Titman, S. 1993. Returns to Buying Winners and Selling Losers: Implications for Stock Market Efficiency. *The Journal of Finance.* 48(1):65-91.

Kanas, A. & Yannapoulos, A. 2001. Comparing linear and nonlinear forecasts for stock returns. *International Review of Economics & Finance.* 10(4):383-398.

Karolyi, G.A. & Van Nieuwerburgh, S. 2020. New Methods for the Cross-Section of Returns. *The Review of Financial Studies.* 33(5):1879-1890.

Kelly, B., Pruitt, S. & Su, Y. Characteristics are covariances: A unified model of risk and return. *Journal of Financial Economics.* 134(3):501-524.

Kryzanowski, L., Galler, M. & Wright, D. W. 1993. Using Artificial Neural Networks to Pick Stocks. *Financial Analysts Journal*. 49(4):21-27.

LeCun, Y., Bottou, L., Orr, G.B. & Muller, K. 1998. Efficient Backprop. In *Neural Networks: Tricks of the Trade.* G. Montavon, G.B. Orr & K. Muller, Eds. Berlin: Springer. 9-48.

Lin, Y., Cunningham III, G.A. & Coggeshall, S.V. 1996. Input variable identification - Fuzzy curves and fuzzy surfaces. *Fuzzy Sets and Systems.* 82(1):65-71.

McCulloch, W.S. & Pitts, W.H. 1943. A Logical Calculus of the Ideas Immanent in Nervous Activity. *The Bulletin of Mathematical Biophysics.* 5(4):115-133.

McLean, R.D. & Pontiff, J. 2016. Does academic research destroy stock return predictability? *The Journal of Finance.* 71(1):5-31.

Nengovhela, M. 2022. *Forecasting stock market returns using machine learning models.* MCom Thesis. Available: https://hdl.handle.net/10210/501443 [2025, June 9].

Olson, D. & Mossman, C. 2003. Neural network forecasts of Canadian stock returns using accounting ratios. *International Journal of Forecasting.* 19(3):453-465.

Phiri, A. 2015. Efficient Market Hypothesis in South Africa: Evidence from Linear and Nonlinear Unit Root Tests. *Managing Global Transitions.* 13(4):369-387.

Pyon, F. 2021. *Artificial neural networks to predict share prices on the Johannesburg stock exchange.* MSc Thesis. Available: http://hdl.handle.net/11427/36046 [2025, June 9].

Qian, E., & Hua, R. 2004. Active Risk and Information Ratio. *Journal of Investment Management.* 2(3):1-15.

Qui, M., Song, Y. & Akagi, F. 2016. Application of artificial neural network for the prediction of stock market returns: The case of the Japanese stock market. *Chaos, Solitons and Fractals.* 85(2016):1-7.

Ramuada, V. C. 2018. *Forecasting Stock Returns: A Comparison of Five Models.* MSc Thesis. Stellenbosch University. Available: http://hdl.handle.net/10019.1/104923 [2025, June 9].

Rasekhschaffe, K.C. & Jones, R.C. 2019. Machine Learning for Stock Selection. *Financial Analysts Journal.* 75(3):70-88.

Rosenberg, B., Reid, K. and Lanstein, R. Persuasive evidence of market inefficiency. *The Journal of Portfolio Management.* 11(3):9-16.

Rosenblatt, F. 1957. *The Perceptron, A Perceiving and Recognizing Automation.* (Research report 85-460-1). Buffalo, New York: Cornell Aeronautical Laboratory, Inc.

Scarpino, M. 2018. *TensorFlow for Dummies*. New Jersey: John Wiley & Sons, Inc.

Sharpe, W.F. 1964. Capital Asset Prices: A Theory of Market Equilibrium Under Conditions of Risk. *The Journal of Finance.* 19(3):435-442.

Tobek, O. & Hronec, M. 2021. Does It Pay to Follow Anomalies Research? Machine Learning Approach with International Evidence. *Journal of Financial Markets.* 56(2021):100588.

Tsai, C., Lin, Y., Yen, D.C. & Chen, Y. 2011. Predicting stock returns by classifier ensembles. *Applied Soft Computing.* 11(2):2452-2459.

Tukey, J.W. 1977. *Exploratory Data Analysis.* Massachusetts: Addison-Wesley.

Van Heerden, J.D. & Van Rensburg, P. 2015. The cross-section of Johannesburg Securities Exchange listed equity returns (1994-2011). *Studies in Economics and Finance.* 32(4):422-444.

Van Rensburg, P. & Robertson, M. 2003. Style characteristics and the cross-section of JSE returns. *Investment Analysts Journal.* 32(57):7-15.

Werbos, P. 1974. *Beyond regression: new tools for prediction and analysis in the behavioral sciences*. Ph.D. Thesis. Harvard University.

Appendix

Appendix A:

