INTERFERENCE REDUCTION IN CLASSIFICATION AND FORECASTING TASKS THROUGH CLUSTER AND TREND ANALYSIS

David Olalekan Afolabi

Department of Computer Science

Faculty of Science and Engineering

University of Liverpool

This thesis is submitted in accordance with the requirements of the University of Liverpool for the degree of Doctor of Philosophy

October 2018

Dedicated to my family.

DECLARATION

This dissertation is the result of completed research work and has been written by myself. It has not been previously submitted to any other university/institution for any degree, or other qualification. In accordance with the department's guidelines, this thesis does not exceed the word limit, and it contains less than 50 figures and tables.

Signed: [illegible]

Date: 27th October 2018

David Olalekan Afolabi

University of Liverpool

Abstract

In order to classify or predict accurately in classification or time series tasks, the model building process is substantially dependent on the quality of data which is available for training such models. Consequently, reduced performance can be observed when input attributes/patterns have a conflicting influence on the learning process either due to intrinsic weak discrimination factor of some specific input attributes or as a result of outliers/anomaly picked up during data acquisition/entry.

Several hypotheses are proposed, defined, and empirically tested to achieve an interference-less machine learning process using meta-assisted learning in data classification and time series forecasting. Meta-learning is a branch of machine learning that focuses on the automatic and flexible learning of informative concepts/knowledge mined from given data in an efficient manner to improve performance whereby such a system includes a process to monitor the learning progress.

The two domains in which this research is focused on are classification tasks and time series forecasting tasks. Within these two domains, two further learning methods are explored whereby both the traditionally flat artificial neural network models and hierarchical structured artificial neural network models are modified to tackle the machine learning interference problem by using derived meta-information to reduce classification and forecasting error. The simulation experiments are performed with the multi-layer perceptron and a variant known as the constructive backpropagation artificial neural network for classification tasks; similarly, the nonlinear autoregressive exogenous model and long short-term memory artificial neural networks are used in time series forecasting tasks.

This thesis is established on the following key hypotheses:

i. Utilising the ‘cluster assumption’ for noise identification and extraction based on the intuition that samples of the dataset with higher similarity are inextricable and therefore should be clustered with other neighbouring samples that have similar labels. Clustered data from algorithms such as density based spatial clustering application with noise are analysed and are essential for the derivation of meta-information.

ii. Detection of repeating trend patterns by decomposing input signal into several building-block components over a range of frequencies enables distinction between information and error/noise/anomaly. To filter or decompose time series trends, we apply the moving average and empirical mode decomposition respectively.

iii. The guided meta-learning process; in which techniques are derived and introduced into the traditional learning process based on the inherent structure/distribution of pattern clusters or component signal trends within the data to tackle the problem of interference and noise within input attributes as the modified machine learner builds an accurate model.

iv. Hierarchical learning of local and global clusters/trends as real-world information tends to be structured in a hierarchy of concepts. Therefore, it is intuitive to learn on small/uncomplicated clusters before tackling a complex/encompassing cluster; or in the case of time series, learning short-term patterns before long-term trends.

This novel approach to noise elimination is shown to statistically increase the performance of a machine learning algorithm which is modified to carry out meta-learning on the training data. It is applicable to various benchmark and real-world datasets with significant improvement on data that contains known/unknown structure or patterns. Therefore the methods put forward in this thesis have the potential to *complement* or *reinforce* existing machining learning algorithms.

ACKNOWLEDGEMENTS

It has been a tremendous privilege to carry out this research surrounded by very supportive colleagues, supervisors and advisors at the University of Liverpool, UK, and Xi'an Jiaotong-Liverpool University, China.

My most profound acknowledgement goes to my supervisor Prof. Sheng-Uei Guan, for his meticulous guidance, research ideas, and feedback. This gratitude also extends to my co-supervisors, Prof. Ka Lok Man and Prof. Prudence W. H. Wong for their immeasurable advice and support throughout my PhD studies.

Finally, I am grateful to close friends and family from Nigeria and all around the world for every bit of support and encouragement I received; my journey was smoother because of you.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS AND ACRONYMS

A	Algorithm space
ALG	Algorithm
ANN	Artificial Neural Network
ARIMAN	Autoregressive Integrated Moving Average-Noise
$Attrib_i$	i^{th} attribute in the input vector
CBP	Constructive Backpropagation
CBR	Case-Based Reasoning
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DTW	Dynamic Time Warping
HMM	Hidden Markov Model
E	Error
EMD	Empirical Mode Decomposition
Eps	radius of the dense region around a point in DBSCAN
F	Feature space
FS	Forward Search
HDBSCAN	Hierarchical Density-Based Clustering of Applications with Noise
ILEL	Interference-less Ensemble Learning
IMF	Intrinsic Mode Functions
L-BFGS	Limited-memory Broyden–Fletcher–Goldfarb–Shanno optimisation algorithm
LMS	Least Mean Square
LS	Least Squares
LSTM	Long Short-Term Memory
LSTM-EMD	Algorithm: long short-term memory model with empirical mode decomposition meta-information

LSTM-MA	Algorithm: long short-term memory model with moving average meta-information
LSTM-ORIG	Algorithm: Traditional long short-term memory approach using the original data
MA	Moving Average
MAD	Mean Absolute Deviation
$MetaAttrib_j$	j^{th} meta-attribute appended into the input vector
meta_info	Meta-information
MinPts	Minimum number of points for a dense region in DBSCAN
MLP	Multi-layer Perceptron
MSE	Mean Squared Error
MT_F	Algorithm: Meta-assisted learning on a flat structure
MT_FC	Algorithm: Meta-assisted learning on a flat structure using output-class partitioning
MT_H	Algorithm: Meta-assisted learning on a hierarchical structure
MT_HC	Algorithm: Meta-assisted learning on a hierarchical structure using output-class partitioning
NAR	Non-linear Autoregressive model
NARX	Non-linear Autoregressive Exogenous model
NARX-EMD	Algorithm: Non-linear autoregressive exogenous model with empirical mode decomposition meta-information
NARX-MA	Algorithm: Non-linear autoregressive exogenous model with moving average meta-information
NAR-ORIG	Algorithm: Traditional non-linear autoregressive using the original data
NLP	Natural Language Processing
NMSE	Normalised Mean Squared Error
ORIG	Algorithm: Traditional machine learning approach using original data in classification tasks

OSM	Organisational Sustainability Modelling
P	Problem space
p-value	calculated probability in statistical hypothesis tests
PCA	Principal Component Analysis
std	Standard Deviation
SVD	Singular Value Decomposition
SVM	Support Vector Machines
SVR	Support Vector Regression
RIN	Result-Integration Network
RNN	Recurrent Neural Network
X_d	training set with *d* attributes in classification tasks
x_t	value at current time *t*
Y	performance space
y	output/actual value

LIST OF APPENDICES

1 Introduction

1.1 Noise in Machine Learning

1.1.1 Formal Definitions

In machine learning, the primary task is to produce a relatively accurate model/function, f, based on the training set, X, such that a predicted value on a test set coincides with the actual variable Y. The process of achieving an accurate model involves the minimisation of a loss function $L(f,(X,Y))$ while ensuring the model is still capable of generalising adequately.

Classification tasks can be formalised as $f: x \rightarrow y$, where the training set with n examples of input/output pairs, $\{(x_1,y_1),\dots,(x_n,y_n)\}$, are necessary to model the best approximate target function. While in time series tasks, it can be formalised as $x_{(t,t-1,\dots,t-d)} \rightarrow x_{t+1}$, where the previous outputs $x_{(t,t-1,\dots,t-d)}$, over a fixed duration, d, is recursively fed to the model to forecast future output x_{t+1} accurately.

1.1.2 Noise

The accuracy of the model can be ensured in a noise-free data without conflicting input patterns. However, in real-world data, such an assumption cannot be assured. Therefore automatic knowledge discovery on noisy data is a non-trivial task, more so as the dimension in many practical problems is high.

To accurately classify or predict in classification or forecasting tasks, the model building process is therefore heavily dependent on the quality of data which is available for the training of such models. Consequently, reduced performance is observed when input attributes/patterns have an adverse influence on the learning process either due to intrinsic weak discrimination factor of specific input attributes or noise picked up during data acquisition/entry. Thus noise can be defined as an undesirable anomaly in the data that has the potential to impact a trivial model or hypothesis negatively, and it is interpreted as follows [1]:

i. Imprecision during input attribute entry
ii. Labelling errors
iii. The omission of additional attributes/features with latent (but potent) representation for individual classes
iv. The inclusion of interfering attributes/features with a weak discrimination factor.

1.2 Implication of the Problem

Some learning problems faced in machine learning have been identified and addressing these issues should have a significant impact on the overall performance of the system.

If a group of labelled patterns or points can be clustered in a region, then an unlabelled data can also be assigned to a given class based on the decision boundary between the classes [2]. Other researchers have shown complexity reduction by recursive segmentation and learning of clusters [3], but in most real-world and artificial datasets the number of data clusters is unknown, and the clusters are not shaped as easily-defined geometric shapes (see Figure 1.1) where its parameters can be useful for class judgement. Similarly, we cannot apply frequency filters to a noisy time series to eliminate noise without eroding crucial components as shown in the sinusoid in Figure 1.2.

Additionally, initialising with ill-conditioned parameters traps the network in one of several local minima [4], and this problem is usually solved by restarting and re-initialising the weights randomly [5]; which is not also guaranteed to find the global minimum.

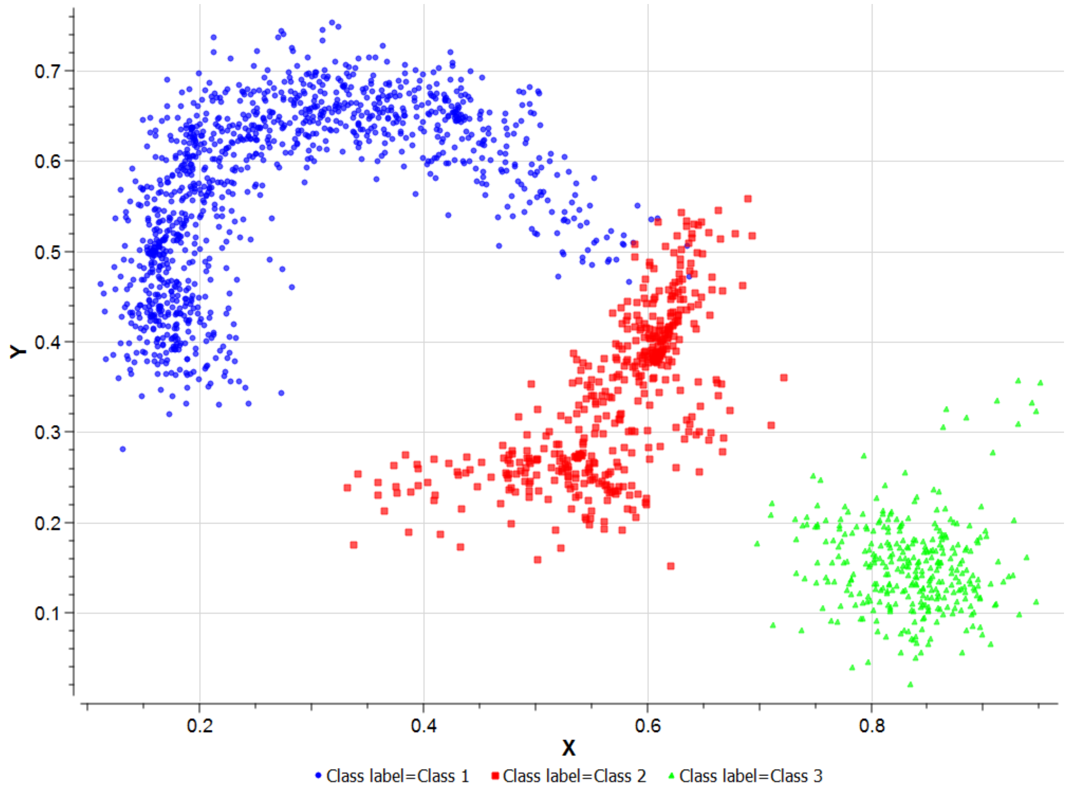


Figure 1.1 Synthetic arbitrary-shaped clusters with high-density regions colour-coded to represent individual classes

Moreover, due to the substantial adaptation of network connection weights and other parameters when incrementally learning from new examples, the information gained from past trained examples is collapsed and lost leading to poor performance [6]. This learning and unlearning process are due to interference. Consequently, the position of points within similar examples in a dataset form clusters that are too complex to be defined by geometric shapes; thus a method for creating irregular-shaped cluster classifier is necessary as they could reveal inconspicuous features of the data.

While producing solutions to the issues mentioned, it is important to note that datasets are structurally different and only algorithms that can automatically learn the salient features in any dataset can be deemed useful. Therefore a possible solution to the adverse effects mentioned above is to generate supportive/collaborative methods thereby leveraging the distinct advantages of both unsupervised and supervised learning. The real-world impact of a solution to this problem can be significant not only on the performance accuracy of the modeller but also by reducing the financial cost, since the alternative—data cleaning 'by hand'—is prone to errors, laborious, expensive and time-consuming [7].

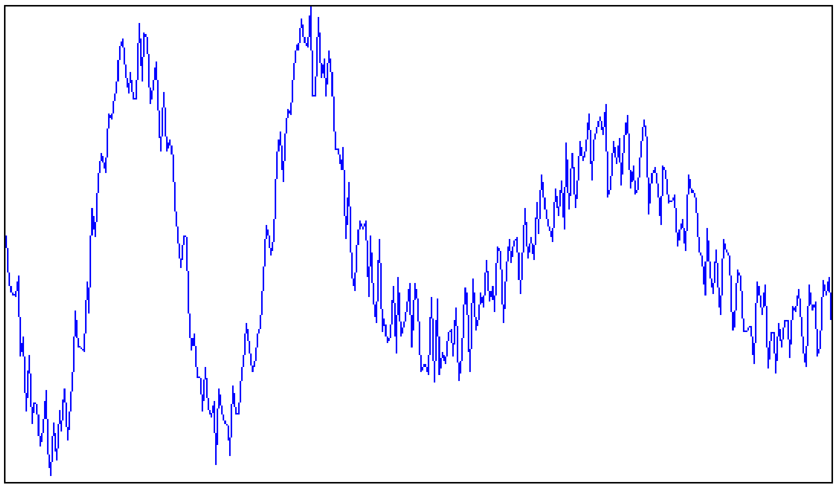


Figure 1.2 Synthetic sinusoid with low- and high-frequency noise

1.3 Focus Area and Roadmap to Derive Learning Algorithms

Several hypotheses are proposed, defined, and empirically tested in order to achieve an interference-less machine learning process. The two domains in which this research focuses on are classification tasks and time series forecasting tasks. Within these two domains, two further learning methods are explored whereby both the traditional flat neural networks and the hierarchical structured neural networks are modified to extract and evaluate meta on the data to enhance classification and forecasting accuracies.

More specifically, to produce a working solution we explored several well-established topics and algorithms including:

- Data clustering
- Time series decomposition
- Meta-learning
- Ensemble learning

On achieving the aforementioned goals, the contributions of this thesis to the art of meta-assisted learning are briefly noted in the following section.

1.4 Overview of the Contribution

- Applying semi-supervised learning technique to organise, predict and make sense of such vast amounts of data will be necessary due to challenges of obtaining labelled/well-structured data especially in the era of big data.
- Accuracy gained by meta-learning is not achieved at a significant time/computational cost increase.
- It is also applicable to systems in which low signal to noise ratio is frequently observed.
- It provides a local and a global view of the data to prevent the machine learner getting trapped at a local optimum, or generalising poorly due to overfitting.
- It has potential application in the field of data mining, function approximation, feature selection, knowledge handling, and training optimisation.

1.5 Thesis Organization

After stating the implication of noise within the data and how it affects learning performance in Chapter 1, the remaining chapters of this thesis are organised as follows:

i. Chapter 2 presents the related work which contains a critical assessment of other surveyed research and identifies their advantages, gaps, and how to complement them.
ii. In Chapter 3, the key hypotheses are proposed, and the rationale for meta-learning is established to provide the motivation for the selected approach and how it can deliver interference-less machine learning via cluster and trend analysis.
iii. Chapter 4 and 5 present the vital information of this thesis by detailing the design, implementation, analysis, interpretation of results, and comparison with existing/traditional models in regards to interference-less learning specific to classification tasks and time series forecasting tasks respectively.
iv. The thesis is concluded in Chapter 6 with insights gained from cluster/trend analysis, the contributions, and potential areas for further research.

2 Related Work

2.1 Literature Review

Interference between input attributes during training of the neural network has been identified by a number of researchers [6], [8]–[10]. In [10], the adverse effect amongst input attributes was reduced by grouping non-interfering attributes thus creating multiple groups which are then tackled by multiple learners and then utilising a Result-Integration Network (RIN) which is shown to produce a better performance. Thus their method for attribute grouping, called interference-less ensemble learning algorithm (ILEL), was tested on datasets with and without attribute interference and concluded that this method is viable only on datasets that had interference. In [10] the formula for attribute interference is:

$$\varphi(i,j) = \begin{cases} 0, & E_{ij} \leq \min(E_i, E_j) \\ 1, & E_{ij} > \min(E_i, E_j) \end{cases} \tag{2.1}$$

where the error score, E_{ij}, is for attributes i and j trained together, and E_i, E_j are the error scores when trained individually.

Furthermore, task decomposition involves dividing up the problem set into sub-problems so that each neural network module solves a fraction of the problem to reveal hidden structure within the full problem set [11]. This task decomposition may involve either input or output space partitioning. In [12], input space partitioning on a supervised constructive backpropagation neural network was used to solve classification problems by dividing the input based on correlating levels of attributes. On the other hand, output space partitioning tries to avoid the inherent interference noticed in larger networks by dividing the data into subsets of classes and training until an error threshold is reached or the error reduction rate is less than 1%, and there is an increase in the validation set error [13]. Likewise, data decomposition into patterns with "easy" or "difficult" clusters was proposed in [3] whereby simpler patterns are recursively learned and removed in order to improve the performance on the remaining complex clusters.

Conversely, Gamberger et al. [14] suggested handling noise beforehand to prevent the adverse influence on the hypothesis building. Their evidence reveals that it is necessary to develop a systematic and generic approach to noise handling through data pre-processing. In the research output of Zhu and Wu [7], they mentioned that an outlier could imply not only noise but also an exceptionally accurate occurrence which on the other hand may not be frequent enough to improve the general accuracy of the model.

We see that in order to propitiate the issue of noise and potential effects in machine learning, we must be able to adequately address the following question proposed by Zhu and Wu in [7].

i. *What's noise in machine learning? What's the inherent relationship between noise and data quality?*

ii. *What are the features of noise, and what's their impact [on] the system performance?*

iii. *What's a general solution in handling noise (especially attribute noise)? Why does it work?*

The approaches above clearly identified the interference due to training all input attributes together, and further highlighted the performance gain achieved by grouping the input attributes. This promotion can be realised by a number of grouping schemes such as contribution ordering (derived from an interference matrix), and unbalanced overlapping (where one attribute can be placed into multiple groups) [10]. From the results reported, the RIN method outperformed the voting and weighting integration techniques, but little detail is provided on the type of network, the structure, and how well it performs on datasets with higher dimensions.

Utilizing a long-term memory (LTM) is suggested in [6] to inhibit the network from adapting to noisy input on a 1-dimensional dataset by retrieving LTM data. However, for this approach to be successful, it requires a considerable amount of data for training and therefore added computational cost. The problem is partially solved in their research by retrieving and adapting only hidden units that were activated. Notwithstanding, a multi-dimensional manifold dataset would make it challenging to distinguish between noise and complex data clusters.

The solution to a complex clustered dataset involves recursively segmenting cluster to be learned by sub-networks with the idea that individual subset will have a smaller mean squared error in training. In the formula below, *P* is the ensemble of *K* subsets, *Tr* and *Val* are training and validation patterns respectively with inputs (*I*) and outputs (*O*).

$$
\begin{aligned}
P &= \{P^1, P^2, \ldots P^K\} \\
P^i &= \{Tr^i, Val^i\} \\
Tr^i &= \{I^i_{tr}, O^i_{tr}\} \\
Val^i &= \{I^i_v, O^i_v\}, \\
&\mathit{where}\ i \in K
\end{aligned}
\tag{2.2}
$$

This divide-and-conquer approach reported better generalisation accuracies but increased training time. Furthermore, patterns located equidistant of two cluster's boundaries (i.e. outliers) are not considered, and such patterns may contain relevant information that defines the border of the class or on the other hand be noise which introduces interference.

Finally, feature reduction can be carried out to reduce the amount of computation by the elimination of the least contributing features [15] like a noise reduction technique. There are several state-of-the-art dimension reduction techniques; among which Principal Component Analysis (PCA) [16], Random Forests, backward feature elimination, forward feature construction, reduction of highly correlated columns, low variance filtering, and removal of data columns are commonly used in data analytics [17]. Singular Value Decomposition (SVD) has also been applied in text classification to both reduce the dimension and eliminate noise significantly [18].

2.2 Related Research

2.2.1 In Classification

Researchers earlier proposed some methods to rectify the impacts of noise in classification tasks. By clearly differentiating the two main types of noise into class noise and attribute noise, Zhu and Wu [7] pointed out that the attribute-class correlation and inter-attribute interactions can discover additional information from the data. They went further by identifying noise as any factor that obscures the correlating relationship between class label and instance attributes, and then concluded that only attributes sensitive to noise need to be handled.

Pruning is another method suggested by Quinlan [19], [20] to prevent overfitting on the training data set and promote better generalisation on the testing data set. In [19], Quinlan inferred that eliminating noisy instances could be counter-productive. In that empirical study, it is concluded that class noise is more harmful to learning than attribute noise. While other researchers [21] have contributed to addressing the issues of sparse data and class noise in the induction of decision trees, others [7] have pointed out that pruning shows limited improvement in situations with a relatively high noise level. Contrarily, Zhu and Wu [7] suggest the removal of instances with class noise and taking action on cleansing attribute noise.

Since outliers can be regarded as instances that do not follow the normal probability distribution, we do not employ the techniques of noisy data elimination, but instead, we characterise each instance by a number of measures (as explained in Chapter 4). Our method is unlike Guyon et al. [22] where class label correction on instances are separated into typical and atypical sets, using an information criterion, which can be susceptible to ordering effects.

2.2.2 In Time Series Forecasting

Data scientists and statisticians apply a standard approach in rectifying the impact of noisy data in time series forecasting tasks called the autoregressive integrated moving average-noise (ARIMAN) model [23]. This model is a development on the popular ARIMA model to tackle the presence of noise within the signal/sequence. Although the ARIMAN model is well developed, it has a limited scope of application. It can be applied in the context of recurring trends or precise seasonality to facilitate accurate forecasting with the presence of noise in the data.

Other data smoothing methods have been explored as a means to detect anomalies or outliers and impute values for the detected noise. An example is the Forward Search (FS) technique which can be applied when other traditional robust estimators fail to detect noise in data with noise level higher than the set threshold [24]. FS subsamples the original dataset with the presumption that the selected sample is error free. The Least Mean Square (LMS) algorithm is then applied to estimate the unknown coefficient from the selected sample for further application through the Least Squares (LS) method, which produces a matrix of residual values as it is repeated on all the observations to produce an LS solution. Some drawbacks of this approach include subsampling methods which can increase the chance of samples containing a significant amount of outliers, and secondly, an excess sensitivity to noise if parameters are not chosen correctly [24].

Since time series data (just like signals) can be decomposed into constituent components, we exploit this characteristic to identify noise within the sequence rather than filtering out high-frequency artefacts based on the assumption that these components are too erratic to be information-bearing. The usual presumption that noise is only contained in the high-frequency components was shown not to be the case in [25]. Noise can also be present in low-frequency components, and this motivates our newly-proposed algorithm, which also prevents the erosion of periodic trends and patterns within the series by learning of local and global trends separately as explained in Chapter 5.

Other researchers in this field have applied the artificial neural network (ANN), dynamic time warping (DTW) [26], hidden Markov model (HMM) [27], fuzzy logic [28], and support vector regression (SVR) [29] for traditional time series forecasting. They can adequately model the training data, but a relatively higher error rate is observed as a result of noise within the original data. Additional research in [30] successfully applied the dynamic time warping algorithm to identify a fixed number of predetermined patterns within a historical time series and transformed them into strings to detect similar patterns for predicting short-term future values. Most of the issues mentioned above in the various models may be attributed to imprecise estimators or presumption on the normal distribution of the data, based on the limited compute capabilities of computer hardware at the time. More recently, the scalability that cloud computing is providing has led to the development of the “as a Service” model whereby more robust and accurate predictors take advantage of data analytics on state-of-the-art models. Examples within financial risk analysis include the application of the Heston model in the Business Intelligence as a Service (BIaaS) [31], and the Financial Modelling and Prediction as a Service (FMPaaS) [32], [33] which uses the Monte Carlo method and Black-Scholes model. This paradigm is applicable to other domains such as cybersecurity [34] and weather forecasting [35] in which the hidden Markov model and ARIMA predictive modelling are applied respectively.

Thus the newly proposed meta-assisted learning algorithm avoids modelling noise along with the data during the pre-processing stage and can be applied to other well-established/standard learning algorithms.

2.3 Potential Research Gaps

From the survey on previous research on noise handling in classification and time series tasks, several research gaps are presented as follows:

i. Machine learning algorithms modelled on noisy data can unknowingly pick up misrepresentation of the data especially when noise is relatively high or when no form of pre-processing or data cleansing is involved.
ii. It is expensive and time consuming for data error correction by experts.
iii. Many of the best-reported classification algorithms on given dataset are designed and tuned to produce the best results due to expertise on that specific dataset, but this knowledge cannot be transferred to other domains.
iv. Similarly, forecasting models focus on only one-step-ahead prediction.
v. Spatial and temporal information or relationship between data points/attributes are ignored when the learning model is built.
vi. Limited focus on exploiting the tendency for real-world information to be hierarchically grouped.

2.4 Publications

Our research work which is established upon existing work has led to some contribution to this field of study and has been published in several conference proceedings and journal articles.

2.4.1 Journal Articles

i. D. Afolabi, S.-U. Guan, K. L. Man, P. W. H. Wong, and X. Zhao, "Hierarchical Meta-Learning in Time Series Forecasting for Improved Interference-Less Machine Learning," *Symmetry*, vol. 9, no. 11, p. 20, Nov. 2017.

2.4.2 Conference Proceedings

i. D. O. Afolabi, S.-U. Guan, F. Liu, K. L. Man, and P. W. H. Wong (2015) "Class Interference Reduction through Meta-attribute Reinforced Learning," *in International Conference on Computing and Technology Innovation,* Luton, UK, 2015.
ii. David O. Afolabi, Sheng-Uei Guan, Bingzhang Wu, and Ka Lok Man (2016) "Evaluating the contribution of meta-attribute to noise reduction in machine learning," *in SZABIST Multidisciplinary International Conference*, Dubai, UAE, Jan. 2016.
iii. D. O. Afolabi, S.-U. Guan, K. L. Man, and P. W. H. Wong, "Meta-learning with Empirical Mode Decomposition for Noise Elimination in Time Series

Forecasting," in *Advanced Multimedia and Ubiquitous Engineering: FutureTech & MUE*, J. J. H. Park, H. Jin, Y.-S. Jeong, and M. K. Khan, Eds. Singapore: Springer Singapore, 2016, pp. 405–413.

iv. D. Afolabi, S.-U. Guan, and K. L. Man, "Meta-information for Data Interference Reduction in Classification and Forecasting," in *Proceedings of the International MultiConference of Engineers and Computer Scientists 2018*, Hong Kong, 2018, vol. II, pp. 711–714.

