FEDERAL UNIVERSITY OF JUIZ DE FORA
FACULTY OF ENGINEERING
GRADUATE PROGRAM IN COMPUTATIONAL MODELING

Kaike Sa Teles Rocha Alves

NFISiS: New Perspectives on Fuzzy Inference Systems Applied to Renewable Energy, Finance, and Cryptocurrency

Juiz de Fora

2025

Kaike Sa Teles Rocha Alves

NFISiS: New Perspectives on Fuzzy Inference Systems Applied to Renewable Energy, Finance, and Cryptocurrency

A thesis submitted to the Graduate Program in Computational Modeling of the Federal University of Juiz de Fora in partial fulfillment of the requirements for the degree of Doctor of Philosophy (PhD) in Computational Modeling.

Supervisor: Prof. Dr. Eduardo Pestana de Aguiar

Co-supervisor: Prof. Dr. Direnc Pekaslan

Juiz de Fora

2025

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF
com os dados fornecidos pelo(a) autor(a)

Alves, Kaike Sa Teles Rocha.
NFISiS: New Perspectives on Fuzzy Inference Systems Applied to Renewable Energy, Finance, and Cryptocurrency / Kaike Sa Teles Rocha Alves. – 2025.
153 f. : il.

Supervisor: Eduardo Pestana de Aguiar
Co-supervisor: Direnc Pekaslan
PhD Thesis – Federal University of Juiz de Fora, Faculty of Engineering. Graduate Program in Computational Modeling, 2025.

1. Fuzzy inference systems. 2. Mamdani. 3. Takagi-Sugeno-Kang. 4. Feature selection. 5. Time series forecasting. I. Aguiar, Eduardo Pestana de, supervisor. II. Pekaslan, Direnc, co-supervisor. III. Título.

Kaíke Sá Teles Rocha Alves

NFISiS: New Perspectives on Fuzzy Inference Systems Applied to Renewable Energy, Finance, and Cryptocurrency

Tese apresentada ao Programa de Pós-Graduação em Modelagem Computacional da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Doutor em Modelagem Computacional. Área de concentração: Modelagem Computacional.

Aprovada em 05 de novembro de 2025.

BANCA EXAMINADORA

Prof. Dr. Eduardo Pestana de Aguiar - Orientador
Universidade Federal de Juiz de Fora

Prof. Dr. Leonardo Goliatt da Fonseca
Universidade Federal de Juiz de Fora

Prof. Dr. Moises Vidal Ribeiro
Universidade Federal de Juiz de Fora

Prof. Dr. Frederico Gadelha Guimarães
Universidade Federal de Minas Gerais

Prof. Dr. Petrônio Cândido de Lima e Silva
Instituto Federal do Norte de Minas Gerais

Prof. Dr. Arthur Caio Vargas e Pinto
Instituto Federal de Minas Gerais

Juiz de Fora, 03/11/2025.

Documento assinado eletronicamente por **Eduardo Pestana de Aguiar**, **Professor(a)**, em 06/11/2025, às 16:19, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Frederico Gadelha Guimarães**, **Usuário Externo**, em 07/11/2025, às 13:23, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Petrônio Cândido de Lima e Silva**, **Usuário Externo**, em 07/11/2025, às 14:48, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Arthur Caio Vargas e Pinto**, **Usuário Externo**, em 07/11/2025, às 18:04, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Moises Vidal Ribeiro**, **Professor(a)**, em 11/11/2025, às 07:28, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro

de 2020.

Documento assinado eletronicamente por **Leonardo Goliatt da Fonseca**, **Professor(a)**, em 13/11/2025, às 14:30, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no Portal do SEI-Ufjf (www2.ufjf.br/SEI) através do ícone Conferência de Documentos, informando o código verificador **2717841** e o código CRC **032D62B2**.

To my beloved wife, Carmen Alves,
and our dear son, Samuel

ACKLOWLEDGMENT

I thank Jesus Christ for all He has done in my life and all His love for me. He is my treasure.

To my beloved wife, for supporting and encouraging me during challenging moments.

To my parents, Gilterney and Rita, for their precious teachings and for supporting my career.

To my sisters, Keyla and Anna Klara, from whom I have learned so much.

To all the friends who have been and still are part of my life.

To my supervisor, Eduardo Aguiar, who gave me valuable advice and assisted me throughout my journey.

To Christian Wagner and Direnc Pekaslan, for the opportunity to study at the University of Nottingham for one year during my Ph.D. and for their supervision during that time.

To the professors of the Department of Computational Modeling at the Federal University of Juiz de Fora.

To all the people who contributed to my progress.

