UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

CS/IT Honours Final Paper 2021

Title: Automated Machine Learning with Genetic Algorithms for Trend Prediction Forecasting in Time Series Data

Author: James Taljard

Project Abbreviation: AUTOML4TP

Supervisor(s): Deshendran Moodley & Josiah Chavula (Co-supervisor)

Category	*Min*	*Max*	Chosen
Requirement Analysis and Design	*0*	*20*	0
Theoretical Analysis	*0*	*25*	0
Experiment Design and Execution	*0*	*20*	20
System Development and Implementation	*0*	*20*	5
Results, Findings and Conclusions	*10*	*20*	20
Aim Formulation and Background Work	*10*	*15*	15
Quality of Paper Writing and Presentation	*10*		10
Quality of Deliverables	*10*		10
Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*)	*0*	*10*	
Total marks		**80**	**80**

Automated Machine Learning with Genetic Algorithms for Trend Prediction Forecasting in Time Series Data

James Taljard
tljjam001@myuct.ac.za
University Of Cape Town
Cape Town, South Africa

ABSTRACT

In recent times, machine learning algorithms have been applied to an array complex real-world problems in various knowledge domains. Automated machine learning aims to automate the selection of these algorithms and their hyperparameters in order to democratise the use of machine learning methods by allowing users to need little-to-no knowledge of machine learning theory in order to effectively use machine learning models. Genetic algorithms have been used extensively to perform hyperparameter optimisation for numerous machine and deep learning models. Research regarding the use of genetic algorithms for automated machine learning usually compares the hyperparameter optimised model to other common models used in similar problem domains instead of directly measuring the performance of the genetic algorithm in terms of hyperparameter selection. In this paper, we explore the use of a genetic algorithm for automated machine learning in the context of trend prediction in time series data, and compare these results to the established automated machine learning algorithm, random search. We apply both of these algorithms to a multilayer perceptron and a simple long short-term memory network. The results of the experiments indicate that a genetic algorithm can achieve better results than random search with both low and high computational budgets, but on average, genetic algorithms take longer to execute and their results are more susceptible to deviation within optimal model hyperparameter configurations.

KEYWORDS

Automated Machine Learning, Genetic Algorithms, Neural Networks, Time Series Forecasting, Trend Prediction

1 INTRODUCTION

In recent years, the use of machine learning algorithms has become prevalent in multiple application domains including natural language processing, computer vision, object detection, and advertising. This is because machine learning, and more specifically deep learning, algorithms can be applied to various problem and data types, while often achieving better results than human experts [7, 18]. However, designing and implementing an effective deep learning model usually requires human machine learning experts to manually select the suitable machine learning algorithm to use as well as tune its hyperparameters, usually through a trail-and-error approach, in order to achieve an optimal model architecture. Common machine learning algorithm hyperparameters include learning rate, batch size, number of hidden neural network (NN) layers, and number of neurons per hidden layer in the NN [11]. These hyperparameters are used to alter the behaviour of the machine learning algorithm in some specific way and must be set by the model designer before training can begin [16].

Recently, there has been an increasing interest in automating the various stages of the development of a machine learning model. These stages include data preparation, algorithm selection, and hyperparameter selection [7]. This automated machine learning (AutoML) has democratised the use of machine learning in various application domains by allowing non-experts to utilize various machine learning algorithms effectively within their knowledge domain [12]. Hyperparameter optimisation (HPO) is a subclass of AutoML which aims to the automate the selection of machine and deep learning algorithm hyperparameters that will give an optimal set of hyperparameters for a given data set and machine learning algorithm. Various methods are used for HPO which include random search, Bayesian optimisation, and meta-heuristic methods like particle swarm optimisation and genetic algorithm [5].

A genetic algorithm (GA) is a subclass of evolutionary algorithms that draw inspiration from the concept of natural selection. A typical GA begins with a group of chromosomes, known as the population; a chromosome is an encoding of a solution to a given problem. The algorithm generates a new population from the old one by generating new chromosomes using the genetic operations of selection, crossover, and mutation. New chromosomes are evaluated by some fitness score in order to determine whether or not it should be added to the population. Once the algorithm has terminated, the fittest chromosome is the final solution to the problem [15, 17, 19]. The use of GAs has become a popular method for HPO since they have the capacity to solve non-convex, non-continuous, non-smooth optimisation problems [18].

A time series is a series of values of a quantity taken at successive time intervals, the time intervals are often equal in length. A trend in time series data is a long-term increase or decrease in the data and understanding these trends can improve our forecasting ability. Due to the complex nature of real-world systems that produce time series data and the problems associated with them, the use of various ML algorithms to perform the task of time series analysis and trend prediction have been demonstrated with varying levels of success [4]. However, the hyperparameters of these algorithms must usually still be manually tuned by ML experts.