3 HYPOTHESES AND RATIONALES

3.1 Sources of Interference

As mentioned in Chapter 1, there are four primary sources of noise in data which cause interference during machine learning, and they can be summarised into two main situations:

i. The imprecision during input attribute or label collection/ data entry.
ii. The omission of attributes with a strong discrimination factor, or the inclusion of weak attributes respectively.

To reduce the impact of the identified interference, we propose several hypotheses, define them, and empirically test them to achieve an interference-less machine learning process. Within the two main application domains (classification tasks and time series forecasting tasks), some learning methods are explored whereby both traditional flat neural networks and hierarchical structured neural networks are modified to perform 'meta-learning' to tackle local and global sources of interference and therefore enhances classification and forecasting accuracies.

Therefore the contribution of this thesis is established on the following key hypotheses:

i. The cluster assumption for meta-learning.
ii. The recurrent trend patterns for meta-learning.
iii. The hierarchical learning of local and global clusters/trends.
iv. Meta-assisted learning for improved noise identification and elimination.

3.2 Key Hypotheses

3.2.1 The Cluster Assumption for Meta-learning

The cluster assumption [2] is based on the hypothesis that samples of the dataset with higher similarity should be clustered with other neighbouring samples having similar labels. If a group of unidentified patterns with high similarity are clustered in a region, the whole cluster can be assigned with the identity of the closest labelled data. This form of semi-supervised learning techniques has a number of advantages such as:

i. Automatic labelling of an enormous set of unlabelled data which is usually the most accessible type of data to collect. This technique is also efficacious to unsupervised learning problems.
ii. Identification of new concepts from unanticipated clusters (i.e. knowledge discovery which can contribute to accurate learning) and anomalies such as data outliers (which can cause learning interference and increase error rate).

Most interference would occur within a single cluster containing various class patterns thus leading to a longer time needed for training and also some amount of unlearning of useful discriminatory factors that could have been learned at an earlier epoch [9]. Therefore by decomposing the entire dataset into arbitrarily shaped clusters, based on the structure of the classification dataset, we tag each training sample via its meta-attribute to guide the model building phase. The Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [36] is utilised for clustering because of its ability to form irregular-shaped clusters which are representative of how real-world multidimensional dataset clusters are shaped.

3.2.2 The Recurrent Trend Patterns for Meta-learning

Identification of characteristic trends within complex time series data is possible through common transforms such as Fourier transform [37] and Empirical Mode Decomposition (EMD) [38]. EMD results in a decomposition of an original signal into several building-block trends from high- to low-frequencies, called the intrinsic mode functions (IMF), which can be summed up to reconstruct the original signal. These derived structural primitive trends contain repeating patterns that can be information bearing, or on the other hand irregular fluctuations that are noise-inducing, but it enables easier detection of components necessary for improved prediction accuracy. This approach offers the following advantages:

i. It allows the generation of meta-information in which noise/interference-inducing components can be eliminated.
ii. Meta-information is introduced into the learning process as an exogenous input to guide the learning process.

3.2.3 The Hierarchical Learning of Local and Global Clusters/Trends

As real-world information tends to be structured in a hierarchy of concepts, it is intuitive to learn on small/simpler clusters before tackling a complex encompassing cluster; or as in time series, learning short-term patterns before long-term trends. Related research has proposed an arbiter meta-learning technique in which random subsets of the training data are used to train several learning algorithm models before integrating the solution to obtain a better or at least equal predictive performance [39]. The technique in [39] can be highly susceptible to randomisation; therefore, we apply the previous two hypotheses to mitigate this effect. This machine learning algorithm is designed in a hierarchically structured manner and presents the following advantages:

i. Prevent interference between class clusters that lie relatively close to each other.
ii. Learning a hierarchy of concepts especially in data with taxonomy tree-like structure.
iii. Provides a local and global view of the data being modelled.
iv. Allows learning of short-term and long-term trends separately in order to avoid erosion of previously learned patterns.

3.2.4 Meta-assisted Learning for Improved Noise Identification and Elimination

Meta-information is derived and introduced into the learning process based on the inherent structure/distribution of pattern clusters or component signal trends within the data to tackle the problem of interference and noise within input attributes. Rather than merely recombining existing input attributes, these meta-information are derived through supervised or unsupervised techniques. In this research, a density-based clustering technique and a signal decomposition technique (for classification and time series tasks respectively) are selected to derive the meta-information which informs the machine learner about inherent characteristics of each training sample and how it relates to other samples/patterns. A detailed introduction to meta-learning is presented in the following section.

3.3 Meta-learning

Meta-learning is a branch of machine learning that focuses on the automatic and flexible learning of informative concepts/knowledge mined from given data (or related data) to efficiently improve performance whereby such system includes a process to monitor the learning progress. A vast plethora of machine learning algorithms exist with both theoretical and empirical results showing no single superior algorithm [40]. This situation has given rise to the algorithm selection problem whereby meta-knowledge from past training experience on similar datasets help to suggest the most suitable algorithm based on selected criteria especially prediction accuracy and training time. This method of learning on all possible domains is computationally expensive, but it reveals subsets where particular algorithms are superior [40].

Based on the framework proposed by Rice [41], Figure 3.1 shows how a problem space **P**, containing problem *x*, is mapped to a feature space **F** via a feature extraction function *f(x)*; and using a selected performance measure *y* from the performance space **Y**, a selection algorithm **S** maps the feature extraction function to an algorithm *a* in the algorithm space **A**. This ensures that the performance, *y*, of algorithm *a* on problem *x* is optimum [42], [43].

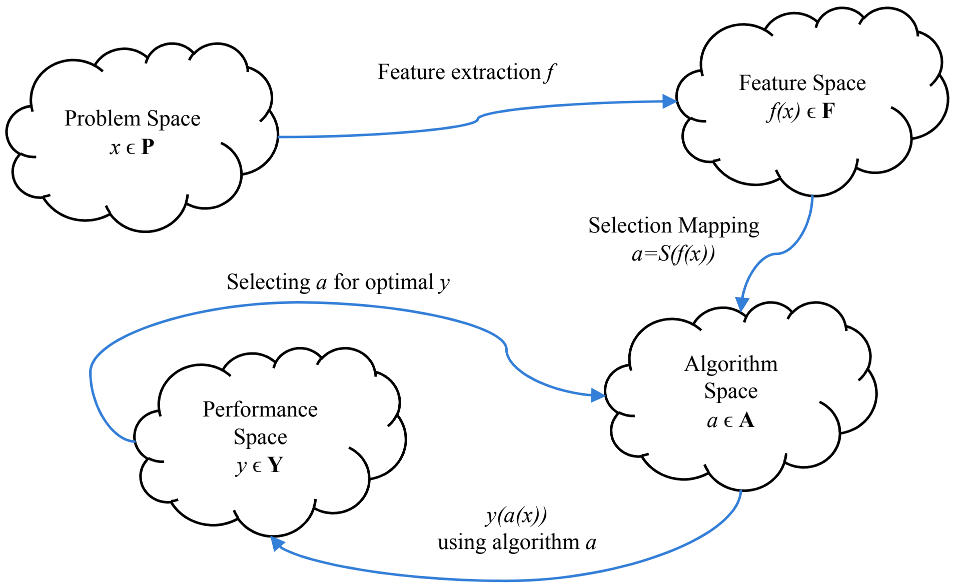


Figure 3.1 Algorithm Selection Framework by Rice; adapted from [42], [43]

3.3.1 Formal Definitions of Meta-learning

A substantial amount of empirical and theoretical research on algorithm selection over several decades has led to various perspectives and definitions of meta-learning. A summary of four prominent definitions in a survey by Lemke et al. [44] are thus presented:

i. A set of principle methods for exploiting meta-knowledge to obtain efficient solutions and models through the adaptation of machine learning and data mining techniques [45].
ii. The meta-learning process involves the monitoring of the automatic learning process within the context of learning problems which are encountered and how it makes behavioural adaptations that can improve performance [46].
iii. According to a domain or task, the aim of meta-learning studies is understanding how learning can become flexible to increase the learning system's efficiency through experience [47].
iv. The primary aim of meta-learning is understanding the interactions between the mechanism of learning and the context in which said mechanism is applicable to [42].

The first two definitions relate to the novel direction we introduced later in this chapter.

3.3.2 Categories of Meta-learning

There are four main categories in which meta-learning characterisations conform to:

i. Statistical and information-theory characterisation.
   Several measures are computed from the data with the assumption that learning algorithms can be classified based on the discrimination ability of these measures. Examples of such measures include the number of classes, the number of attributes, the ratio of nominal/numeric attributes, signal-to-noise ratio, and mutual information to mention a few.
ii. Landmarking.
   Several fundamental machine learning algorithms are trained in the premise that the learners' space can be partitioned into various areas of expertise. The performance of a landmark algorithm on a given problem space is used as an indicator for the performance of other similar variants of the algorithm.
iii. Decision tree-based characteristics.
   Suitable learning algorithms are recommended based on results derived from a decision tree that is induced by several measures which were computed from the data such as depth, balance and shape.
iv. Higher-order inductive learning [48].
   By utilising induced trees for task characterisation, rather than collecting pre-computed characteristics information manually from methods such as statistical

measures, they can perform higher-order inductive learning to produce rich structures.

3.3.3 Applications

Meta-learning applies to the following domains; which are optimisation, classification, regression, time series, constraint satisfaction and sorting [43]:

Optimisation involves searching for the best solution in a task in order to maximise the benefits while satisfying given constraints. Due to time, memory, and computational limitations of searching for a solution in a huge problem space, algorithm selection can be utilised in selecting suitable solutions based on relevant metrics which characterise the complexity of the optimisation problem.

Besides the classic application of algorithm selection in classification problems based on the measures mentioned in section 3.3.2i, an additional task of optimal parameter selection can be explored using meta-learning. Many other forms of meta-learning have been developed, for example, to determine the best kernel for Support Vector Machines (SVM) [49] or the best weight optimiser for an artificial neural network (ANN).

With regards to regression, new meta-features are needed to adequately characterise the data, such as variation, outliers, variables correlation among others. In this domain, algorithm selection and parameter selection have been applied in several studies [43], [50]. Similarly, in time-series forecasting where a model is built to capture the trends of a continuous data sequence based on preceding output (and input in some time series tasks), the most suitable algorithm is selected based on a number of measures such as mean absolute deviation (MAD), mean squared error (MSE) and normalised mean squared error (NMSE). The features used in [51] to characterise this type of data include the autocorrelation coefficient, the coefficient of determination, granularity (quarterly or yearly), MSE from the regression model, and the number of turning points. Other studies mentioned in [43] included trend, seasonality, skewness, kurtosis, and serial correlation. In such an application, a decision tree learner is used to generate rules which aid in algorithm recommendation for a given task.

In constraint satisfaction, there has been a focus on solving NP-hard problems with applications in routing, timetabling, and statistical experimental design. In this task, the algorithm portfolios mapping is learned based on the logarithm of runtime (which is the computation time to solve a given problem) [52]. A set of characteristic features such as the number of clauses and variables, ratios of positive and negative literals in each clause, and search-space size estimates is used to understand the relationship between the computation time (target) and constraint problem in [43].

Sorting tasks have been a trendy research area with an overabundance of algorithms developed over several decades. Therefore, this domain can also benefit from the framework described above. Features usually taken into consideration for sorting-algorithm selection include length of the sequence, number of inversions, distance of inversion, maximum distance required to move an element into a sorted position, minimum number of exchanges for a complete sort, minimum number of elements that must be removed to sort a subsequence, and number of sorted lists constructed by a specific algorithm when applied to the sorting problem [53].

3.3.4 Divergence and Contributions

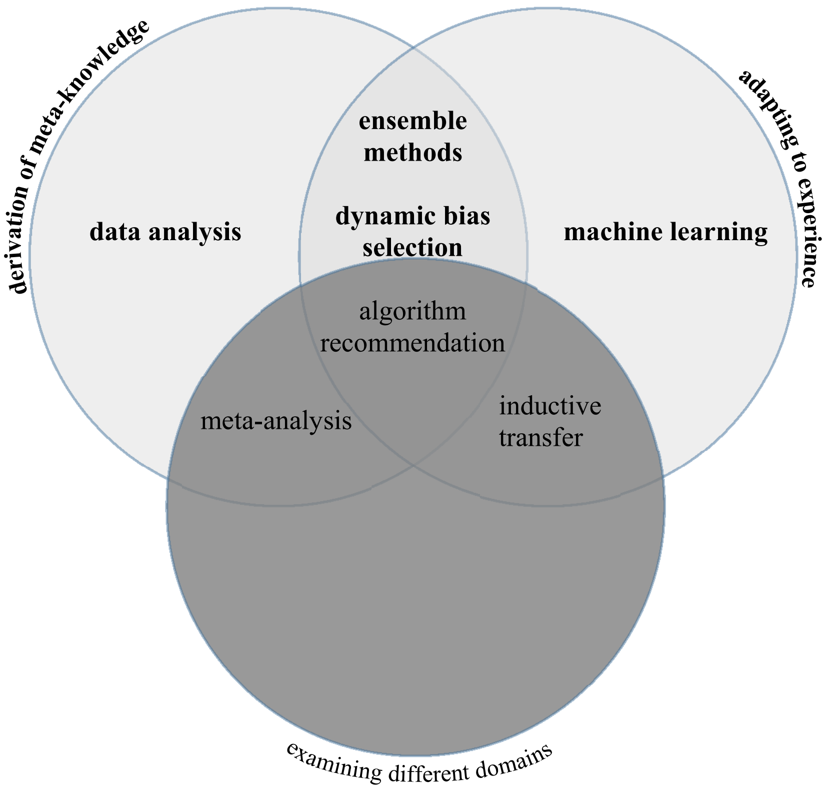


Figure 3.2 Components of a meta-learning system adapted from Lemke et al. [44]

A wide variety of literature in meta-learning learning have made contributions into to three areas which combine to form a defined notion of the meta-learning problem. These three fundamental concepts [44] as shown in Figure 3.2 are:

- Adapting to experience
- Derivation of meta-knowledge
- Examining different domains

These component concepts overlap and denote some areas of considerable research interest by many researchers while overlooking the potential of applying machine learning to the machine learning problem itself. A related area which is model combination involves ensemble learning to carefully select or combine the output of several machine learning algorithms. They include popular methods such as bagging [54], boosting [55] and more precisely—stacked generalisation [56] and cascade generalisation [57]. The central area of interest of this thesis within this domain is data analysis and machine learning. Therefore ensemble methods and dynamic bias selection are vital aspects of our framework to achieve a meta-assisted learning algorithm for interference-less classification and forecasting.

As explored in [58], the case-based reasoning (CBR) approaches dataset representation within the category of statistical and information-theory characterisation; whereby some useful characteristics of the available data are calculated from its general, numerical, and symbolic attributes for specific application goals. During data acquisition, selecting the attributes can be potentially complex, as too many conflicting data or too few non-representative data may be acquired.

Kalousis and Hilario [59] stated that a set of attributes which produce the best performance in one inducer could perform differently with another inducer. This observation is based on the "no free lunch theorem" in [60]. They further mention that the explanation and understanding of dataset properties is an area in the field of meta-learning with least research attention. The filter and wrapper approach are two main machine learning methods for feature selection [59]. In the filter feature selection technique, properties of the dataset are utilised to elect a subset of features based on measures of interdependence or distance without accounting for the bias of the learning algorithm that will be used to build the model. The filter approach can be significantly faster than the wrapper approach. The wrapper feature selection has a more active approach to electing features with the aim of improving the accuracy of the learning algorithm with an extensive or heuristic search pattern. The search pattern of backward elimination (starting from a full set) or forward search (starting from an empty set) can be prohibitively computationally intensive.

3.4 Potential Real-World Applications and Advantages

i. Applying semi-supervised learning technique to organise, predict and make sense of such massive amounts of data will be necessary due to challenges of obtaining labelled and well-structured data especially in the era of big data.

ii. Performance-accuracy increase by meta-assisted learning is not achieved at a significant time/computational cost.

iii. It is also applicable to systems in which low signal to noise ratio is frequently observed.

iv. It provides local and global views into the data to prevent the machine learner getting trapped at a local optimum rather than finding a global optimum.

v. It has potential application in the field of data mining, function approximation, feature selection, knowledge handling, and training optimisation.

4 Interference-less Learning in Classification Tasks

In classification tasks, a number of machine learning techniques are commonly applied to accurately predict a target class after training on a set of examples. Interference introduced by noisy input data or attributes causes classifiers to perform poorly. It is common for real-world datasets to contain inconsistent attribute values, which leads to a longer time required for training and also some amount of unlearning of useful discriminatory features that could have been learned at an earlier epoch [9]. In [15], we stated how some techniques to mitigate interference in artificial neural networks (ANNs) such as input data partitioning, analysis of inter-attribute interference, and incremental attribute learning are applied in existing research outputs [6], [8], [61]. Unfortunately, there are no standardised methods to improve the general performance of machine learning algorithms besides hand-tuning the parameters of a specific model on specific datasets. Therefore, to tackle attribute interference without dependence on the type of model used we have applied meta-learning to pre-process the dataset. A modified representation of the data based on the density distribution of the training set is derived, and the empirical results support the applicability of the proposed method.

4.1 Background

Real-World data categories or clusters are not necessarily formed as definable geometric shapes in a spatial dimension. From [3], it is stated that complexity reduction can be achieved by recursive segmentation and learning of formed clusters. Therefore using partitioning algorithms such as k-means, expectation–maximization algorithm, hyperplanes or hypersphere [62] may not be sufficient to define the irregularly shaped cluster boundary as shown in Figure 4.1. Conversely, a viable solution derived from hyperplanes, hypersphere, or any of the earlier mentioned algorithms can get too complicated to characterise an arbitrarily shaped group of related data points accurately.

We, therefore, perform cluster analysis on the data to generate meta-information based on the spatial distribution or density of class clusters.

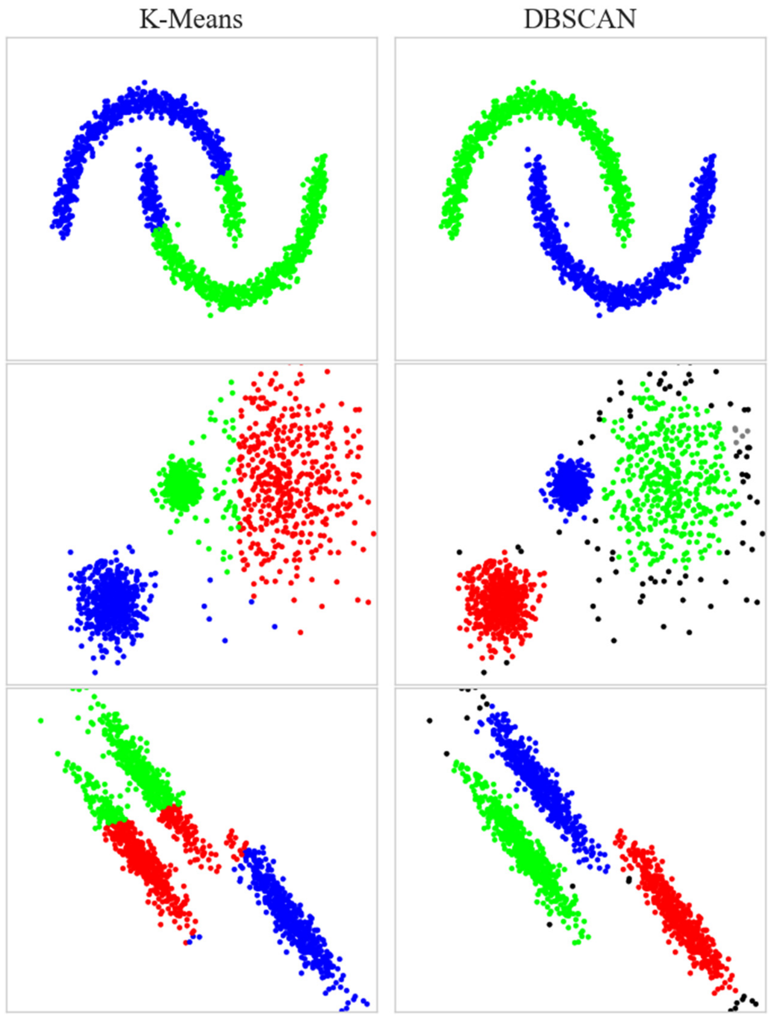


Figure 4.1 Partitioning versus clustering on synthetic data [63]

We define meta-attributes as a set of information-bearing attributes generated based on structural patterns, and distinctive features observed in the data which supports better performance of predictive models. Rather than simply recombining existing input attributes, these meta-attributes generated through supervised and unsupervised techniques are added to the existing input columns of the data. For example, a d-dimensional dataset which has an original input attribute vector X_d = {Attrib$_1$, Attrib$_2$, Attrib$_3$, ..., Attrib$_d$}; and after generating meta-attributes (with j-dimension), the updated dataset becomes either [15]:

Appended:

$$X_{d+j} = \{Attrib_1, Attrib_2, \ldots, Attrib_d\} \cup \{metaAttrib_1, metaAttrib_2, \ldots, metaAttrib_j\}$$

or Substituted:

$$X_d = \{Attrib_1, \ldots, Attrib_{d-j-1}, Attrib_{d-j}\} \cup \{metaAttrib_1, metaAttrib_2, \ldots, metaAttrib_j\}$$

We used the appending method in this research in other to avoid omission of latent information in attributes deemed less important. Figure 4.2 provides a simplified overview of how we achieve it.

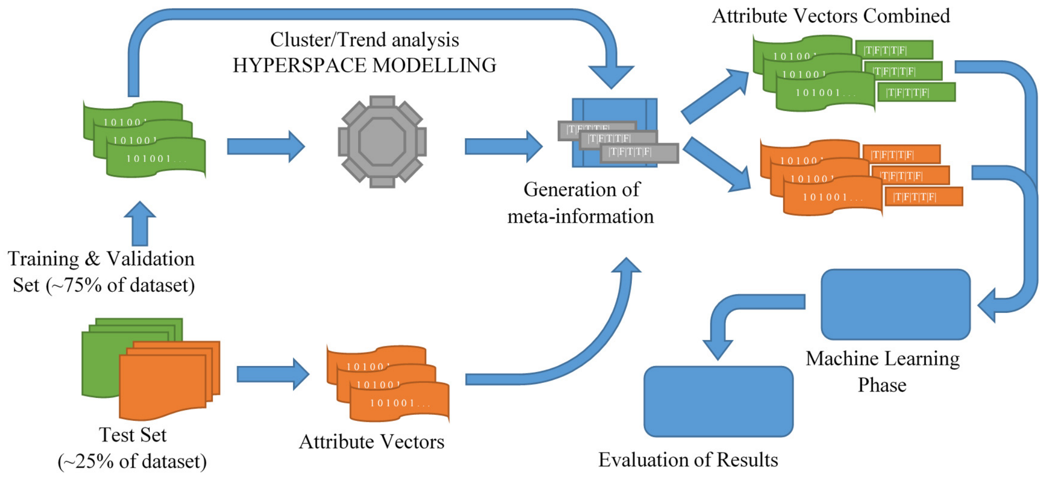

Attribute Vectors Combined
Cluster/Trend analysis
HYPERSPACE MODELLING
Generation of meta-information
Training & Validation Set (~75% of dataset)
Machine Learning Phase
Test Set (~25% of dataset)
Attribute Vectors
Evaluation of Results

Figure 4.2 Meta-information generation work-flow

Two widely used classification models are modified and tested for meta-assisted learning. They are the Constructive Backpropagation (CBP) [9], and the Multi-layer Perceptron (MLP) optimised by Limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm (L-BFGS) [64]. These algorithms are selected for their mainstream use, and ease of implementation and configuration.

In Constructive Backpropagation (CBP), the number of hidden neurons are automatically determined during training by evaluating the performance after each training epoch of the neural network. The self-growing neural network incrementally grows its width by the following steps [9]:

STEP 1: Initialise a network with no hidden units and minimise the sum squared error by adjusting the weights. The weight values are then fixed at the end of the training:

$$Error = \sum_{i=1}^{m}\sum_{j=1}^{o}(a_{ij} - d_{ij})^2 \tag{4.1}$$

where m is the number of training examples, o is the number of outputs to be learned, a_{ij} is the actual output and d_{ij} is the desired output.

STEP 2: Create the (n+1)th hidden unit to be added to the network and train the new hidden unit with the modified sum of square error formula:

$$ModifiedError_{n+1} = \sum_{i=1}^{m}\sum_{j=1}^{o}(f(\sum_{k=0}^{n}u_{kj}a_{ik} + u_{(n+1)j}a_{i(n+1)}) - d_{ij})^2 \tag{4.2}$$

where u_{kj} is the connection from the kth hidden unit to the jth output unit, and $u_{(n+1)j}$ is the connection from the (n+1)th hidden unit to the jth output unit.

STEP 3: The weights of the new network obtain in step 2 are now fixed.

STEP 4: The network performance is then evaluated and if acceptable no further hidden units are added, else repeat step 2.

The supervised learning algorithm, multi-layer perceptron (MLP), contains several layers between the input and output layers as shown in Figure 4.3. These middle layers referred to as the hidden layers are comprised of non-linear function approximators. Similar to the CBP, it utilises backpropagation for training.

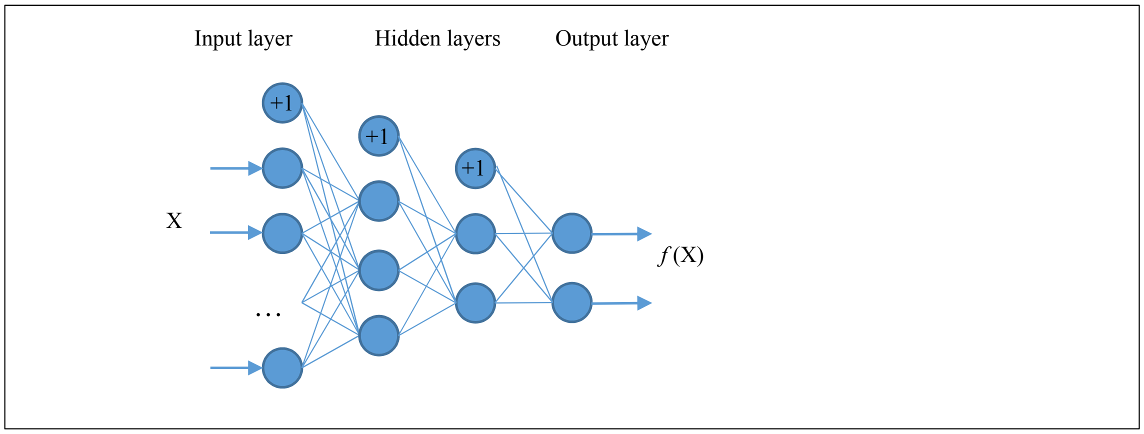


Figure 4.3 Multi-layer perceptron network architecture

4.1.1 Meta-information: Density-Based Spatial Clustering Application with Noise

For classification tasks, the Density-Based Clustering of Applications with Noise (DBSCAN) [36], [65] clustering technique is utilised for meta-information generation. DBSCAN algorithm requires a radius and minimum-number-of-points parameters to form clusters within the data (see Algorithm 4.1). The radius specifies a region which can be identified as dense, based on the number of points located within this area and the minimum number of points. DBSCAN designates either of this labels to each instance/point in the data: core, border, or noise point. A 'core' label is assigned when there is at least the minimum number of points (instances) within that radius. A 'border' label is assigned if it does not have the minimum number of data points around it but is within the radius of at least one core point. Finally, a point can be labelled as 'noise' when there are no other points within its radius. Also, a unique cluster-identifier is given to all points within every formed cluster.