I acknowledge the Federal University of Juiz de Fora for essential support during this work and thank the anonymous referees for their valuable comments.

I also acknowledge the financial support provided by CAPES (Finance Code 001) and FAPEMIG.

No man ever steps in the same river twice,
for it's not the same river and he's not the same man.
Heraclitus

ABSTRACT

Fuzzy inference systems are a special class of machine learning models known for balancing accuracy with interpretability. The two main types of fuzzy inference systems are Mamdani and Takagi-Sugeno-Kang. While Mamdani models prioritize interpretability, Takagi-Sugeno-Kang models achieve higher accuracy by approximating nonlinear systems through a collection of linear subsystems. However, designing fuzzy rules lacks a standardized method, often leading to models with excessive hyperparameters, high complexity, and no direct control over the number of rules. To address such challenges, this work introduces two new data-driven approaches for designing Mamdani and Takagi-Sugeno-Kang rules. Among the advantages of the proposed fuzzy inference systems with the new mechanism for rule generation, the following can be highlighted: reduced complexity, fewer hyperparameters, enhanced interpretability, and direct control over the number of final rules. Additionally, feature selection techniques, including genetic algorithms and ensemble methods, are integrated to improve the models' ability to handle large datasets, optimize performance, increase interpretability, and prevent overfitting. The introduced models result in the NFISiS, a new collection of data-driven fuzzy inference systems for time series forecasting. The proposed models are evaluated using benchmark time series, renewable energy, financial, and cryptocurrency datasets in terms of errors and the number of rules. Their performance is compared against state-of-the-art machine learning models, including classical approaches, deep learning architectures, and rule-based evolving Fuzzy Systems. The results indicate that the proposed models effectively handle complex, non-stationary datasets, such as those in finance and cryptocurrency. All proposed models are available as a Python package, which can be installed via pip: `pip install nfisis`(`https://pypi.org/project/nfisis`). Furthermore, the source code used for the simulations are available at `https://github.com/kaikerochaalves/nfisis_thesis.git`.

Keywords: Fuzzy inference system. Mamdani. Takagi-Sugeno-Kang. Feature selection. Time series forecasting.

RESUMO

Sistemas de inferência *fuzzy* consistem em uma classe especial de modelos de aprendizado de máquina conhecidos por balancear acurácia com interpretabilidade. Os dois principais tipos de sistemas de inferência *fuzzy* são Mamdani e Takagi-Sugeno-Kang. Enquanto os modelos Mamdani priorizam a interpretabilidade, os modelos Takagi-Sugeno-Kang normalmente obtêm menores erros devido a sua habilidade de aproximar sistemas não lineares por meio de uma coleção de subsistemas lineares. Contudo, a criação do conjunto de regras *fuzzy* não possuem uma abordagem padronizada, o que frequentemente resulta em modelos com excesso de hiperparâmetros, alta complexidade e sem controle direto sobre o número de regras. Para superar tais desafios, este trabalho introduz duas novas abordagens para geração de uma base de regras *fuzzy* baseadas em dados, tanto para modelos Mamdani quanto para os Takagi-Sugeno-Kang. Entre as vantagens dos sistemas propostos, destacam-se: complexidade reduzida, menor número de hiperparâmetros, maior interpretabilidade e controle direto sobre o número final de regras. Adicionalmente, técnicas de seleção de atributos, incluindo algoritmos genéticos e métodos ensemble, são integradas para melhorar a capacidade dos modelos em lidar com grandes conjuntos de dados, otimizar o desempenho, aumentar a interpretabilidade e evitar o sobreajuste (*overfitting*). Os modelos introduzidos resultam no NFISiS, uma nova coleção de sistemas de inferência *fuzzy* dirigida a dados para a previsão de séries temporais. Os modelos propostos são avaliados utilizando séries temporais sintéticas, bem como conjuntos de dados de energia renovável, dados financeiros e de criptomoedas. Os resultados são avaliados em termos de erros e número de regras. O desempenho dos modelos propostos é comparado com o de modelos de aprendizado de máquina no estado da arte, incluindo abordagens clássicas, arquiteturas de aprendizado profundo (*deep learning*) e sistemas *fuzzy* evolutivos baseados em regras. Os resultados indicam que os modelos propostos lidam eficazmente com conjuntos de dados complexos e não estacionários, como os das áreas financeira e o de criptomoedas. Todos os modelos propostos estão disponíveis como um pacote Python, que pode ser instalado via pip: `pip install nfisis`(`https://pypi.org/project/nfisis`). Além disso, o código-fonte utilizado para as simulações está disponível em `https://github.com/kaikerochaalves/nfisis_thesis.git`.