A.1 Data Preprocessing

```
import pandas as pd
import numpy as np
import csv

# set seed to ensure results are replicable:
np.random.seed(1)

# Import the Raw CSV File
raw_df = pd.read_csv("/…/monthly_data_JSE_2005_2020 (version 1).csv")

# Drop irrelevant variables
to_drop = ['temp_id',
           'month_num',
           'ticker',
           'sharenum',
           'share_name',
           'exchange_name',
           'exchange_abbreviation',
           'exchange_code']

raw_df.drop(to_drop, inplace = True, axis = 1)


# Drop rows with missing forward returns information
rtns_df = raw_df.dropna(how = "any",
             subset=['returnsfwd'])

# Remove shares that do not trade
threshold = 0.0001 #0.01% threshold
rtns_df.loc[:, 'voltshout'] = rtns_df.loc[:,'voltshout'].copy()/1000000
rtns_df = rtns_df[rtns_df.loc[:,'voltshout'] >= threshold].copy()

# Drop columns missing more than 10% of observations
row_count = rtns_df.shape[0]
rtns_df1 = rtns_df.dropna(axis = 1, thresh = int(0.9*row_count)).copy()


# Append the additional variables
to_add = ['c24mepsp',
          'epsg24']
add_df = rtns_df[to_add]


# Remove correlated and other variables
to_drop1 = ['at', …
            'voltshout']
rtns_df1.drop(to_drop1, inplace=True, axis=1)


# Join the two dataframes
rtns_df2 = rtns_df1.join(add_df)
```

```
# clean for observations that do not make sense
# Also firms with negative earnings yield and CFTP - INCORRECT TO DO THIS

rtns_df2.drop(rtns_df2[rtns_df2['lmv'] <= 0].index, inplace = True)
rtns_df2.drop(rtns_df2[rtns_df2['lnp'] <= 0].index, inplace = True)
rtns_df2.drop(rtns_df2[rtns_df2['ey'] <= 0].index, inplace = True)
rtns_df2.drop(rtns_df2[rtns_df2['cftp'] <= 0].index, inplace = True)

# Convert the columns to appropriate dtypes
rtns_df2['month'] = pd.to_datetime(rtns_df2['month'], infer_datetime_format=True)
rtns_df2.dtypes

# Transform certain variables to better reflect best practice in research
rtns_df3.loc[:, 'bvtm'] = np.log(rtns_df3.loc[:, 'bvtm'])

# Replace inf and -inf with 0 - only due to the previous transformation
rtns_df3.replace([np.inf, -np.inf], 0, inplace=True)

# Drop rows with less than 50% valid data
rtns_df4 = rtns_df3.dropna(how='any',axis = 0, thresh = rtns_df3.shape[1]*0.5).copy()

# Remove other variables
to_drop_sector = ['sector']
rtns_df4.drop(to_drop_sector, inplace=True, axis=1)
rtns_df4.to_csv('rtns_df4.csv')

# Data Exploration - Prior to replacing missing values or Tukey 2.2
file_path = "/…/rtns_df4.csv"
dirty_df = pd.read_csv(file_path,
                parse_dates=['month'],
                index_col=['month'])
dirty_df.drop('Unnamed: 0', inplace = True, axis = 1)
dirty_df.sort_values(by=['month'], inplace=True)

# remove scode
dirty_df.drop('scode', inplace=True, axis=1)

# split the dataframe into time periods
train_dfdirty = dirty_df["2005-01-01":"2015-12-01"]
test_dfdirty = dirty_df["2016-01-01":]

# Replace missing values with median each month
rtns_df5 = dirty_df.groupby(pd.Grouper(freq='M')).transform(lambda x: x.fillna(x.median()))

# Replace Outliers in X
rtns_df4Y = rtns_df5.pop('returnsfwd').copy()

# Replace Outliers - Tukey 2.2 - Training and Test Data - Replaced cross-sectionally each
month to avoid look ahead bias

def iqr(df):

    df_copy = df.copy()
```

```
    bounds = {}

    for month in df_copy.index.get_level_values('month').unique():
        df_month = df_copy.loc[df_copy.index.get_level_values('month') == month]

        for column in df_copy.columns:

            Q1 = np.percentile(df_month[column], 25)
            Q3 = np.percentile(df_month[column], 75)
            IQR = Q3-Q1
            C = 2.2*IQR

            LB = Q1 - C
            UB = Q3 + C

            df_copy.loc[df_copy.index.get_level_values('month') == month, column] = np.clip(
                df_month[column],
                LB,
                UB
            )

            bounds[(month, column)] = (LB, UB)

    return df_copy, bounds

rtns_df5, bounds = iqr(rtns_df5)
out_df = pd.concat([rtns_df5,rtns_df4Y],axis=1)

# Seperate the Features from Target
y_df = out_df.pop('returnsfwd')

# Normalise the Data Cross-Sectionally each month
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

standardized_X = out_df.groupby(out_df.index).apply(lambda x: scaler.fit_transform(x))

out_X = pd.DataFrame(np.vstack(standardized_X), index=out_df.index, columns=out_df.columns)

full_df = pd.concat([out_X,y_df],axis=1)
full_df.to_csv('full_df.csv')
```

A.2 OLS Model Code

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
import statsmodels.formula.api as smf
import scipy.stats

file_path = "/.../full_df.csv"
df = pd.read_csv(file_path,
                 parse_dates=['month'],
                 index_col=['month'])
df.sort_values(by=['month'], inplace=True)

df = df["2005-01-01":"2015-12-01"]

# Set up Fama-Macbeth Regression
def ols_coef(x,formula):
    return smf.ols(formula,data=x).fit().params

def fm_summary(p):
    uni_sum = p.describe().T
    uni_sum['std_error'] = uni_sum['std']/np.sqrt(uni_sum['count'])
    uni_sum['tstat'] = uni_sum['mean']/uni_sum['std_error']
    return uni_sum[['mean','std_error','tstat']]

# define dependent and independent variables
dependent_variable = 'returnsfwd'

explanatory_variables = ['bvtm', …
                         'retvar12']

# Univariate OLS Models
results_dict = {}

for var in explanatory_variables:
    formula = f"{dependent_variable} ~ 1 + {var}"
    model_results = df.groupby('month').apply(ols_coef, formula)
    results_dict[var] = model_results

# Fama-Macbeth Summary
univar_df = pd.DataFrame(columns=['mean','std_error','tstat'])

for var, model_results in results_dict.items():
    summary_stats = fm_summary(model_results)
    univar_df.loc[var] = summary_stats.iloc[1]