Algorithm 4.1: DBSCAN simplified (refer to [15], [65], [66] for details)

```
Input:  dataset X, MinPts, Eps.
Output: Cluster ID vector C, Type vector V.
C = 0
for each point in X,
     count = number of neighbouring points within radius Eps.
     C = next cluster
     if  count>= MinPts, then
          assign point and neighbour data points to cluster C and set
          point type V as Core.
     elseif (count<MinPts)and(count>0), then
          assign point to cluster C and set its type V as Border.
     else
          set point type V as Noise.
     endif
endfor
return C, V
```

The major drawback in using DBSCAN is the difficulty of selecting the optimal parameter values in order to get the best clustering results [67]. Thus a parameter-search algorithm was used to test for the best formation of clusters. *Eps* is derived with the "volume of an n-ball" equation below [68]:

$$R_n(V) = \frac{\Gamma(\frac{n}{2}+1)^{1/n}}{\sqrt{\pi}} V^{1/n} \tag{4.3}$$

where *R* is the radius *Eps*, *V* is volume, *Γ* is the Leonhard Euler's gamma function, and *n* is the number of attributes in the dataset being processed [66]. In the parameter-search algorithm below (see Algorithm 4.2), the minimum number of points (*MinPts*), needed to form a cluster is increased if the percentage of border points exceeds the percentage of noise, or the percentage of core point is greater than a heuristic set value of 75%. If the number of clusters formed is unchanged after ten iterations, the search is ended and the *bestMinPts* selected. In [15], special cases are observed in which the percentage of core points identified falls below the set threshold without finding *bestMinPts*. Thus an alternative minimum-number-of-points for clustering is selected, and it is based on the first parameters that generate a percentage of border points that is higher than the percentage of noise.

Algorithm 4.2: DBSCAN Optimal Parameters-Search

```
Input:  dataset X
Output: MinPts, Eps
edit = True
threshold = 75

for MinPts 2:(25% of the number of instances in X)
    DBSCAN(X, MinPts)
    if ((number of clusters>1) && (%Noise<=best%Noise)
       && (%Border>%Noise||%Core>threshold))
           then bestMinPts=MinPts
    endif
    if (%Border > %Noise && edit)//executes only once
    //sets MinPts when Border points first exceeds Noise
        failSafeMinPts = MinPts
        edit = False
    endif
    if (number of cluster formed does not change after 10 iterations)
        break //break for loop
    endif
endfor
return best MinPts with corresponding Eps
```

The optimal output of Algorithm 4.1 is then used to cluster the training data to derive additional information about the structure of the grouped point/concepts and their outliers. After which each example in the test set is assigned the label (i.e. cluster_id) of the closest core point if it is within the radius *Eps*, otherwise it is labelled as an outlier. These meta-information are:

i. CLUSTER_ID: Depending on the number of clusters formed, an integer is assigned to each cluster, except noise/outlier points.

ii. POINT_TYPE: Each instance is labelled with either 1, 0, or -1 corresponding to the DBSCAN cluster point type which are core point, border point, or noise respectively.

4.1.2 Classification Architecture for Meta-assisted Learning

We propose and implement a traditionally flat model, a hierarchically structured model, and also perform "output-class" partitioning for each one of these structures to transform the data into useful/meaningful representations such that any classification model can benefit from the organised local details and global context within the dataset.

Output-class partitioning employs a "divide-and-conquer" approach to improve parallelisation and reduce internal interference significantly by dividing the dataset into distinct classes to perform machine learning on each sub-data before being merged to get the final output [13]. We, therefore, apply a similar technique before cluster analysis on the data to discover how multiple clusters within a single class are spatially distributed to reveal local details and identify potential anomalies/noise.

4.2 Method

4.2.1 Flat Structure Meta-information Learning

The flat structure meta-assisted learning process is illustrated in Figure 4.4 and Figure 4.5, in which we begin by splitting the dataset into training and test sets. Then cluster analysis is performed on only the training set to generate meta-information for each instance. The artificial neural network is then modelled on this modified dataset. In order to predict the class of the test instances, the meta-information is first applied to the test set via the nearest neighbour technique, whereby the meta-attribute of the closest training instance is assigned to each selected test instance. Performance evaluation of the modified model is then used to assess the enhancement over an exact copy of the original model.

The primary difference between the two flat structure methods is the application of output partitioning. The two algorithms are:

i. Meta-assisted learning on a flat structure (MT_F)
ii. Meta-assisted learning on a flat structure using output-class partitioning (MT_FC)

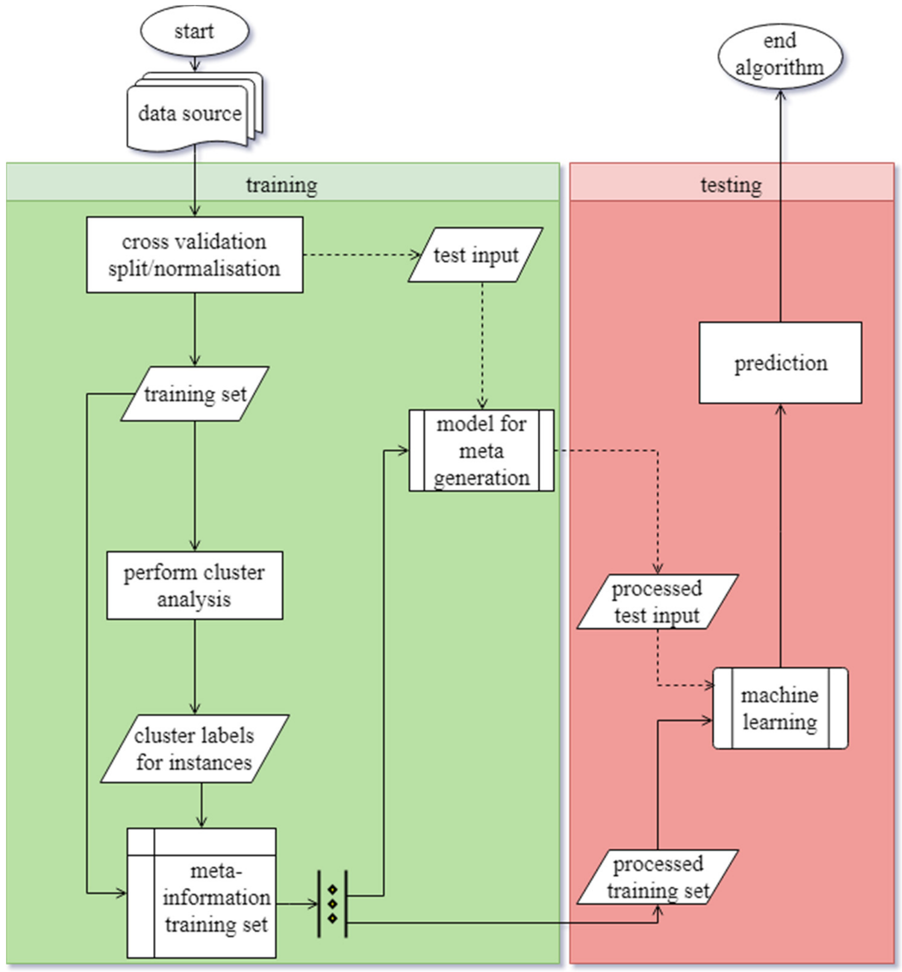


Figure 4.4 Meta-assisted learning on a flat structure model (MT_F)

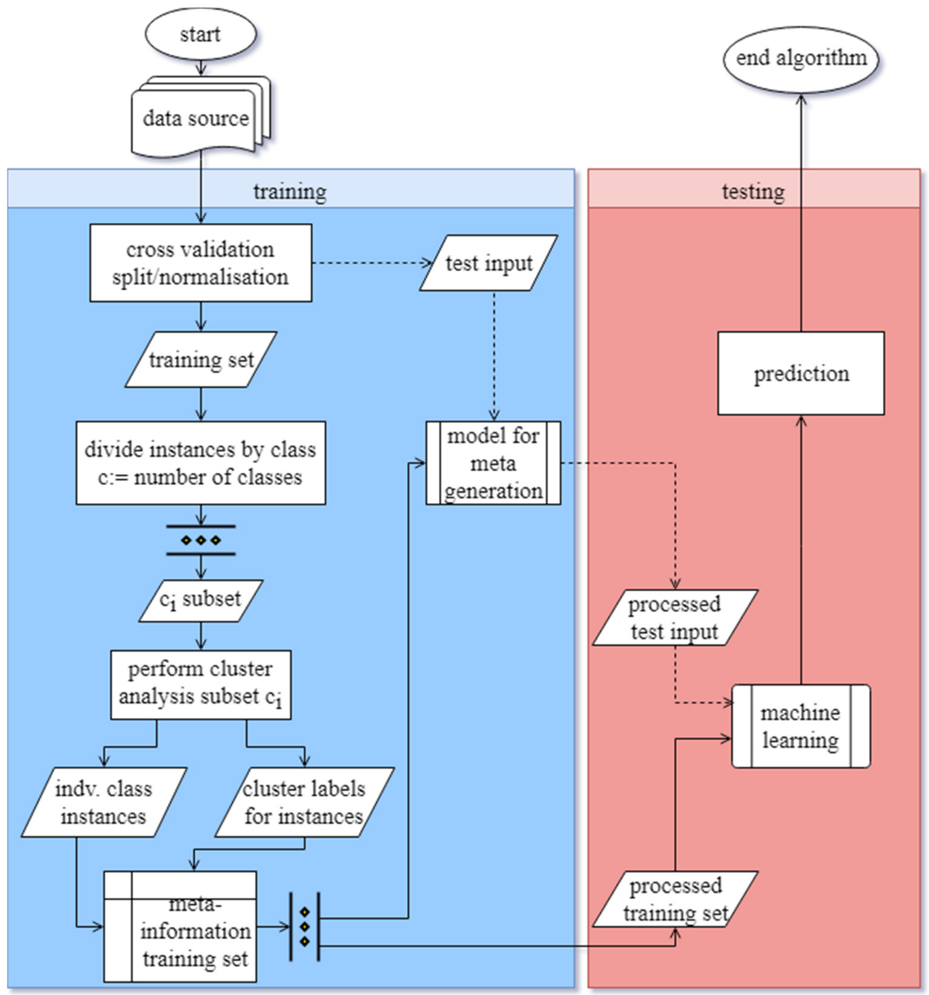

start
data source
training
testing
end algorithm
cross validation split/normalisation
test input
training set
prediction
divide instances by class c:= number of classes
model for meta generation
c_i subset
processed test input
perform cluster analysis subset c_i
machine learning
indv. class instances
cluster labels for instances
meta-information training set
processed training set

Figure 4.5 Meta-assisted learning on a flat structure model using output-class partitioning (MT_FC)

4.2.2 Hierarchical Meta-information Learning

For the hierarchical structure, we use similar methods whereby we apply cluster analysis and output-class partitioning during the meta-information generation phase. The sub-models in the middle layer produce intermediate results from the meta-information-modified dataset and the original dataset to get a robust solution to the classification task; especially in situations where instance clusters do not reveal any useful information. Finally, a top layer neural network is used to integrate the results from the sub-models. The two algorithms as shown in Figure 4.6 and Figure 4.7 are:

1. Meta-assisted learning on a hierarchical structure (MT_H)
2. Meta-assisted learning on a hierarchical structure using output-class partitioning (MT_HC)

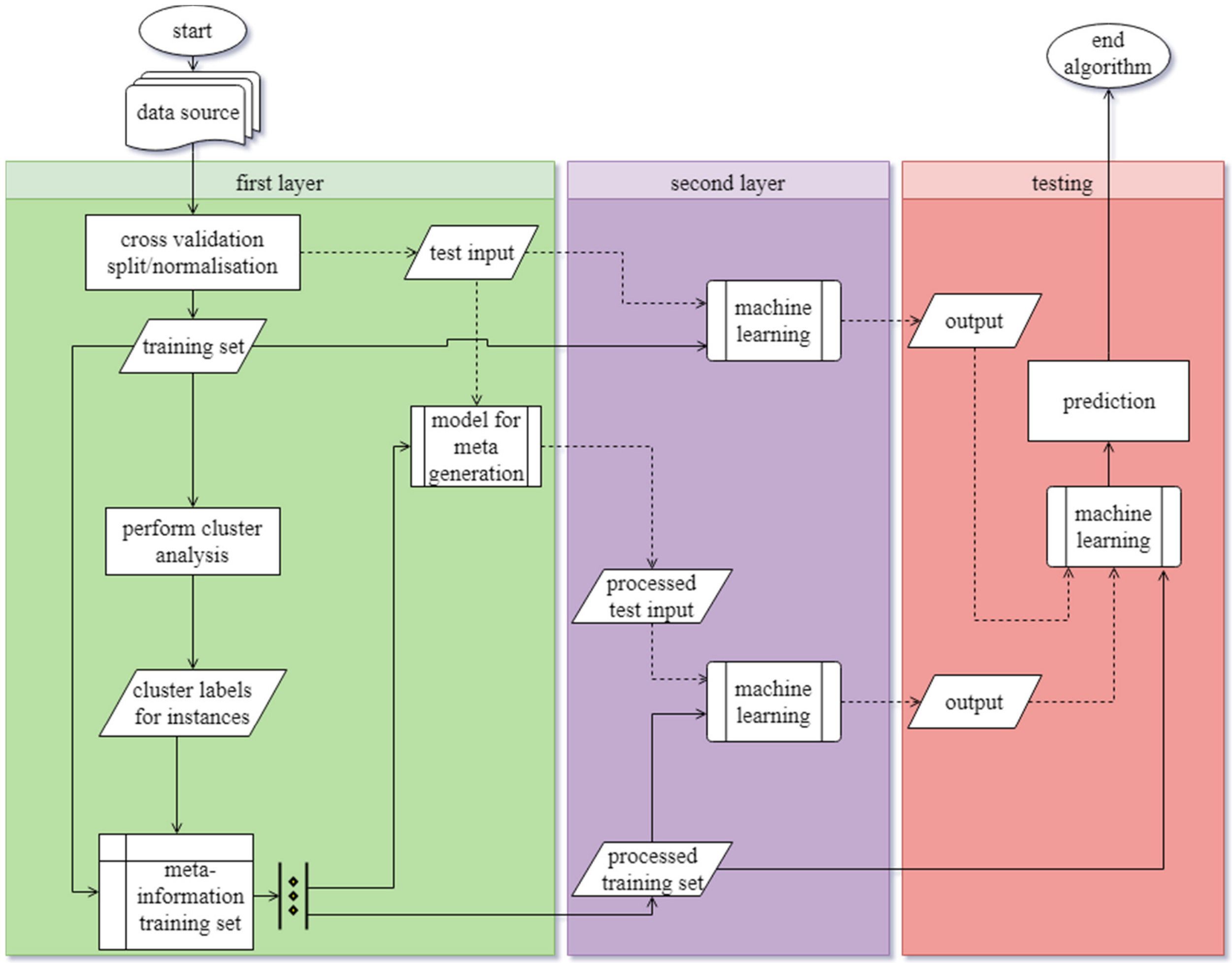


Figure 4.6 Meta-assisted learning on a hierarchical structure model (MT_H)

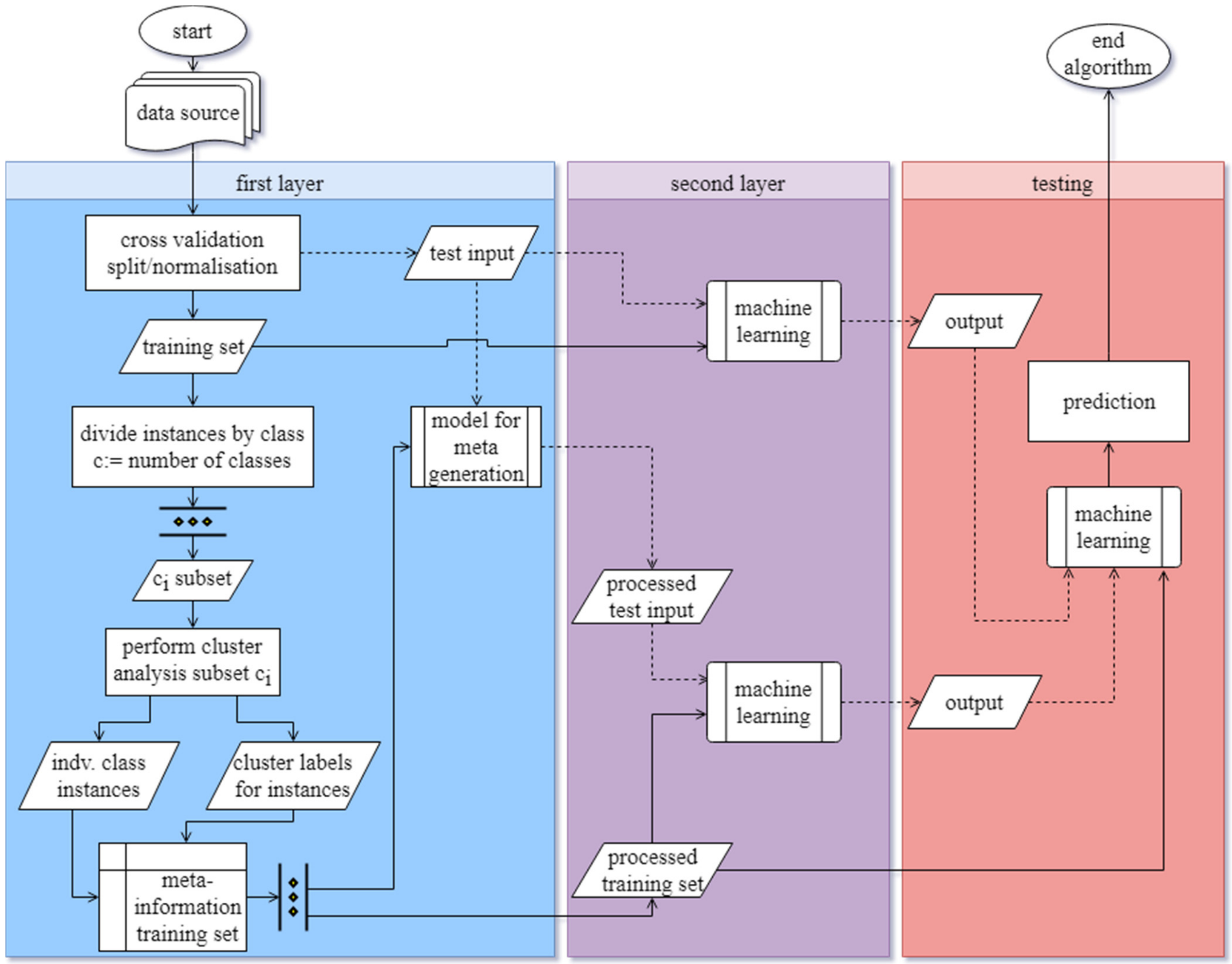


Figure 4.7 Meta-assisted learning on a hierarchical structure model using output-class partitioning (MT_HC)

4.2.3 Benchmark Datasets

The empirical evaluation of the proposed algorithms was performed on 18 benchmark datasets which are retrieved from real-world and synthetic data. These datasets include low, medium, and high dimensionalities, with the number of output classes ranging from 2 to 11 for classification tasks. Table 4.1 summarises the properties of each dataset namely: concentric circles (CCLS) [62], mice protein expression (CORTEX) [69], glass identification (GLASS) [62], SPECTF heart (HEART) [70], hepatitis (HEP) [71], hill-valley (HILL) [71], high time resolution universe pulsar survey (HTRU_2) [72], ionosphere (IONO) [73], iris plant. (IRIS) [74], seeds (SEED) [75], connectionist bench sonar, mines vs. rocks (SONAR) [76], user knowledge modelling (USRKNWL) [77], statlog vehicle silhouettes (VEHICLE) [78], connectionist bench vowel recognition - deterding data (VOWEL) [71], [79], wine (WINE) [80], breast cancer wisconsin original (WIS) [81], yeast (YEAST1) [71], [79], and zoo (ZOO) [71].

4.3 Results and Analysis

Each dataset was split for 5-fold cross-validation and then normalised. The results in Table 4.1 are the average performance accuracy in 100 runs for each of the four designed algorithms which are compared with the original artificial neural network. The ANN were all initialised with two hidden layers of 15 and 10 units respectively, and corresponding input and output node based on the number of attributes and classes of the dataset. As mentioned earlier, the L-BFGS [64] optimiser is used in all ANNs, and the early stopping technique is utilised to prevent poor generalisation. The tests were timed and other essential metrics recorded and presented in Appendix A. These include 25th, 50th, and 75th percentile, and the *p*-value and percentage improvement of the best algorithm versus the others. For a more obvious comparison in Table 4.1, the percentage accuracy of all the algorithms in each dataset row has been colour-coded with the colour-gradient from red through yellow to green, where deep red cells signify worst performance and deep green reveal the best algorithm. The box plots from Figure 4.8 to Figure 4.16 indicate the distribution of the observed accuracies and reveals outliers which are representative of the algorithm's general performance and stability over several random initialisations. They are labelled as:

i. Traditional approach using original data (ORIG): blue
ii. Meta-assisted learning on a flat structure (MT_F): green

iii. Meta-assisted learning on a flat structure using output-class partitioning (MT_FC): red

iv. Meta-assisted learning on a hierarchical structure (MT_H): purple

v. Meta-assisted learning on a hierarchical structure using output-class partitioning (MT_HC): dark yellow

By assigning a weighted score of one to the algorithm with the highest percentage on each tested dataset, it can be seen in Table 4.1 that meta-assisted learning algorithms MT_FC and MT_H performed relative well after the noise/outlier within the data was identified. The percentage improvement of the meta-assisted learning algorithm over the unaided classifier is also shown in Table 4.1.

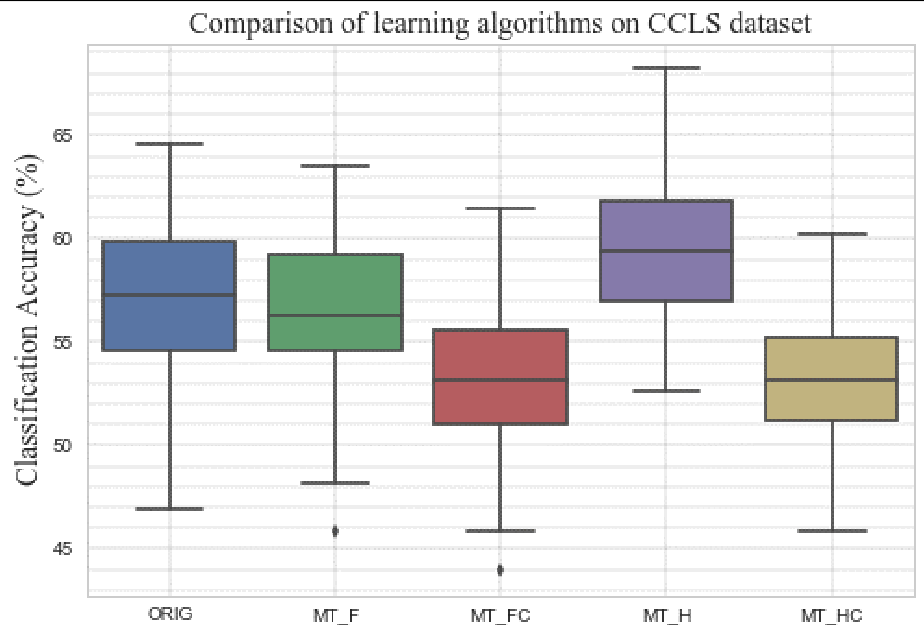
In the box plots, we see shorter vertical lines (whiskers) which indicate a smaller standard deviation on the hierarchical design, signifying less variation. From these figures, we can clearly identify which datasets benefit from cluster analysis on output-class partitioned data. From the HILL dataset box plot, we see that the boxes are flat because a significant number of observed accuracy is 100%, on the other hand, only MT_FC performed flawlessly without any outlier in the plot.

In general, we see fewer outliers compared to the existing traditional method for classification with data noise/anomaly. The *p*-value columns in Appendix Table 8.1 to Table 8.6 show the statistical significance for the best performing algorithm in each dataset. A *p*-value less than the 0.05 threshold—highlighted purple in the tables—is considered to be statistically significant; therefore we can reject the null hypothesis thus accepting the alternative. In other words, we can show that the meta-assisted learning algorithm's benefits are valid. More specifically, by tallying the number of occurrences when each algorithm is shown to be statistically valid, MT_FC and MT_H rank the best, followed by MT_HC, MT_F, and then lastly the default/traditional algorithm.

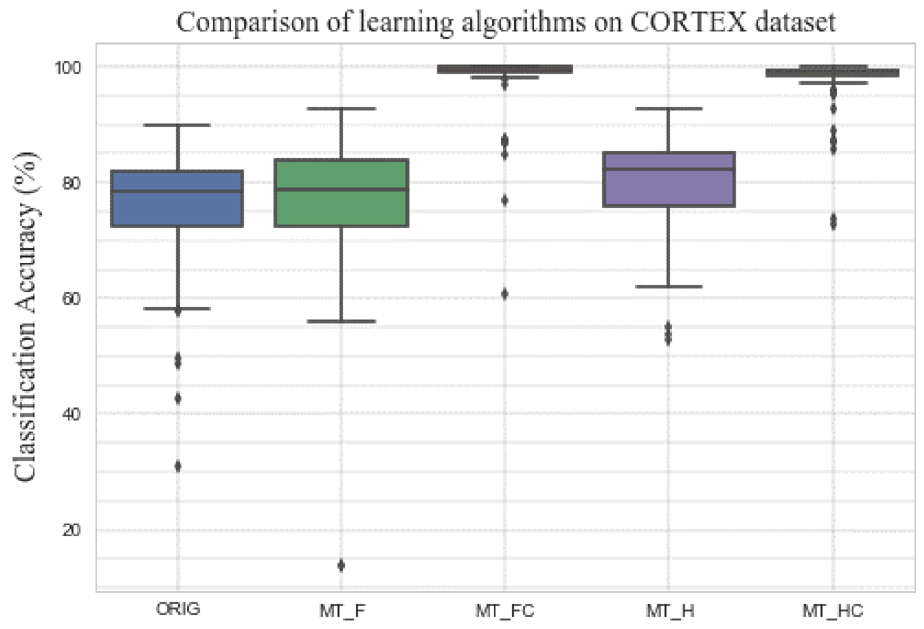
It is important to point out that applying output class partitioning before cluster analysis produces a significant enhancement to the flat architecture, MT_FC. Applying output-class partitioning to the hierarchical architecture does not produce similar benefits possibly because output-class partitioning becomes redundant in this kind of pre-structured learning. Notwithstanding, a few of the dataset benefited from the output-class and hierarchical learning. Therefore, it can be possible to combine the contributions from the meta-assisted learning from MT_H and MT_HC algorithms by determining when clustering with output-class partitioning is advantageous.