Palavras-chave: Sistema de inferência *fuzzy*. Mamdani. Takagi-Sugeno-Kang. Seleção de atributos. Previsão de séries temporais.

LIST OF ABBREVIATIONS AND SYMBOLS

ADF	Augmented Dickey-Fuller
ANFIS	Adaptive neural-fuzzy inference system
ANN	Artificial neural network
ARIMA	Autoregressive integrated moving average
ACO-SVM	Ant colony optimization support vector machine
CT-NET	Convolutional-transformer-based network
DENFIS	Dynamic evolving neural-fuzzy inference system
DL	Deep learning
DT	Decision tree
E-SVR&RF	Ensemble learning-Support Vector Regression & Random Forest
EFuNN	evolving fuzzy neural network
ePL	evolving Participatory learning
ePL+	enhanced Evolving Participatory Learning
ePL-KRLS-DISCO	evolving participatory learning with kernel recursive least squares and distance correlation
eTS	evolving Takagi-Sugeno
exTS	evolving extended Takagi-Sugeno
GA	Genetic algorithm
GBM	Gradient boosting machines
GK	Gustafson Kessel
GP	Grid partitioning
IS	Interval size
KNN	K-nearest neighbors
LS-SVM	Least-squares support-vector machines
LSTM	Long short-term memory
$\bar{m}_{\Delta}$	Maximum variation of the target value
$\underline{m}_{\Delta}$	Minimum variation of the target value
MAE	Mean absolute error
NASDAQ	National Association of Securities Dealers Automated Quotations
NDEI	Non-dimensional index error
NTSK	New Takagi-Sugeno-Kang
OptiFel	Optimizing fuzzy model
P	Covariance matrix of the adaptive filtering
p	Number of attributes in the input vector
PV	Photovoltaic
R	Fuzzy rule
$R_{\max}$	Maximum number of rules
RF	Random forest
RLS	Recursive least squares
RMSE	Root-mean-square error

S&P 500	Standard & Poor's 500
Simpl_eTS	Simplified evolving Takagi-Sugeno
SA	Simulated annealing
SVM	Support vector machine
TAIEX	Taiwan Stock Exchange Capitalization Weighted Stock Index
TSK	Takagi-Sugeno-Kang
TSEC	Taiwan Stock Exchange Corporation
v	Mean of the Gaussian fuzzy set
w	Firing strength
wRLS	Weighted recursive least squares
exTS	evolving extended Takagi-Sugeno
x	Input vector containing the attributes
X	Universe of discourse for x
XGBoost	Extreme gradient boosting
y	Actual output or target value
$\hat{y}$	Estimated or predicted output
θ	Parameters of the consequent
μ	Membership function
λ	Forgetting factor
σ	Standard deviation of the Gaussian fuzzy set

CONTENTS

1 INTRODUCTION

Deep learning (DL) techniques have become increasingly popular due to technological advances and the growing complexity of databases. However, DL models are more intricate and susceptible to several issues, such as vanishing gradients, overfitting, long training times [1], and a lack of interpretability due to their black-box nature [2, 3]. In contrast, fuzzy inference systems (FIS), introduced by Zadeh [4], constitute a class of machine learning models that balance accuracy and interpretability. These systems have been successfully applied in various fields, including finance, business, and management [5, 6, 7, 8], renewable energy [9, 10], medicine [11, 12], engineering [13, 14], chemistry [15, 16], scheduling [17], pattern recognition [18], and fault detection [19].

There are two primary types of fuzzy models: Mamdani [20] and Takagi-Sugeno-Kang (TSK) [21, 22]. These fuzzy rules are divided into two main parts: the antecedent and the consequent. In these models, data are modeled through rules and fuzzy sets. The antecedent part models the inputs, while the consequent part models the output. Although both Mamdani and TSK models share the same approach in the antecedent part, they differ in their consequent parts. Specifically, Mamdani systems use fuzzy sets in the consequent, which tends to provide more interpretable results [23]. Miller defines interpretability as the ability to explain or justify the reasoning behind a given result to humans [24, 25].

On the other hand, TSK implements polynomial functions in the consequent, which typically provide more accurate results and require fewer rules to describe highly nonlinear and complex systems, generally fewer than those of Mamdani models [26, 27]. The core idea behind TSK models is to approximate a nonlinear system by combining multiple local linear subsystems. The output is estimated as a weighted combination of the local outputs from each rule [28, 29].