# Create a df of the significant features

# Calculate the critical t value for DOF = number of monthly observations
dof = len(df.index.unique())-1
confidence = 0.05

t_critical = scipy.stats.t.ppf(q=1-confidence/2,df=dof)
```

```
# Check which variables are significant
sig_vars = []

for column, row in univar_df.iterrows():
    t_stat = row['tstat']
    if abs(t_stat) >= t_critical:
        sig_vars.append(column)

sig_df = df[sig_vars]


# Check correlation of significant features
sig_df_corr = sig_df.corr()

# Drop lowest t-stat variables with correlation > 0.7
to_drop = ['mom12']
sig_df = sig_df.drop(to_drop, axis = 1).copy()


# Perform Multivariate Regression
y = dependent_variable
X = sig_df.columns.tolist()

all_variables = [y] + X
full_df = df[all_variables]

m_multivariate = (full_df.groupby('month')
                 .apply(ols_coef, f'{y} ~ 1 + {" + ".join(X)}'))

multivar_df = fm_summary(m_multivariate).iloc[1:]


# Create a df of the jointly significant features
dof = len(df.index.unique())-1
confidence = 0.05

t_critical = scipy.stats.t.ppf(q=1-confidence/2,df=dof)
joint_sig_vars = []

for column, row in multivar_df.iterrows():
    t_stat = row['tstat']
    if abs(t_stat) >= t_critical:
        joint_sig_vars.append(column)


# Perform Joint Multivariate Regression
y = dependent_variable
joint_X = joint_sig_df.columns.tolist()

joint_all_variables = [y] + joint_X
joint_full_df = df[joint_all_variables]

m_joint_multivariate = (joint_full_df.groupby('month')
                 .apply(ols_coef, f'{y} ~ 1 + {" + ".join(joint_X)}'))

joint_multivar_df = fm_summary(m_joint_multivariate).iloc[1:]
```

```
# Predictions - the below is repeated separately for the Joint Variables

# Use the mean value to calculate the expected return of the dataframe over time

sig_expected_return = (sig_df*multivar_df['mean']).sum(axis=1) + multivar_average_intercept

# Create a df of actual and forecast results

all_train_results_df = pd.DataFrame({'Forecast': sig_expected_return, 'Actual':
df['returnsfwd']})


# Out of sample Data

file_path = "/Users/joshu/miniconda3/envs/tf-gpu/Thesis/full_df.csv"
sig_test_df = pd.read_csv(file_path,
                parse_dates=['month'],
                index_col=['month'])
sig_test_df.sort_values(by=['month'], inplace=True)

sig_test_df = sig_test_df["2016-01-01":]

sig_y_test_df = sig_test_df['returnsfwd']

sig_test_df = sig_test_df[sig_vars]

# Predictions

sig_test_expected_return = (sig_test_df*multivar_df['mean']).sum(axis=1) +
multivar_average_intercept

all_test_results_df = pd.DataFrame({'Forecast': sig_test_expected_return, 'Actual':
sig_y_test_df})
```

A.3 Artificial Neural Network Code

```
# This code provides an example of a 3 hidden layer model for the reduced variables subset.
import pandas as pd
import numpy as np
import csv
import pydot
import tensorflow as tf
import statsmodels.api as sm
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.layers import Dropout
from tensorflow.keras.models import Sequential
from tensorflow.keras.initializers import he_normal
from keras.regularizers import l1,l2
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import KFold
from keras.callbacks import EarlyStopping
from keras.losses import Loss

# Import the Raw CSV File
file_path = "/…/full_df.csv"
df = pd.read_csv(file_path,
                 parse_dates=['month'],
                 index_col=['month'])
df.sort_values(by=['month'], inplace=True)

# Import the df for Jointly significant variables
# Train

file_path = "/…/joint_train_df.csv"
sig_train_df = pd.read_csv(file_path,
                 parse_dates=['month'],
                 index_col=['month'])
sig_train_df.sort_values(by=['month'], inplace=True)

# Test
file_path = "/…/joint_test_df.csv"
sig_test_df = pd.read_csv(file_path,
                 parse_dates=['month'],
                 index_col=['month'])
sig_test_df.sort_values(by=['month'], inplace=True)

# Split the dataset
train_df = df["2005-01-01":"2015-12-01"]
test_df = df["2016-01-01":]

# Split features from labels
train_feat = sig_train_df["2005-01-01":"2015-12-01"]
test_feat = sig_test_df
train_labels = train_df.pop('returnsfwd')
test_labels = test_df.pop('returnsfwd')
X_train, y_train = train_feat, train_labels
```

```
# Build the model - allows for various structures - this is a 3 layer model

def train_ann(X_train, y_train, X_val, y_val, nodes, regularizer, learning_rate, epochs,
batch_size, seed):
    tf.random.set_seed(seed)

    model = keras.Sequential([
        keras.layers.Input(shape=(X_train.shape[1],)),
        keras.layers.BatchNormalization(),
        keras.layers.Dense(nodes, activation='relu', kernel_regularizer=regularizer,
kernel_initializer=he_normal(seed=seed)),
        keras.layers.Dropout(0.1),
        keras.layers.BatchNormalization(),
        keras.layers.Dense(nodes // 2, activation='relu', kernel_regularizer=regularizer,
kernel_initializer=he_normal(seed=seed)),
        keras.layers.Dropout(0.1),
        keras.layers.BatchNormalization(),
        keras.layers.Dense(nodes // 4, activation='relu', kernel_regularizer=regularizer,
kernel_initializer=he_normal(seed=seed)),
        keras.layers.Dense(1)
    ])

    optimizer = keras.optimizers.Adam(learning_rate=learning_rate)

    model.compile(optimizer=optimizer, loss='mean_squared_error')

    early_stopping = keras.callbacks.EarlyStopping(monitor='val_loss', patience=10,
mode='min', verbose=1, restore_best_weights=True)

    history = model.fit(
     X_train, y_train,
        epochs=epochs,
        batch_size=batch_size,
        validation_data=(X_val, y_val),
        callbacks=[early_stopping],
        verbose=2
    )

    return model, history.history


# Define the parameters

node_counts = [8, 16, 32, 64, 128, 256]
regularizers = [None]
learning_rates = [0.00001]
epochs = [500]
batch_sizes = [64]
random_seeds = [1,2,3]