Table 4.1 The Mean Performance Accuracy of Algorithms

Dataset	Examples	Attributes	Classes	ORIG	MT_F	MT_FC	MT_H	MT_HC	%Δ vs ORIG
CCLS	959	2	2	57.273	56.565	53.103	59.369	53.171	3.66
CORTEX	1080	77	8	75.764	76.667	98.102	80.213	97.824	29.48
GLASS	214	9	6	57.433	58.643	72.045	59.552	71.442	25.44
HEART	270	13	2	81.981	81.426	65.722	82.056	66.556	0.09
HEP	80	19	2	78.237	77.949	77.625	78.258	77.666	0.03
HILL	1212	100	2	98.859	99.422	100.000	99.463	99.010	1.15
HTRU_2	17898	8	2	97.501	97.561	96.145	97.483	96.156	0.06
IONO	350	34	2	90.457	91.914	91.803	92.401	92.363	2.15
IRIS	150	4	3	95.200	95.133	96.233	95.533	96.300	1.16
SEED	210	7	3	90.190	91.310	92.167	91.571	91.786	2.19
SONAR	208	60	2	77.691	78.245	78.220	77.638	78.667	1.26
USRKNWL	403	5	4	60.539	60.878	51.303	60.844	50.986	0.56
VEHICLE	846	18	4	62.970	63.009	66.817	68.289	66.748	8.45
VOWEL	990	10	11	74.374	75.788	88.949	65.874	80.051	19.60
WINE	178	13	3	74.491	72.883	82.563	77.310	82.310	10.84
WIS	616	9	2	88.288	87.499	89.327	87.828	89.396	1.26
YEAST1	1484	8	2	76.556	76.587	72.305	76.924	72.321	0.48
ZOO	101	16	7	90.376	91.961	92.135	89.373	90.821	1.95
WEIGHTED SCORE				0	2	7	6	3	
AVERAGE TRAINING TIME				0.203	0.740	0.596	1.135	0.865	