The focus of this study is to compare the effectiveness of genetic algorithms for automated machine learning against a more novel approach to AutoML, random search.

The aim of the research is two-fold. Firstly, to analyse and compare the effectiveness of automated machine learning with genetic algorithms against a more novel approach - random search. Secondly, to analyse the effectiveness of the use of automated machine

learning in the domain of trend prediction forecasting in time series data. In order to address these aims of the research, we seek to answer two research questions:

(1) Does a genetic algorithm perform better than random search for the task of automated machine learning in the context of trend prediction forecasting for time series data?
(2) Does AutoML, using either algorithm, produce satisfactory results when applied to an MLP and an LSTM, in the problem domain of trend prediction forecasting?

To this end, we will apply these AutoML algorithms to the task of trend prediction forecasting, making use of two neural networks: a multilayer perceptron (MLP), and a simple long short-term memory (LSTM) network.

The remaining sections of the paper are organised as follows. In Section 2, we provide context to the problem by discussing background and related work with regard to the use of AutoML for trend prediction forecasting and the use of GAs for AutoML. In Section 3, we discuss the design and implementation of the AutoML pipeline, the design and implementation of the experiment, as well as key evaluation metrics. In Section 4, we present the results of the experiments as well as discuss the key finding from these experiments. Section 5 presents the conclusions drawn from the key findings of the research. Section 6 presents the limitations of the work and describes potential future work in this research area.

2 BACKGROUND AND RELATED WORK

2.1 AutoML for Trend Prediction Forecasting in Time Series Data

Literature regarding the use of AutoML for trend prediction in time series data is limited. Kouassi and Moodley [11] showed that the application of AutoML, specifically the BOHB framework, for HPO and model selection for deep learning models can perform effectively when compared to deep learning algorithms with hand-tuned hyperparameters. The bulk of the literature focuses on the use of AutoML for HPO with various machine and deep learning models for time series forecasting. AutoML was applied to various machine learning models including long short-term memory, recurrent neural network, support vector regression, multilayer perceptrons, and gated recurrent units. Of these machine learning algorithms, LSTM and GRU appeared to be most applicable to time series data [2, 3, 9, 14]. However, most of the research focused on the performance of AutoML and a particular machine learning algorithm applied to time-series forecasting and not the performance of several AutoML techniques when applied to various machine learning algorithms when applied to time series data.

2.2 Genetic Algorithm for AutoML

Genetic algorithms are a subclass of evolutionary algorithms that draw inspiration from the process of natural selection. Due to there ability to solve non-convex, non-continuous, non-smooth optimisation problems, genetic algorithms have been effectively applied for hyperparameter optimisation, and AutoML in general. The majority of literature reviewed focused on the use of GAs for HPO for a particular machine learning algorithm as opposed to the larger combined hyperparameter and algorithm selection (CASH) problem. In almost all cases, a hybrid GA model outperformed models with hand-tuned hyperparameters [3, 9].

3 DESIGN AND IMPLEMENTATION

This section presents the design and theory of the AutoML pipeline, as well as the discuss the procedure and implementation of the AutoML experiments.

3.1 Neural Networks and Hyperparameter Search Space

In this paper, the hyperparameter optimisation algorithms are applied to two neural networks: an MLP and an LSTM. In this section, we discuss the design of these networks and their respective hyperparameter search spaces.

3.1.1 Multilayer Perceptron. The MLP consists of a simple feed-forward neural network with N fully-connected layer(s) where $N \in [1, 3]$. Each layer is followed by a ReLU activation function, except for after the output layer where a linear activation. A ReLU activation function was used in order to introduce non-linearity and for its simple computation. A dropout layer is added after each layer, except the final, in order to prevent overfitting. Each layer, n_i where $i \in [1, N]$, is fitted with $p_i \in [2, 64]$ nodes where $i \in [1, N]$.

3.1.2 Long Short-Term Memory. A simple long short-term memory network is implemented in order to evaluate the hyperparameter optimisation techniques. The LSTM network consists N LSTM layer(s), where $N \in [1, 2]$, and $p \in [2, 100]$ cells in each layer. Each layer is followed by a ReLU activation function as well as a dropout layer in order to prevent the model from overfitting. Finally, the LSTM is fitted to a fully-connected neural network layer in order to allow for multiple regression required for this problem.