Despite the clear advantages of FIS, their practical design and optimization present significant challenges, which form the central motivation for this thesis. There is no standardized method for defining rules, leading to several limitations in the existing approaches: (i) rule generation mechanisms are often complex, requiring hybridizations that increase the number of hyperparameters; (ii) usually there are no direct control over the final number of rules, forcing an unpredictable trade-off between accuracy and interpretability; and (iii) the TSK model, while accurate, suffers from a lack of interpretability in its consequent part, which uses polynomial functions. These gaps highlight a clear need for a new family of FIS that are simultaneously interpretable, controllable, and less complex to configure.

For this reason, this thesis presents the outcome of in-depth research into FIS by proposing the new fuzzy inference systems (NFISiS), a collection of novel data-driven

fuzzy modeling approaches that define fuzzy rules based on a specified number of rules and construct the fuzzy sets based on the target value. NFISiS offers three key advantages: first, it allows users to specify the exact number of rules, enabling them to balance accuracy and interpretability according to their needs. Second, by defining rules based on the target value, the model establishes clearer correlations between the target and its attributes. Finally, NFISiS reduces the number of hyperparameters and overall model complexity, making it well-suited for real-world applications.

NFISiS includes both Mamdani and TSK variants, referred to as the new Mamdani fuzzy inference system (NMFIS) and the new Takagi-Sugeno-Kang system (NTSK) [30, 31], respectively. Additionally, NMFIS is further categorized into the new Mamdani classifier (NMC) and the new Mamdani regressor (NMR). As will be discussed later, due to its structural formulation, NTSK is only applicable to regression problems. Since NTSK computes the output using polynomial functions, two recursive adaptive filtering techniques are implemented in the consequent part of NTSK to estimate the consequent parameters: the recursive least squares (RLS) and the weighted recursive least squares (wRLS).

Additionally, with the growing complexity of databases, one challenge remains: which attributes should be selected to optimize the model's performance while considering the trade-off between accuracy and interpretability? The answer to this question depends on feature selection techniques, which can be divided into five main types: filter, wrapper, embedded, hybrid, and ensemble [32]. The first one, filter methods, are statistically based approaches, but they neglect the integration between the selected subset and the performance of the learning algorithm [33]. In contrast, wrapper methods use a metric that measures the learning algorithm's performance to identify the feature set that leads to the best results. However, since it is infeasible to test all possible combinations of features, heuristic and metaheuristic approaches are employed, such as randomized search [34], genetic algorithm (GA) [35], and ant colony optimization [36].

In embedded methods, the feature selection technique is integrated into the learning algorithm by adjusting the model's internal parameters. Decision trees, random forest (RF), and gradient boosting are examples of embedded feature selection techniques [37]. Hybrid methods combine multiple feature selection approaches in a multi-step process [38]. Finally, ensemble methods implement multiple weak learning algorithms and combine their results to achieve better performance than any individual model [39, 40]. The key difference between hybrid and ensemble methods is that hybrid models use multiple models to obtain a single result, while ensemble methods generate one output per model and combine the outputs into a final result.

In this scenario of increased complexity of the datasets and aiming to improve the performance of the introduced data-driven approaches to real-world applications, two feature selection paradigms, a wrapper and an ensemble, are combined with NMR and

NTSK. The goal is to produce FIS capable of handling large datasets while ensuring optimized performance, enhanced interpretability, and a reduced proneness to overfitting. The wrapper technique consists of a GA approach to search for a subset of input attributes that minimizes the error, deriving two models: the genetic new Mamdani regressor (GEN-NMR) and the genetic new Takagi-Sugeno-Kang (GEN-NTSK). In addition, the NFISiS framework incorporates three ensemble-based models: the random new Mamdani regressor (R-NMR), the random new Takagi-Sugeno-Kang (R-NTSK), and the random forest new Takagi-Sugeno-Kang (RF-NTSK). To evaluate the performance of the proposed models, renewable energy, financial, and cryptocurrency datasets are used. These datasets were chosen because of their complexity and relevance in many real-world applications.

Photovoltaic (PV) energy is expected to become one of the primary sources of energy worldwide, as it is abundant, affordable, and easily scalable [41]. However, the energy supplied by PV modules can be intermittent due to the varying nature of weather conditions [42]. Forecasting the power generated in a PV plant is crucial, as it helps grid operators, plant managers, and energy markets anticipate and manage the variability inherent in PV energy generation. Accurate forecasts optimize energy production, ensure a stable and reliable power supply, and contribute to cost reduction, efficient resource planning, and the overall sustainability of the energy landscape. Furthermore, the chaotic nature and uncertainties associated with PV energy make it difficult to obtain accurate results using physical equations. This challenge has led to the increased use of machine learning models for predicting PV energy generation [43]. On the other hand, obtaining returns on investments in the stock market is challenging due to its complex nature. As a result, machine learning models assist investors in developing investment strategies [44, 45].