# define folds

n_splits = 5
```

```
# Model Training Loop
# Model trains for each random seed and fold
# With each training loop in- and out-of-sample predictions are made
# For each structure, the average of the forecasts from each random seed and fold combination
is saved

forecast_train_dict = {}
forecast_test_dict = {}

total_iterations = len(node_counts) * len(regularizers) * len(learning_rates) * len(epochs) *
len(batch_sizes)
iteration_count = 0

# Training Loop

for nodes in node_counts:
    for regularizer in regularizers:
        for learning_rate in learning_rates:
            for epoch in epochs:
                for batch_size in batch_sizes:

                    train_predictions_all_seeds_train = []
                    test_predictions_all_seeds_test = []

                    for seed in random_seeds:
                        kf = KFold(n_splits=n_splits, shuffle=True, random_state=seed)
                        train_predictions_seed_train = []
                        test_predictions_seed_test = []

                        X_train, y_train, X_test, y_test = train_feat, train_labels,
test_feat, test_labels

                        for train_idx, val_idx in kf.split(X_train):
                            X_train_fold, X_val_fold = X_train.iloc[train_idx],
X_train.iloc[val_idx]
                            y_train_fold, y_val_fold = y_train.iloc[train_idx],
y_train.iloc[val_idx]

                            model, history = train_ann(X_train_fold, y_train_fold, X_val_fold,
y_val_fold, nodes, regularizer, learning_rate, epoch, batch_size, seed)

                            # Train predictions
                            train_pred_train = model.predict(X_train)
                            train_pred_test = model.predict(X_test)

                            train_predictions_seed_train.append(train_pred_train)
                            test_predictions_seed_test.append(train_pred_test)

                        # Save the predictions for each seed

train_predictions_all_seeds_train.append(np.mean(train_predictions_seed_train, axis=0))

test_predictions_all_seeds_test.append(np.mean(test_predictions_seed_test, axis=0))

                    # Average predictions across seeds
                    avg_train_predictions_train = np.mean(train_predictions_all_seeds_train,
axis=0)
```

```
                    avg_test_predictions_test = np.mean(test_predictions_all_seeds_test,
axis=0)

                    model_name = f"Nodes_{nodes}_LR_{learning_rate}"

                    forecast_train_dict[model_name] = avg_train_predictions_train
                    forecast_test_dict[model_name] = avg_test_predictions_test

                    iteration_count += 1

                    print(f"Iteration {iteration_count}/{total_iterations}")

Forecast_Train_df = pd.DataFrame(index = train_labels.index)
Forecast_Test_df = pd.DataFrame(index = test_labels.index)

for model_name, forecast_values in forecast_train_dict.items():
    Forecast_Train_df[model_name] = forecast_values.flatten()

for model_name, forecast_values in forecast_test_dict.items():
    Forecast_Test_df[model_name] = forecast_values.flatten()