3.1.3 Network Training. Both the MLP and LSTM are trained using the same hyperparameters. The Mean Squared Error (MSE) is used as the loss function during training, where both the trend duration and the trend slope are equally weighted in the calculation. The Adam optimiser is used as the optimisation algorithm. This allows for an adaptive learning rate as well as being computationally efficient [10]. L2 regularisation is employed to add a cost for larger weight values within the network in order to prevent overfitting. The network weights are initialised randomly according to a uniform distribution, such that weight

$$w_{jk}^{l} \sim U(-\sigma, \sigma)$$

with

$$\sigma = \frac{1}{\sqrt{p}}$$

where p is the number of input features.

3.1.4 Hyperparameter Search Space. The hyperparameter search space for each NN consists of both shared hyperparameters and hyperparameters unique to each NN. The hyperparameters regarding the network topology, the number of hidden/LSTM layer and the number of nodes/cells per layer, are unique to each NN and have been discussed in each NN's respective subsection. Shared hyperparameters include the learning rate, $\boldsymbol{\alpha}$, the dropout probability, $\mathbf{p}$,

the number of training epochs, $\mathbf{n}$, and the regularisation parameter, $\boldsymbol{\lambda}$. The possible values for these hyperparameters considered during the experiment are as follows:

$$\boldsymbol{\alpha} \in \{\ 0.5, 0.1, 0.05, 0.01, 0.005, 0.001, 0.0001, 0.00001\ \}$$
$$\mathbf{p} \in \left\{ \frac{x}{20} \;\middle|\; x \in [0, 20] \right\}$$
$$\mathbf{n} \in \{\ 50, 100, 200, 500, 1000, 2000\ \}$$
$$\boldsymbol{\lambda} \in \{\ 0, 0.1, 0.01, 0.001, 0.0001, 0.00001\ \}$$

Discrete values for these hyperparameters are used in order to reduce the overall search space and the selected values are commonly used values for their respective hyperparameter. Despite this, the hyperparameter search space for each NN is large with the LSTM having 1197504 unique hyperparameter combinations and the MLP having 4536852768 unique hyperparameter combinations. The smaller LSTM hyperparameter search space balances the longer training time required to train the LSTM network.

3.2 Automated Machine Learning Methods

In this section, we discuss the automated machine learning algorithms used to perform the hyperparameter optimisation of our NN models.

3.2.1 Random Search. In order to serve as a benchmark to compare against our GA, a random search algorithm is implemented to perform the hyperparameter optimisation for the neural networks. The random search algorithm works by sampling random points, according to a uniform distribution, from the hyperparameter search space. A NN is built and trained with the selected hyperparameters and the validation error is used to evaluate the NN against other hyperparameter configurations. The model configuration with the lowest validation error is recorded by the random search algorithm. The random search algorithm continues in this fashion for a prescribed number of iterations, after which, the best hyperparameter configuration is returned.

3.2.2 Genetic Algorithm. A genetic algorithm is a meta-heuristic a optimisation technique inspired by the process of natural selection first described by Holland. GAs perform well in finding optimal, or near-optimal, solutions in large search spaces within a reasonable computational limit [9, 14].

Figure 1 presents the overall procedure of the genetic algorithm. We begin with an initial population of p chromosomes, where p is the size of the population - a hyperparameter of the GA. Each chromosome is a set of values, or genes, which uniquely correspond to dimensions in the search space. In the case of our research, these chromosome will represent a set of NN hyperparameters. The fitness, or performance, of each chromosome is determined through a fitness function. Using the operations of selection, crossover, and mutation, GA generates a new population from the previous generation. These steps are repeated until a termination criterion is reached. Each chromosome is represented as a set of genes, where each gene corresponds to a particular hyperparameter. The value of each gene is an index corresponding to a value of the respective hyperparameter instead of using the actual hyperparameter value.

Initial Population An initial population is sampled randomly from the hyperparameter search space according to a uniform distribution. The size of the population is an important hyperparameter of the GA algorithm and is discussed below.

Fitness Function In order to evaluate the performance, or fitness, of each chromosome, we fit an NN with the selected hyperparameters and record the validation MSE. This validation MSE is used as the fitness function for the GA.

Selection In order to select individuals used to produce the subsequent generation, GA uses a selection procedure. This selection procedure has an important role in the convergence of the GA. In this paper, a tournament selection method is used in which individuals with the best fitness score are used to generate the subsequent generation.

Crossover After selection is performed, GA performs crossover between two parent chromosomes in order to generate new offspring chromosomes. Crossover is performed according to some probability. Due to the nature of the chromosome encoding, simulated binary crossover is used to perform the crossover procedure [6].

Mutation After crossover is performed, mutation is performed. The mutation operator randomly changes one or more genes within a chromosome within the current generation with some probability p. Mutation is introduced in order to prevent the model from converging to a local/sub-optimal optima. This probability is a significant factor in the behaviour of the GA as too high a mutation probability reduces GA to reduce random search while too small a probability results in sub-optimal convergence [6, 13].