To measure the models' performance on the PV datasets, the data are partitioned into training, validation, and test sets. The hyperparameters are defined using the validation set, and the final errors are reported on the test set. On the other hand, for the cryptocurrency and financial datasets, sliding-window cross-validation is used, and the results are reported as the mean and standard deviation of the results of each subset. Furthermore, the number of fuzzy rules is reported for the fuzzy models. The results of the proposed models are compared with those of classical models, DL, and evolving Fuzzy Systems (eFSs) to enrich the work and provide more comprehensive conclusions. Statistical tests are included to validate the results. Finally, a detailed discussion regarding rules, interpretability, and computational cost is presented.

1.1 RESEARCH QUESTIONS

To address the identified gaps, this thesis seeks to answer the following primary research questions:

- How can novel data-driven algorithms for Mamdani and TSK models be designed to provide direct, explicit control over the number of rules, thereby enabling a clearer balance between predictive accuracy and model interpretability?
- To what extent do adaptive filtering techniques, specifically RLS and wRLS, enhance the parameter estimation and predictive performance of the proposed TSK models compared to standard methods?
- How does the integration of GA-based wrapper and ensemble-based feature selection methods impact the performance, interpretability, and generalization of the proposed fuzzy systems when applied to high-dimensional data?
- How does the performance of the proposed NFISiS framework compare against state-of-the-art classical, DL, and eFSs when applied to complex, real-world time series in renewable energy, finance, and cryptocurrency?

1.2 OBJECTIVES

To answer these research questions, the main objective of this thesis is to develop and evaluate a new family of fuzzy inference systems, called NFISiS, designed to address key limitations in existing data-driven fuzzy modeling techniques. To achieve this goal, the following specific objectives are defined:

- To develop novel data-driven algorithms for constructing both Mamdani and TSK fuzzy models, providing direct control over the number of rules to balance model accuracy and interpretability.
- To investigate the effectiveness of RLS and wRLS adaptive filtering techniques for estimating the consequent parameters in the proposed TSK models.
- To incorporate a GA-based wrapper method and an ensemble approach into the proposed FIS to perform automatic feature selection, enhancing model performance and interpretability.
- To implement the proposed models in a user-friendly, open-source Python library named `nfisis` to promote reproducibility and facilitate further research.
- To validate the performance of the proposed models through extensive simulations on complex, real-world time series datasets from the domains of renewable energy, finance, and cryptocurrency.
- To conduct a comprehensive comparative analysis of the proposed models against classical, DL, and state-of-the-art eFSs to benchmark their effectiveness.

1.3 MAIN CONTRIBUTIONS

This work introduces and discusses NFISiS, a collection of six novel FIS. The contributions of this work are summarized as follows:

- New data-driven Mamdani and TSK models are proposed.
- Two adaptive filtering approaches, the RLS and wRLS, are tested for estimating the polynomial functions in the consequent part of TSK-based models.
- GA and ensemble approaches are incorporated into the proposed data-driven FIS for feature selection.
- The development and public release of `nfisis`, an open-source Python library containing the implementations of all proposed models and feature selection techniques.
- The proposed models are extensively tested on PV, financial, and cryptocurrency datasets.
- The presented models include direct control over the number of rules, enhanced interpretability and explainability, a simplified structure with fewer hyperparameters, lower prediction errors, and strong performance when dealing with complex series.
- Extensive simulations are conducted not only to validate the performance of the introduced models but also to compare them with classical models, DL, and eFSs, which are widely referenced in the literature.

1.4 ORGANIZATION OF THE MANUSCRIPT

The remainder of this document is organized as follows:

- Chapter 2 presents a detailed literature review on Mamdani, TSK, GA, and ensembles. Additionally, it examines previous research on PV, financial, and cryptocurrency applications. Whenever possible, the review highlights the advantages and limitations of existing approaches.
- Chapter 3 provides background knowledge and mathematical formalisms for FIS, GA, and ensemble methods.
- Chapter 4 introduces NFISiS, emphasizing its novel approaches to designing Mamdani and TSK data-driven algorithms and their inference processes. Additionally, this chapter outlines the benefits of the models' integrated GA and ensemble approaches. Finally, pseudo-code for the introduced algorithms is provided.

- Chapter 5 details the datasets and their statistical characteristics, presents the tools and devices utilized, introduces about the developed library, and outlines the error metrics and simulation setup.
- Chapter 6 reports the simulation results and the statistical tests. Furthermore, technical aspects are discussed, such as interpretability, computational performance, and the stability of the proposed stochastic algorithms. Finally, observed limitations are presented.
- Finally, Chapter 7 concludes the work, summarizes the findings, and proposes directions for future research.

