Automated Machine Learning for Trend Prediction

Adam Lewison
lwsada002@myuct.ac.za
University Of Cape Town
Cape Town, South Africa

ABSTRACT

Machine learning techniques and algorithms are used in almost every application domain such as image recognition, object detection, and financial applications. Building a high-quality machine learning model relies on human expertise. Automated Machine Learning (AutoML) is the process of automating different stages of the Machine Learning Pipeline. This automation has been shown to produce superior Machine Learning Models. This research includes experimentation of a variety of Evolutionary Algorithms used as an optimisation technique to solve a Black Box Optimization Problem - specifically selecting both a Machine Learning Algorithm and its Hyper-Parameters, this constitutes an Automated Machine Learning method known as *Combined Algorithm And Hyper-parameter Selection* (CASH). The application domain over which this was tested is predicting future trends given raw time series data of financial equities.

KEYWORDS

Automated Machine Learning, Deep Learning, Time Series Trend Prediction, Evolutionary Algorithms

1 INTRODUCTION

Many people in the world today would be better off if they were able to make accurate predictions about future values. This idea has been formalised as Time Series Forecasting. A time series is a sequence of real-valued observations - for example: the historical prices of a particular share. Due to the importance of time series forecasting in many areas: a lot of research has gone into the development of statistical models that are able to outperform its predecessor on a time series forecasting task [1].

This paper will deal with a sub-section of time series forecasting known as trend prediction. In trend prediction we aim to forecast the future trend i.e. the upward or downward pattern instead of an exact future value. In many instances it is more valuable business intelligence to know the future trend of a time series rather than to forecast a future point-value of the time series [25].

Originally, Time Series Trend Prediction methods included Hidden Markov Models and multi-step ahead prediction. However, underlying the raw data of a time series is a complex data generating system. This system is the result of of uncertain behaviours and dynamic causalities of the underlying multivariate real-world processes [13]. This results in time series exhibiting characteristics of uncertainties, multivariate correlation, non-linearity and chaotic fluctuation [3]. It was shown that machine learning methods are more suitable to adequately model such sytems [3]. Artifical Neural Networks (ANNs) and State Vector Machines (SVMs) have been widely applied for time series prediction in finance, for example in stock price prediction [12].

Due to the superior results of Machine Learning Models - the study of Machine Learning including Deep Learning has become increasingly popular among researchers and its application is constantly expanding and of great value to businesses.

Building a high-quality machine learning model relies on human expertise. The human expert (usually a data scientist) will approach a machine learning problem as outlined in Figure 1. The decisions of the data scientist in each of these steps affect the performance and quality of the developed model [7]. Furthermore, users of machine learning tools are often non-experts who require off-the-shelf solutions [7]. The automation of different steps of the machine learning pipeline reduces the human effort necessary for applying machine learning, improves the performance of machine learning algorithms as well as improves the reproducibility and fairness of scientific studies [6]. Automated Machine Learning (AutoML) is beneficial for the ML expert due to the automation of tasks like hyperparameter optimization (HPO) and also for the Domain expert who is now able to build their own ML pipeline without a data scientist.

In the experimentation documented in paper, the author employed AutoML in order to make trend predictions on given raw time series data of financial equities. A particularly significant part of the Machine Learning Pipe-Line is what is sometimes refereed to as *Model Generation* as depicted in 1. In the *Model Generation* stage the human expert has to make decisions about which Machine Learning Algorithm to use along with its Hyper-Parameters. We used AutoML to automate this process in what is known as Combined Algorithm and Hyper-Parameter Selection (CASH).

Indeed, the search space that this problem presents is extremely large due to the vast number of machine learning algorithms to choose from as well as the many Hyper-Parameters associated which each model. Moreover, to test each and every combination over this search space is a relatively expensive computation . It is for these reasons that it is of the utmost importance to have an effective optimisation technique when attempting to minimise the expensive objective function over the large search space. The research of this paper attempts to find the right optimization technique for this task. After a thorough examination of the available literature, it appeared that a category of meta-heuristic optimization techniques based on evolutionary theory known as *Evolutionary Algorithms* show good potential for this task [27]. The objective of this research is to compare the most popular [1] evolution based optimisation techniques with each other and with a chosen benchmark of *Random Search*.

This paper will first explore the previous work which has showed that machine learning is a more effective method on time series forecasting than its substitute (Section 2.2) and the previous work on using machine learning methodology for trend prediction. Secondly,

[1] Albeit, An Evolutionary Algorithm called Particle Swarm Optimisation could not be included in this research. See Section 6.1

this paper will explore the rise of Automated Machine Learning in general as an improvement on manual machine learning (Section 2.3) and discuss what there . The author will then elucidate each step of the experimentation conducted in this research. Namely: Which Data-Sets were used; how the data was processed from raw time-series to trend sequences; the search space over which the objective function will be optimized, including which machine learning models are under consideration and their respective hyper-parameters; the optimization techniques which are being tested in this experiment and finally how the results were evaluated. The author will discuss the results of the output of the experiments conducted. And finally, the author will conclude with a brief discussion of the achievements of this research as well as a discussion of the limitations that were presented and future work that has become apparently worthwhile by this research.

2 RELATED WORK

2.1 Raw Time Series To Trend Sequence Conversion

The work of Keogh et al. presented four algorithms to segment raw time series data into a piece-wise linear function. [15]. These four algorithms are: sliding windows, bottom up, top down as well as a novel algorithm, The Sliding Window and Bottom-up (SWAB) algorithm. For each of these four algorithms, that paper brought forward two methods for approximating a straight line, namely: linear interpolation and linear regression. Further details of how this methodology was implemented in our research is found in Section 3.2.