Termination Termination of the GA is performed after a certain number of generations have been produced and evaluated.

3.3 Experiment Design

3.3.1 Data Preparation. In order to conduct trend prediction on time series data, the time series data has to be segmented into trend lines. In order to do this, we implement a sliding window, or bottom-up, piece-wise linear approximation method similar to the method used by Kouassi and Moodley. We begin with raw time series data in the form:

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

where x_t is a real-valued observation at time t.

We apply the sliding window algorithm to X in order to obtain a trend sequence T.

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

where ℓ_i is the duration of the trend, or the number of data points x_t covered by trend i, and s_i is the slope of trend $k \in [1, K]$ where K is the number of trend lines.

In order to utilise the trend sequence in a supervised learning problem, T is segmented in an overlapping fashion to produce T_S,

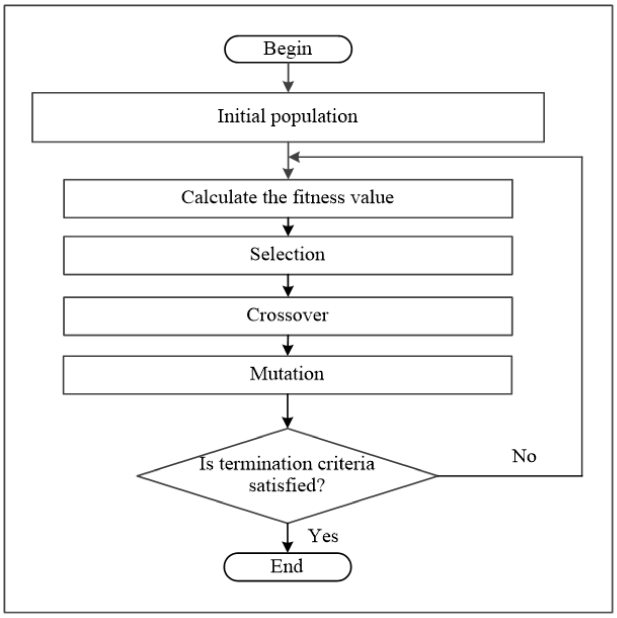

Figure 1: An overview of the genetic algorithm procedure [1]

where

$$T_S = [[[< \ell_1, S_1 >, ..., < \ell_n, S_n >], [< \ell_{n+1}, S_{n+1} >]], \\ [[< \ell_2, S_2 >, ..., < \ell_{n+1}, S_{n+1} >], [< \ell_{n+2}, S_{n+2} >]], \\ ... \\ [[< \ell_{K-n}, S_{K-n} >, ..., < \ell_{K-1}, S_{K-1} >], [< \ell_K, S_K >]]]$$

and n is the number of trend lines to be used as input features to the NN, or the length of the trend sequence.

We use a trend sequence of length 4 as an input feature to our NNs, and use these 4 trend lines in order to perform a one-step ahead prediction. Since a trend line consists of two data points, slope and duration, we have a total of 8 input features for both NNs and two output nodes (one corresponding to the predicted slope and one to the predicted duration of the trend line).

The sliding window algorithm accepts the maximum error to be used for regression during the piecewise linear approximation. This maximum error value is selected manually in order to ensure that the trend lines were were sufficiently long in duration, but also such that the trend lines do not over-generalise the time series data.

3.3.2 Datasets. Experiments are performed using two time series datasets of stock closing prices.

NYSE Daily Closing Price The first data set is the daily closing prices of the NYSE from 31/12/1965 to 22/06/2021. The data set was obtained from Yahoo Finance and contains 13964 data points. Once this trend series has been transformed into trend sequences, there are 315 trend sequence data points.

Nasdaq Daily Closing Price The second data set is the daily closing prices of the Nasdaq stock from 05/02/1971 to 22/06/2021. The data set was obtained from Yahoo Finance and contains 12706 data points. Once this trend series has been transformed into trend sequences, there are 334 trend sequence data points.

Datasets are partitioned into train, validation, test sets using a 78/11/11 split. An 80/10/10 split is what we set out to use, but the slight change in values is to ensure the respective size of the train/validation/test sets can be divided into the batch size with minimal remainder.

3.3.3 Evaluation Techniques. In order to evaluate our model during the hyperparameter optimisation process, we monitor the equally weighted slope and duration MSE achieved by the model on the validation set. Validation is performed after every training epoch with the best validation MSE, and respective network parameters, being recorded. The network parameters that achieved the lowest validation error is returned as well as validation MSE after the model has finished training. Thus, if we train the network for 2000 epochs and the lowest validation error is recorded during the 1000-th epoch, the network configuration and parameters returned by the model is the configuration and parameters at 1000-th epoch. This is done in order to prevent overfitting when the number of training epochs is high.