2 LITERATURE REVIEW

This chapter presents a comprehensive review of the two FIS types, Mamdani and TSK, highlighting their advantages and limitations. Subsequently, it provides an overview of GA and ensemble methods, the two feature selection techniques implemented in this work. Finally, it examines forecasting techniques applied to PV, financial, and cryptocurrency datasets, which constitute the applications selected for this work.

2.1 FUZZY INFERENCE SYSTEMS

This section reviews the literature on FIS, presenting the strengths and weaknesses of the Mamdani and TSK for time series forecasting.

2.1.1 MAMDANI FUZZY MODELS

Anderson and Hall [46] proposed the Mamdani-style Fuzzy Inference System (MR. FIS), which generates Mamdani rules based on the knowledge embedded in a neural network. Duan and Chung [47] introduced a Mamdani-type Multistage Fuzzy Neural Network to address the curse of dimensionality problem in single-stage fuzzy networks. Their findings indicate that the proposed model achieves faster convergence, greater robustness, and reduced complexity. Ying [48] extended a single-input single-output (SISO) Mamdani controller to a two-input two-output (TITO) and analyzed the resulting performance. Uncu et al. [49] developed a Mamdani FIS that implements fuzzy c-means to define Type-2 fuzzy sets, concluding that the proposed method yields more accurate results. Kaymak et al. [50] introduced a probabilistic Mamdani fuzzy system, while Chai et al. [51] implemented the Mamdani model-based Adaptive Neural Fuzzy Inference System (M-ANFIS). Their work demonstrated that M-ANFIS outperforms the standard adaptive neural-fuzzy inference system (ANFIS) in terms of hyperparameter efficiency, training data requirements, runtime, and error reduction. Tung and Quek [52] discussed the Mamdani-Takagi-Sugeno linguistic neural-fuzzy inference system (MTS-LiNFIS), a hybrid neuro-fuzzy system that combines the explanatory trait of Mamdani with the accuracy of TSK.

Gacto et al. [53] proposed the METSK-HDe, a fuzzy system that uses an evolutionary algorithm to build the rule-based structure and a rule selection approach in the post-processing stage. Gou et al. [54] implemented a modified fuzzy c-means to improve the fuzzy rules of the Wang-Mendel algorithm even in the presence of noise and outliers. Asadi [55] implemented a GA to extract the rules of a Mamdani FIS. Pekaslan et al. [56] developed the ADONiS model, which adjusts the standard deviation of the Gaussian fuzzy sets to enhance robustness and adaptability under uncertainty. Kacimi et al. [57] utilized particle swarm optimization (PSO) for the automatic learning and self-adaptation

of Mamdani rules. Zhang and Wagner [58] applied Markov blankets to establish causal relationships between input and output variables to limit the number of rules by exploring the concept of causality. Navarro-Almanza et al. [59] used a GA to build a fuzzy linguistic interpretable model from an optimized Mamdani neuro-fuzzy model. Zhang et al. [60] proposed a fuzzy algorithm that clusters the dataset to perform the regression task.

2.1.2 TSK FUZZY MODELS

Since the introduction of TSK, many researchers have explored and expanded upon this topic. Fantuzzi and Rovatti [61] investigated the approximation capability of homogeneous TSK models. Joh et al. [62] addressed stability issues in TSK systems, proposing a method to find a common symmetric positive-definite matrix for linear discrete TSK fuzzy models. Similarly, Nguyen et al. [63] provided a comprehensive review of the stability of fuzzy control systems. From an interpretability perspective, Johansen et al. [27] analyzed TSK models and provided practical examples to enhance their comprehensibility. Ying [64] established sufficient conditions for system configuration, confirming TSK models as universal approximators within single-input single-output (SISO) systems.

Several evolving neural-fuzzy models have been proposed, including the adaptive neuro-fuzzy inference system (ANFIS) [65], the evolving fuzzy neural network (EFuNN) [66], and the dynamic evolving neural-fuzzy inference system (DENFIS) [67]. DENFIS incrementally creates and updates fuzzy rules in an online manner as new data arrives. The limitations of DENFIS include: (i) it can generate a high number of rules; (ii) it requires substantial training data to minimize errors; and (iii) it may produce significant errors if the training data inadequately represents the input domain.