2.2 Machine Learning for Trend Prediction

An essential area of the study of time series is an area called *forecasting*. Time Series Forecasting entails the estimation of future values of a time based sequence of historical data (time series) [1]. There is a wide variety of research demonstrating the efficacy of the use of Machine Learning methods for time series *forecasting* [1, 18, 19, 25].

The use of machine learning methodology for time series *trend prediction* is less commonplace among the literature. However, there is research available in this area. For example, in recent work [18] demonstrated superior results in time series trend prediction using a novel hybrid neural network for trend learning.

2.3 Automated Machine Learning

As is the case with all machine learning models, the performance depends on the selection of the right model architecture and training algorithm [30]. Hence, automating and optimising these choices has shown to provide superior results in time series problems [17].

As described in [31], Automated Machine Learning (AutoML) has been evolving for a number of decades. As early as the 1990s, solutions were available for Hyper-Parameter Optimisation for selected classification algorithms using the brute-force [21] grid search technique. This was quickly improved using a greedy best-first approach adaptation of grid search [16]. By the early 2000s, the first approaches for automatic feature selection were introduced into AutoML literature [22]. In 2009 *Full model selection* [9] attempted to completely automate the full ML pipeline - that is: data preprocessing, feature selection and classification while optimizing the Hyper-Parameters of each step. A more detailed description of the ML pipeline is covered by [11]. Starting from 2011 Bayesian Optimisation for Hyper-Parameter tuning [2] as well as model selection [24]. The work of [24] developed *Auto-Weka* framework for Combined Algorithm And Hyper-parameter Selection (CASH). Automatic Feature engineering without domain knowledge was introduced into AutoML in 2015 [14]. In recent years there has been a primary research focus on Neural Architecture Search (NAS) which automates the stages of model generation and model evaluation, many papers have done an in-depth survey of NAS [8, 26]. There is also much research into automation of the rest of the ML pipeline [7, 11]. The rest of this paper will focus on the review of the research and development of the Combined Algorithm And Hyper-parameter Selection problem.

2.4 AutoML for Trend Prediction

This work along with that of Kouassi and Moodley [17] is unique in the employment of Automated Machine Learning Methodology in order to achieve superior accuracy in making predictions on trends. The recent research of Kouassi and Moodley [17] conclusively showed that employment of Automated Machine Learning Methodology on making predictions on time series trends provides better results that any previous work.

Their work included Bayesian Optimisation and Hyper-band (BOHB) as both an optimizer to run Hyper-Parameter Optimization separately on three Deep Learning Algorithms, namely: Multi-Layer Perceptron (MLP), Convolutional Neural Network (CNN) and Long short-term memory (LSTM). Additionally, their work used BOHB as a solver to optimize the Combined Algorithm and Hyper-Parameter Selection (CASH) problem using the aforementioned Deep Learning Algorithms in the search space. Their results, in general, showed a marginal improvement (lower Root Mean Sqaured Error) when using BOHB as a CASH solver as opposed to only a Hyper-Parameter Optimizer.

Bayesian Optimisation and Hyper-band (BOHB) was the only optimisation technique used in their research. Furthermore, no other literature was found on the use of Evolutionary Algorithms for AutoML for Trend Prediction.

2.5 Evolutionary Algorithms for AutoML

Even though no literature was found on the use of Evolutionary Algorithms for AutoML for Trend Prediction - there is vast research demonstrating the efficacy of Evolutionary Algorithms in AutoML. The literature compilation of Yao et al. provides many examples of these optimization techniques used for different AutoML problems [28].

In this research we will look more deeply at an evolutionary algorithm known as Differential Evolution

2.5.1 Differential Evolution. Differential Evolution (DE) is a type of Evolutionary Algorithm i.e. a nature-inspired black-box global optimization method.

As with all Evolutionary Algorithms: the basic structure of this algorithm is a *generation* of a certain size (*Population Size*) where

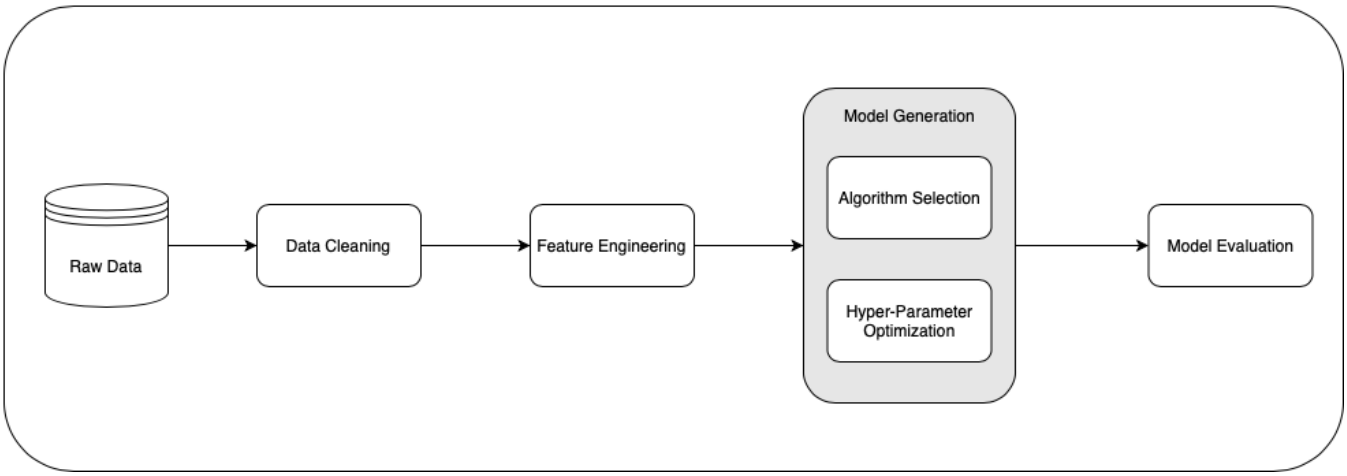

Figure 1: Machine Learning pipeline as described by [7, 11, 21, 31]. The goal of Combined Algorithm and Hyper-Parameter Selection is to automate the *Model Generation* step of this pipeline.

each member of the generation is a vector of parameter values. The following generation is supposed to *evolve* - i.e. improve (closer the the optimal position) - over its predecessor using some technique that is determined by the design of the Evolutionary Algorithm.