Forecast_Train_df.to_csv("Forecast_Train_joint_predictions_3D.csv")
Forecast_Test_df.to_csv("Forecast_Test_joint_predictions_3D.csv")
```

A.4 Performance Metrics

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import csv
import pydot

# Import the Raw CSV File
file_path = "/…/full_df.csv"
df = pd.read_csv(file_path,
                 parse_dates=['month'],
                 index_col=['month'])
df.sort_values(by=['month'], inplace=True)

# Import the df for Reduced significant variables
# Train
file_path = "/…/sig_df.csv"
sig_train_df = pd.read_csv(file_path,
                 parse_dates=['month'],
                 index_col=['month'])
sig_train_df.sort_values(by=['month'], inplace=True)

# Test
file_path = "/…/sig_test_df.csv"
sig_test_df = pd.read_csv(file_path,
                 parse_dates=['month'],
                 index_col=['month'])
sig_test_df.sort_values(by=['month'], inplace=True)

# Split the dataset
train_df = df["2005-01-01":"2015-12-01"]
test_df = df["2016-01-01":]

# Split features from labels
train_feat = sig_train_df["2005-01-01":"2015-12-01"]
test_feat = sig_test_df

train_labels = train_df.pop('returnsfwd')
test_labels = test_df.pop('returnsfwd')

X_train, y_train = train_feat, train_labels


forecast_train_df = pd.read_csv("/…/Forecast_Train_joint_predictions_3D.csv",
parse_dates=['month'], index_col='month')
forecast_test_df = pd.read_csv("/…/Forecast_Test_joint_predictions_3D.csv",
parse_dates=['month'], index_col='month')

forecast_train_df.index = pd.to_datetime(forecast_train_df.index)
forecast_test_df.index = pd.to_datetime(forecast_test_df.index)


forecast_train_df.columns = forecast_train_df.columns.str.replace('Nodes_',
'').str.replace('_LR_1e-05', '')
forecast_test_df.columns = forecast_test_df.columns.str.replace('Nodes_',
'').str.replace('_LR_1e-05', '')
```

```
# Define functions to calculate the desired metrics

# Regression Output
def regression_metrics(df, actual_column, forecast_column):
    X = sm.add_constant(df[forecast_column])
    y = df[actual_column]

    model = sm.OLS(y, X).fit()

    r2 = model.rsquared
    intercept = model.params['const']
    coefficient = model.params[forecast_column]
    t_stat = model.tvalues[forecast_column]
    correlation = df[[actual_column, forecast_column]].corr().iloc[0, 1]

    # Adjusted R2
    n = len(y)
    k = len(X.columns)

    adj_r2 = 1 - (1 - r2) * ((n - 1) / (n - k - 1))

    # Demeaned R2
    demeaned_r2 = 1 - ((y - model.predict(X)).pow(2).sum() / (y.pow(2).sum()))

    return {
        'R2': r2,
        'Adjusted R2': adj_r2,
        'Demeaned R2': demeaned_r2,
        'Intercept': intercept,
        'Coefficient': coefficient,
        'T-stat': t_stat,
        'Correlation': correlation}

# Average Slope Coefficient and T-Stat

def avg_regression_metrics(df, actual_column, forecast_column):

    df['Month'] = df.index.to_period("M")

    def calc_regression_metrics(group):

        X = sm.add_constant(group[forecast_column])
        y = group[actual_column]

        model = sm.OLS(y, X).fit()

        beta = model.params[forecast_column]

        return pd.Series({'Slope Coefficient': beta, 'Slope T-stat': np.nan})

    metrics_output = df.groupby('Month').apply(calc_regression_metrics)
    metrics = df.groupby('Month').apply(calc_regression_metrics).mean()

    # slope t-stat

    mean_beta = metrics['Slope Coefficient']
    std_beta = metrics_output['Slope Coefficient'].std()
```

```
    num_months = len(df['Month'].unique())
    T_beta = np.sqrt(num_months)

    metrics['Slope T-stat'] = mean_beta / (std_beta/T_beta)

    # Cumulative Slope Coefficient
    metrics['Cumulative Slope Coefficient'] = (1 + metrics_output['Slope
Coefficient']).cumprod().iloc[-1]

    return metrics

# Information Coefficient

def average_IC(df, actual_column, forecast_column):

    df['Month'] = df.index.to_period("M")

    monthly_ic = df.groupby('Month').apply(lambda x:
x[forecast_column].corr(x[actual_column]))

    # Average IC
    average_ic = monthly_ic.mean()

    # Cumulative IC
    cumulative_ic = (1 + monthly_ic).cumprod().iloc[-1]

    # Standard Deviation
    std_ic = monthly_ic.std()

    # Qian and Hua IR
    qh_ir = average_ic/std_ic

    return {'Average IC': average_ic,
            'Cumulative IC': cumulative_ic,
            'Standard Deviation': std_ic,
            'Qian and Hua IR': qh_ir}

df_train = forecast_train_df
df_test = forecast_test_df

all_results = []

for column in df_train.columns:
    df_train_column = pd.DataFrame({'Actual': train_labels, 'Forecast': df_train[column]})
    df_test_column = pd.DataFrame({'Actual': test_labels, 'Forecast': df_test[column]})

    regression_metric_train = regression_metrics(df_train_column, 'Actual', 'Forecast')
    avg_regression_metric_train = avg_regression_metrics(df_train_column, 'Actual',
'Forecast')
    average_ic_train = average_IC(df_train_column, 'Actual', 'Forecast')

    regression_metric_test = regression_metrics(df_test_column, 'Actual', 'Forecast')
    avg_regression_metric_test = avg_regression_metrics(df_test_column, 'Actual', 'Forecast')
    average_ic_test = average_IC(df_test_column, 'Actual', 'Forecast')

    metrics_train = {
```

```
        'Period': 'Train Period',
        'Column': column,
        **regression_metric_train,
        **avg_regression_metric_train,
        **average_ic_train
    }

    metrics_test = {
        'Period': 'Test Period',
        'Column': column,
        **regression_metric_test,
        **avg_regression_metric_test,
        **average_ic_test
    }

    all_results.append(metrics_train)
    all_results.append(metrics_test)

# Create a DataFrame from the results
performance_df = pd.DataFrame(all_results)


# Calculate Cumulative IC
def cum_metrics(df, actual_column, forecast_column):

    df['Month'] = df.index.to_period("M")

    # Information Coefficient
    monthly_ic = df.groupby('Month').apply(lambda x:
x[actual_column].corr(x[forecast_column]))

    cumulative_ic = (1 + monthly_ic).cumprod()

    return pd.DataFrame({f'{column}': cumulative_ic.values}, index=cumulative_ic.index)


df_train = forecast_train_df
df_test = forecast_test_df

cum_results_train = []
cum_results_test = []

for column in df_train.columns:
    df_train_column = pd.DataFrame({'Actual': train_labels, 'Forecast': df_train[column]})
    df_test_column = pd.DataFrame({'Actual': test_labels, 'Forecast': df_test[column]})

    cum_metric_train = cum_metrics(df_train_column, 'Actual', 'Forecast')
    cum_metric_test = cum_metrics(df_test_column, 'Actual', 'Forecast')

    cum_results_train.append(cum_metric_train)
    cum_results_test.append(cum_metric_test)

cum_df_train = pd.concat(cum_results_train, axis = 1)
cum_df_test = pd.concat(cum_results_test, axis = 1)

# print(cum_df_train)
# print(cum_df_test)
```

```
plt.figure(figsize=(10, 6))
for column in df_train.columns:
    col_name = column
    plt.plot(cum_df_train.index.to_timestamp(), cum_df_train[col_name], label=col_name)

plt.title('Cumulative IC - Train Period')
plt.xlabel('Date')
plt.ylabel('Cumulative IC')
plt.axhline(y=1, color='black', linewidth=0.5)
plt.legend()
plt.show()

plt.figure(figsize=(10, 6))
for column in df_test.columns:
    col_name = column
    plt.plot(cum_df_test.index.to_timestamp(), cum_df_test[col_name], label=col_name)

#plt.title('Cumulative IC - Test Period')
plt.xlabel('Date')
plt.ylabel('Cumulative IC')
plt.axhline(y=1, color='black', linewidth=0.5)
plt.legend()
plt.show()

#print(plt.rcParams['axes.prop_cycle'].by_key()['color'])


# Graph of portfolio constructed by taking top forecasts
def portfolio_returns(df, actual_column, forecast_column, percentile = 25):

    df_sorted = df.groupby(df.index).apply(lambda x: x.sort_values(by=forecast_column,
ascending=False))

    top_percent = df_sorted.groupby(df_sorted.index).apply(lambda x: x.head(int(len(x) *
(percentile / 100))))
    bottom_percent = df_sorted.groupby(df_sorted.index).apply(lambda x: x.tail(int(len(x) *
(percentile / 100))))

    portfolio_returns_top = top_percent.groupby(top_percent.index)[actual_column].mean()
    portfolio_returns_bottom =
bottom_percent.groupby(bottom_percent.index)[actual_column].mean()

    # Calculate the cumulative payoff
    cumulative_payoff_top = (1 + portfolio_returns_top).cumprod()
    cumulative_payoff_bottom = (1 + portfolio_returns_bottom).cumprod()

    # Create a time series DataFrame for the cumulative payoff
    time_series_df = pd.DataFrame({
        f'Cumulative_Payoff_Top_{column}': cumulative_payoff_top,
        f'Cumulative_Payoff_Bottom_{column}': cumulative_payoff_bottom,
        f'Portfolio_Returns_Top_{column}': portfolio_returns_top,
        f'Portfolio_Returns_Bottom_{column}': portfolio_returns_bottom
    })

    time_series_df.index = df.index.unique()

    return time_series_df
```

```
def benchmark_returns(df, actual_column):

    benchmark_returns = df.groupby(df.index)[actual_column].mean()

    cumulative_payoff = (1 + benchmark_returns).cumprod()

    benchmark_df = pd.DataFrame({
        f'Cumulative_Benchmark_Payoff': cumulative_payoff,
        f'Benchmark_Returns': benchmark_returns
    })

    return benchmark_df
df_train = forecast_train_df
df_test = forecast_test_df

df_train_labels = pd.DataFrame({'Actual': train_labels})
df_test_labels = pd.DataFrame({'Actual': test_labels})

portfolio_results_train = []
portfolio_results_test = []

# Forecasts

for column in df_train.columns:
    df_train_column = pd.DataFrame({'Actual': train_labels, 'Forecast': df_train[column]})
    df_test_column = pd.DataFrame({'Actual': test_labels, 'Forecast': df_test[column]})

    return_train = portfolio_returns(df_train_column, 'Actual', 'Forecast', percentile = 25)
    return_test = portfolio_returns(df_test_column, 'Actual', 'Forecast', percentile = 25)

    portfolio_results_train.append(return_train)
    portfolio_results_test.append(return_test)

portfolio_results_train_df = pd.concat(portfolio_results_train, axis = 1)
portfolio_results_test_df = pd.concat(portfolio_results_test, axis = 1)

unique_dates_train = portfolio_results_train_df.index.get_level_values(0).unique()
unique_dates_test = portfolio_results_test_df.index.get_level_values(0).unique()

# Benchmark

benchmark_results_train = []
benchmark_results_test = []

benchmark_return_train = benchmark_returns(df_train_labels, 'Actual')
benchmark_return_test = benchmark_returns(df_test_labels, 'Actual')

benchmark_results_train.append(benchmark_return_train)
benchmark_results_test.append(benchmark_return_test)

benchmark_results_train_df = pd.concat(benchmark_results_train, axis = 1)
benchmark_results_test_df = pd.concat(benchmark_results_test, axis = 1)

# Train
```

```
plt.figure(figsize=(10, 6))
plt.plot(benchmark_results_train_df.index,
benchmark_results_train_df['Cumulative_Benchmark_Payoff'], color='black', linestyle='-',
label='Benchmark')
for i, column in enumerate(df_train.columns):
    col_name_top = f'Cumulative_Payoff_Top_{column}'
    col_name_bottom = f'Cumulative_Payoff_Bottom_{column}'

    colours = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b']
    line_top = '-'
    line_bottom = '--'

    plt.plot(unique_dates_train, portfolio_results_train_df[col_name_top], color=colours[i],
linestyle=line_top, label=f'{column}')
    plt.plot(unique_dates_train, portfolio_results_train_df[col_name_bottom],
color=colours[i], linestyle=line_bottom)

plt.title('Cumulative Payoff - Train Period')
plt.xlabel('Date')
plt.ylabel('Cumulative Payoff')
plt.axhline(y=1, color='black', linewidth=0.5)
plt.legend()
plt.show()

# Test

plt.figure(figsize=(10, 6))
plt.plot(benchmark_results_test_df.index,
benchmark_results_test_df['Cumulative_Benchmark_Payoff'], color='black', linestyle='-',
label='Benchmark')
for i, column in enumerate(df_train.columns):
    col_name_top = f'Cumulative_Payoff_Top_{column}'
    col_name_bottom = f'Cumulative_Payoff_Bottom_{column}'

    colours = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b']
    line_top = '-'
    line_bottom = '--'

    plt.plot(unique_dates_test, portfolio_results_test_df[col_name_top], color=colours[i],
linestyle=line_top, label=f'{column}')
    plt.plot(unique_dates_test, portfolio_results_test_df[col_name_bottom], color=colours[i],
linestyle=line_bottom)

#plt.title('Cumulative Payoff - Test Period')
plt.xlabel('Date')
plt.ylabel('Cumulative Payoff')
plt.axhline(y=1, color='black', linewidth=0.5)
plt.legend(loc='upper left', handlelength=2)
plt.show()
```