In the context of evolving models, Angelov and Filev [68] proposed the evolving Takagi-Sugeno (eTS) model, a rule-based eFS. The eTS model dynamically updates its rules and parameters as new data becomes available, enabling it to detect data shifts and drifts. Inspired by eTS, various extensions were developed, such as Simpl_eTS [69], exTS [70], FLEXFIS [71], ePL [72], ePL+ [73], eMG [74], and ePL-KRLS-DISCO [75]. Despite their advantages, these models often require tuning various hyperparameters for rule creation, deletion, and update rates. Additionally, their continuous rule revision process can reduce explainability by producing models with too few or too many number of rules across different stages.

Kukolj and Levi [26] proposed a neural network-based TSK model with improved transparency, fewer rules, and higher accuracy, though they did not include a discussion of hyperparameters or provide implementation details. In subsequent work, Vernieuwe et al. [76] explored clustering methods and introduced ClusterFinder, an algorithm that identifies optimized fuzzy sets. Kung and Su [77] implemented an approach that defines the fuzzy sets using a c-regression model; however, this approach offers no direct control

over the number of rules and fails to show how the number of clusters affects model error and rule count. Du and Zhang [78] implemented a GA to optimize the number of rules and the parameters of the membership functions in a TSK model and perform feature selection. Nevertheless, the number of rules and computation time increase substantially with the dimensionality of the input.

Rezaee and Zarandi [79] proposed a hybrid approach that starts with a single rule and incrementally adds rules until a stopping criterion is met. Xu et al. [80] manually designed TSK rules for a two-wheeled mobile robot using domain knowledge. Cheung et al. [81] introduced the optimizing fuzzy model (OptiFel), which utilizes a heterogeneous multi-swarm PSO algorithm to address limitations of classical PSO, such as premature convergence and susceptibility to local optima, thereby improving TSK model accuracy. Precup et al. [82] implemented two optimization algorithms to optimize TSK rules: simulated annealing (SA) and PSO. The authors report three main advantages of the proposed models: (i) the simplicity of the models' structure; (ii) their consistency across both training and testing data; and (iii) the transparency of the overall fuzzy modeling approach.

Vrkalovic et al. [83] compared the performance of three optimization algorithms in designing TSK rules: PSO, SA, and gravitational search algorithm (GSA). Tsai and Chen [84] introduced a new identification method for TSK that uses the Xie-Beni index algorithm to define the number of rules and implemented PSO and orthogonal least-squares (OLS) to compute the matrices and parameters of the consequent part. Lai et al. [85] proposed a technique to learn TSK rules from data with missing values. More recently, Zander et al. [86] proposed reinforcement learning (RL) to optimize TSK rules. In a related way, Fang et al. [87] presented a self-learning TSK that implements RL in the learning process. Zhang et al. [88] developed an ensemble model named D-RSP-TSKE that utilizes a deep reconciled and self-paced TSK fuzzy system for imbalanced data classification. The authors report the model achieves improved interpretability.

2.2 FEATURE SELECTION

This subsection reviews the literature on the two feature selection approaches implemented in this work: GA and ensembles.

2.2.1 GENETIC ALGORITHMS

Holland made a foundational contribution to evolutionary algorithms with the introduction of GA in his books [89, 90]. Although he did not introduce the concept of evolutionary algorithms, he was the first to use this term [91]. Since then, many researchers have applied GA to a wide variety of problems. Shi et al. [92] implemented the PSO-GA, a meta-heuristic approach that achieves better results than using either

PSO or GA individually. Juang [93] discussed the hybrid GA particle swarm optimization (HGAPSO) in the design of a recurrent neural network. Wang [94] presented the hybrid genetic algorithm–neural network strategy (GA–NN). Sheikh et al. [95] utilized GA in the context of clustering. Syarif et al. [96] developed a GA to optimize SVM parameters.

Ding and Fu [97] introduced GAKFCM, a combination of an improved GA with kernel-based fuzzy c-means to obtain clusters with improved performance. Sehgal et al. [98] implemented a GA to optimize the parameters of a deep deterministic policy gradient. Zivkovic et al. [99] discussed the implementation of a GA to optimize the hyperparameters of a neuro-fuzzy inference system. Gao et al. [100] proposed DockingGA, a model that combines a GA with transformer neural networks to generate molecules with better binding affinity. Furthermore, some researchers have utilized GA in the context of feature selection. Maleki et al. [101] used a GA for feature selection in medical images. Guha et al. [102] proposed the combination of a GA with the great deluge algorithm to obtain increased exploitation capability. Halim et al. [103] successfully applied an enhanced GA for feature selection. Altarabichi et al. [104] implemented a fast GA for attribute selection using decision trees.