In order to evaluate the optimal network configuration return by the AutoML algorithm, the optimal network is evaluated on an unseen test set and the equally weighted slope and duration test MSE is returned and recorded. The data is not shuffled in order to preserve the temporal order of the data.

3.3.4 Experiments. Experiments were run in order to measure the performance of GA and random search with both low and high budgets.

		RS	GA				B
Budget	**NN**	**Iter**	**PS**	**Gen**	**MR**	**CrP**	**BS**
High	**LSTM**	900	30	30	0.02	0.7	32
	MLP	3000	50	60	0.02	0.7	32
Low	**Both**	200	10	20	0.05	0.7	32

Table 1: This table presents the hyperparameters used for both the genetic algorithm (GA) and random search (RS).

High Budget Experiments High budget experiments are used to evaluate the AutoML algorithm given significant computational resources and/or time. During these experiments, a large number of iterations for each AutoML algorithm is performed. The same number of iterations is selected for each AutoML algorithm in order to be able to draw comparisons between the two algorithms given the same opportunity to explore the hyperparameter search space.
For random search we simply state the number of iterations to be performed, whereas, in the GA we use both the population size and number of generations to control the number of iterations performed. In the high budget experiments regarding the LSTM network, a lower number of iterations is used due to the longer training times of the LSTM and the smaller hyperparameter search space.

Low Budget Experiments Low budget experiments were also conducted in order to assess the AutoML algorithms given less computational resources and/or time. These experiments are performed using 200 iterations of the AutoML algorithm.

A total of 8 experiments were run - one for each AutoML algorithm, dataset, and neural network - for each high and low budget experiments. Thus, a total of 128 experiments were run in order to obtain the results. Each experiment was run 8 times and the average of these results are reported. An average statistic was used since NNs can produce vastly different results when evaluated with the same parameters on the same dataset due to the random initialisation of the network weights [11]. Similarly, both GA and random search algorithm incorporate elements of randomisation which mean that they too are susceptible to variation between experiment runs.

Table 1 presents the hyperparameters for the given algorithms used to run the experiments. Experiments are divided between high and low budgets. The NN used during the experiment is listed since the LSTM and MLP have differing budgets for the high budget experiments. The number of iterations (**Iter**) is the only hyperparameter unique to the random search (**RS**) algorithm. The population size (**PS**), number of generations (**Gen**), mutation rate (**MR**), and crossover probability (**CrP**) are unique to GA. Finally, the batch size (**B**) is a hyperparameter common amongst all experiments.

3.3.5 Implementation. The aforementioned experiments were implemented in Python mainly using PyTorch machine learning library to build the NN models and PyMoo to implement the GA. The random search algorithm is implemented from scratch due to its simplicity.

The experiments were run on a number of different machine architectures including a Apple M1 chip and an Intel i5 quad-core CPU, both with 8 gigabytes of RAM. Additionally, some of the longer experiments were run on Google Colab which has a backend to the Google Cloud Compute Engine in order to make use of their GPUs with the CUDA capability provided by PyTorch.

		GA		RS		Best	
		M	**L**	**M**	**L**	**M**	**L**
NQ	**V**	**6039**	**15161**	8040	15767	**GA**	**GA**
	D	702	506	**662**	**244**	**RS**	**RS**
	T	**7553**	6991	8281	**6504**	**GA**	**RS**
	M	416	609	**173**	**226**	**RS**	**RS**
NY	**V**	**443**	**424**	468	432	**GA**	**GA**
	D	16	**10**	**5**	25	**RS**	**GA**
	T	**20646**	18336	24577	**18276**	**GA**	**RS**
	M	601	613	**215**	**215**	**RS**	**RS**

Table 2: Results from 8 high budget experiment runs. The equally weighted slope and duration validation (V) and test MSE (T) are given as well as the validation MSE standard deviation (D) and average time taken (M) for the experiment to run. Results for both the Nasdaq (NQ) and NYSE (NY) datasets are included.

4 RESULTS AND DISCUSSIONS

In this section, the results of the experiments are presented, interpreted, and discussions regarding these results are held.

4.1 Interpreting the Results

The results of the experiments need to be interpreted keeping the following points in mind. Firstly, the since the validation error is used to evaluate the models during the AutoML procedure, performance of the AutoML algorithm will be based on these values. Secondly, the test error is a measure of how well the model with the best hyperparameter configuration selected by the AutoML algorithm performs on out-of-sample data. Since the validation error is only a proxy for the true out-of-sample error, if the model hyperparameter configuration with the lowest validation error does not generalise the data well enough, the test error will be poor. It is important to note this as it directly pertains to the research objectives previously outlined. Finally, since the experiments were run on different computer architectures, the time taken to complete a particular experiment differs greatly.