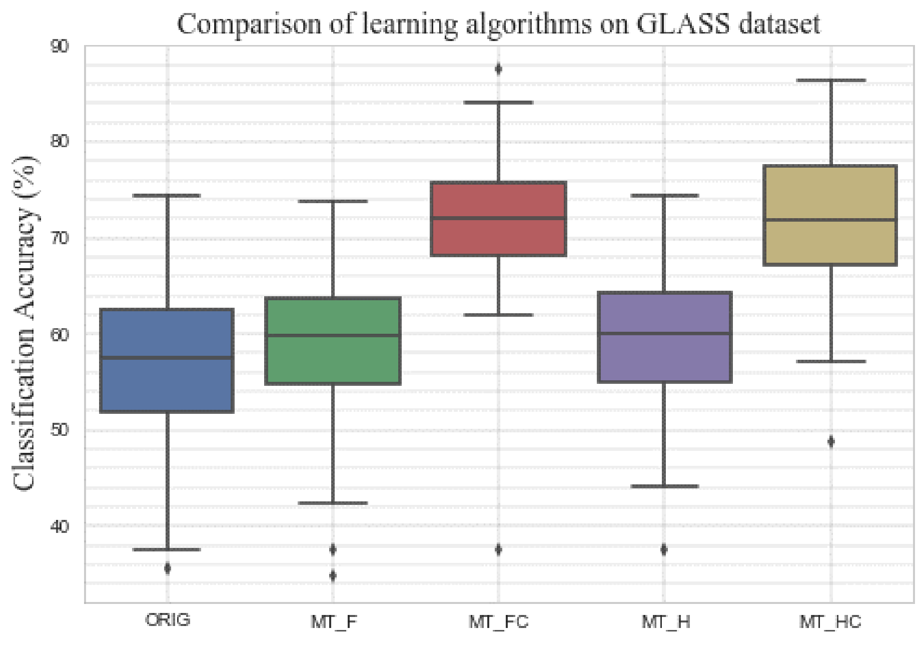


a

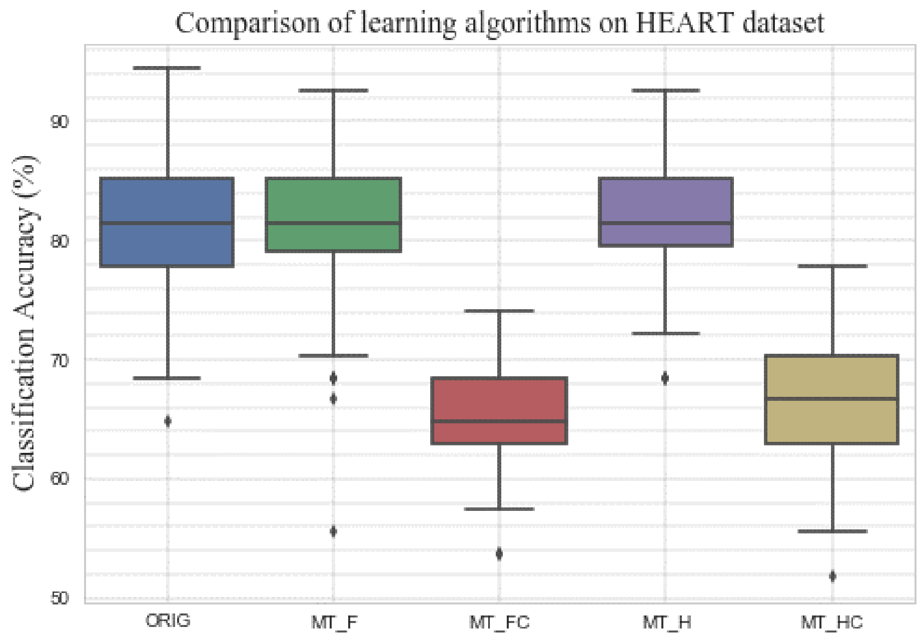


b

Figure 4.8. Box plot of classification accuracy on datasets a-b

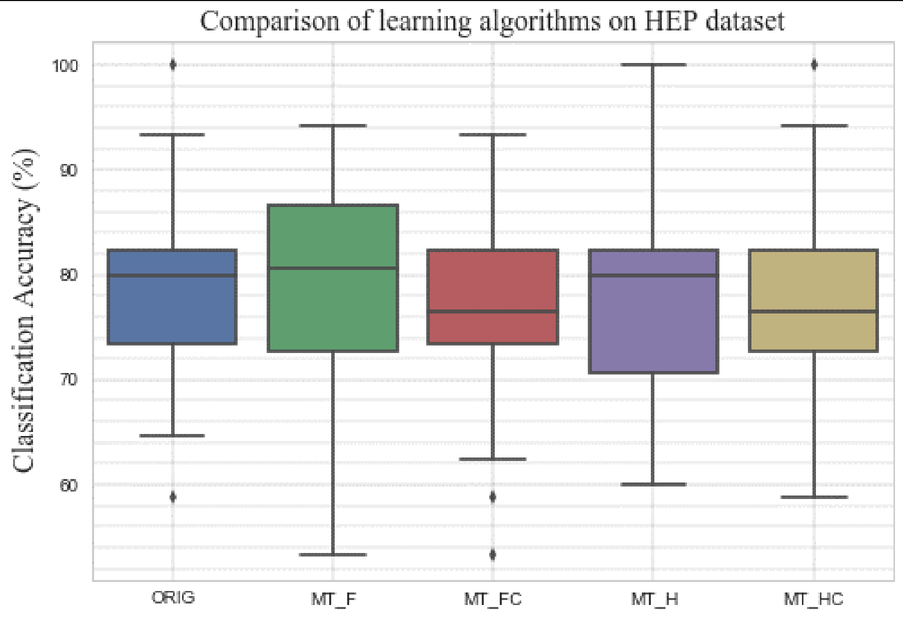


c

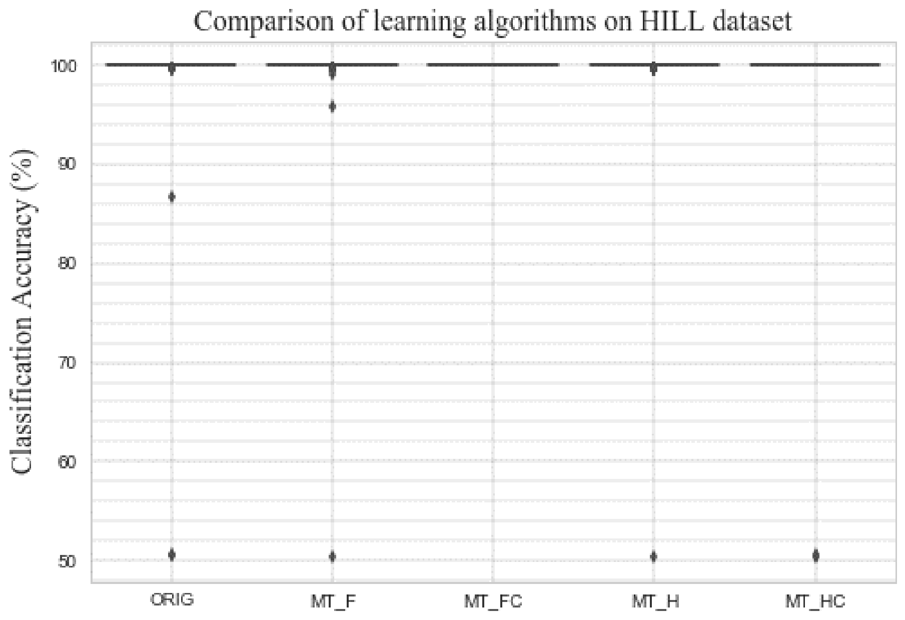


d

Figure 4.9. Box plot of classification accuracy on datasets c-d

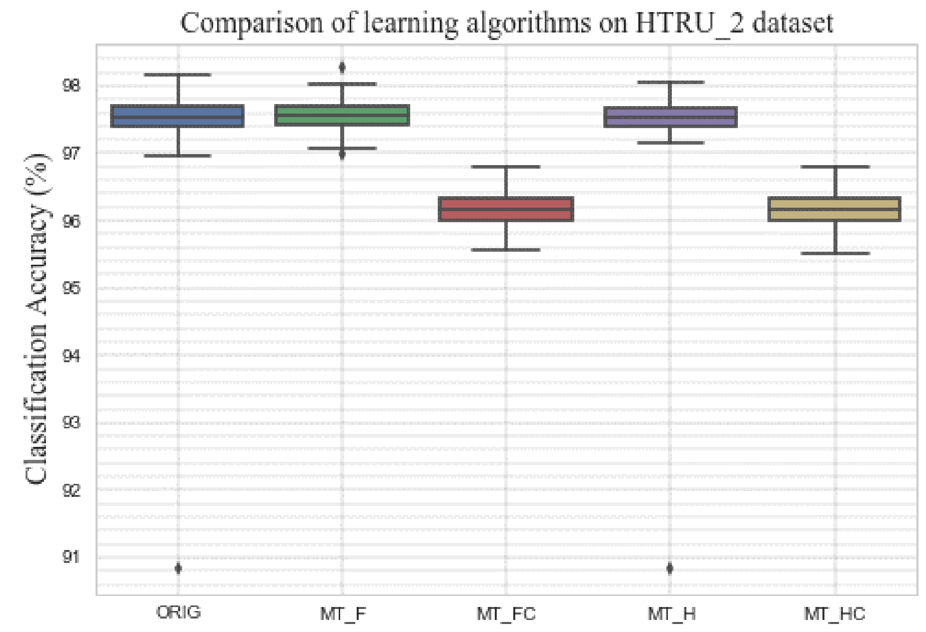


e

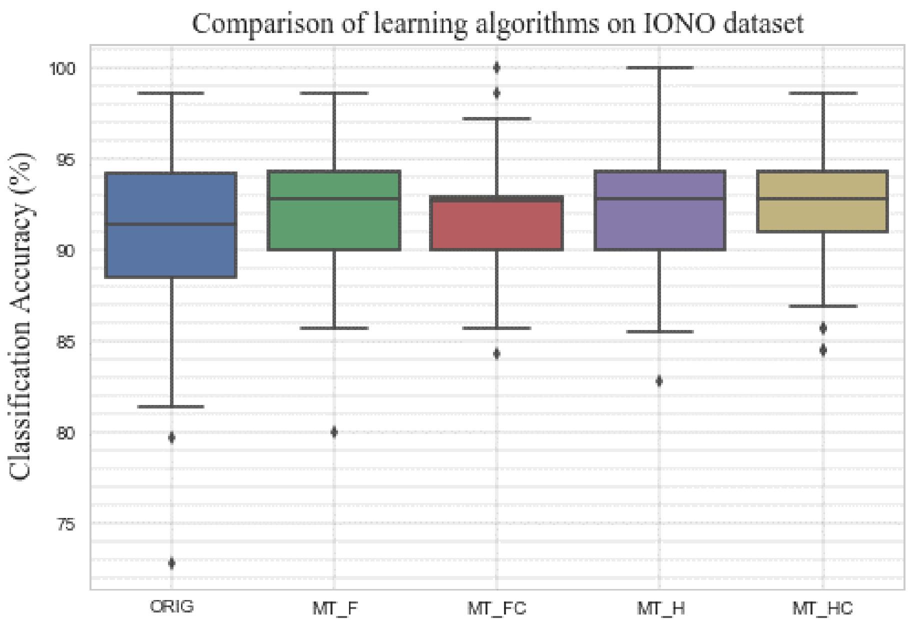


f

Figure 4.10. Box plot of classification accuracy on datasets e-f

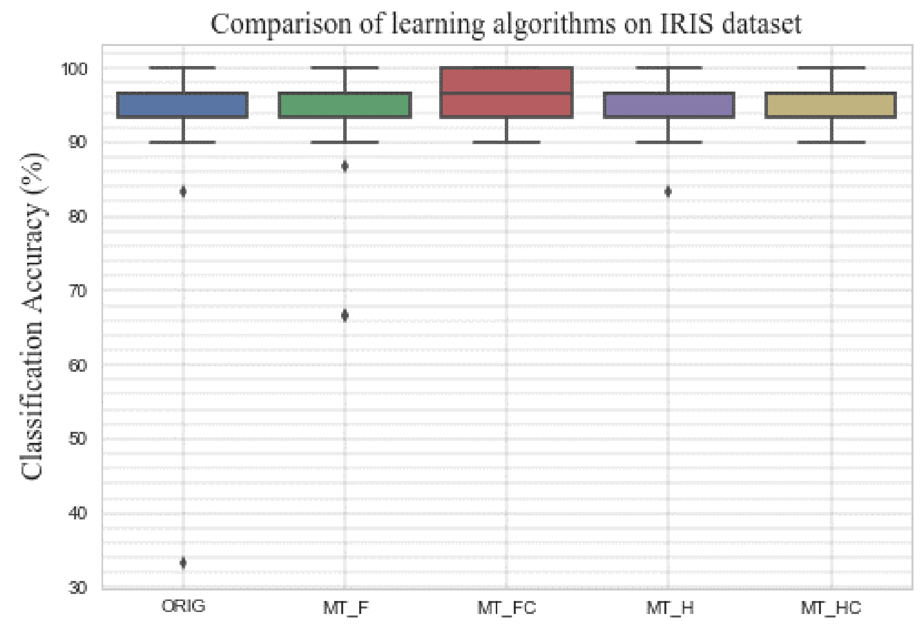


g

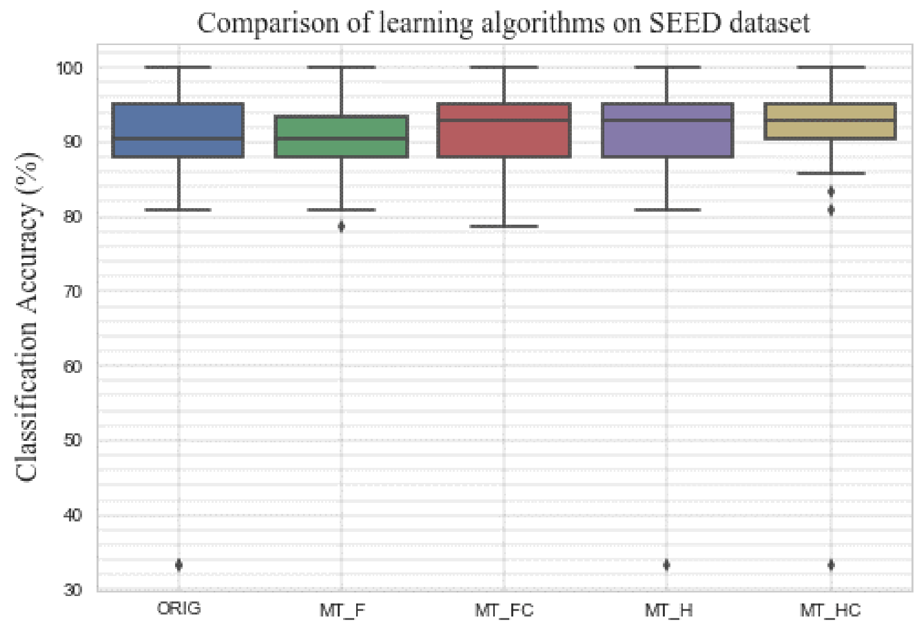


h

Figure 4.11. Box plot of classification accuracy on datasets g-h

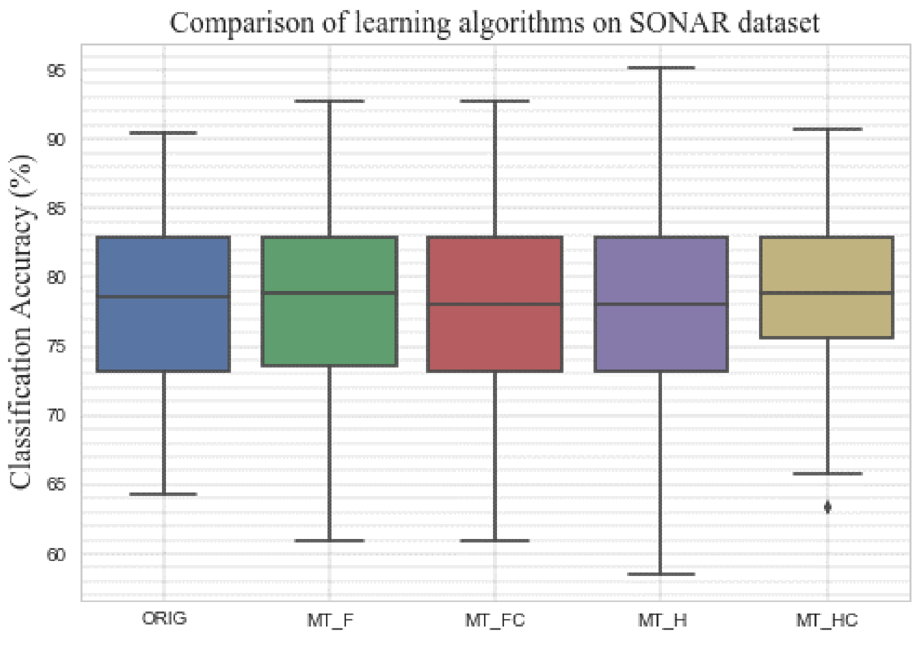


i

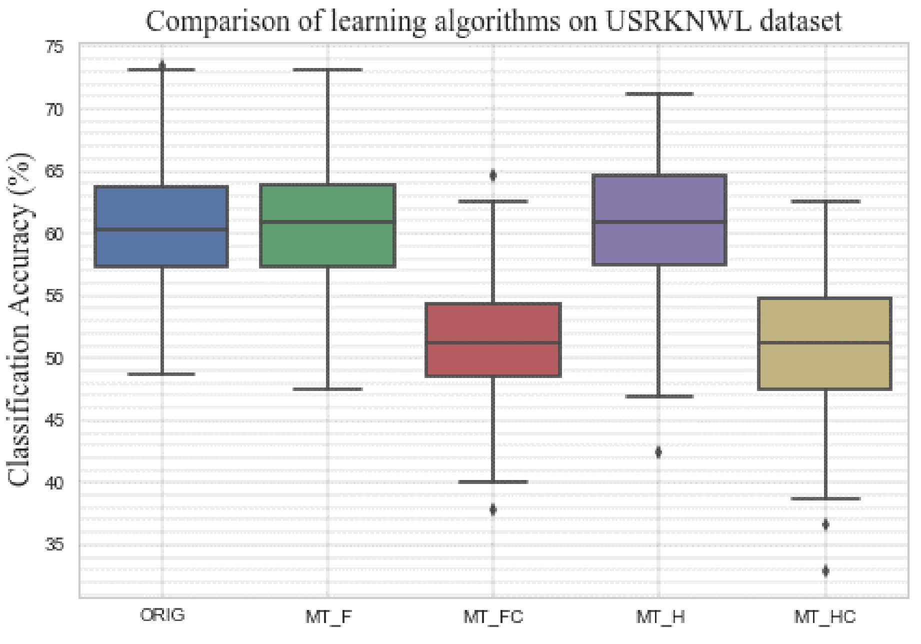


j

Figure 4.12. Box plot of classification accuracy on datasets i-j

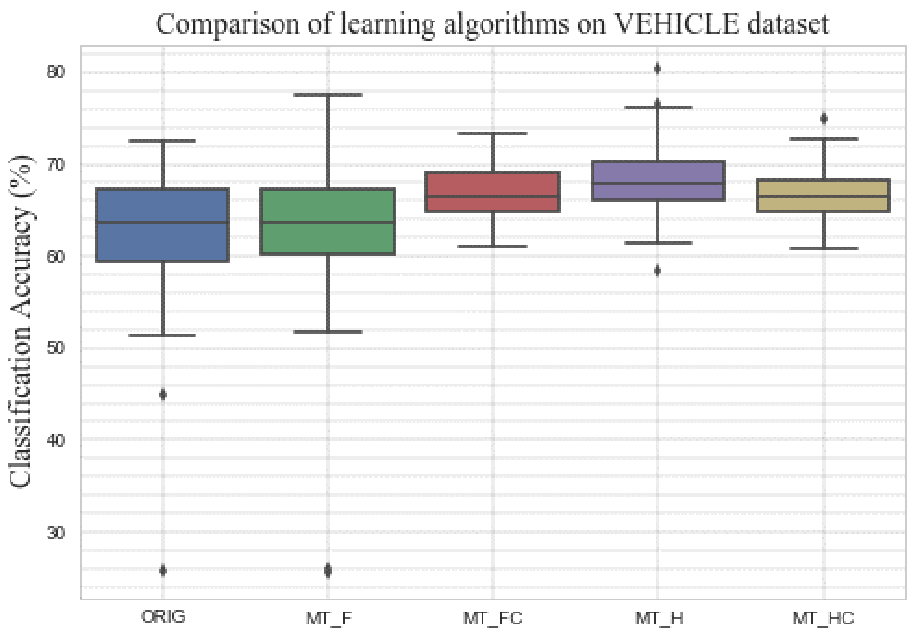


k

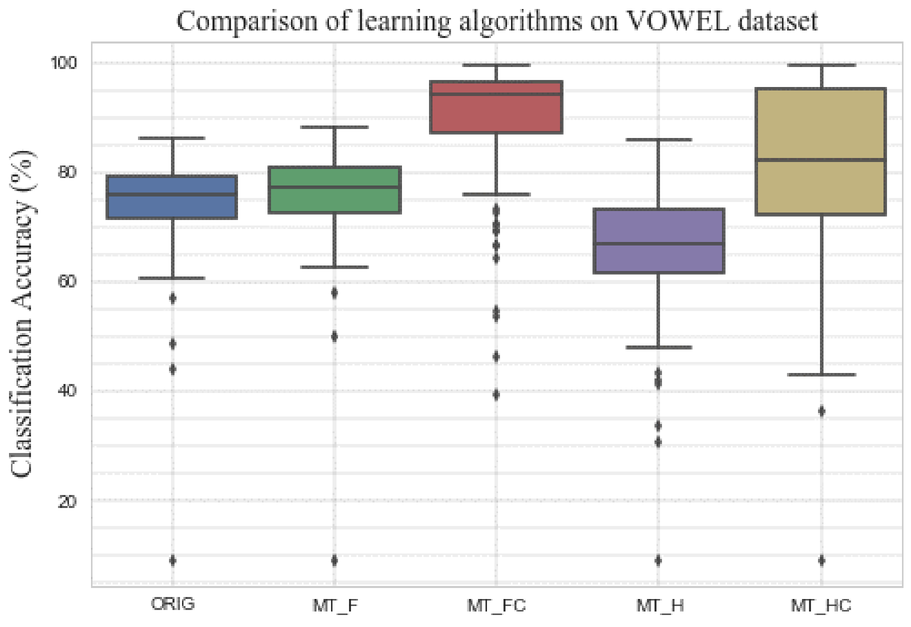


l

Figure 4.13. Box plot of classification accuracy on datasets k-l

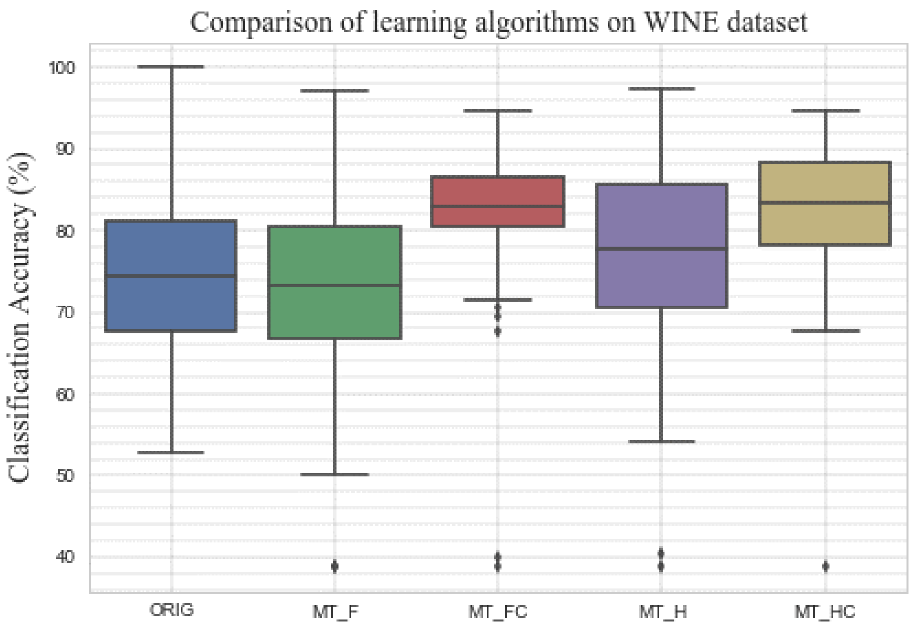


m

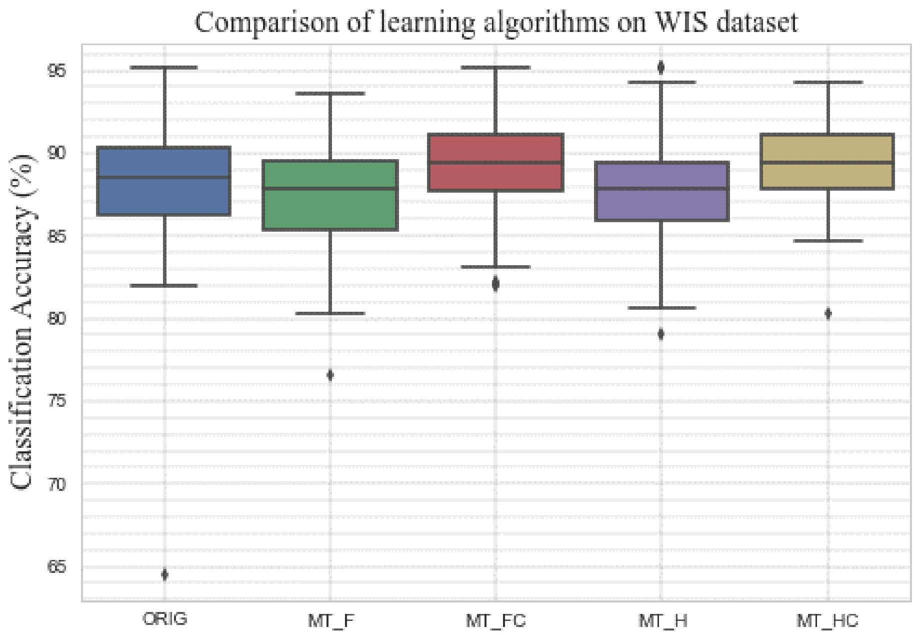


n

Figure 4.14. Box plot of classification accuracy on datasets m-n

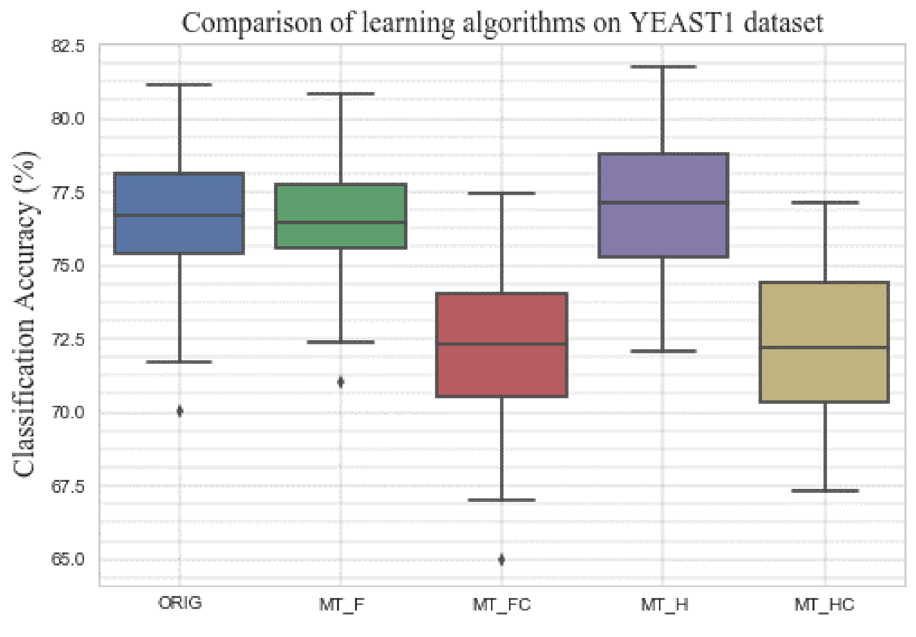


o

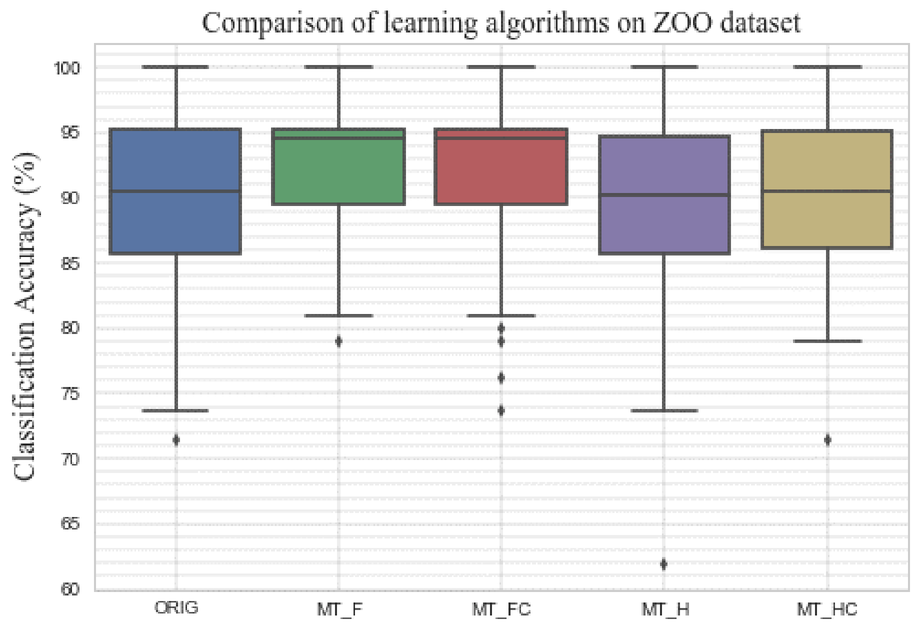


p

Figure 4.15. Box plot of classification accuracy on datasets o-p



q



r

Figure 4.16. Box plot of classification accuracy on datasets q-r

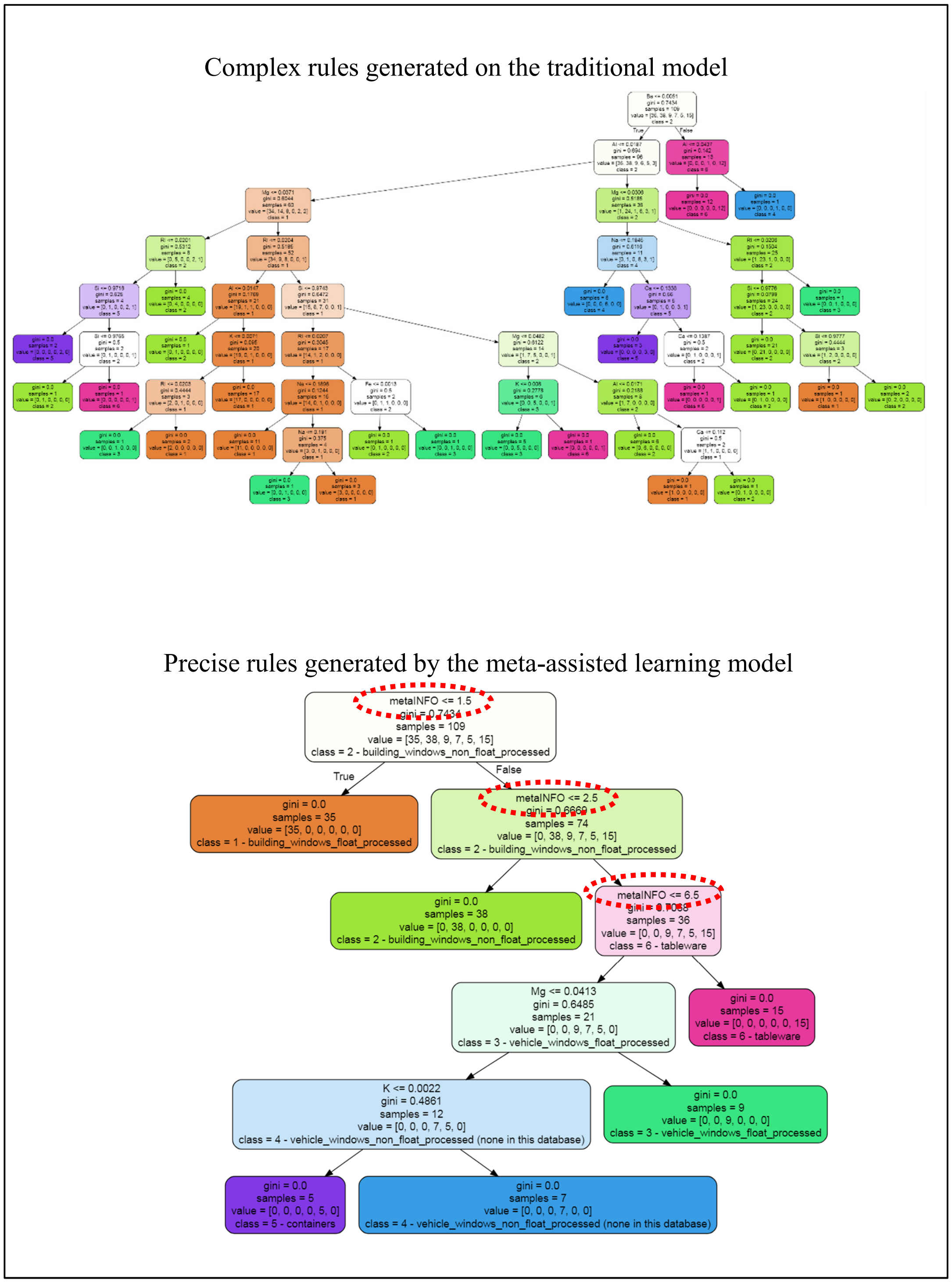

Complex rules generated on the traditional model
Precise rules generated by the meta-assisted learning model
metalNFO <= 1.5
gini = 0.7434
samples = 109
value = [35, 38, 9, 7, 5, 15]
class = 2 - building_windows_non_float_processed
True
False
gini = 0.0
samples = 35
value = [35, 0, 0, 0, 0, 0]
class = 1 - building_windows_float_processed
metalNFO <= 2.5
gini = 0.6669
samples = 74
value = [0, 38, 9, 7, 5, 15]
class = 2 - building_windows_non_float_processed
gini = 0.0
samples = 38
value = [0, 38, 0, 0, 0, 0]
class = 2 - building_windows_non_float_processed
metalNFO <= 6.5
gini = 0.7068
samples = 36
value = [0, 0, 9, 7, 5, 15]
class = 6 - tableware
Mg <= 0.0413
gini = 0.6485
samples = 21
value = [0, 0, 9, 7, 5, 0]
class = 3 - vehicle_windows_float_processed
gini = 0.0
samples = 15
value = [0, 0, 0, 0, 0, 15]
class = 6 - tableware
K <= 0.0022
gini = 0.4861
samples = 12
value = [0, 0, 0, 7, 5, 0]
class = 4 - vehicle_windows_non_float_processed (none in this database)
gini = 0.0
samples = 9
value = [0, 0, 9, 0, 0, 0]
class = 3 - vehicle_windows_float_processed
gini = 0.0
samples = 5
value = [0, 0, 0, 0, 5, 0]
class = 5 - containers
gini = 0.0
samples = 7
value = [0, 0, 0, 7, 0, 0]
class = 4 - vehicle_windows_non_float_processed (none in this database)

Figure 4.17. Decision Trees generated from fitting the model on the GLASS dataset

4.4 Concluding Remarks

In this work, the application of meta-assisted learning for noise reduction in classification task is shown to improve the performance accuracy. It shows how cluster analysis of the data in a flat architecture (with output-class partitioning) or hierarchical architecture can complement existing approaches to machine learning when the data contains noisy or interfering attributes. Therefore rather than deleting the noisy data as recommended in related research, we propose identifying structural patterns and uniquely labelling anomalies and outliers as well as groups of points that represent distinct concepts with meta-information to enable any selected machine learning algorithm to build a more accurate model based on the modified data. A white box model such as Decision Trees [63], [82] enables interpretation of the model to reveal how the meta-information aided the learning process which validates our hypothesis. As shown in Figure 4.17, the meta-assisted learning model generated simpler and accurate set of rules with the meta-information used as one of the input features thereby eliminating interference within the original input attributes.

To further the research direction proposed in this chapter, we expect that an automated decision protocol to determine when to apply output-class partitioning and selecting between flat and hierarchical architectures; such as the voting ensemble used in [83] to select base classifiers by their level of confidence on the classification task. This protocol can improve the applicability and eliminate extra processing time which we observed. Additionally, exploring the viability of the Hierarchical Density-Based Clustering of Applications with Noise (HDBSCAN) [84] to extract variable density clusters based on the stability of the points within the cluster for meta-assisted learning.

5 Interference-less Learning in Forecasting tasks

Forecasting involves building a model on a sequence of past values with the aim of accurately predicting the next value in the sequence. Examples of real-world univariate and multivariate forecasting are the stock market prediction, weather or natural phenomena prediction, human activity classification, sign language identification, energy management, and control engineering. In such predictions, the accuracy performance correlates to the quality of the data being modelled; in essence, the usefulness of the prediction heavily depends on the training sample and its clarity. The primary aim of this research is to develop a systematic technique to enhance the machine learning process by eliminating noise which is observable in short- and long-term trends of the time series.

We, therefore, developed a set of algorithms to exploit component trend patterns with a high information-bearing capacity and eliminate other low information-bearing components to prevent the model from learning inconsistent trends.

5.1 Background

Noise in time series forecasting can have a cumulative impairment on the prediction of values *n* steps ahead; therefore, algorithms that can build an accurate model based upon an available data sequence *n* steps before whilst reducing the impact of high- or low-frequency noises within the data can be useful in predicting short- or long-term values. The impact of noise is noticeable when predicting values many steps ahead; it is, therefore, essential to identify low- or high-frequency components that are inconsistent. In existing research in [85], the importance of noise management was stated, and from our preliminary research in [86] we observed that noise could occur in both low- or high-frequency components of the time series.

In univariate time series prediction, the non-linear autoregressive model (NAR) is commonly used when a sequence of data has only one series, *y(t)*; the future value is predicted using only *d* number of past values, where the function *f* can be estimated with a multilayer perceptron (MLP) [87]:

$$y(t) = f\big(y(t-1), \dots, y(t-d)\big) \tag{5.1}$$

On the other hand, when time series forecasting has one or more input series (in a vector $\vec{x}$) which are necessary for prediction thereby utilising both input vector and outputs of past values over a fixed window, d, a non-linear autoregressive exogenous model (NARX) is required:

$$y(t) = f\big(\vec{x}(t-1), \dots, \vec{x}(t-d), y(t-1), \dots, y(t-d)\big) \tag{5.2}$$

For example, rather than using the past week's temperature alone to predict tomorrow's temperature, a more accurate prediction can be achieved if multiple factors such as wind speed, humidity and seasonal trends are applied as inputs. This multivariate data is then used to train the artificial neural network in conjunction with the temperatures of past days or weeks. Figure 5.1 shows a fundamental NARX model structure with a delayed or lagged feedback from the output connected back to the input unit to create a feedback loop.

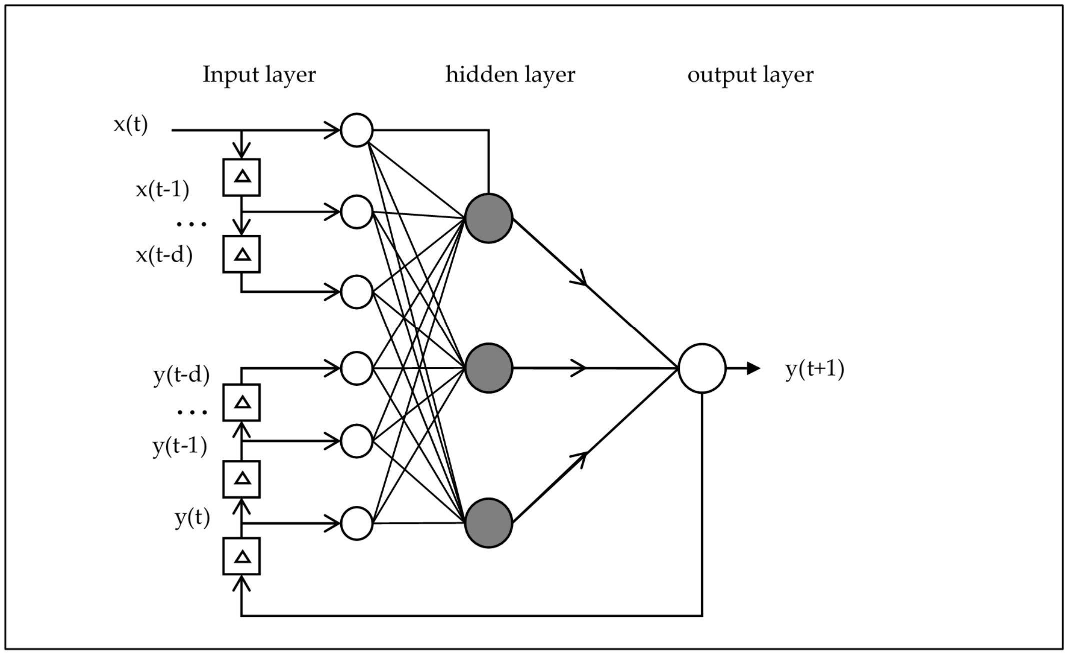


Figure 5.1. Nonlinear autoregressive exogenous network architecture

The NARX model can also be expressed in a state-space form as discussed in [87], wherein additional properties can be illustrated in the form:

$$y_i(t+1) = \begin{cases} \Psi[x(t), y(t)] & i = 1 \\ y_{i-1}(\mathrm{t}) & i = 2, \dots, d \end{cases} \tag{5.3}$$

where y is a scalar output at a given time, the subscript term i controls the limit of recursion within the predefined delay window, function Ψ represents MLP mapping, and d is the order (or output delay window). By deriving the Jacobian of the state-space map of Equation 5.3, it is revealed that the NARX network will deteriorate due to long-term dependencies and vanishing gradients [87]. It may, therefore, be intuitive to learn short-term and long-term patterns separately in a hierarchical manner to prevent interference; this is further discussed in our proposed learning algorithm.

Another predominant model in time series forecasting is called the long short-term memory (LSTM) and is a variation of the recurrent neural network (RNN) [88]. The LSTM is an artificial neural network with a chain-like connection that enables it to loop information between sigmoid function processing units, which are comprised of an input gate, forget gate, and an output gate. These gates control the LSTM's states to adequately deal with sequential data, thus making it popular in the field of natural language processing (NLP). The RNN and its variants such as LSTM, and gate recurrent unit (GRU) can be characterised as having a sequential architecture [88].

As LSTM models a data sequence, x, it can be expressed by the following equations [88]:

$$\mathrm{i}_t = \sigma\left(x_t \mathrm{U}^i + \mathrm{h}_{t-1}\mathrm{W}^i + \mathrm{b}_i\right) \tag{5.4}$$

$$\mathrm{f}_t = \sigma\left(x_t \mathrm{U}^f + \mathrm{h}_{t-1}\mathrm{W}^f + \mathrm{b}_f\right) \tag{5.5}$$

$$\mathrm{o}_t = \sigma(x_t \mathrm{U}^o + \mathrm{h}_{t-1}\mathrm{W}^o + \mathrm{b}_o) \tag{5.6}$$

$$\mathrm{q}_t = \tanh\left(x_t \mathrm{U}^q + \mathrm{h}_{t-1}\mathrm{W}^q + \mathrm{b}_q\right) \tag{5.7}$$

$$\mathrm{p}_t = \mathrm{f}_t * \mathrm{p}_{t-1} + \mathrm{i}_t * \mathrm{q}_t \tag{5.8}$$

$$\mathrm{h}_t = \mathrm{o}_t * \tanh(\mathrm{p}_t) \tag{5.9}$$

The hidden state h_t at current time step t is generated by each temporary result from the equations above, where U and W are distinct parameter/weight matrix for the input gate i_t, forget gate f_t, and output gate o_t. All the gates have a bias, b, and are generated from a sigmoid function using the ensemble of input x_t and past hidden state h_{t-1}. The final hidden state is derived from temporary result q_t using a *tanh* non-linearity function and an updated history p_t. Figure 5.2 is adapted from [89] and simplifies the description showing a single unit of an LSTM network. The LSTM allows multiple input series, and therefore this algorithm is used as one of our building block models based on the default configurations from the scikit-learn API [63].

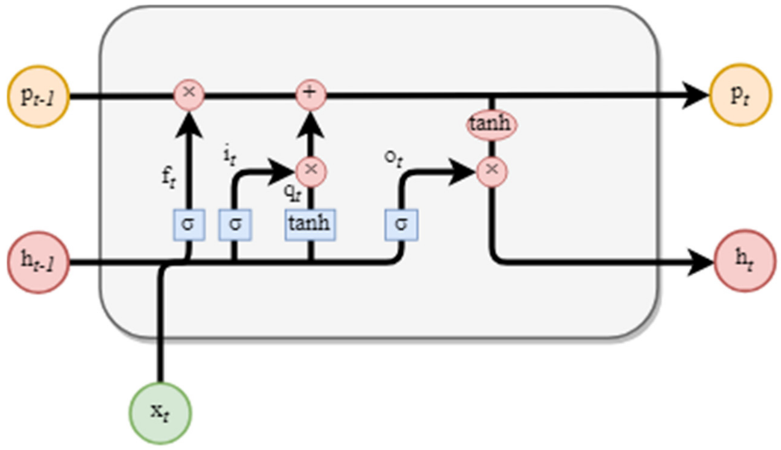


Figure 5.2. Simplified single unit of an LSTM model

5.1.1 Meta-information: Moving Average

Moving average [37] is a statistical data analysis technique used to smoothen a sequence of data points. It uses a window or period over which a fixed number of data points are averaged in steps from the initial to the final values. With this method, high-frequency fluctuations are erased depending on the size of the rolling window; therefore, finding the optimal window size is heavily dependent on the use-case. Variations of the moving average commonly used in the technical analysis of data include cumulative moving average, weighted moving average, and the exponential moving average, among others. In this research, we applied the simple moving average, which is an unweighted mean of the data within each window period. It is selected because the resultant trend has the highest visual similarity to the original series by calculating the rolling mean over a relatively small window size.

$$moving\ average(t) = \frac{\sum_{i=0}^{d-1} y(t-i)}{d} \tag{5.10}$$

where *d* is the window period, and *y(t)* the *t*-th element in the time series. While this method may be able to eliminate short-term noise from the data, it does not take into account the effects of long-term noise and the erosion of critical short-term trends on the model. Figure 5.3 shows the smoothening effect of moving average over a running window of five steps on the first 100 data points. This method can be directly used for noise reduction during data pre-processing but as shown in the graph below some crucial information is eroded. Nonetheless, by applying moving average information to meta-assisted learning, we guide the model to learn the general trend of the data sequence.

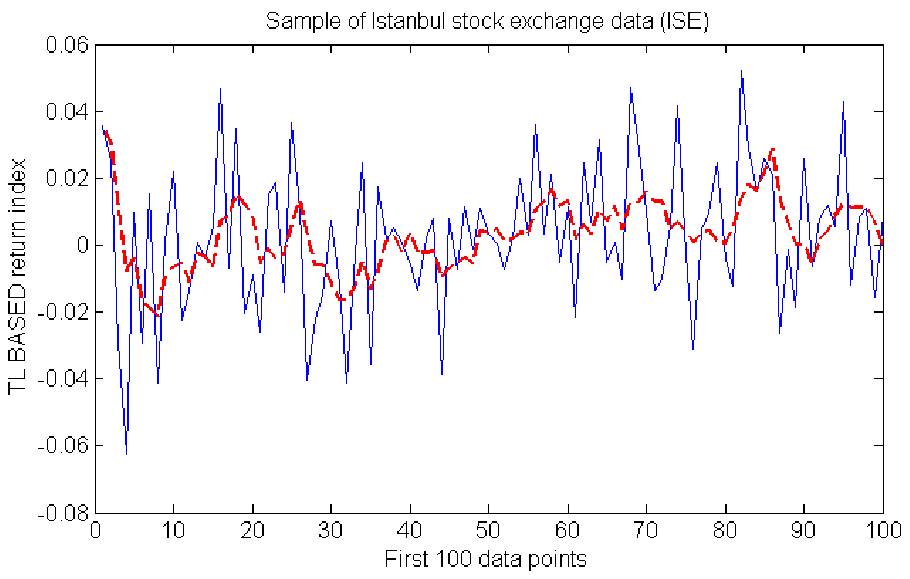


Figure 5.3. The smoothed plot in red of first 100 data points from Moving Average method

5.1.2 Meta-information: Empirical Mode Decomposition

Empirical mode decomposition (EMD) was developed as a fundamental part of the Hilbert–Huang transform to iteratively divide a signal into component signals called intrinsic mode functions (IMF) [38]. EMD is capable of producing structural primitives [85] or building-block trends, based on the hypothesis that past patterns are highly likely to recur. These decomposed components enable easier detection of new and unobserved information-bearing patterns or, on the other hand, noise. It is a robust algorithm with several improvements which include stopping criterion, extrema interpolation, and boundary effect [90] to enable accurate reconstruction of the original signal by summing up each derived IMF.

The procedure of detecting intrinsic oscillations in the signal/time series involves firstly identifying local extrema, after which an iterative sifting process is used to derive each intrinsic mode function [25]. The iterative sifting process ends after one of the two stopping rules is attained:

Either the absolute value of a potential IMF, h_i, is less than the tolerance level,

$$|h_i(t)| < tolerance\ level \tag{5.11}$$

or when the variation in a successive IMF candidate is within the tolerance level as expressed in the equation below [91].

$$\sum_t \left(\frac{h_i(t)-h_{i-1}(t)}{h_{i-1}(t)}\right)^2 < tolerance\ level \tag{5.12}$$

The method for extrapolating the endpoints of the signal has significant importance due to its effect on the accuracy of signal reconstruction [90]. Since we chose to work with the EMD implementation of [90], their solution to this effect is to consider endpoints as maxima and minima simultaneously according to the nearest extremum, thereby enforcing all IMFs to be zero at those points and ensuring alternation between maxima and minima.

The complete empirical mode decomposition can be achieved as follows [86]:

STEP 1: Let $h_i(t) = y(t)$, and $i = 1$.

STEP 2: Find some local minima and maxima in $h_i(t)$.

STEP 3: Connect all identified maxima and minima by a cubic spline as the upper envelope $up_i(t)$ and lower envelope $low_i(t)$, and calculate local mean as $m_i(t)=[up_i(t) + low_i(t)]/2$.

STEP 4: Update as $h_i(t) = h_i(t) - m_i(t)$.

STEP 5: Ensure $h_i(t)$ fulfils the requirement of IMF. If not, then redo STEP 2 to STEP 5. If done, then $IMF_i(t) = h_i(t)$, $i = i + 1$ and
$h_i(t) = y(t) - IMF_{i-1}(t)$.

STEP 6: Check if $h_i(t)$ is a monotonic function or not. If it is not, then redo STEP 2 to STEP 5 for the next IMF. If it is monotonic then
$h_i(t) = rc(t)$ and end the EMD process.