After an initial initialisation of vectors - Differential Evolution works through a simple cycle of the following stages

(1) Difference vector based mutation - Every vector in the current generation is combined with a another vector produced via a special operation known as *differential mutation* to form a *trial* vector.
(2) Crossover/Recombination - an operation in place to enhance the diversity of the population.
(3) Selection - selection of the "fittest" vectors to keep the population size constant

After Selection happens in step three - the selected vectors then mutate (step one) and the cycle repeats. [5]

Differential Evolution has demonstrated a lot of success and finished third at the First International Contest on Evolutionary Optimization in May 1996, and in that same contest - it was found that DE was the best evolutionary algorithm for solving the real-valued test function suite. Furthermore [5]

DE has been shown to competitively be able to determining the coefficients of the 33-dimensional Chebyshev polynomial. Chebyshev polynomials is considered a difficult task for a global optimization method.

More recently, DE showed to outperform SMAC - a Bayesian Optimization based Optimization technique - when used for Hyper-Parameter Optimization. [23]

3 DESIGN AND IMPLEMENTATION

3.1 Data-Sets

Experiments were carried out over three data-sets. The data-sets come from the category of Financial Time Series and record the closing price of a particular Equity across different markets around the globe. Each of the data-sets were obtained from *Yahoo Finance* [2]. Below is an overview of each data-set used:

- The New York Stock Exchange (NYSE) data-set contains 13964 data points of the closing price of the New York Stock Exchange. The range of this data is from 31/12/1965 to 22/06/2021. The New York Stock Exchange is based in New York, United State Of America.
- The SATRIX 40 (STX40) data-set contains 3454 data points of the daily closing price of the SATRIX 40 stock index. The range of this data is from 02/01/2008 to 22/01/2021. The SATRIX 40 is based on shares in the Johannesburg Stock Exchange - Johannebsurg, South Africa.
- The NASDAQ data-set contains 12706 data points of the daily closing price of the NASDAQ Stock Exchange. The range of this data is from 05/02/1971 to 22/01/2021

3.2 Data Pre-Processing

The goal of data pre-processing in this experiment is as follows

(1) Remove all null values from the time series
(2) Convert the the time series data to a sequence of trend lines
(3) Create input-output pairs of that can be used to train the machine learning model.
(4) Partition the data into a training set, a validation set and a test set.

In order to achieve Step 2 above, we made use of the work of Keogh et al. who conducted research on algorithms for Segmenting Time Series (See Section 2.1). We made use of the sliding window algorithm described in their research in combination with linear regression in order to create a trend sequence out of the time series.

Formally, given any time series X such that

$$X = [x_1, ..., X_T]$$

where x_t is a real-valued observation at time t. We convert time series data X into a trend sequence T by performing a linear

[2] https://finance.yahoo.com

regression on the first feasible subset of X. We expand the window one data point at a time until the Root Mean Squared Error passes some threshold *max_error*.

We record the gradient and the duration of each line created in order to have a trend sequence T

$$T = [< \ell_1, s_1 >, ..., < \ell_k, s_k >]$$

where ℓ_i is the number of data points x_t covered by trend i and s_i is the gradient of the line k.

The value of *max_error* was determined for each data-set by manual tuning and observing how well the sequence of trend line captures the trends of the time series.

After step two is completed: we now have a trend sequence T. We need to convert T to input-output pairs that are suitable to train a supervised machine learning algorithm.

To this end, we use another standard sliding window algorithm.

This sliding window process is explained well by way of diagram in Figure 2 below

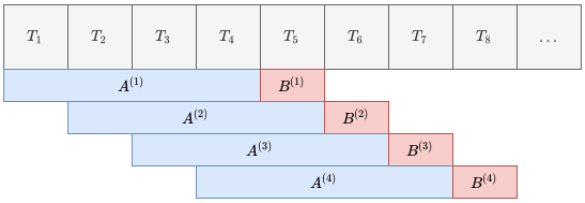

Figure 2: The Sliding Window algorithm responsible for the creation of input-output pairs. Both the inputs and outputs are in the form of matrices.

As can be seen in Figure 2 - the algorithm creates a matrix $A^{(t)}$ where

$$A^{(t)} = \begin{bmatrix} T_{t-3} \\ T_{t-2} \\ T_{t-1} \\ T_t \end{bmatrix} = \begin{bmatrix} \ell_{t-3} & s_{t-3} \\ \ell_{t-2} & s_{t-2} \\ \ell_{t-1} & s_{t-1} \\ \ell_t & s_t \end{bmatrix} \quad (1)$$

and is a matrix of four trend lines. Matrix $A^{(t)}$ is considered the input-feature and $B^{(t)}$ where

$$B^{(t)} = \begin{bmatrix} T_{t+1} \end{bmatrix} = \begin{bmatrix} \ell_{t+1} & s_{t+1} \end{bmatrix} \quad (2)$$

is the corresponding label.

The window then slides one position to the right and continues in this fashion until it has produced a complete set of input-output features S

$$S = [(A^{(1)}, B^{(1)}), \ldots, (A^{(n)}, B^{(n)})] \quad (3)$$

A window size of *four* was chosen.

Finally, we partition S into three data sets as shown in Figure 3

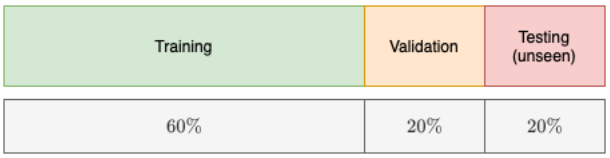

Figure 3: Partitioning ratios of data in training, validation and testing sets.