4.2 Experiment Results

The results for both the high budget and low budget experiments are presented in Table 2 and Table 3, respectively. Note: the lower the MSE, the better. For each NN, we list the AutoML algorithm that performs the best according to each metric - these values are also presented in bold face.

The results from the high budget experiments indicate that GA outperformed random search in terms of validation error - the key evaluation metric of the two algorithms - in both the Nasdaq and NYSE datasets and both the MLP and LSTM. In terms of model deviation, in most cases the random search produced a more stable optimal hyperparameter solution. In terms of the predictive performance of the optimal model selected by the AutoML

		GA		RS		Best	
		M	L	M	L	M	L
NQ	V	11986	**15561**	**9758**	16515	RS	GA
	D	4322	1262	**581**	**329**	RS	RS
	T	**6816**	**6895**	7653	6974	GA	GA
	M	32	108	**23**	**54**	RS	RS
NY	V	**483**	**435**	491	474	GA	GA
	D	7.8	**28**	**7.6**	51	RS	GA
	T	**18815**	18230	19017	**18155**	GA	RS
	M	31	77	**21**	**51**	RS	RS

Table 3: Results from 8 low budget experiment runs for. The equally weighted slope and duration validation (V) and test MSE (T) are given as well as the validation MSE standard deviation (D) and average time taken (M) for the experiment to run. Results for both the Nasdaq (NQ) and NYSE (NY) datasets are included.

algorithm, for both datasets, the MLP has a better out-of-sample performance when optimised by the GA while the LSTM has a better out-of-sample performance when optimised by the random search algorithm. Finally, it is evident that given the same number of iterations, GA takes significantly longer to compute than the random search algorithm in the high budget experiments.

The results of the low budget experiments are very similar to that of the high budget experiments in terms of the best performing AutoML algorithms for each NN. The only difference in terms of the validation error is that random search outperformed GA for the MLP on the NASDAQ dataset. In terms of the out-of-sample performance, GA found the best MLP hyperparameter configuration for both datasets while this result was split for the LSTM between GA and random search for the Nasdaq and NYSE datasets, respectively. Finally, the time taken to execute these two algorithms is still lower for random search, but the difference in execution time between the two AutoML algorithms is significantly lower.

4.3 GA for AutoML

The results of both the high and low budget experiments indicate that GA has potential to produce better results than random search, especially given a larger budget. However, it is evident that the validation error standard deviation for random search is lower for almost all NNs and datasets aside from one (the LSTM on the NYSE dataset). This could be a result of a number of factors, the most prevalent being the initial population, population size, and the mutation rate.

4.3.1 Initial Population and Population Size. Since new offspring are generally created from existing chromosomes, the initialisation of the initial population has great implications for subsequent offspring created. Population size also plays an important role. If population size is too small, we restrict the search space and could result in a local optima, while if population size is too large, computational expense becomes an issue [6]. The experiments indicate that a population size of 10 is too small for this problem, while both 30 and 50 are sufficiently large for both the LSTM and MLP, respectively.

4.3.2 Mutation Rate. The mutation rate plays an important role in ensuring the GA does not converge to local optima by randomly moving along the search space. Since a high mutation rate causes the GA to breakdown to random search, a low probability of mutation was used for the GA in the experiments. The mutation rate was set to 0.02, which is within the ranges commonly used in literature [14]. However, due to the variability amongst the GA experiments optimal solution, a higher mutation rate might be better, especially when the population size is small.

In the high budget experiments, the execution time of the GA is significantly longer than that of the random search algorithm. This long runtime can be attributed to both the termination condition of the GA and as well as the way that GA works. Termination of the GA was done after a certain number of generations. In the high budget experiments, this meant that the algorithm began to sufficiently converge before the final generation. In most cases, the optimal number of training epochs was 1000 which meant that the training time of most of the chromosomes ended up being relatively long. Since GA had converged to this (or a similar) solution relatively early, we find most offspring eventually are trained using this number of epochs. In order to address this issue, it is suggest that a different terminating condition is used when searching over large search spaces in conjunction with large population sizes. This could be when the best fitness function, or average fitness of the generation, falls below a threshold, or when the best fitness has not changed after a certain number of generations. This would give the GA enough opportunity to explore the search space, while not wasting time once the algorithm converges. It is important to note that appropriate mutation and crossover rates need to be used in conjunction with this in order to allow the GA to explore the search space sufficiently.