An example of the EMD outcome is shown in Figure 5.4, and the result consists of c IMF functions and a residue, $rc(t)$, which is expressed in the following equation as:

$$y(t) = \sum_{i=1}^{c} IMF_i(t) + rc(t) \tag{5.13}$$

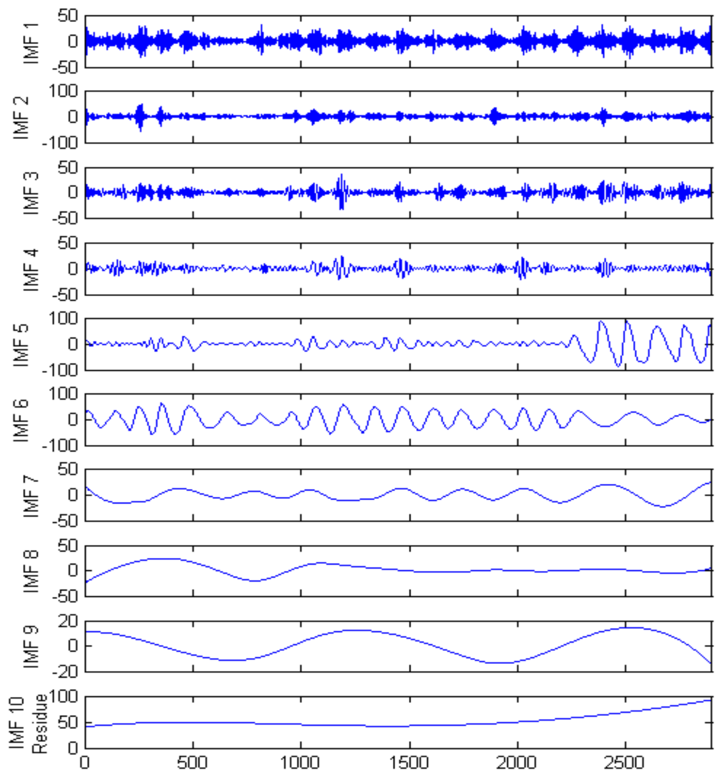


Figure 5.4 Empirical mode decomposition showing individual IMF of the Sunspot data (SOL)

This meta-information is used as the exogenous input in the model, and as inspired by the incremental contribution based learning model in [92], we can devise an algorithm to selectively include attributes—in this case: component IMFs—that enhance the machine learning process. As will be further discussed in section 5.1.3, we proposed and implemented a traditionally flat model and further developed a hierarchically structured model to capture inconspicuous local details and global context within the data.

5.1.3 Forecasting Architecture for Meta-assisted Learning

In the default flat architecture, only single-layer of the aforementioned machine learning algorithms is initialised, trained, and tested for forecasting. This preliminary design was proposed for interference-less forecasting, and some useful insights were derived as they revealed the limitations of only having a global perspective when analysing time series data. Initially, we expected noise would mostly be within high-frequency component IMF, but we discovered in [86], after progressively excluding some low-frequency IMF, that the mean square error (MSE) dropped. Therefore, instead of sequentially removing IMF components from high- to low-frequency, we revised the algorithm into a hierarchical structure to train several sub-NARX ANN on the first level with only one IMF excluded at a time for training and re-evaluation and then eliminated the IMF with the least MSE reduction. Secondly, as interference may not be persistent throughout the entire data series, we divided each part of the training set into sequential batches before identifying the noisy components.

Hence, using a hierarchical structure for trend discovery and noise elimination enables the algorithm to tackle short- and long-term noise within the time series data. The lower layer ANN learns specific time division/seasonality of the data sequence and outputs learned concepts to the higher layer model as it assembles all the information from the sub-layers with respect to the global view on original data.

5.2 Method

5.2.1 Flat Structure Meta-information Learning

The process of including meta-information to the learning process in this flat model is as follows:

Algorithm 5.1: Flat structure in meta-information for noise reduction

```
Input: Time series data, x; Delay window, d; Machine learning model,
m = {NARX, LSTM}
Output: Prediction, $y_t = x_{(t+1)}$
1     perform meta-information analysis on the training set to
      generate denoising meta_info;
2     sequentially process data meta_info into
      [$meta\_info_t$, $meta\_info_{t-1}$,…, $meta\_info_{t-d}$]
3     sequentially process data x into [$x_t$, $x_{t-1}$,…,$x_{t-d}$] and $x_{t+1}$ as
      training input and target respectively
4     modify input as a concatenation:
      [$(x, meta\_info)_t$, $(x, meta\_info)_{t-1}$,…, $(x, meta\_info, x)_{t-d}$]
5     initialise model m and train using modified input
6     evaluate prediction accuracy
```

5.2.2 Hierarchical Structure Meta-information Learning

As shown in Figure 5.5, the training series is first divided into partitions (which are further decomposed as in the case of applying EMD for meta-information generation). Machine learning on the first layer of the hierarchy is carried out in parallel by each sub-ANN model (for each partition's set of IMFs) to discern which component IMF to use in *meta_info* building. After the sub-level training and generation of meta-information is completed, an ANN on the next hierarchy level is initialised for training with the original time series along with the *meta_info* (de-noising knowledge) output from EMD_meta_info algorithm below. The *meta_info* is utilised as an exogenous input for the performance boost we present in the results and analysis section.

Algorithm 5.2: EMD_meta-info

```
Input: Time Series Data, x; Heuristic Data Division Value, h.
Output: Denoised meta_info;
1     sequentially divide x into h approximately equal partitions
2     for each partition p to hth
3     |     run EMD routine
4     |     initialise sub-NARX ANN and train by exempting one IMF at
            a time
5     |     identify IMF elimination that contributes to lowest MSE
6     |     sum valid IMF to create meta_info for pth partition
7     endfor
8     append meta_info from each partition sequentially to form
      series length equal to x.
9     return meta_info
```

After the noise-inducing components in each division have been identified and rejected by the first layer of machine learning units for local trends, a secondary layer, which learns the global trend, is then used to produce the final prediction based on both the meta-information output of the previous layer and also the original time series data.

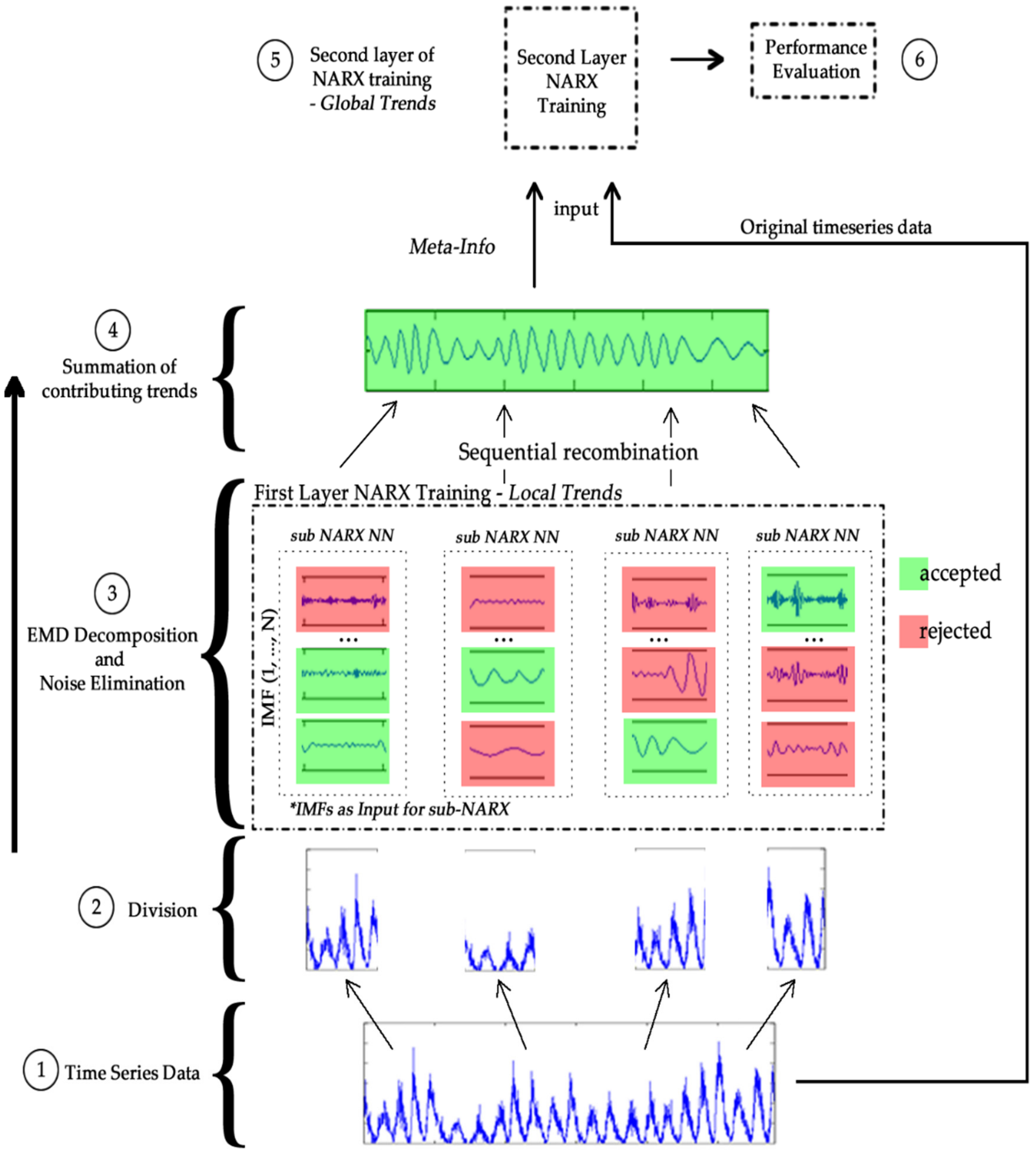

5
Second layer of NARX training - Global Trends
Second Layer NARX Training
Performance Evaluation
6
input
Meta-Info
Original timeseries data
4
Summation of contributing trends
Sequential recombination
First Layer NARX Training - Local Trends
sub NARX NN
sub NARX NN
sub NARX NN
sub NARX NN
🟩 accepted
🟥 rejected
3
EMD Decomposition and Noise Elimination
IMF (1, ..., N)
*IMFs as Input for sub-NARX
2
Division
1
Time Series Data

Figure 5.5. Hierarchical meta-learning using the non-linear autoregressive exogenous neural network–empirical mode decomposition meta-information

5.2.3 Benchmark Datasets

We make use of four data sets from financial markets and physical sciences. The data sequence was divided into training, validation, and testing sets sequentially. Meaning that the first 70% of the data sequence was partitioned for training, and then the next 15% was allocated for validation, while the final 15% was for testing the performance of the derived machine learning algorithm for comparison with the traditional approach we tested. In Figure 5.6 an output response plot shows how much deviation (error) the predicted value has from the target value (identified as orange vertical lines); meaning that time steps with more prominent orange lines have an inaccurate prediction. More importantly, this graph highlights how the data was divided into the blue, green, and red sections denoting the training, validation, and test data respectively. A window size (or lag/delay) is required in time series learning, which is the number of data points made available to the model before the actual data point that is being predicted. In the case of this experiment, two window sizes—5 and 10—are explored in order to identify trends within the data. The window size assists in detecting weekly or fortnightly patterns, especially in the stock market data.

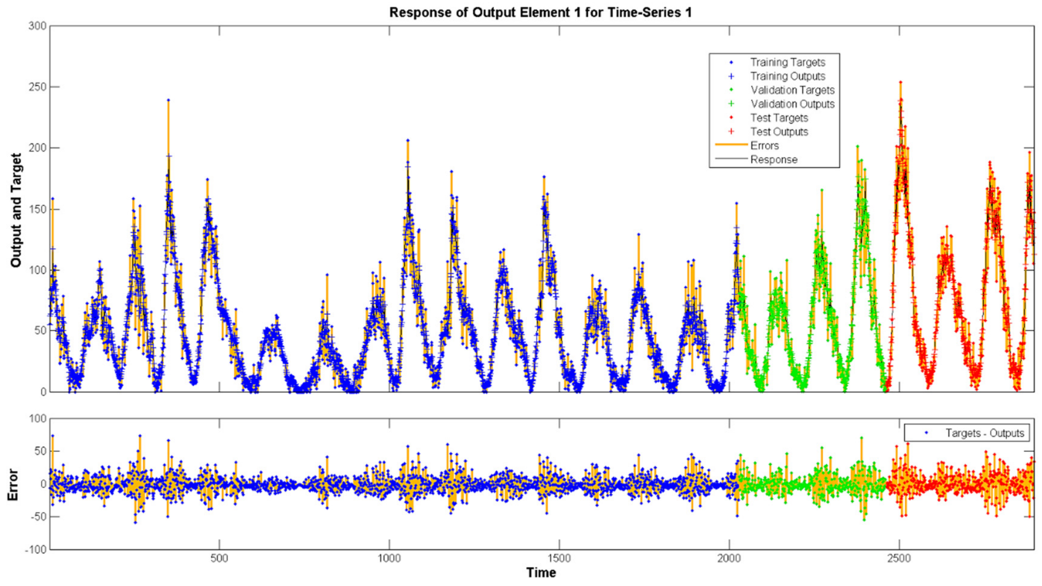


Figure 5.6. Output response (highlighting the data division method)

The first time series is the stock data from Apple Inc. It is retrieved from Yahoo Finance website in a similar method as [93], and it contains the daily closing prices from 01/01/2006 to 01/01/2015 having minimum and maximum values of 50.67 and 702.1 respectively (Figure 5.7a). This dataset will be referred to as AAPL in subsequent sections. Secondly, the sunspot dataset is derived from a recurrent temporary natural phenomenon that causes visible dark spots on the sun's surface due to its magnetic activity. A past study has been carried out to identify the trends inherent in these solar cycles [94]. We have selected this long-duration dataset, which contains a monthly record over a period of 240 years from [94]. It has 2899 entries with values ranging between 0 and 254 as shown in Figure 5.7b and this time series will be referred to as SOL. The third-time series task involves data that was used at the Santa Fe time series competition and generated from the transition of far-infrared laser intensity pulsations from periodic to chaotic [95]. The time series (referred to as SFL) is within the range of 2 to 255, and 1000 points with several bell-shaped oscillations, as seen in Figure 5.7c. Finally, the Istanbul Stock Exchange (referred to as ISE) time series is obtained from [96], where they explored its relationship to other international stock indices using a hybrid radial basis function neural network [96]. In our research, we utilised a single data column from that dataset called "TL based return index". It contains 536 records over a period of about two years from 05/01/2009 to 22/02/2011 with minimum and maximum values of −0.0622 and 0.0690 shown in Figure 5.7d.

5.3 Results and Analysis

The average simulation time over 100 repetitions was recorded, including the average training error, average test error, lowest (i.e., best) test error, and the p-value of each algorithm's test errors versus that of the algorithm with the lowest observed average test error. The maximum training epoch was fixed at 250. The early stopping technique was applied to terminate training and prevent overfitting (i.e. improve generalisation) by revert the artificial neural network to the state with the minimum error if the validation error increases consecutively for ten iterations.

The performance function used in the experiment is the mean square error, and the results here are expressed as the normalised mean square error (NMSE), which produces a non-negative mean from the summation of squared errors. Where σ^2 is the variance of a series of length n, y_i is the target, and $\hat{y}_i$ is the predicted value expressed in the equation below. It represents the absolute deviation of the prediction from the target; therefore, a perfect model will have an NMSE of 0.

$$NMSE = \frac{1}{\sigma^2 n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{5.14}$$

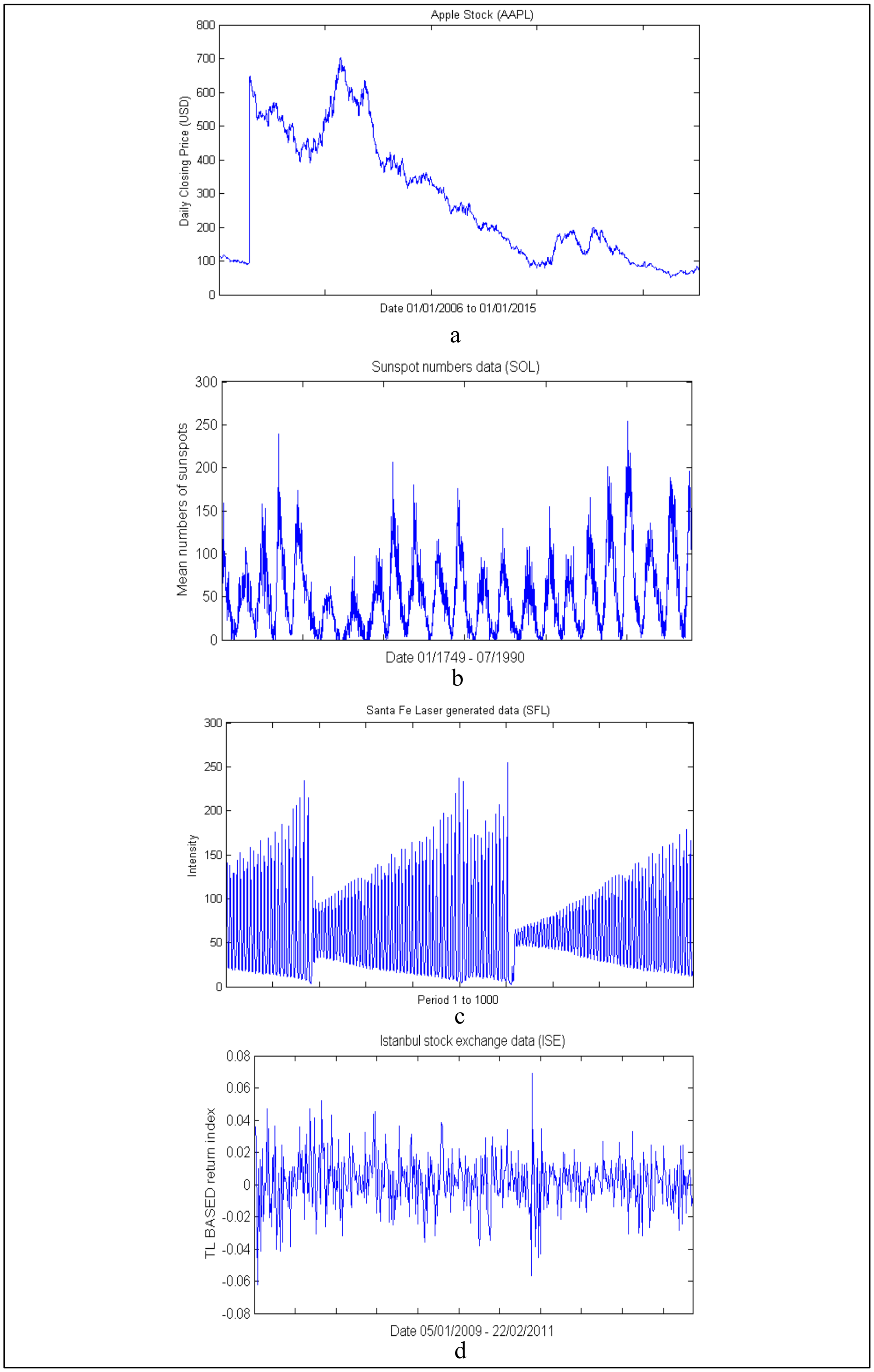


Figure 5.7 Time series data plot of a) Apple stock data (APPL), b) Sunspot numbers (SOL), c) Santa Fe laser (SFL), and d) Istanbul stock data (ISE)

In comparing the algorithms using meta-information generated by MA, and EMD, we included the normalised mean square error results when the division was set to 10 and 50 to explore the effect on the prediction accuracy from both the NARX and the LSTM model. The division value controls the number of partitions on which the first layer of noise elimination is executed upon; therefore we explore the range of values to identify the effect of performing excessive noise reduction as the value increases. Each row labelled in the tables (see Appendix Table 8.7 to Table 8.14) has been colour coded with a gradient from red through yellow to green, where deep red cells show poor performance (with high NMSE), and bold values with deep green denote the best performance (with low NMSE). To explore the effect of making a prediction several steps ahead, rather than a potentially trivial one-step-ahead prediction in related research, we employed the data partitioning scheme mentioned in the method section to create the training, validation, and test series.

The six algorithms tested in meta-assisted learning for time series forecasting are:

i. Traditional non-linear autoregressive and long short-term memory approach using the original data: NAR-ORIG and LSTM-ORIG
ii. Non-linear autoregressive exogenous model with moving average and empirical mode decomposition meta-information: NARX-MA and NARX-EMD
iii. Long short-term memory model with moving average and empirical mode decomposition meta-information: LSTM-MA and LSTM-EMD

The result on AAPL from Table 8.7 and Figure 5.9a, reveals that NARX-EMD has a lower NMSE in comparison to the conventional NAR-ORIG method and the NARX-MA, both in training accuracy and testing accuracy for delay windows of 5 and 10. The reduction of error in both training and testing scenarios indicates a suitable generalisation of the model. Overall, when the NARX-EMD is given a wider delay window, some performance increase is noted, even though a similar observation can be seen for the other two methods. Additional numbers of neurons did not show a consistent performance boost. The NAR-ORIG and NARX-MA models were considerably influenced by noise, as their training errors are higher, and overfitting is observed with more neurons. NARX-MA test performance is the worst of the three on AAPL. In Table 8.11 and Figure 5.9c, it can be seen that the LSTM-MA algorithm's performance accuracy and standard deviation are statistically better than those of LSTM-ORIG and LSTM-EMD methods; additionally, the feedback lag of 10 steps resulted in a lower NMSE across all algorithms, while overfitting is prominent with lag 5. The best performing NARX-

EMD algorithm (i.e., lag: 5) with a lower NMSE required approximately half the time of LSTM-MA. Finally, the training time does not reveal any correlation with accuracy except that networks with a higher number of hidden neurons spent more time on training.

SOL time series shows that excessive noise reduction has an adverse effect on the performance when many batches are created with each containing only a small series of data. Overfitting on the training data is consistently noticed in NAR-ORIG, NARX-MA, and NARX-EMD with 50 hidden neurons. We observe similar results as in AAPL when NAR-ORIG underperforms, and NARX-EMD confirms the noise detection and elimination hypothesis supersedes NAR-MA by decomposing the time series to find and remove component trends that do not contain information necessary for accurate prediction. There is no significant difference when the model is built upon a delay window of 5 or 10. The seasonality of the SOL data series can be visually observed in Figure 5.7b. The EMD_meta-info algorithm exploits data characteristics such as seasonality to produce a lower test error in both NARX and LSTM forecasting models (Table 8.8 and Table 8.12). There was no significant impact of lag in both of the NARX-EMD results, but LSTM-EMD with a lag of 10 significantly outperformed the model with a shorter lag.

The NARX-EMD is unable to learn the SFL time series with better accuracy than the other two non-linear autoregressive methods tested. Even with a window lag of 5 and 10, the training NMSE is significantly higher when compared to NAR-ORIG and NARX-MA. On this time series task, each algorithm performed better with a longer delay window of 10 steps and more artificial neural processing units. To understand the reason for NARX-EMD's performance on the SFL time series data, the empirical mode decomposition is plotted in Figure 5.8 and compared with that of SOL in Figure 5.4. As shown in Figure 5.8, the SFL generated wave is irregular, and the decomposition yields no intrinsic pattern as compared to SOL in Figure 5.4. Notably, the first IMF in Figure 5.8 has a high similarity with the original time series in Figure 5.7c, which means the decomposition process was comparatively unsuccessful. Periodically repeating trends are necessary for the learner to build an accurate model. This outcome reconfirms that the hypothesis for interference removal necessitates an accurate decomposition of the data.

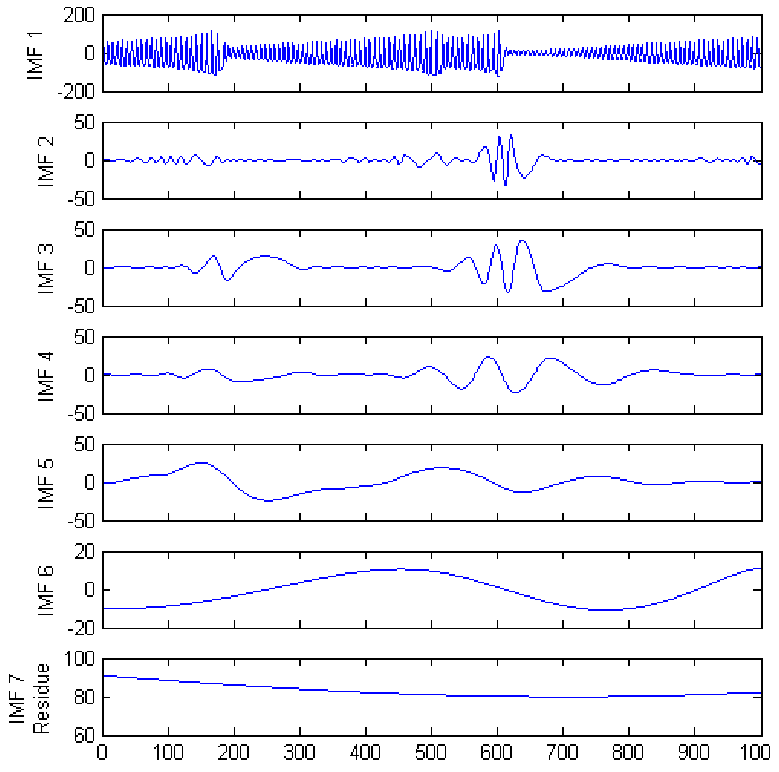


Figure 5.8. Empirical mode decomposition showing individual IMF of the Santa Fe Laser data (SFL)

The ISE time series IMF decomposition derived periodically repeating trends which are information-bearing, and this explains the substantial gain in accuracy in both modified NARX and LSTM algorithms. The best result is obtained with a window size of 10 (a fortnight in trading days) rather than a weekly base (window size 5) on the NARX-EMD (see Figure 5.9 a, d). In Table 8.10, the NARX-EMD's normalised mean square error is considerably lower than its LSTM-EMD counterpart (see Table 8.14). Furthermore, the training duration of the NARX-EMD learner was relatively similar to the traditional NAR-ORIG method and about ten times faster than LSTM-ORIG on average.