3.3 Machine Learning Algorithms

In our experimentation we consider two Machine Learning Algorithms: The Multi-Layer Perceptron (MLP) and A Long short-term memory (LSTM). Both the MLP and the LSTM fall into a more specific category known as Deep Neural Networks, however for the sake of consistency, we will refer to them in this research by the name of their parent category: Machine Learning Algorithms. The configuration of these algorithms used in our experimentation is outlined below:

- The MLP consists of N fully connected neural network layers, where $N \in [1, 5]$. Each layer has K neurons, where $K \in [4, 50]$. The *ReLu* function was used as the activation between the neurons of different layers. This activation function enables the neural-network to capture non-linear patterns. A dropout layer is prepended to the output layer to prevent overfitting. A fully connected NN layer is added to the end of the LSTM where the output of this layer is the next trend.
- A Long short-term memory (LSTM) - a type of artificial recurrent neural network (RNN) consisting of N LSTM layers, where $N \in [1, 5]$. Each layer has K neurons, where $K \in [10, 200]$. The *sigmoid* function was used as the activation between the neurons of different layers. This activation function enables the neural-network to capture non-linear patterns. The dropout rate, which helps to prevent overfitting is given as one of the Hyper-Parameters to be tuned. A fully connected NN layer is added to the end of the LSTM where the output of this layer is the next trend.

The dropout rate of each algorithm is one of {0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}.

The use case in our experimentation is supervised machine learning which means that each algorithm is trained using a set of labeled data. The training data partition (Figure 3) of S (Equation 3) is used to train the machine learning algorithms. Both the MLP and the LSTM are configured to take as input a 4×2 matrix and ouputs a 4×1 matrix in agreement with the data pre-processing stage.

The equally weighted average slope and duration mean square error (Equation 4) is then used as a loss function during training using the Adam Optimizer.

$$\frac{1}{2T} \sum_{i=1}^{T} ((\ell_i - \hat{\ell}_i)^2 + (s_i - \hat{s}_i)^2) \quad (4)$$

3.4 Search Space

The nature of the Combined Algorithm and Hyper-Parameter Selection problem dictates the structure of the search space. Namely the

search space will consist of the machine learning models mention above (Section 3.3) and their respective Hyper-Parameters (albeit a chosen subset of thereof). We were able to leverage off of manual experimentation in [17] in order to determine The machine learning models used in this paper. The respective Hyper-Parameters are described in Section 3.3

Hyper-Parameter	Value Type
Algorithm	Categorical
Number of LSTM Layers	Discrete
Number of Hidden Layers	Discrete
Number of Hidden neurons of layer_i	Discrete
Learning Rate	Continous
Dropout Rate	Discrete

Table 1: Summary of the Hyper-Parameters available in the search space

The learning rate of each algorithm is one of {0,0001, 0.001, 0.01, 0,1}.

Kouassi and Moodley also included a Convolutional Neural Network (CNN) in their search space however this was not employed in our work as it was evidenced over there that a CNN does not show to be significantly different to an MLP or LSTM in this problem domain however its inclusion would vastly increase the size of the search space.

3.5 Problem Optimization

In line with the objectives of this research, we will test various optimization techniques in order to minimize the loss function over the search space (outlined in Section 3.4). The following optimization techniques will be used:

- Random Search (Random). Random Search is a type of brute-force method to minimize any objective function. An implementation of Random Search was coded manually in Python in a simple way that makes use of Python's *random* class in order to select hyper-parameter configurations that are in the search space. Random Search is a popular choice as a benchmark in order to compare to other Optimization Techniques.
- Differential Evolution (DE) - The Python package *PyMoo*[3] was used in order to implement Differential Evolution. A more in depth analysis of the configuration of the Differential Algorithm for this experimentation is elaborated in Section 4
- Pattern Search (PS) - *PyMoo* was also used in order to implement Pattern Search. The standard configuration of Pattern Search was used as a factory setting Evolutionary Algorithm in order to use as a bench mark to compare to Differential Evolution.

The loss function that these algorithms seek to minimize - by way of tweaking the machine learning algorithm and hyper-parameters - is the equally weighted average slope and duration mean square error (Equation 5) on the *validation* set (Figure 3).

[3]https://pymoo.org

$$\frac{1}{2T}\sum_{i=1}^{T}((\ell_i - \hat{\ell}_i)^2 + (s_i - \hat{s}_i)^2) \tag{5}$$

Combined (equally-weighted) Slope (s_i) and Duration (ℓ_i) Mean Squared Error. Where a hat ($\hat{y}$) represents the predicted value and no hat (y) represents the observed value.

3.6 Budget

In order to make meaningful comparisons between the competing optimization techniques - it is necessary to fix a budget and only allow each optimization technique the same fixed budget. We use the total number of calls to the objective function in order to quantify the budget.

It is the nature of Evolutionary Algorithms to employ the concept of a population [29], where each generation is an improvement on the previous generation. Each member of the population must make a call to the objective function. Therefore, when working with evolutionary algorithms, in order to fix the number of calls to objective function - the number of generations must be adjusted based on the population size.

A summary of the information used is found in Table 2

OT	Population Size	Number of Generations	Total function calls
Random	1	540	540
PS	45	12	540
DE	20	27	540

Table 2: Abbreviations used for neatness: OT = Optimization Technique, PS = Pattern Search, DE = Differential Evolution

3.7 Evaluation

In order that the results of this research be comparable with that of Kouassi and Moodley: the equally weighted average slope and duration mean square error (Equation 6) on the *test* (out-of-sample) set (Figure 3) is used as the evaluation metric

$$\sqrt{\frac{1}{2T}\sum_{i=1}^{T}((\ell_i - \hat{\ell}_i)^2 + (s_i - \hat{s}_i)^2)} \tag{6}$$

Combined (equally-weighted) Slope (s_i) and Duration (ℓ_i) Root Mean Square Error.