4.4 GA for AutoML for Trend Prediction

In order to address our second research objective, we utilised a test set to measure the out-of-sample performance of the NN with the best network hyperparameter configuration selected by the AutoML algorithm. From the results of the experiments, it appears that both the MLP and the LSTM perform well on unseen data from the Nasdaq data set. Across both the high and low budget experiments, GA slightly outperformed random search in terms of selecting a network configuration that performs best on unseen data. However, due to reasons discussed in section 6, the validity of our models should be scrutinised.

5 CONCLUSIONS

The purpose of this research was to determine the effectiveness of genetic algorithms for the purpose of automated machine learning, more specifically, hyperparameter optimisation. Additionally, we set out to determine the predictive performance of the optimal models selected by the AutoML algorithms in the context of trend prediction.

In order to address these objectives, 4 sets of experiments were run on two different datasets. These datasets were financial data that were converted to trendlines using a bottom-up piecewise linear approximation method. The experiments were divided into high and low budget sets in order to determine the effect of budget

on model performance and computation time. For both low and high budget experiments, a random search and a genetic algorithm were used to perform hyperparameter optimisation on two types of neural networks: a multilayer perceptron and a simple long short-term memory unit. Each experiments were run 8 times in order to establish robust statistics that were used to evaluate the performance of the AutoML algorithm.

The experiments indicated that GA performed well against random search with respect to hyperparameter optimisation. This was based on the fact that in almost all cases, GA produced hyperparameter configurations that achieved a lower validation error than configurations found by the random search algorithm. This is significant as the validation error of the neural network was used as the objective function we were seeking to optimise.

In terms of predictive performance, the genetic algorithm produced models with a lower test error, especially on the Nasdaq dataset. However, due to the poor robustness of our validation procedure(see below), the resulting models do not generalise the data well, especially on the NYSE dataset.

The experiments also revealed that a higher mutation rate could be used to reduce the deviation amongst optimal model performances selected by the genetic algorithm. In order to reduce the execution times of high budget genetic algorithm executions, a termination condition other than the number of generations evaluated, or a combination of multiple termination procedures, is recommended.

6 LIMITATIONS AND FUTURE WORK

6.1 Limitations

6.1.1 Data Preparation. Two main issues arose from the data preparation procedure. Firstly, the slope of the trend line was used instead of the angle. This meant the range of the slope was $[-\infty, \infty]$ instead of $[-90, 90]$. This meant that steep trend lines have extremely large slope values. Depending on where these trend lines lie in the training, validation, or test set, they can greatly affect the performance metrics of the model.

Secondly, the data should have been scaled-down in order to encourage smaller weight values especially when used in conjunction with L2 regularisation. This is especially true due the issue regarding the slope value.

6.1.2 Model Validation. A simple validation set was used in order to approximate our NN model's out-of-sample performance. Due to the size of our dataset, and complexity of the problem, a cross-validation procedure would have provided us with a more robust statistic to measure out-of-sample performance. In addition to this, a walk-forward validation approach instead of conventional cross-validation would have been more applicable to our problem as the walk-forward approach maintains temporal order of the data.

6.1.3 Hyperparameter Search Space Configuration. The experiments in this research paper were limited to the purpose of HPO, not the larger CASH problem, due to the fact that we were not able to have a search space that is conditional on the machine or deep learning algorithm selected by the AutoML algorithm. Due to the large search spaces associated with the CASH problem, GAs ability to explore large search spaces could be extremely beneficial.

6.1.4 Parallelisation. Parallelising the algorithms where possible could be an effective way to speed-up computation time. In addition to this, utilising GPUs or TPUs can greatly improve training times.

6.2 Future Work

Due to the lack of work regarding trend series forecasting using AutoML, research of any kind in this area would be valuable. The implementation of a GA to solve the larger CASH problem could provide great results for large search spaces. In order to obtain more reliable results, more experiments like the ones discussed in this paper need to be performed on a larger number and variety of time series data as well as using different machine and deep learning models.

ACKNOWLEDGMENTS

The author would like to very much thank Deshendran Moodley and Josiah Chavula for their calm guidance during our meetings. I would like to thank my research partner, Adam Lewison, for his hard work and help when needed.