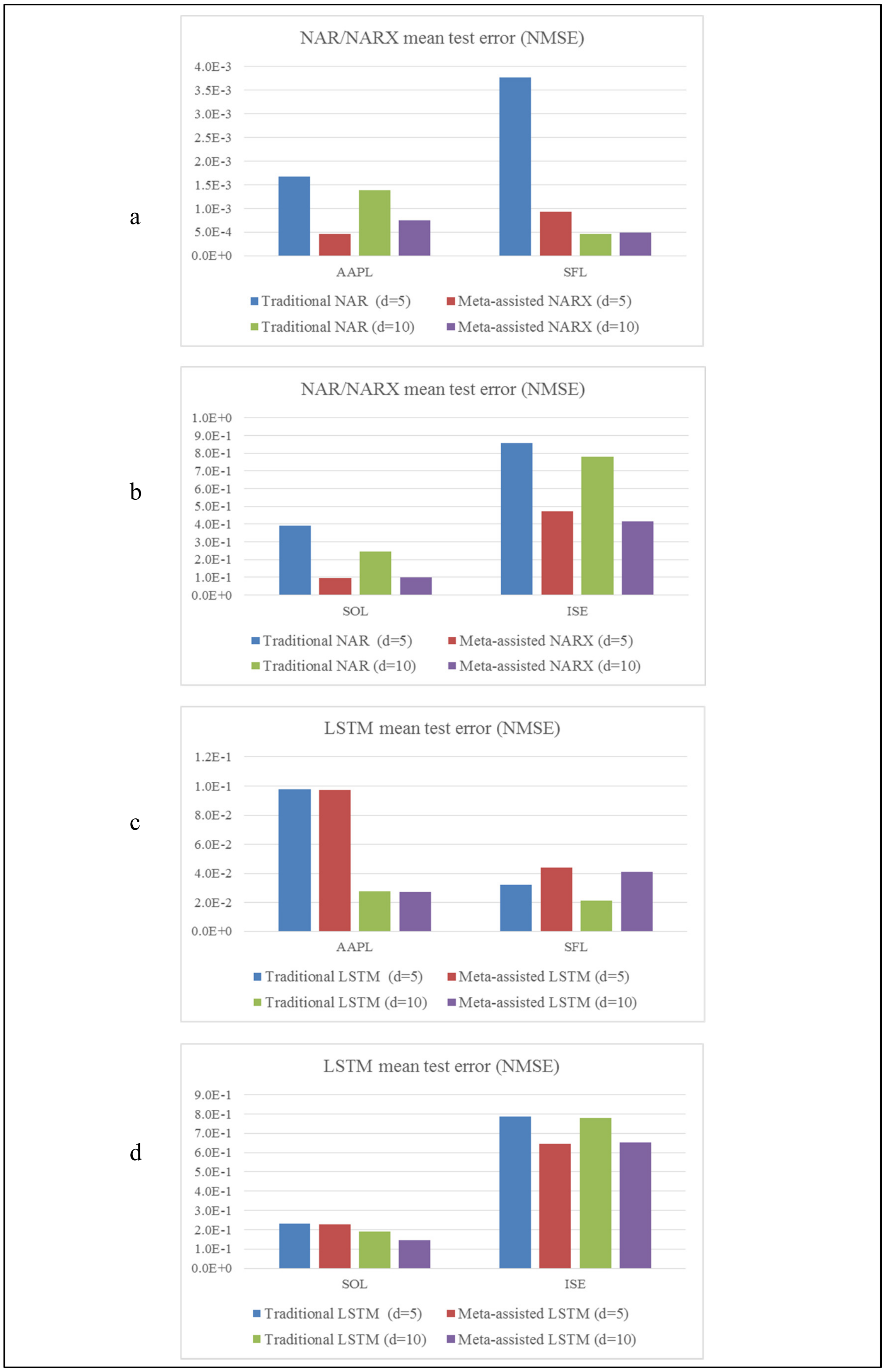


Figure 5.9. The normalised mean square error of traditional and meta-assisted forecasting algorithms (lower is better)

5.4 Concluding Remarks

In the case of time series tasks where only a sequence of past output data is available, we have developed a different approach to interference elimination. After decomposing fixed periods within the time series using EMD, we can identify noisy components before summing up the remaining beneficial IMFs for each period and finally re-joining the meta-information sequentially for the higher hierarchy artificial neural network training, as we have demonstrated with NARX and LSTM models.

We have shown the benefits of the new approach over the conventional non-linear autoregressive and long short-term memory forecasters in cases where the time series can be adequately decomposed into meaningful trends using empirical mode decomposition.

The significant contribution of our algorithm is its resilience to noise/anomalies that may have occurred both within a short period and over an extended period in the series. Utilising a hierarchically structured order for training sub-networks on small divisions (local learners) before another learner is trained with the complete series (global learner) provides the machine learner with a better overview of the data. Therefore meta-assisted learning is advantageous when the selection of optimal window size is non-trivial. This proven concept of “divide-and-conquer” was similarly applied in [97] whereby researchers embed several LSTM layers to convert high-dimensional stock market data into low-dimensional data and generate “stock vector” information to enhance the predictive performance of the neural network.

Finally, the pre-processing time and training time for the first level sub-networks did not increase the total training time significantly because each division can be trained concurrently. This parallelisation inherent within the design of the meta-assisted learning approach affords implementers the ability to deploy models on emergent distributed cloud computing services such as Organisational Sustainability Modelling (OSM) [98] for interference-less real-time analytics especially in situations with a vast amount of time series data. Further research directions to be explored include automatic detection of patterns present in IMFs, and using such information for the selection of the best meta-learning algorithm and its hyper-parameters for a hierarchical structure.

6 CONCLUSION

6.1 Reflection

To make appropriate use of the rapidly growing data we collect daily, we propose techniques for building enhanced classification and forecasting models capable of analysing the data for anomalies and creating a hierarchy of structured concepts even with the presence of noise. With the primary aim of eliminating interference due to attribute noise, we embarked on this research and focused on two important domains of great relevance in the era of big data.

Following the four vital hypotheses: cluster assumption for meta-learning, recurrent trend patterns for meta-learning, hierarchical learning of local and global clusters/trends, and meta-assisted learning for improved noise identification and elimination, we established several algorithms that generate meta-information to make sense of the dither within the data. Existing machine learning models then utilise this structured training data, and our empirical findings reveal that even data with noise/outliers show improvement in prediction accuracy, stable performance with lower standard deviation, and in select cases faster learning time.

6.1.1 On Classification Tasks

In classification tasks we compared meta-assisted learning based on the density meta-information in flat and hierarchical models with and without output class partitioning:

i. Meta-assisted learning on a flat structure (MT_F)
ii. Meta-assisted learning on a flat structure using output-class partitioning (MT_FC)
iii. Meta-assisted learning on a hierarchical structure (MT_H)
iv. Meta-assisted learning on a hierarchical structure using output-class partitioning (MT_HC)

It is therefore recommended from the result to select structured meta-assisted learning either by clustering with output-class partitioning for flat models (MT_FC) or without output-class partitioning for hierarchical models (MT_H) because it can infer the class partition due to the depth of the network structure. Therefore it is crucial to apply a form of cluster analysis that can produce arbitrary-shaped groups of data instances with similar concepts and also tag outliers. Currently, the appropriate circumstance to apply output-class partitioning is not known, so it may be imperative to apply an ANN in the top hierarchy to select between two sub-ANNs that models the data with and without output-class partitioning.

6.1.2 On Forecasting Tasks

In time series forecasting we applied meta-assisted learning to NARX and LSTM time series models that utilise meta-information relating to the component trends:

i. Non-linear autoregressive exogenous model with moving average and empirical mode decomposition (NARX-MA and NARX-EMD)
ii. Long short-term memory model with moving average and empirical mode decomposition (LSTM-MA and LSTM-EMD)

We tested the hypotheses on two state-of-the-art time series models, and according to the primary aim of this research, the designed algorithm detects anomaly to improve the general the performance of the machine learner. It is observed that the algorithms that incorporate the meta-assisted learning via empirical mode decomposition can be superior and can avoid the impact of interfering trends especially in conditions where the decomposition is successful.

Therefore as stated in the landmarking category of meta-learning, this is a potential indicator for the enhanced performance on variants of the algorithm/technique we have proposed in other similar state-of-the-art machine learners.

6.2 Contribution

In summary, this thesis presents some empirical validation to the proposed guideline on how to achieve interference-less machine learning when outlier/noise is present in the data. Thus providing answers to some of the questions posed by related research work in this area which include:

- Successful application of the semi-supervised technique to structure data before learning.

- Designing the meta-information generation routines of the algorithms with parallelisation to reduce the time and computational cost of pre-processing for noise handling.
- Providing local details and global overview of the data during the model building stage to avoid flattening of structured data.
- Eliminating the burden of enforcing cumbersome input validation to maintain data integrity which signifies the impact of applying these algorithms to real-world practical problems.

6.3 Future Work

By reporting the viability of meta-assisted learning in classification tasks and forecasting tasks, we show that the meta-information can be harnessed from the data.

A potential aspect for further investigation would be on the duality of meta-information whereby the types of meta extracted from classification problems can also be applied to time series forecasting, vice versa, and also to other domains in which machine learning can be applied. Identifying such meta-information that exhibits spatial and temporal relationship may be advantageous.

Furthermore, a systematic method to monitor the learning process and fall back to the default algorithm (or alternate between output-class partitioning) can reduce the extra pre-processing time we observed especially in data with a significantly low amount of outliers/anomaly.

Finally, formalisation of cluster/trend evaluation can help to identify important factors for selecting meta-generation methods. To fit the scope of our proposed roadmap we had to restrict this metric to focus on improving the performance accuracy of the model. Nonetheless, clustering metrics like homogeneity, completeness, and stratification characteristics, or time series metrics like correlation, trend drift, and seasonality characteristics, among other recent advances in data analytics, can provide protocols for selecting meta for a noise-tolerant machine learning.

REFERENCES

[1] E. Alpaydin, *Introduction to machine learning*. Cambridge, Mass.: MIT Press, 2010.

[2] P. K. Mallapragada, Rong Jin, A. K. Jain, and Yi Liu, "SemiBoost: Boosting for Semi-Supervised Learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 11, pp. 2000–2014, Nov. 2009.

[3] K. Ramanathan and S. U. Guan, "Clustering and combinatorial optimization in recursive supervised learning," *J. Comb. Optim.*, vol. 13, no. 2, pp. 137–152, Dec. 2006.

[4] P. Van Der Smagt and G. Hirzinger, "Solving the ill-conditioning in neural network learning," in *Neural Networks: Tricks of the Trade*, Springer, 1998, pp. 193–206.

[5] W. Duch and J. Korczak, "Optimization and global minimization methods suitable for neural networks," *Neural Comput. Surv.*, vol. 2, pp. 163–212, 1998.

[6] M. Kobyashi, A. Zamani, S. Ozawa, and S. Abe, "Reducing computations in incremental learning for feedforward neural network with long-term memory," in *International Joint Conference on Neural Networks, 2001. Proceedings. IJCNN '01*, 2001, vol. 3, pp. 1989–1994 vol.3.

[7] X. Zhu and X. Wu, "Class noise vs. attribute noise: A quantitative study," *Artif. Intell. Rev.*, vol. 22, no. 3, pp. 177–210, 2004.

[8] S. Weaver, L. Baird, and M. Polycarpou, "Using localizing learning to improve supervised learning algorithms," *IEEE Trans. Neural Netw.*, vol. 12, no. 5, pp. 1037–1046, 2001.

[9] J. Hua Ang, S.-U. Guan, K. C. Tan, and A. A. Mamun, "Interference-less neural network training," *Neurocomputing*, vol. 71, no. 16–18, pp. 3509–3524, Oct. 2008.

[10] M. Li, S. Guan, L. Zhao, W. Li, and T. Wang, "Learning of neural network with reduced interference-An ensemble approach," *Int. J. Mach. Learn. Comput.*, vol. 2, no. 6, pp. 786–790, 2012.

[11] S.-U. Guan and S. Li, "Parallel growing and training of neural networks using output parallelism," *Neural Netw. IEEE Trans. On*, vol. 13, no. 3, pp. 542–550, 2002.

[12] S. J. Guo, S.-U. Guan, S. Yang, W. F. Li, L. F. Zhao, and J. H. Song, "Input Partitioning Based on Correlation for Neural Network Learning," *J. Clean Energy Technol.*, pp. 335–338, 2013.

[13] S.-U. Guan, Q. Yinan, S. Kiat Tan, and S. Li, "Output partitioning of neural networks," *Neurocomputing*, vol. 68, pp. 38–53, Oct. 2005.

[14] D. Gamberger, N. Lavrac, and S. Dzeroski, "Noise detection and elimination in data preprocessing: Experiments in medical domains," *Appl. Artif. Intell.*, vol. 14, no. 2, pp. 205–223, Feb. 2000.

[15] D. O. Afolabi, S.-U. Guan, F. Liu, K. L. Man, and P. W. H. Wong, "Class Interference Reduction through Meta-attribute Reinforced Learning," in

International Conference on Computing and Technology Innovation, Luton, UK, 2015.

[16] Y.-J. Lee, Y.-R. Yeh, and Y.-C. F. Wang, “Anomaly Detection via Online Oversampling Principal Component Analysis,” *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 7, pp. 1460–1470, Jul. 2013.

[17] R. Silipo, I. Adae, A. Hart, and M. Berthold, “Seven Techniques for Data Dimensionality Reduction,” KNIME, 2014.

[18] P. Symeonidis, I. Kehayov, and Y. Manolopoulos, “Text classification by aggregation of SVD eigenvectors,” in *Advances in Databases and Information Systems*, 2012, pp. 385–398.

[19] J. R. Quinlan, “Induction of decision trees,” *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, Mar. 1986.

[20] J. R. Quinlan, “The Effect of Noise on Concept Learning,” in *Machine Learning, An Artificial Intelligence Approach (Volume II)*, Morgan Kaufmann, 1986, pp. 149–166.

[21] C. Schaffer, “Sparse data and the effect of overfitting avoidance in decision tree induction,” in *Proceedings of the tenth national conference on Artificial intelligence*, 1992, pp. 147–152.

[22] I. Guyon, N. Matic, and V. Vapnik, “Discovering Informative Patterns and Data Cleaning,” *Adv. Knowl. Discov. Data Min.*, pp. 181–203, 1996.

[23] W. Wong and R. B. Miller, “Repeated Time Series Analysis of ARIMA–Noise Models,” *J. Bus. Econ. Stat.*, vol. 8, no. 2, pp. 243–250, Apr. 1990.

[24] M. Pesentia and M. Pirasa, “A Modified Forward Search Approach Applied to Time Series Analysis,” in *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Beijing, China, 2008, vol. XXXVII, pp. 787–792.

[25] D. Afolabi, S.-U. Guan, K. L. Man, P. W. H. Wong, and X. Zhao, “Hierarchical Meta-Learning in Time Series Forecasting for Improved Interference-Less Machine Learning,” *Symmetry*, vol. 9, no. 11, p. 20, Nov. 2017.

[26] D. J. Bemdt and J. Clifford, “Using Dynamic Time Warping to Find Patterns in Time Series,” in *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, Seattle, WA, USA, 1994, pp. 359–370.

[27] Y. Bengio and P. Frasconi, “Input-output HMMs for sequence processing,” *IEEE Trans. Neural Netw.*, vol. 7, no. 5, pp. 1231–1249, 1996.

[28] M. Štěpnička, V. Pavliska, V. Novák, I. Perfilieva, L. Vavříčková, and I. Tomanová, “Time Series Analysis and Prediction Based on Fuzzy Rules and the Fuzzy Transform,” in *Proceedings of the Joint 2009 International Fuzzy Systems Association World Congress and 2009 European Society of Fuzzy Logic and Technology Conference*, Lisbon, Portugal, 2009, pp. 483–488.

[29] S. R. Gunn, “Support Vector Machines for Classification and Regression,” *ISIS Tech. Rep.*, vol. 14, pp. 85–86, May 1998.

[30] S. Jeon, B. Hong, and V. Chang, “Pattern graph tracking-based stock price prediction using big data,” *Future Gener. Comput. Syst.*, vol. 80, pp. 171–187, Mar. 2018.

[31] V. Chang, “The Business Intelligence As a Service in the Cloud,” *Future Gener Comput Syst*, vol. 37, no. C, pp. 512–534, Jul. 2014.

[32] M. Ramachandran and V. Chang, "Financial Software as a Service – A Paradigm for Risk Modelling and Analytics," p. 25.

[33] V. Chang and M. Ramachandran, "Financial Modeling and Prediction as a Service," *J. Grid Comput.*, vol. 15, no. 2, pp. 177–195, Jun. 2017.

[34] A. S. Sohal, R. Sandhu, S. K. Sood, and V. Chang, "A Cybersecurity Framework to Identify Malicious Edge Device in Fog Computing and Cloud-of-things Environments," *Comput Secur*, vol. 74, no. C, pp. 340–354, May 2018.

[35] V. Chang, "Towards Data Analysis for Weather Cloud Computing," *Know-Based Syst*, vol. 127, no. C, pp. 29–45, Jul. 2017.

[36] X. Xu, M. Ester, H.-P. Kriegel, and J. Sander, "A distribution-based clustering algorithm for mining in large spatial databases," in *Data Engineering, 1998. Proceedings on 14th International Conference*, 1998, pp. 324–331.

[37] W. W.-S. Wei, *Time Series Analysis: Univariate and Multivariate Methods*, 2nd ed. USA: Addison-Wesley Pearson Higher Ed, 2006.

[38] N. E. Huang *et al.*, "The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis," in *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 1998, vol. 454, pp. 903–995.

[39] B. Merk, C. V. Bratu, and R. Potolea, "Meta-learning enhancements by data partitioning," in *Intelligent Computer Communication and Processing, 2009. ICCP 2009. IEEE 5th International Conference on*, 2009, pp. 59–62.

[40] D. Ler, I. Koprinska, and S. Chawla, "A Landmarker Selection Algorithm Based on Correlation and Efficiency Criteria," in *AI 2004: Advances in Artificial Intelligence*, vol. 3339, G. I. Webb and X. Yu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 296–306.

[41] J. R. Rice, "The Algorithm Selection Problem," in *Advances in Computers*, vol. 15, 1976, pp. 65–118.

[42] C. Giraud-Carrier, "Metalearning-a tutorial," in *Tutorial at the 2008 International Conference on Machine Learning and Applications, ICMLA*, 2008, pp. 11–13.

[43] K. A. Smith-Miles, "Cross-disciplinary perspectives on meta-learning for algorithm selection," *ACM Comput. Surv.*, vol. 41, no. 1, pp. 1–25, Dec. 2008.

[44] C. Lemke, M. Budka, and B. Gabrys, "Metalearning: a survey of trends and technologies," *Artif. Intell. Rev.*, vol. 44, no. 1, pp. 117–130, Jun. 2015.

[45] P. Brazdil, *Metalearning: Applications to Data Mining*. Berlin, Heidelberg: Springer-Verlag, 2009.

[46] J. Vanschoren, "Understanding Machine Learning Performance with Experiment Databases," Arenberg Doctoral School of Science, Engineering & Technology, Katholieke Universiteit Leuven, Belgium, 2010.

[47] R. Vilalta and Y. Drissi, "A perspective view and survey of meta-learning," *Artif. Intell. Rev.*, p. 95, 2002.

[48] H. Bensusan, C. Giraud-Carrier, and C. Kennedy, "A Higher-order Approach to Meta-learning," University of Bristol, Bristol, UK, UK, 2000.

[49] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[50] C. Soares, P. B. Brazdil, and P. Kuba, “A Meta-Learning Method to Select the Kernel Width in Support Vector Regression,” *Mach. Learn.*, vol. 54, no. 3, pp. 195–209, Mar. 2004.

[51] B. Arinze, “Selecting appropriate forecasting models using rule induction,” *Omega*, vol. 22, no. 6, pp. 647–658, 1994.

[52] K. Leyton-Brown, E. Nudelman, and Y. Shoham, “Learning the empirical hardness of optimization problems: The case of combinatorial auctions,” in *International Conference on Principles and Practice of Constraint Programming*, 2002, pp. 556–572.

[53] V. Estivill-Castro and D. Wood, “A survey of adaptive sorting algorithms,” *ACM Comput. Surv. CSUR*, vol. 24, no. 4, pp. 441–476, 1992.

[54] L. Breiman, “Bagging predictors,” *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.

[55] Y. Freund and R. E. Schapire, “A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting,” *J Comput Syst Sci*, vol. 55, no. 1, pp. 119–139, Aug. 1997.

[56] D. H. Wolpert, “Stacked Generalization,” *Neural Netw.*, vol. 5, pp. 241–259, 1992.

[57] J. Gama and P. Brazdil, “Cascade Generalization,” *Mach. Learn.*, vol. 41, no. 3, pp. 315–343, Dec. 2000.

[58] G. Lindner, D. Ag, and R. Studer, “AST: Support for Algorithm Selection with a CBR Approach,” in *Recent Advances in Meta-Learning and Future Work*, 1999, pp. 418–423.

[59] A. Kalousis and M. Hilario, “Feature Selection for Meta-learning,” in *Proceedings of the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, London, UK, UK, 2001, pp. 222–233.

[60] D. H. Wolpert, “The lack of a priori distinctions between learning algorithms,” *Neural Comput.*, vol. 8, no. 7, pp. 1341–1390, 1996.

[61] S.-U. Guan and J. Liu, “Incremental ordered neural network training,” *J. Intell. Syst.*, vol. 12, no. 3, pp. 137–172, 2002.

[62] J. Song, S.-U. Guan, and B. Zheng, “Incremental Hyper-Sphere Partitioning for Classification,” *Int. J. Appl. Evol. Comput. IJAEC*, vol. 5, no. 2, pp. 72–88, 2014.

[63] F. Pedregosa *et al.*, “Scikit-learn: Machine Learning in Python,” *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[64] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, “A Limited Memory Algorithm for Bound Constrained Optimization,” *SIAM J. Sci. Comput.*, vol. 16, no. 5, pp. 1190–1208, 1995.

[65] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise.,” in *Kdd*, 1996, vol. 96, pp. 226–231.

[66] M. Daszykowski, B. Walczak, and D. L. Massart, “Looking for natural patterns in data: Part 1. Density-based approach,” *Chemom. Intell. Lab. Syst.*, vol. 56, no. 2, pp. 83–92, 2001.

[67] G. Gan and C. Ma Wu, Jianhong, *Data clustering: theory, algorithms, and applications*. Philadelphia, Pa.; Alexandria, Va.: SIAM, Society for Industrial and Applied Mathematics ; American Statistical Association, 2007.

[68] F. W. J. Olver, D. W. Lozier, R. F. Boisvert, and C. W. Clark, Eds., *NIST Handbook of Mathematical Functions*. New York, NY: Cambridge University Press, 2010.

[69] C. Higuera, K. J. Gardiner, and K. J. Cios, “Self-Organizing Feature Maps Identify Proteins Critical to Learning in a Mouse Model of Down Syndrome,” *PLOS ONE*, vol. 10, no. 6, pp. 1–28, 2015.

[70] L. A. Kurgan, K. J. Cios, R. Tadeusiewicz, M. R. Ogiela, and L. S. Goodenday, “Knowledge discovery approach to automated cardiac SPECT diagnosis,” *Artif. Intell. Med.*, vol. 23, p. 149, 2001.

[71] K. Bache and M. Lichman, *UCI Machine Learning Repository*. University of California, Irvine, School of Information and Computer Sciences, 2013.

[72] R. J. Lyon, B. W. Stappers, S. Cooper, J. M. Brooke, and J. D. Knowles, “Fifty years of pulsar candidate selection: from simple filters to a new principled real-time classification approach,” *Mon. Not. R. Astron. Soc.*, vol. 459, no. 1, pp. 1104–1123, 2016.

[73] V. G. Sigillito, S. P. Wing, L. V. Hutton, and K. B. Baker, “Classification of radar returns from the ionosphere using neural networks,” *Johns Hopkins APL Tech. Dig.*, vol. 10, no. 3, pp. 262–266, 1989.

[74] R. A. Fisher, “The use of multiple measurements in taxonomic problems,” *Ann. Hum. Genet.*, vol. 7, no. 2, pp. 179–188, 1936.

[75] M. Charytanowicz, J. Niewczas, P. Kulczycki, P. A. Kowalski, S. Łukasik, and S. Żak, “Complete gradient clustering algorithm for features analysis of x-ray images,” in *Information technologies in biomedicine*, Springer, 2010, pp. 15–24.

[76] R. P. Gorman and T. J. Sejnowski, “Analysis of hidden units in a layered network trained to classify sonar targets,” *Neural Netw.*, vol. 1, no. 1, pp. 75–89, 1988.

[77] H. T. Kahraman, S. Sagiroglu, and I. Colak, “The development of intuitive knowledge classifier and the modeling of domain dependent data,” *Knowl.-Based Syst.*, vol. 37, pp. 283–295, 2013.

[78] J. P. Siebert, “Vehicle recognition using rule based methods,” 1987.

[79] J. Alcalá-Fdez *et al.*, “KEEL: a software tool to assess evolutionary algorithms for data mining problems,” *Soft Comput*, vol. 13, no. 3, pp. 307–318, Oct. 2008.

[80] S. Aeberhard, D. Coomans, and O. De Vel, “Comparison of classifiers in high dimensional settings,” *Dept Math Stat. James Cook Univ N. Qld. Aust. Tech Rep*, vol. 92, p. 02, 1992.

[81] W. Wolberg and O. Mangasarian, “Multisurface method of pattern separation for medical diagnosis applied to breast cytology,” in *Proceedings of the National Academy of Sciences*, 1990, pp. 9193–9196.

[82] J. R. Quinlan, “Simplifying decision trees,” *Int. J. Man-Mach. Stud.*, vol. 27, no. 3, pp. 221–234, 1987.

[83] D. Morrison, R. Wang, and L. C. De Silva, “Ensemble methods for spoken emotion recognition in call-centres,” *Speech Commun.*, vol. 49, no. 2, pp. 98–112, Feb. 2007.

[84] L. McInnes and J. Healy, “Accelerated Hierarchical Density Based Clustering,” in *2017 IEEE International Conference on Data Mining Workshops*, 2017, pp. 33–42.

[85] S. Singh, “Noise impact on time-series forecasting using an intelligent pattern matching technique,” *Pattern Recognit.*, vol. 32, no. 8, pp. 1389–1398, 1999.

[86] D. O. Afolabi, S.-U. Guan, K. L. Man, and P. W. H. Wong, “Meta-learning with Empirical Mode Decomposition for Noise Elimination in Time Series Forecasting,” in *Advanced Multimedia and Ubiquitous Engineering: FutureTech & MUE*, J. J. H. Park, H. Jin, Y.-S. Jeong, and M. K. Khan, Eds. Singapore: Springer Singapore, 2016, pp. 405–413.

[87] T. Lin, B. G. Horne, P. Tino, and C. L. Giles, “Learning long-term dependencies in NARX recurrent neural networks,” *IEEE Trans. Neural Netw.*, vol. 7, no. 6, pp. 1329–1338, 1996.

[88] W. Yin, K. Kann, M. Yu, and H. Schütze, “Comparative Study of CNN and RNN for Natural Language Processing,” *CoRR*, vol. abs/1702.01923, 2017.

[89] “Understanding LSTM Networks -- colah’s blog.” [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Accessed: 09-May-2018].

[90] R. T. Rato, M. D. Ortigueira, and A. G. Batista, “On the HHT, its problems, and some solutions,” *Mech. Syst. Signal Process.*, vol. 22, no. 6, pp. 1374–1394, Aug. 2008.

[91] D. Kim and H.-S. Oh, “EMD: A package for empirical mode decomposition and hilbert spectrum,” *R J.*, vol. 1, no. 1, pp. 40–46, 2009.

[92] S.-U. Guan, J. Liu, and Y. Qi, “An incremental approach to contribution-based feature selection,” *J. Intell. Syst.*, vol. 13, no. 1, pp. 15–42, 2004.