This method of evaluation is useful in order to evaluate the efficacy of the CASH method as a technique in Automated Machine Learning or more specifically the use of the CASH method to automate machine learning algorithms to better make trend predictions. However, in order to compare the efficacy of optimization techniques as a CASH solver - we note that the objective function which is being minimised is the mean squared error on the *validation* set. Thus in order to evaluate the virtue of on optimization technique compared to another, we need to compare how quickly the algorithm is able to find smaller values of the objective function.

4 CONFIGURATION OF DIFFERENTIAL EVOLUTION ALGORITHM

The Differential Evolution Optimization Technique has three key control-parameters. Varying the configuration of these control-parameters has an effect on the efficacy of the algorithm when an applied as a solver to the CASH problem. Thus it was necessary to run experimentation to determine the effect of each control parameter when used as a CASH solver.

The following are the control-parameters under consideration

- *Population Size.* The Population Size can be any number that divides the total number of calls to the objective function. i.e. $total\_calls = pop\_size \times num\_generations$
- *Weighting Factor (F)* The domain of F is $F \in [0, 1]$
- *Crossover Constant (CR)* can be interpreted as the probability that a single member of a population exchanges variable values from the member from which it is derived. The domain of CR is $CR \in [0, 1]$

Seeing that our goal of configuring the Differential Evolution Algorithm is to ensure its optimal performance in minimising the objective function - we consider the validation mean square error (See above Section 3) as a performance metric.

The following graphs show the change in performance when the F and CR values are varied. In order to isolate the effect of the change in F or CR value respectively - the graph value shown is an average taken over all the other parameter values where the parameter at hand is kept fixed.

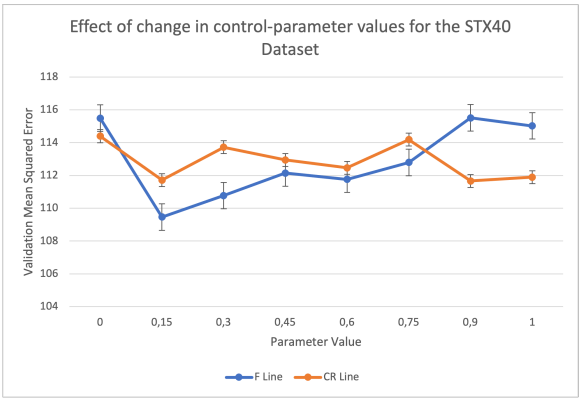

Figure 4: Graph showing the effect of tuning the *F* and the *CR* control-parameters of the Differential Evolution Optimization Technique for the STX40 dataset. The Validation Mean Square Error is an average over all *Population Sizes* under consideration.

Figure 4 shows that average optimal F value for the *STX40* dataset is 0.15 while that of CR is 0.9 however the value of 0.9 is not statistically significantly different (See standard error bars) compared to a CR of 0.15 as well.

When looking at the same graph for the NASDAQ dataset does not show the same pattern (Figure 5.

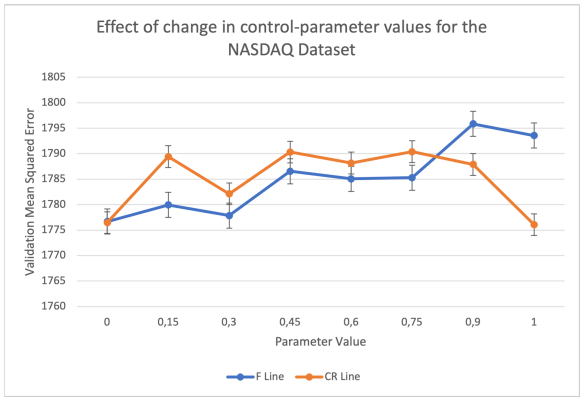

Figure 5: Tuning control-parameter graphs for the NASDAQ dataset

At this point in our investigation it appeared that the F and CR did not show to have a strong impact.

For this reason we look at the effect of change in *Population Size*:

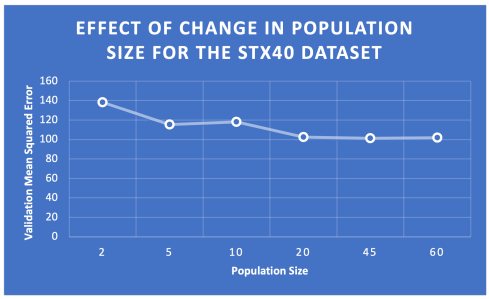

Figure 6: Graph showing the effect of tuning the *Population Size* of the Differential Evolution Optimization Technique for the STX40 dataset. The Validation Mean Square Error is an average over all *F* and *CR* values under consideration.

From Figure 6 we see a significant improvement in a *Population Size* of 20 compared to that of 2. This is improvement is a 28.57% (40/140) reduction in MSE.

Similarly, when looking at the same graph for the NYSE dataset (Figure 7). There appears to be a significant decrease in a *Population Size* of 20 compared to that of 2.

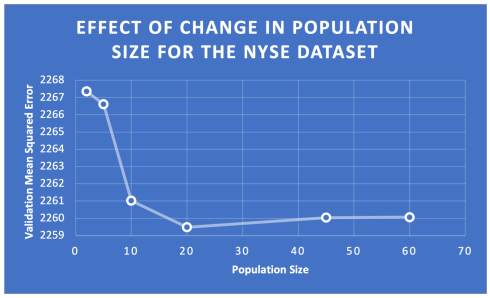

Figure 7: Graph showing the effect of tuning the *Population Size* of the Differential Evolution Optimization Technique for the NYSE dataset. The Validation Mean Square Error is an average over all *F* and *CR* values under consideration.