REFERENCES

[1] Musatafa Abbas Albadr, Sabrina Tiun, Masri Ayob, and Fahad AL-Dhief. 2020. Genetic Algorithm Based on Natural Selection Theory for Optimization Problems. *Symmetry 2020, Vol. 12, Page 1758* 12, 11 (oct 2020), 1758. https://doi.org/10.3390/SYM12111758

[2] Abdulaziz Almalaq and Jun Jason Zhang. 2019. Evolutionary Deep Learning-Based Energy Consumption Prediction for Buildings. *IEEE Access* 7 (2019), 1520–1531. https://doi.org/10.1109/ACCESS.2018.2887023

[3] Salah Bouktif, Ali Fiaz, Ali Ouni, and Mohamed Adel Serhani. 2018. Optimal deep learning LSTM model for electric load forecasting using feature selection and genetic algorithm: Comparison with machine learning approaches. *Energies* 11, 7 (2018). https://doi.org/10.3390/en11071636

[4] Dattatray P. Gandhmal and K. Kumar. 2019. Systematic analysis and review of stock market prediction techniques. , 100190 pages. https://doi.org/10.1016/j.cosrev.2019.08.001

[5] Baosu Guo, Jingwen Hu, Wenwen Wu, Qingjin Peng, and Fenghe Wu. 2019. The Tabu genetic algorithm: A novel method for hyper-parameter optimization of learning algorithms. *Electronics (Switzerland)* 8, 5 (may 2019), 579. https://doi.org/10.3390/electronics8050579

[6] Ahmad Hassanat, Khalid Almohammadi, Esra'a Alkafaween, Eman Abunawas, Awni Hammouri, and V. B.Surya Prasath. 2019. Choosing mutation and crossover ratios for genetic algorithms-a review with a new dynamic approach. *Information (Switzerland)* 10, 12 (2019). https://doi.org/10.3390/info10120390

[7] Xin He, Kaiyong Zhao, and Xiaowen Chu. 2021. AutoML: A survey of the state-of-the-art. *Knowledge-Based Systems* 212 (2021), 106622. https://doi.org/10.1016/j.knosys.2020.106622 arXiv:1908.00709

[8] John H. (John Henry) Holland. 1992. Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence. (1992), 211.

[9] Trang Thi Kieu Tran, Taesam Lee, Ju Young Shin, Jong Suk Kim, and Mohamad Kamruzzaman. 2020. Deep learning-based maximum temperature forecasting assisted with meta-learning for hyperparameter optimization. *Atmosphere* 11, 5 (2020), 1–21. https://doi.org/10.3390/ATMOS11050487

[10] Diederik P Kingma and Jimmy Lei Ba. [n.d.]. ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION. ([n. d.]). arXiv:1412.6980v9

[11] Kouame Hermann Kouassi and Deshendran Moodley. 2020. *Automatic deep learning for trend prediction in time series data.* arXiv:2009.08510 http://arxiv.org/abs/2009.08510

[12] Jason Liang, Elliot Meyerson, Babak Hodjat, Dan Fink, Karl Mutch, and Risto Miikkulainen. 2019. Evolutionary Neural AutoML for Deep Learning. 9 (2019). https://doi.org/10.1145/3321707.3321721

[13] Michael Lynch. 2010. Evolution of the mutation rate. *Trends in Genetics* 26, 8 (aug 2010), 345–352. https://doi.org/10.1016/J.TIG.2010.05.003

[14] Jiseong Noh, Hyun Ji Park, Jong Soo Kim, and Seung June Hwang. 2020. Gated recurrent unit with genetic algorithm for product demand forecasting in supply chain management. *Mathematics* 8, 4 (2020). https://doi.org/10.3390/math8040565

[15] Laurits Tani, Diana Rand, Christian Veelken, and Mario Kadastik. 2021. Evolutionary algorithms for hyperparameter optimization in machine learning for

application in high energy physics. *European Physical Journal C* 81, 2 (feb 2021), 170. https://doi.org/10.1140/epjc/s10052-021-08950-y arXiv:2011.04434

[16] Jonathan Waring, Charlotta Lindvall, and Renato Umeton. 2020. Automated machine learning: Review of the state-of-the-art and opportunities for healthcare. , 101822 pages. https://doi.org/10.1016/j.artmed.2020.101822

[17] Darrell Whitley. 1993. *A Genetic Algorithm Tutorial.* 40 pages.

[18] Li Yang and Abdallah Shami. 2020. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing* 415 (2020), 295–316. https://doi.org/10.1016/j.neucom.2020.07.061 arXiv:2007.15745

[19] Steven R. Young, Derek C. Rose, Thomas P. Karnowski, Seung Hwan Lim, and Robert M. Patton. 2015. Optimizing deep learning hyper-parameters through an evolutionary algorithm. *Proceedings of MLHPC 2015: Machine Learning in High-Performance Computing Environments - Held in conjunction with SC 2015: The International Conference for High Performance Computing, Networking, Storage and Analysis* (nov 2015). https://doi.org/10.1145/2834892.2834896