[93] Y. Wei and V. Chaudhary, “The Influence of Sample Reconstruction on Stock Trend Prediction via NARX Neural Network,” in *Proceedings of the 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, Miami, FL, USA, 2015, pp. 52–57.

[94] P. Hertz and E. D. Feigelson, “A sample of astronomical time series,” in *Applications of Time Series Analysis in Astronomy and Meteorology*, T. Subba Rao, M. B. Priestley, and O. Lessi, Eds. London: Chapman & Hall, 1997, pp. 340–356.

[95] A. S. Weigend, “The Future of Time Series,” in *International Conference On Neural Information Processing*, Seoul, Korea, 1994, vol. 3(2), pp. 853–856.

[96] O. Akbilgic, H. Bozdogan, and M. E. Balaban, “A novel Hybrid RBF Neural Networks model as a forecaster,” *Stat. Comput.*, vol. 24, no. 3, pp. 365–375, May 2014.

[97] X. Pang, Y. Zhou, P. Wang, W. Lin, and V. Chang, “An innovative neural network approach for stock market prediction,” *J. Supercomput.*, Jan. 2018.

[98] V. Chang, R. J. Walters, and G. B. Wills, “Organisational sustainability modelling—An emerging service and analytics model for evaluating Cloud Computing adoption with two case studies,” *Int. J. Inf. Manag.*, vol. 36, no. 1, pp. 167–179, Feb. 2016.

APPENDICES

Appendix A: Results in Classification Tasks

Table 8.1 A comparison of algorithms performance metrics (A)

dataset	ALG	mean ACC	±std	min	Q1 (25%)	Q2 (50%)	Q3 (75%)	max	*p*-value vs best	time (s)	%Δ vs best
CCLS	ORIG	57.273	3.801	46.875	54.628	57.292	59.896	64.583	3.41E-05	0.145	3.66
	MT_F	56.565	3.670	45.833	54.557	56.250	59.215	63.542	2.74E-08	0.160	4.96
	MT_FC	53.103	3.377	43.979	51.042	53.125	55.555	61.458	5.04E-30	0.126	11.80
	MT_H	59.369	3.161	52.604	56.994	59.375	61.830	68.229	best	0.437	best
	MT_HC	53.171	2.967	45.833	51.242	53.125	55.208	60.209	2.51E-32	0.278	11.66
CORTEX	ORIG	75.764	10.009	31.019	72.454	78.472	81.944	89.815	1.74E-48	0.266	29.48
	MT_F	76.667	11.808	13.889	72.569	78.935	83.796	92.593	3.79E-39	0.578	27.96
	MT_FC	98.102	5.353	60.648	99.074	99.537	100.000	100.000	best	0.338	best
	MT_H	80.213	7.816	52.778	75.810	82.407	85.185	92.593	3.68E-46	1.156	22.30
	MT_HC	97.824	4.547	72.685	98.495	99.074	99.537	100.000	6.93E-01	0.861	0.28
GLASS	ORIG	57.433	7.598	35.714	51.995	57.500	62.573	74.419	6.62E-33	0.102	25.44
	MT_F	58.643	7.246	34.884	54.886	59.762	63.799	73.810	2.59E-30	0.102	22.85
	MT_FC	72.045	6.633	37.500	68.182	72.093	75.714	87.500	best	0.168	best
	MT_H	59.552	6.634	37.500	54.940	60.000	64.286	74.419	2.60E-29	0.317	20.98
	MT_HC	71.442	7.195	48.889	67.248	71.761	77.500	86.364	5.39E-01	0.337	0.84

Table 8.2 A comparison of algorithms performance metrics (B)

dataset	ALG	mean ACC	±std	min	Q1 (25%)	Q2 (50%)	Q3 (75%)	max	*p*-value vs best	time (s)	%Δ vs best
HEART	ORIG	81.981	5.589	64.815	77.778	81.481	85.185	94.444	9.22E-01	0.096	0.09
	MT_F	81.426	5.999	55.556	79.167	81.481	85.185	92.593	4.24E-01	0.101	0.77
	MT_FC	65.722	4.256	53.704	62.963	64.815	68.519	74.074	2.81E-62	0.115	24.85
	MT_H	82.056	5.075	68.519	79.630	81.481	85.185	92.593	best	0.287	best
	MT_HC	66.556	5.048	51.852	62.963	66.667	70.370	77.778	4.12E-54	0.218	23.29
HEP	ORIG	78.237	7.565	58.824	73.333	80.000	82.353	100.000	9.85E-01	0.064	0.03
	MT_F	77.949	9.280	53.333	72.647	80.625	86.667	94.118	8.01E-01	0.173	0.40
	MT_FC	77.625	8.021	53.333	73.333	76.471	82.353	93.333	5.77E-01	0.137	0.82
	MT_H	78.258	7.991	60.000	70.588	80.000	82.353	100.000	best	0.294	best
	MT_HC	77.666	8.040	58.824	72.647	76.471	82.353	100.000	6.02E-01	0.215	0.76
HILL	ORIG	98.859	7.052	50.617	100.000	100.000	100.000	100.000	1.07E-01	0.135	1.15
	MT_F	99.422	4.969	50.413	100.000	100.000	100.000	100.000	2.46E-01	0.343	0.58
	MT_FC	100.000	0.000	100.000	100.000	100.000	100.000	100.000	best	0.208	best
	MT_H	99.463	4.956	50.413	100.000	100.000	100.000	100.000	2.80E-01	0.482	0.54
	MT_HC	99.010	6.963	50.413	100.000	100.000	100.000	100.000	1.57E-01	0.352	1.00

Table 8.3 A comparison of algorithms performance metrics (C)

dataset	ALG	mean ACC	±std	min	Q1 (25%)	Q2 (50%)	Q3 (75%)	max	*p*-value vs best	time (s)	%Δ vs best
HTRU_2	ORIG	97.501	0.713	90.838	97.402	97.542	97.709	98.156	4.24E-01	1.540	0.06
	MT_F	97.561	0.222	96.983	97.422	97.569	97.709	98.268	best	9.449	best
	MT_FC	96.145	0.272	95.556	95.997	96.157	96.341	96.788	1.78E-97	7.545	1.47
	MT_H	97.483	0.699	90.838	97.402	97.542	97.661	98.045	2.89E-01	12.534	0.08
	MT_HC	96.156	0.250	95.503	95.999	96.173	96.341	96.788	1.66E-100	9.231	1.46
IONO	ORIG	90.457	4.332	72.857	88.530	91.429	94.203	98.571	3.34E-04	0.063	2.15
	MT_F	91.914	3.178	80.000	90.000	92.805	94.286	98.592	2.74E-01	0.187	0.53
	MT_FC	91.803	3.051	84.286	90.000	92.754	92.958	100.000	1.70E-01	0.168	0.65
	MT_H	92.401	3.094	82.857	90.000	92.857	94.286	100.000	best	0.261	best
	MT_HC	92.363	3.036	84.507	91.013	92.857	94.286	98.571	9.30E-01	0.243	0.04
IRIS	ORIG	95.200	7.048	33.333	93.333	96.667	96.667	100.000	1.47E-01	0.063	1.16
	MT_F	95.133	5.023	66.667	93.333	96.667	96.667	100.000	4.24E-02	0.195	1.23
	MT_FC	96.233	2.868	90.000	93.333	96.667	100.000	100.000	8.66E-01	0.126	0.07
	MT_H	95.533	3.489	83.333	93.333	96.667	96.667	100.000	8.45E-02	0.315	0.80
	MT_HC	96.300	2.717	90.000	93.333	96.667	96.667	100.000	best	0.202	best

Table 8.4 A comparison of algorithms performance metrics (D)

dataset	ALG	mean ACC	±std	min	Q1 (25%)	Q2 (50%)	Q3 (75%)	max	*p*-value vs best	time (s)	%Δ vs best
SEED	ORIG	90.190	9.337	33.333	88.095	90.476	95.238	100.000	5.58E-02	0.096	2.19
	MT_F	91.310	4.108	78.571	88.095	90.476	93.452	100.000	1.50E-01	0.098	0.94
	MT_FC	92.167	4.285	78.571	88.095	92.857	95.238	100.000	best	0.132	best
	MT_H	91.571	7.158	33.333	88.095	92.857	95.238	100.000	4.76E-01	0.272	0.65
	MT_HC	91.786	6.939	33.333	90.476	92.857	95.238	100.000	6.41E-01	0.238	0.42
SONAR	ORIG	77.691	6.335	64.286	73.171	78.571	82.927	90.476	2.50E-01	0.099	1.26
	MT_F	78.245	6.715	60.976	73.650	78.821	82.927	92.683	6.30E-01	0.210	0.54
	MT_FC	78.220	6.691	60.976	73.171	78.049	82.927	92.683	6.09E-01	0.203	0.57
	MT_H	77.638	6.668	58.537	73.171	78.049	82.927	95.122	2.39E-01	0.332	1.33
	MT_HC	78.667	5.621	63.415	75.610	78.821	82.927	90.698	best	0.326	best
USRKNWL	ORIG	60.539	5.289	48.750	57.317	60.380	63.750	73.418	6.46E-01	0.124	0.56
	MT_F	60.878	5.125	47.500	57.317	60.976	63.952	73.171	best	0.234	best
	MT_FC	51.303	5.011	37.805	48.588	51.235	54.430	64.634	1.96E-29	0.233	18.66
	MT_H	60.844	5.498	42.500	57.500	60.976	64.634	71.250	9.64E-01	0.487	0.06
	MT_HC	50.986	5.653	32.911	47.561	51.220	54.878	62.500	3.15E-28	0.489	19.40

Table 8.5 A comparison of algorithms performance metrics (E)

dataset	ALG	mean ACC	±std	min	Q1 (25%)	Q2 (50%)	Q3 (75%)	max	*p*-value vs best	time (s)	%Δ vs best
VEHICLE	ORIG	62.970	6.611	25.731	59.463	63.743	67.303	72.515	2.96E-11	0.175	8.45
	MT_F	63.009	7.304	25.595	60.325	63.717	67.303	77.515	7.60E-10	0.217	8.38
	MT_FC	66.817	3.044	60.947	64.881	66.567	69.151	73.373	2.22E-03	0.144	2.20
	MT_H	68.289	3.643	58.480	66.082	67.836	70.235	80.357	best	0.596	best
	MT_HC	66.748	2.704	60.819	64.912	66.567	68.303	74.854	8.24E-04	0.362	2.31
VOWEL	ORIG	74.374	9.368	9.091	71.717	75.758	79.293	86.364	6.02E-18	0.242	19.60
	MT_F	75.788	9.378	9.091	72.601	77.273	80.808	88.384	2.45E-15	0.379	17.37
	MT_FC	88.949	12.099	39.394	87.247	94.192	96.465	99.495	best	0.360	best
	MT_H	65.874	11.875	9.091	61.490	66.919	73.359	85.859	3.23E-30	0.861	35.03
	MT_HC	80.051	17.517	9.091	72.096	82.323	95.076	99.495	4.37E-05	0.837	11.12
WINE	ORIG	74.491	10.199	52.778	67.647	74.286	81.081	100.000	5.02E-09	0.088	10.84
	MT_F	72.883	11.461	38.889	66.667	73.251	80.556	97.059	1.08E-10	0.089	13.28
	MT_FC	82.563	8.375	38.889	80.417	82.857	86.486	94.595	best	0.131	best
	MT_H	77.310	11.141	38.889	70.588	77.778	85.714	97.297	2.16E-04	0.263	6.80
	MT_HC	82.310	7.730	38.889	78.228	83.333	88.235	94.595	8.25E-01	0.241	0.31

Table 8.6 A comparison of algorithms performance metrics (F)

dataset	ALG	mean ACC	±std	min	Q1 (25%)	Q2 (50%)	Q3 (75%)	max	*p*-value vs best	time (s)	%Δ vs best
WIS	ORIG	88.288	3.719	64.516	86.290	88.525	90.323	95.161	1.46E-02	0.126	1.26
	MT_F	87.499	3.172	76.613	85.366	87.805	89.516	93.548	5.50E-06	0.244	2.17
	MT_FC	89.327	2.864	81.967	87.780	89.431	91.129	95.161	8.56E-01	0.231	0.08
	MT_H	87.828	3.015	79.032	85.920	87.805	89.452	95.161	9.57E-05	0.486	1.79
	MT_HC	89.396	2.535	80.328	87.805	89.431	91.129	94.309	best	0.468	best
YEAST1	ORIG	76.556	1.925	70.034	75.421	76.728	78.114	81.145	2.28E-01	0.186	0.48
	MT_F	76.587	1.927	71.044	75.612	76.431	77.778	80.808	2.69E-01	0.396	0.44
	MT_FC	72.305	2.388	64.983	70.549	72.297	74.074	77.441	9.51E-31	0.191	6.39
	MT_H	76.924	2.350	72.054	75.274	77.104	78.788	81.757	best	0.793	best
	MT_HC	72.321	2.266	67.340	70.370	72.176	74.411	77.104	1.03E-31	0.404	6.36
ZOO	ORIG	90.376	6.485	71.429	85.714	90.476	95.238	100.000	4.58E-02	0.047	1.95
	MT_F	91.961	5.010	78.947	89.474	94.591	95.238	100.000	8.22E-01	0.160	0.19
	MT_FC	92.135	5.875	73.684	89.474	94.591	95.238	100.000	best	0.171	best
	MT_H	89.373	6.162	61.905	85.714	90.238	94.737	100.000	1.38E-03	0.261	3.09
	MT_HC	90.821	5.618	71.429	86.201	90.476	95.060	100.000	1.08E-01	0.271	1.45

Appendix B: Results in Forecasting Tasks

Forecasting experiment results using the non-linear autoregressive (NAR/NARX) model.

Table 8.7 Algorithms normalised mean square error (NMSE) performance comparison on Apple stock (AAPL)

Delay Window	lag: 5						lag: 10					
Algorithm	NAR-ORIG		NARX-MA		NARX-EMD		NAR-ORIG		NARX-MA		NARX-EMD	
Division/ANN neurons	10	50	10	50	10	50	10	50	10	50	10	50
Avg. Training Time (s)	2.113	1.551	2.766	3.493	3.041	4.269	2.456	2.288	3.157	5.394	3.224	7.332
Avg. Training Error	0.0071252	0.0070175	0.0070970	0.0069618	0.0042749	**0.0008638**	0.0070956	0.0069004	0.0070665	0.0068903	0.0033542	0.0010875
Avg. Test Error ± standard deviation	0.0016678 ±0.0009144	0.0037132 ±0.0011296	0.0017700 ±0.0010237	0.0040955 ±0.0010815	0.0007602 ±0.0011614	**0.0004631** **±0.0003893**	0.0013870 ±0.0010336	0.0042773 ±0.0021347	0.0014124 ±0.0013610	0.0046522 ±0.0020343	0.0007500 ±0.0013131	0.0008448 ±0.0015741
Minimum Test Error	0.0003573	0.0009636	0.0004469	0.0022597	0.0000508	0.0000798	0.0003758	0.0015388	0.0004493	0.0019610	**0.0000438**	0.0000701
p-value vs. best	1.81×10^{-25}	1.87×10^{-68}	6.71×10^{-25}	6.34×10^{-79}	1.67×10^{-02}	best	1.39×10^{-14}	5.01×10^{-42}	2.46×10^{-10}	9.11×10^{-50}	3.84×10^{-02}	2.02×10^{-02}

Table 8.8 Algorithms normalised mean square error (NMSE) performance comparison on Sunspot (SOL)

Delay Window	lag: 5						lag: 10					
Algorithm	NAR-ORIG		NARX-MA		NARX-EMD		NAR-ORIG		NARX-MA		NARX-EMD	
Division/ANN neurons	10	50	10	50	10	50	10	50	10	50	10	50
Avg. Training Time (s)	2.255	1.624	2.963	3.891	2.957	3.922	2.237	2.3	2.76	5.58	2.959	5.985
Avg. Training Error	0.1116833	0.1005595	0.1087313	0.0949837	**0.0394043**	0.0558120	0.1085390	0.0960426	0.1074148	0.0879532	0.0398371	0.0561283
Avg. Test Error ± standard deviation	0.3913484 ±0.7046393	0.8481608 ±0.4306742	0.3934549 ±1.1453019	0.8978903 ±1.1651971	**0.0974777 ±0.0870907**	0.6780297 ±0.5060787	0.2442256 ±0.0750881	0.6417264 ±0.4650035	0.2905302 ±0.1672015	0.6722254 ±0.6007605	0.0980438 ±0.0369733	0.6016824 ±0.4468190
Minimum Test Error	0.1883577	0.2696093	0.1931820	0.2058434	**0.0553965**	0.1643110	0.1886322	0.2116681	0.1804997	0.1979736	0.0609981	0.1521699
p-value vs. best	5.60×10^{-05}	1.49×10^{-40}	1.11×10^{-02}	1.10×10^{-10}	best	5.12×10^{-23}	2.05×10^{-27}	1.30×10^{-23}	7.19×10^{-20}	1.20×10^{-17}	9.53×10^{-01}	2.48×10^{-22}

Table 8.9 Algorithms normalised mean square error (NMSE) performance comparison on Santa Fe laser (SFL)

Delay Window	lag: 5						lag: 10					
Algorithm	NAR-ORIG		NARX-MA		NARX-EMD		NAR-ORIG		NARX-MA		NARX-EMD	
Division/ANN neurons	10	50	10	50	10	50	10	50	10	50	10	50
Avg. Training Time (s)	1.622	0.993	1.720	3.449	1.210	4.019	1.546	2.366	2.482	4.959	1.491	4.194
Avg. Training Error	0.0132433	0.0225547	0.0237647	0.0017928	0.0666883	0.0121078	0.0106071	**0.0004086**	0.0217637	0.0004789	0.0491124	0.0090752
Avg. Test Error ± standard deviation	0.0043834 ±0.0140755	0.0037581 ±0.0022762	0.0115036 ±0.0677390	0.0009243 ±0.0018475	0.0298414 ±0.0555601	0.0083077 ±0.0398612	0.0041026 ±0.0108263	**0.0004567 ±0.0001875**	0.0052185 ±0.0094118	0.0004919 ±0.0003914	0.0264680 ±0.0613395	0.0057052 ±0.0298941
Minimum Test Error	0.0007238	0.0017372	0.0004752	0.0003155	0.0020193	0.0008517	0.0003111	0.0002870	**0.0002730**	0.0002746	0.0006813	0.0003910
p-value vs. best	6.04×10^{-03}	1.39×10^{-32}	1.06×10^{-01}	1.30×10^{-02}	3.68×10^{-07}	5.14×10^{-02}	9.66×10^{-04}	best	1.08×10^{-06}	4.21×10^{-01}	3.73×10^{-05}	8.22×10^{-02}

Table 8.10 Algorithms normalised mean square error (NMSE) performance comparison on Istanbul Stock Exchange (ISE)

Delay Window	lag: 5						lag: 10					
Algorithm	NAR-ORIG		NARX-MA		NARX-EMD		NAR-ORIG		NARX-MA		NARX-EMD	
Division/ANN neurons	10	50	10	50	10	50	10	50	10	50	10	50
Avg. Training Time (s)	0.616	0.451	0.637	0.797	0.682	0.748	0.591	0.667	0.640	1.077	0.591	1.032
Avg. Training Error	1.0466828	0.8795638	1.0366650	0.8866556	**0.5577968**	0.7428110	1.0480104	0.8458651	1.0099425	0.8149923	0.6001350	0.8643662
Avg. Test Error ± standard deviation	0.8579863 ±0.0516803	0.9902402 ±0.1252702	0.8677769 ±0.0810234	0.9540938 ±0.1115357	0.4721791 ±0.1063513	1.0481726 ±0.2227919	0.7809116 ±0.0359027	0.9983421 ±0.1287332	0.7878874 ±0.0559552	0.8587937 ±0.0783127	**0.4165572 ±0.0513170**	0.8576149 ±0.0742370
Minimum Test Error	0.7707221	0.8026880	0.7634348	0.8174138	0.3335602	0.7699630	0.7152155	0.7854705	0.6834771	0.7091668	**0.3318121**	0.7032872
p-value vs. best	2.20×10^{-129}	7.32×10^{-101}	4.92×10^{-109}	2.15×10^{-103}	5.16×10^{-06}	1.64×10^{-69}	4.78×10^{-126}	3.92×10^{-100}	4.19×10^{-112}	2.41×10^{-109}	best	4.79×10^{-112}

Experiment results using the long short-term memory (LSTM) recurrent neural network.

Table 8.11 Algorithms normalised mean square error (NMSE) performance comparison on Apple stock (AAPL)

Delay Window	lag: 5						lag: 10					
Algorithm	LSTM-ORIG		LSTM-MA		LSTM-EMD		LSTM-ORIG		LSTM-MA		LSTM-EMD	
Division/ANN neurons	10	50	10	50	10	50	10	50	10	50	10	50
Avg. Training Time (s)	8.588	24.525	8.019	8.541	8.462	8.093	8.548	19.212	7.979	7.945	8.611	8.008
Avg. Training Error	0.1102562	0.0414285	0.1166426	0.0454050	0.1012255	0.1214790	0.0533478	**0.0355548**	0.0640460	0.0447802	0.0521724	0.0941153
Avg. Test Error ± standard deviation	0.1485295 ±0.0504588	0.0975778 ±0.0262894	0.1430151 ±0.0508515	0.0971847 ±0.0254819	0.1389314 ±0.0397565	0.2434674 ±0.0399610	0.0606585 ±0.0320497	0.0277204 ±0.0077224	0.0666034 ±0.0353700	**0.0272315 ±0.0099083**	0.0573101 ±0.0283412	0.0905659 ±0.0267799
Minimum Test Error	0.0456379	0.0314639	0.0302435	0.0458273	0.0450077	0.1499620	0.0029272	0.0112347	**0.0017439**	0.0054977	0.0038684	0.0398468
p-value vs. best	4.20×10^{-59}	6.08×10^{-63}	9.90×10^{-56}	2.32×10^{-64}	1.32×10^{-68}	8.59×10^{-118}	4.54×10^{-19}	6.99×10^{-01}	2.83×10^{-21}	best	3.17×10^{-19}	2.88×10^{-55}

Table 8.12 Algorithms normalised mean square error (NMSE) performance comparison on Sunspot (SOL)

Delay Window	lag: 5						lag: 10					
Algorithm	LSTM-ORIG		LSTM-MA		LSTM-EMD		LSTM-ORIG		LSTM-MA		LSTM-EMD	
Division/ANN neurons	10	50	10	50	10	50	10	50	10	50	10	50
Avg. Training Time (s)	28.97	14.047	26.731	9.322	11.165	14.471	25.122	14.363	24.434	14.118	10.443	14.044
Avg. Training Error	0.1328087	0.1319767	0.1446349	0.1762698	0.1340337	0.1248077	0.1173348	0.1148004	0.1161796	0.1103128	**0.0902734**	0.0916366
Avg. Test Error ± standard deviation	0.2399828 ±0.0261120	0.2326393 ±0.0289071	0.2931989 ±0.0783809	0.3847042 ±0.1012203	0.2736516 ±0.0599508	0.2270046 ±0.0092213	0.1970842 ±0.0038476	0.1904281 ±0.0024817	0.1925080 ±0.0089283	0.1775317 ±0.0041043	**0.1476351 ±0.0187093**	0.2451787 ±0.0128595
Minimum Test Error	0.2170601	0.2168433	0.2207024	0.2125954	0.2065012	0.2056850	0.1874098	0.1834852	0.1709400	0.1698466	**0.0876853**	0.2181468
p-value vs. best	3.01×10^{-72}	5.09×10^{-62}	1.80×10^{-43}	1.35×10^{-57}	2.61×10^{-49}	1.28×10^{-92}	3.88×10^{-65}	1.27×10^{-56}	8.78×10^{-54}	4.26×10^{-36}	best	6.25×10^{-102}

Table 8.13 Algorithms normalised mean square error (NMSE) performance comparison on Santa Fe laser (SFL)

Delay Window	lag: 5						lag: 10					
Algorithm	LSTM-ORIG		LSTM-MA		LSTM-EMD		LSTM-ORIG		LSTM-MA		LSTM-EMD	
Division/ANN neurons	10	50	10	50	10	50	10	50	10	50	10	50
Avg. Training Time (s)	8.611	14.114	9.332	13.055	14.9695	11.737	9.95	14.079	9.667	11.7	8.717	14.801
Avg. Training Error	0.1801327	0.1110482	0.2093012	0.1134647	0.3633793	0.1815522	0.1286195	**0.0688271**	0.2099519	0.1354106	0.2999538	0.1037100
Avg. Test Error ± standard deviation	0.0559024 ±0.0109374	0.0320430 ±0.0024095	0.0852827 ±0.0300794	0.0439690 ±0.0263654	0.2403362 ±0.0383493	0.0501263 ±0.0092680	0.0461984 ±0.0277388	**0.0215419 ±0.0157497**	0.0846316 ±0.0376178	0.0512063 ±0.0184436	0.1679626 ±0.0297202	0.0413049 ±0.0068129
Minimum Test Error	0.0320886	0.0275861	0.0280625	0.0183325	0.1163166	0.0346829	0.0151599	**0.0062204**	0.0420030	0.0337207	0.0699817	0.0239597
p-value vs. best	4.82×10^{-43}	4.65×10^{-10}	1.47×10^{-45}	8.34×10^{-12}	3.52×10^{-118}	3.37×10^{-36}	6.70×10^{-13}	best	1.12×10^{-35}	8.37×10^{-26}	6.01×10^{-103}	1.19×10^{-23}

Table 8.14 Algorithms normalised mean square error (NMSE) performance comparison on Istanbul Stock Exchange (ISE)

Delay Window	lag: 5						lag: 10					
Algorithm	LSTM-ORIG		LSTM-MA		LSTM-EMD		LSTM-ORIG		LSTM-MA		LSTM-EMD	
Division/ANN neurons	10	50	10	50	10	50	10	50	10	50	10	50
Avg. Training Time (s)	10.196	9.938	14.837	9.787	10.333	11.108	9.47	9.876	9.626	9.654	9.733	10.293
Avg. Training Error	1.2221676	1.1722678	1.1575247	1.1340869	1.0323971	1.1280384	1.1382452	1.1299286	1.1250142	1.1185877	**0.9930821**	1.0977961
Avg. Test Error ± standard deviation	0.8087027 ±0.0319960	0.7857503 ±0.0253995	0.7791352 ±0.0138709	0.7713591 ±0.0026795	**0.6444975 ±0.0672950**	0.7715180 ±0.0032700	0.7830149 ±0.0070671	0.7808031 ±0.0080943	0.6521218 ±0.0564859	0.7650546 ±0.0046042	0.6521219 ±0.0564861	0.7560338 ±0.0085550
Minimum Test Error	0.7701950	0.7690272	0.7541847	0.7637071	**0.4343486**	0.7632934	0.7688600	0.7757211	0.4754810	0.7484215	0.4754810	0.7414659
p-value vs. best	7.19×10^{-55}	4.45×10^{-48}	5.92×10^{-48}	9.55×10^{-46}	best	8.55×10^{-46}	1.81×10^{-50}	1.95×10^{-49}	3.89×10^{-01}	6.62×10^{-43}	3.89×10^{-01}	1.27×10^{-38}