And finally, the same graph for the NASDAQ dataset (Figure 8) shows that there is a general pattern where higher population sizes have better performance.

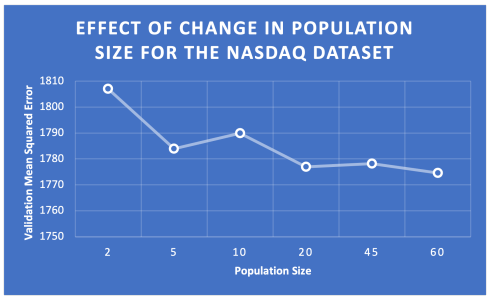

Figure 8: Graph showing the effect of tuning the *Population Size* of the Differential Evolution Optimization Technique for the NASDAQ dataset. The Validation Mean Square Error is an average over all *F* and *CR* values under consideration.

Selecting 20 as a good performing population size, however one that is still not too close to the overall budget (thus allowing a higher number of generations). We now can re-look at the effect of the *F* and *CR* values by way of a heat map.

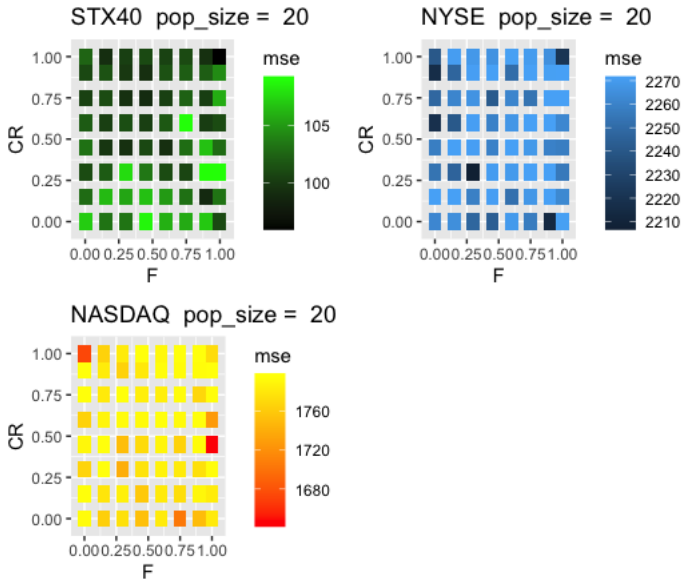

Figure 9: Heat Map to identify significantly better performing regions of *F* and *CR* values

By inspection it appears that a *CR* value of 1.0 performs well across all datasets. This is consistent with the information shown in Figure 4 and Figure 5 above. While the algorithm appears to perform more consistently well with lower values of *F*.

In line with these findings - the author chose to run the Differential Evolution Algorithm using a Population Size of 20, an *F* value of 0.15 and a *CR* value of 1.

5 RESULTS AND DISCUSSION

Table 3 presents an overview of the results of the experimentation performed in this research.

On the STX40 dataset, an index of the Johannesburg Stock Exchange, DE outputted an RMSE of 27.93. Meaning that it was the worst performing compared to it's benchmarks. However, it is an insignificant difference.

This RMSE value is directly comparable to the result of the JSE data-set as found in [17]. Although they are not the same data-set, both are an index of shares on the Johannesburg Stock Exchange and thus we would assume the trend behaviour to be similar. That research achieved an RMSE 16.21 which is substantially better than our results - 41.96% improvement. The author suspects that the cause of this is over-fitting - see Section 6.1.

By inspection of Figure 10 and Figure 11 - which gives us insight into how quickly the optimization technique is able to minimize the objective function - we can see that DE found a Validation MSE below 97 for the first time after 64 seconds while that of Random Search took 23 seconds and that of PS took 49 seconds. Both DE and PS were unable to converge within 540 function calls.

		NASDAQ	NYSE	STX40	Average
Random	S	150.86	130.76	38.29	106.64
	D	11.49	4.3	5.56	7.12
	A	**106.99**	93.51	27.36	75.95
DE	S	151.33	130.75	38.44	106.84
	D	6.54	4.46	8.19	6.4
	A	107.11	92.51	27.93	75.85
PS	S	151.34	130.76	38.26	106.79
	D	7.74	4.16	5.3	5.73
	A	107.15	**92.5**	**27.31**	**75.65**
Average	S	151.18	130.76	38.33	106.76
	D	8.59	4.31	6.35	6.42
	A	107.08	92.84	27.53	75.82

Table 3: Tabulated Summary of Results of experiments. The table shows the out-of-sample Root Mean Square Error (RMSE) of each Optimization Technique (OT) over each data-set. The table also shows the break up of RMSE between error in Slope (S), error in Length/Duration (D) and combined (equally-weighted) error (A). The best performing combined error for Optimization Technique is bolded in each column.

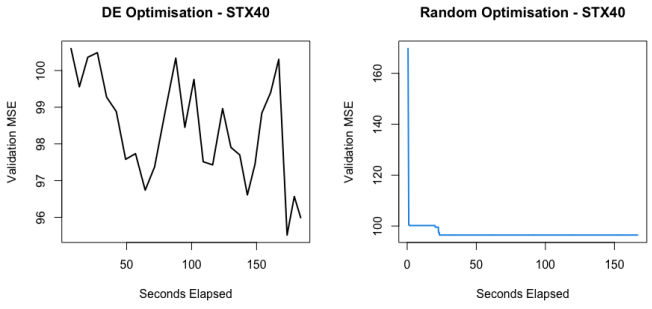

Figure 10: An insight into the optimization of the Differential Evolution Optimization technique compared to the Random Search Optimization technique. The graphs show the decreasing validation mean squared error as time progresses for the STX40 dataset.