Appendix B:

Table B.1: Ranking of Neural Network Model Performance within Hidden Layer Counts

Panels A to E show the ranking for performance metrics of each node count variant across different hidden layer counts for the All Variables and Reduced Variables subsets.

Panel A.1 – Single Layer – All Variables

	Rank					
Nodes in First Hidden Layer	**8**	**16**	**32**	**64**	**128**	**256**
Demeaned R2	2	4	1	6	5	3
T-stat	5	3	6	1	2	4
Average IC	4	6	5	2	3	1
Cumulative IC	4	6	5	2	3	1
Qian and Hua IR	5	6	4	2	3	1
Average Slope Coefficient	5	6	4	1	3	2
Slope T-stat	3	6	5	1	4	2
Annualized Return – Top Quartile	2	6	4	3	5	1
Returns/Risk – Top Quartile	2	6	4	3	5	1
Annualized Return – Bottom Quartile	5	3	4	1	6	2
Returns/Risk – Bottom Quartile	5	3	4	1	6	2
Average	**3.8**	**5.0**	**4.2**	**2.1**	**4.1**	**1.8**
Rank	**3**	**6**	**5**	**2**	**4**	**1**

Panel A.2 – Single Layer – Reduced Variables

	Rank					
Nodes in First Hidden Layer	**8**	**16**	**32**	**64**	**128**	**256**
Demeaned R2	5	2	4	1	6	3
T-stat	3	5	1	6	2	4
Average IC	3	6	1	5	2	4
Cumulative IC	3	6	1	5	2	4
Qian and Hua IR	3	6	1	5	2	4
Average Slope Coefficient	4	6	1	5	2	3
Slope T-stat	3	6	1	5	2	4
Annualized Return – Top Quartile	3	5	1	6	2	4
Returns/Risk – Top Quartile	3	4	1	6	2	5
Annualized Return – Bottom Quartile	3	4	2	6	1	5
Returns/Risk – Bottom Quartile	3	4	2	6	1	5
Average	**3.3**	**4.9**	**1.5**	**5.1**	**2.2**	**4.1**
Rank	**3**	**5**	**1**	**6**	**2**	**4**

Panel B.1 – Two Layer – All Variables

	Rank					
Nodes in First Hidden Layer	**8**	**16**	**32**	**64**	**128**	**256**
Demeaned R2	2		3	5	3	1
T-stat	4		3	1	2	5
Average IC	4		1	2	3	5
Cumulative IC	4		1	2	3	5
Qian and Hua IR	4		1	2	3	5
Average Slope Coefficient	5		3	2	1	4
Slope T-stat	4		1	2	3	5
Annualized Return – Top Quartile	2		1	3	4	5
Returns/Risk – Top Quartile	2		1	3	4	5
Annualized Return – Bottom Quartile	5		1	3	2	4
Returns/Risk – Bottom Quartile	5		1	2	3	4
Average	**3.7**		**1.5**	**2.5**	**2.8**	**4.4**
Rank	**4**		**1**	**2**	**3**	**5**

Panel B.2 – Two Layer – Reduced Variables

	Rank					
Nodes in First Hidden Layer	**8**	**16**	**32**	**64**	**128**	**256**
Demeaned R2	2		2	5	2	1
T-stat	3		1	4	2	5
Average IC	5		3	1	4	2
Cumulative IC	5		3	1	4	2
Qian and Hua IR	5		3	1	4	2
Average Slope Coefficient	5		2	1	4	2
Slope T-stat	5		3	1	4	2
Annualized Return – Top Quartile	2		5	1	4	3
Returns/Risk – Top Quartile	2		5	1	4	3
Annualized Return – Bottom Quartile	4		2	1	5	3
Returns/Risk – Bottom Quartile	4		2	1	5	3
Average	**3.8**		**2.8**	**1.6**	**3.8**	**2.5**
Rank	**4**		**3**	**1**	**4**	**2**

Panel C.1 – Three Layer – All Variables

	Rank					
Nodes in First Hidden Layer	**8**	**16**	**32**	**64**	**128**	**256**
Demeaned R2	1			3	4	2
T-stat	4			1	2	3
Average IC	4			1	2	3
Cumulative IC	4			1	2	3
Qian and Hua IR	4			1	2	3
Average Slope Coefficient	3			1	2	4
Slope T-stat	4			1	2	3
Annualized Return – Top Quartile	3			1	2	4
Returns/Risk – Top Quartile	3			1	2	4
Annualized Return – Bottom Quartile	4			2	1	3
Returns/Risk – Bottom Quartile	4			2	1	3
Average	**3.5**			**1.4**	**2.0**	**3.2**
Rank	**4**			**1**	**2**	**3**

Panel C.2 – Three Layer – Reduced Variables

	Rank					
Nodes in First Hidden Layer	**8**	**16**	**32**	**64**	**128**	**256**
Demeaned R2		2		1	2	
T-stat		2		1	3	
Average IC		2		1	3	
Cumulative IC		2		1	3	
Qian and Hua IR		2		1	3	
Average Slope Coefficient		1		2	3	
Slope T-stat		2		1	3	
Annualized Return – Top Quartile		2		1	3	
Returns/Risk – Top Quartile		2		1	3	
Annualized Return – Bottom Quartile		2		1	3	
Returns/Risk – Bottom Quartile		2		1	3	
Average		**1.9**		**1.1**	**2.9**	
Rank		**2**		**1**	**3**	