Figure 11: Differential Evolution Optimization technique compared to the Pattern Search Optimization technique on STX40 dataset

Looking now at the NYSE data-set: DE was outperformed again by PS (albeit by less than 0.1%), however DE was better than a Random Search.

This dataset is directly comparable to the NYSE data-set used in the research of Kouassi and Moodley. The performance obtained by DE in our experimentation is 92.51 while that of Kouassi and Moodley was 43.49 which is substantially better than our results - 112.71% improvement.

Looking at figure 12 we can see that PS was far quicker and finished termination after 143 seconds while DE finished termination after 591 seconds. The best function value found by PS was 2222.96 and was found after 90 seconds. DE was only able to minimize down to 2270.89 and this happened after 259 seconds. However, it appears that DE was able to converge to a small range of values where PS did not converge. However, DE did not find the minimum found by PS. After 259 seconds - the time it took for DE to find its minimum - Random Search had found a minimum of 2271.89.

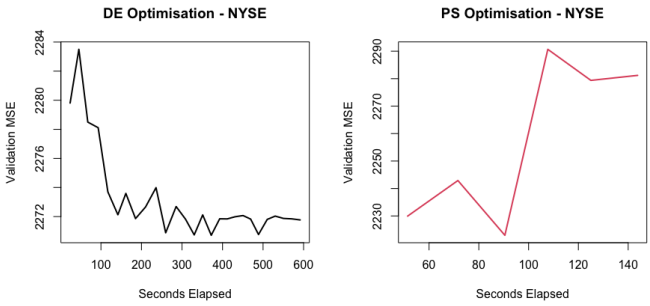

Figure 12: An insight into the optimization of the Differential Evolution Optimization technique compared to the Pattern Search Optimization technique. The graphs show the decreasing validation mean squared error as time progresses for the NYSE dataset.

Looking at the NASDAQ dataset - both DE and PS was outperformed by a Random Search. However, if we look at the graphs which show how quickly the algorithms were able to minimize

the objective function we see that DE outperformed both PS and Random Search in this regard.

DE found a minimum MSE of 1742.96 shortly after 200 seconds. PS found a minimum MSE of 1769.53 after 237 seconds while Random Search found a minimum MSE 1759.92 after 74 seconds. Neither DE or PS was able to converge.

6 CONCLUSIONS

The objective of this research was to compare the most popular evolution based optimisation techniques with each other and with a chosen benchmark of *Random Search* as a solver to the large and expensive optimization problem known as Combined Algorithm and Hyper-Parameter Selection. The domain over which this experimentation was conducted is to making trend predictions on financial equities.

In terms of making future trend prediction akin to out-of-sample data: the results of these experiments were poor in comparison to that of Kouassi and Moodley. However, as mentioned above, the optimization techniques here focused on minimizing the validation set mean squared error without any consideration for over-fitting. It should be noted that more work was put in place in the work of Kouassi and Moodley in order to prevent over-fitting and this may explain the far superior out-of-sample RMSE (See Section 6.1)..

Furthermore, we could not show that Differential Evolution was superior by any metric in comparison to a standard out-of-the-box Pattern Search or even a simple Random Search Method. However, it must be noted that the search space is very large in comparison to a budget of 540 calls to the objective function. The author suspects this is the reason we were not able to observe the DE optimization technique converge to a minimum. Further experimentation should be done before the reader should make these conclusions (See Section 6.1).

We recommend that the reader makes use of the methodology applied in the research of Kouassi and Moodley [17]. As it is the opinion of the author that the out-of-sample RMSE achieved in their research is state-of-the-art and is the best methodology to date for Automated Machine Learning for Time Series Trend Prediction.

6.1 Limitations and Future Work

The following is a list from the author of limitations stumbled upon over the course of thos research and recommendations for future research that is worthwhile to carry out based upone these limitations:

- Due to the nature of Black-Box Optimization with expensive objective functions - as in the case in this research - expensive computing resources are required as well as a large amount of time in order to properly allow Optimization Techniques
- Based on the review of the literature, there was evidence that the evolutionary algorithm known as Particle Swarm Optimization (PSO) is one of the best performing optimization techniques [31]. In the course of this research - many Python libraries were attempted in order to solve our optimization problem using PSO - however a number of libraries produced significant errors (For example *PSPSO* [10]) while other libraries [4, 20] do not provide support for categorical or discrete variables. It is outside the scope of this research to develop a custom built Particle Swarm Optimization therefore the author strongly encourages for future research that Particle Swarm Optimization is applied to the above experimentation and the results are compared.
- This research should be repeated with more steps in place to prevent over-fitting. The author suggests the following method:
  - Penalising validation mean square error based on complexity.
- It would be valuable to perform the same research using time series data-sets from a different industry.

ACKNOWLEDGMENTS

The author would like to thank Dr Deshendran Moodley for his insight and guidance.

REFERENCES

[1] Salihu Aish Abdulkarim. 2018. *Time Series Forecasting using Dynamic Particle Swarm Optimizer Trained Neural Networks.* Ph.D. Dissertation. http://hdl.handle.net/2263/70388

[2] James Bergstra, R Bardenet, Yoshua Bengio, Balázs Kégl, and Rémi Bardenet. 2011. Algorithms for Hyper-Parameter Optimization. https://hal.inria.fr/hal-00642998

[3] Changqing Cheng, Akkarapol Sa-Ngasoongsong, Omer Beyca, Trung Le, Hui Yang, Zhenyu Kong, and Satish Bukkapatnam. 2015. Time Series Forecasting for Nonlinear and Nonstationary Processes: A Review and Comparative Study. *IIE Transactions* (01 2015). https://doi.org/10.1080/0740817X.2014.999180

[4] Marc Claesen, Jaak Simm, Dusan Popovic, Yves Moreau, and Bart De Moor. 2014. Easy Hyperparameter Search Using Optunity. arXiv:1412.1114 [cs.LG]