Panel D.1 – Four Layer – All Variables

	Rank					
Nodes in First Hidden Layer	**16**	**32**	**64**	**128**	**256**	**512**
Demeaned R2		3	2	5	3	1
T-stat		3	4	1	2	5
Average IC		3	1	5	4	2
Cumulative IC		3	1	5	4	2
Qian and Hua IR		3	1	5	4	2
Average Slope Coefficient		5	1	4	3	2
Slope T-stat		2	1	5	3	4
Annualized Return – Top Quartile		2	1	4	3	5
Returns/Risk – Top Quartile		2	1	3	4	5
Annualized Return – Bottom Quartile		4	1	5	2	3
Returns/Risk – Bottom Quartile		4	1	5	2	3
Average		**3.1**	**1.4**	**4.3**	**3.1**	**3.1**
Rank		**2**	**1**	**5**	**2**	**2**

Panel D.2 – Four Layer – Reduced Variables

	Rank					
Nodes in First Hidden Layer	**16**	**32**	**64**	**128**	**256**	**512**
Demeaned R2	1		4	2	3	
T-stat	4		2	1	3	
Average IC	4		2	1	3	
Cumulative IC	4		2	1	3	
Qian and Hua IR	4		2	1	3	
Average Slope Coefficient	2		4	1	3	
Slope T-stat	4		2	1	3	
Annualized Return – Top Quartile	4		2	1	3	
Returns/Risk – Top Quartile	4		2	1	3	
Annualized Return – Bottom Quartile	2		4	1	3	
Returns/Risk – Bottom Quartile	1		4	2	3	
Average	**3.1**		**2.7**	**1.2**	**3.0**	
Rank	**4**		**2**	**1**	**3**	

Panel E.1 – Five Layer – All Variables

	Rank					
Nodes in First Hidden Layer	**32**	**64**	**128**	**256**	**512**	**1024**
Demeaned R2			3	4	2	1
T-stat			1	2	3	4
Average IC			1	2	3	4
Cumulative IC			1	2	3	4
Qian and Hua IR			1	2	3	4
Average Slope Coefficient			2	1	3	4
Slope T-stat			2	1	3	4
Annualized Return – Top Quartile			4	1	2	3
Returns/Risk – Top Quartile			4	1	2	3
Annualized Return – Bottom Quartile			1	3	2	4
Returns/Risk – Bottom Quartile			1	3	2	4
Average			**1.9**	**2.0**	**2.5**	**3.5**
Rank			**1**	**2**	**3**	**4**

Panel E.2 – Five Layer – Reduced Variables

	Rank					
Nodes in First Hidden Layer	**32**	**64**	**128**	**256**	**512**	**1024**
Demeaned R2		1	2	2	4	4
T-stat		5	3	4	1	2
Average IC		5	3	4	1	2
Cumulative IC		5	3	4	2	1
Qian and Hua IR		5	3	4	1	2
Average Slope Coefficient		5	3	4	1	2
Slope T-stat		5	3	4	1	2
Annualized Return – Top Quartile		4	1	5	2	3
Returns/Risk – Top Quartile		4	1	5	2	3
Annualized Return – Bottom Quartile		5	4	1	2	3
Returns/Risk – Bottom Quartile		5	4	1	2	3
Average		**4.5**	**2.7**	**3.5**	**1.7**	**2.5**
Rank		**5**	**3**	**4**	**1**	**2**

Table B.2 – Rankings of Best Models for each Hidden Layer Count

Panels A and B show the rankings across all performance metrics of the top models from each hidden layer count relative to each other for both the All Variables and Reduced Variables models.

Panel A – All Variables

			Rank		
Hidden Layers	**1**	**2**	**3**	**4**	**5**
Nodes in First Hidden Layer	**256**	**32**	**64**	**64**	**128**
Demeaned R2	1	4	3	2	5
T-stat	5	3	1	4	2
Average IC	2	5	1	4	3
Cumulative IC	2	5	1	3	3
Qian and Hua IR	2	5	1	4	3
Average Slope Coefficient	4	5	1	2	3
Slope T-stat	4	5	1	2	3
Annualized Return – Top Quartile	4	2	1	3	5
Returns/Risk – Top Quartile	4	2	1	3	5
Annualized Return – Bottom Quartile	4	5	3	2	1
Returns/Risk – Bottom Quartile	4	5	3	2	1
Average	**3.3**	**4.2**	**1.5**	**2.8**	**3.1**
Rank	**4**	**5**	**1**	**2**	**3**

Panel B – Reduced Variables

			Rank		
Hidden Layers	**1**	**2**	**3**	**4**	**5**
Nodes in First Hidden Layer	**32**	**64**	**64**	**128**	**512**
Demeaned R2	3	5	1	2	4
T-stat	3	5	1	2	4
Average IC	4	2	1	2	5
Cumulative IC	4	3	1	2	5
Qian and Hua IR	4	3	1	2	5
Average Slope Coefficient	4	3	1	2	5
Slope T-stat	3	4	1	2	5
Annualized Return – Top Quartile	3	2	1	4	5
Returns/Risk – Top Quartile	3	2	1	4	5
Annualized Return – Bottom Quartile	3	2	1	4	5
Returns/Risk – Bottom Quartile	3	2	1	4	5
Average	**3.4**	**3.0**	**1.0**	**2.7**	**4.8**
Rank	**4**	**3**	**1**	**2**	**5**