[5] Swagatam Das and Ponnuthurai Nagaratnam Suganthan. 2011. Differential Evolution: A Survey of the State-of-the-Art. *IEEE Transactions on Evolutionary Computation* 15, 1 (2011), 4–31. https://doi.org/10.1109/TEVC.2010.2059031

[6] Frank. editor. Hutter, Lars. editor. Kotthoff, and Joaquin. editor. Vanschoren. 2019. *Automated Machine Learning.* Springer International Publishing. 101822 pages. https://doi.org/10.1007/978-3-030-05318-5

[7] Radwa Elshawi, Mohamed Maher, and Sherif Sakr. 2019. Automated Machine Learning: State-of-The-Art and Open Challenges. (6 2019). http://arxiv.org/abs/1906.02287

[8] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2019. Neural Architecture Search: A Survey. arXiv:1808.05377 [stat.ML]

[9] Hugo Jair Escalante, Manuel Montes, and Luis Villaseñor. 2009. Particle Swarm Model Selection for Authorship Verification. https://doi.org/10.1007/978-3-642-10268-4_66

[10] Ali Haidar, Matthew Field, Jonathan Sykes, Martin Carolan, and Lois Holloway. 2021. PSPSO: A package for parameters selection using particle swarm optimization. *SoftwareX* 15 (2021), 100706.

[11] Xin He, Kaiyong Zhao, and Xiaowen Chu. 2021. AutoML: A survey of the state-of-the-art. *Knowledge-Based Systems* 212 (1 2021). https://doi.org/10.1016/j.knosys.2020.106622

[12] Huseyin Ince. 2007. Kernel principal component analysis and support vector machines for stock price prediction. *Iie Transactions* 39 (03 2007). https://doi.org/10.1080/07408170600897486

[13] Peng Jiang, Cheng Chen, and Xiao Liu. 2016. Time series prediction for evolutions of complex systems: A deep learning approach. *2016 IEEE International Conference on Control and Robotics Engineering (ICCRE).* https://doi.org/10.1109/ICCRE.2016.7476150

[14] James Max Kanter and Kalyan Veeramachaneni. 2015. Deep feature synthesis: Towards automating data science endeavors. *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA).* https://doi.org/10.1109/DSAA.2015.7344858

[15] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. 2001. An online algorithm for segmenting time series. In *Proceedings 2001 IEEE international conference on data mining.* IEEE, 289–296.

[16] Ron Kohavi and George H. John. 1995. Automatic Parameter Selection by Minimizing Estimated Error. https://doi.org/10.1016/B978-1-55860-377-6.50045-1

[17] Kouame Hermann Kouassi and Deshendran Moodley. 2020. Automatic deep learning for trend prediction in time series data. (9 2020). http://arxiv.org/abs/2009.08510

[18] Tao Lin, Tian Guo, and Karl Aberer. 2017. Hybrid Neural Networks for Learning the Trend in Time Series. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17.* 2273–2279. https://doi.org/10.24963/ijcai.2017/316

[19] Zachary C. Lipton, David C. Kale, Charles Elkan, and Randall Wetzel. 2017. Learning to Diagnose with LSTM Recurrent Neural Networks. arXiv:1511.03677 [cs.LG]

[20] Lester James Miranda. 2018. PySwarms: a research toolkit for Particle Swarm Optimization in Python. *Journal of Open Source Software* 3, 21 (2018), 433.

[21] Randal S. Olson, Nathan Bartley, Ryan J. Urbanowicz, and Jason H. Moore. 2016. Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science. arXiv:1603.06212 [cs.NE]

[22] B. Samanta. 2004. Gear fault detection using artificial neural networks and support vector machines with genetic algorithms. *Mechanical Systems and Signal Processing* 18 (5 2004). Issue 3. https://doi.org/10.1016/S0888-3270(03)00020-7

[23] Mischa Schmidt, Shahd Safarani, Julia Gastinger, Tobias Jacobs, Sébastien Nicolas, and Anett Schülke. 2019. On the Performance of Differential Evolution for Hyperparameter Tuning. In *2019 International Joint Conference on Neural Networks (IJCNN)*. 1–8. https://doi.org/10.1109/IJCNN.2019.8851978

[24] Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. 2013. Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms. *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. https://doi.org/10.1145/2487575.2487629

[25] Min Wen, Ping Li, Lingfei Zhang, and Yan Chen. 2019. Stock Market Trend Prediction Using High-Order Information of Time Series. *IEEE Access* 7 (2019). https://doi.org/10.1109/ACCESS.2019.2901842

[26] Martin Wistuba, Ambrish Rawat, and Tejaswini Pedapati. 2019. A Survey on Neural Architecture Search. arXiv:1905.01392 [cs.LG]

[27] Li Yang and Abdallah Shami. 2020. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing* 415 (11 2020), 295–316. https://doi.org/10.1016/j.neucom.2020.07.061
.

[28] Quanming Yao, Mengshuo Wang, Yuqiang Chen, Wenyuan Dai, Yu-Feng Li, Wei-Wei Tu, Qiang Yang, and Yang Yu. 2018. Taking Human out of Learning Applications: A Survey on Automated Machine Learning. (10 2018). http://arxiv.org/abs/1810.13306

[29] Xinjie Yu and Mitsuo Gen. 2010. *Introduction to evolutionary algorithms*. Springer Science & Business Media.

[30] Peter Zhang, Eddy Patuwo, and Michael Hu. 1998. Forecasting With Artificial Neural Networks: The State of the Art. *International Journal of Forecasting* 14 (03 1998), 35–62. https://doi.org/10.1016/S0169-2070(97)00044-7

[31] Marc-André Zöller and Marco F. Huber. 2021. Benchmark and Survey of Automated Machine Learning Frameworks. arXiv:1904.12054 [cs.LG]