2.2.2 ENSEMBLE MODELS

Sagi and Rokach [105] classified ensemble learning into two main frameworks: dependent and independent. In the dependent framework, the learners are interdependent. The learners are built in sequence, and the outputs of the k-th estimator are used to guide the learning of the estimator at step $k+1$. On the other hand, in the independent framework, the estimators are built independently of one another. AdaBoost, introduced by Freund and Schapire [106], exemplifies the dependent framework by iteratively assigning weights to samples based on the current model's errors and subsequently inducing a new estimator. This process continues until a specified criterion is met. AdaBoost boasts several advantages, including high precision, adaptability to various data types, reduced susceptibility to overfitting, and ease of implementation. However, it also has notable limitations, such as sensitivity to noise and outliers, high computational demands, and unsuitability for complex tasks like speech and image recognition [107]. To address these issues, several extensions have been proposed, including soft margin AdaBoost [108], modest AdaBoost [109], SpatialBoost [110], and scalable variants like AdaBoost.PL and LogitBoost.PL [111].

On the other hand, RF, initially conceptualized by Ho [112] and later popularized by Breiman [113], is one of the main examples of independent ensembles. RF's simplicity and strong performance make it particularly effective, often achieving lower errors with minimal parameter tuning. The technique involves creating multiple decision trees using subsamples of the training data and aggregating their predictions. Research shows that RF is more

robust and stable compared to methods like support vector machines (SVM), extreme learning machines, and neural networks, especially for small training sets [114]. A related method, Extremely Randomized Trees (Extra Trees), was proposed by Geurts et al. [115], differing from RF by not employing bootstrap sampling. However, large forests are required to reduce bias and variance. Friedman [116] introduced Gradient Boosting Machines (GBM), which build models iteratively based on the gradient of the loss function from previous learners. While GBM can deliver high accuracy, its hyperparameters are sensitive and require careful tuning to avoid overfitting or underfitting. To improve scalability and reduce computational complexity, variants such as Extreme Gradient Boosting (XGBoost) [117] and LightGBM [118] were developed, offering significant efficiency gains. Recently, DL-based ensembles have gained attention. Researchers have explored ensembles using Long Short-Term Memory networks (LSTM) [119, 120, 121, 122, 123], Recurrent Neural Networks (RNN) [124, 125], and Gated Recurrent Units (GRU) [126]. Additionally, ensembles that combine multiple DL models [127, 128] have demonstrated promising results across various domains.

2.3 PV ENERGY FORECASTING, FINANCIAL SERIES, AND CRYPTOCURRENCY

This section reviews the literature on PV, financial, and cryptocurrency datasets, the three real-world applications implemented for the simulations.

Given the critical importance of reliability in PV energy management, many researchers have addressed this topic extensively. Wan et al. [129], Barbieri et al. [130], Das et al. [131], and Alcañiz et al. [41] presented extensive literature reviews on this topic. Munsif et al. [132] proposed the convolutional-transformer-based network (CT-NET) for power forecasting. Jailani et al. [17] implemented LSTM for solar energy forecasting. Artificial neural networks (ANN) [133, 134] and ANN with feature selection [135, 136] have also been applied. Sharma et al. [137] utilized a Levenberg Marquardt artificial neural network (LM-ANN) employing the gradient descent (GD) optimization technique for solar energy forecasting. Key advantages of ANNs include: (i) self-adaptive ability; (ii) fault-tolerance; (iii) robustness; and (iv) strong inference capabilities [138]. However, due to their architecture, ANNs can exhibit high complexity. Support vector machines (SVM) have also been explored for PV energy forecasting [139], praised for their non-linear modeling capacity and independence from prior knowledge [140]. Numerous hybrid approaches using SVM have emerged, such as PSO-SVM [141], GASVM [142], and ACO-SVM [143]. Fuzzy models have been widely applied to the power prediction of PV systems [144, 145, 146, 147, 148]. A notable advantage of fuzzy models is their ability to handle uncertainties in real-world data [149]. Within the domain of fuzzy models, ANFIS typically yields errors substantially lower than those of standalone fuzzy models. However, its computational complexity also increases, especially compared to SVM [131, 150].

In financial forecasting, DL models are increasingly prevalent, despite their black-box nature [151, 152, 153]. LSTM networks have proven particularly effective in addressing long-term dependencies in financial data [154, 155]. Ballings et al. [156] compared several classical models and reported the superiority of RF and SVM. Ensemble models have been widely applied to financial time series. Zhu and He [157] showed the superior performance of an XGBoost ensemble over ARIMA and LSTM in predicting the stock market price of Amazon. Xu et al. [158] implemented the ensemble learning-support vector regression & random forest (E-SVR&RF) to predict the stock closing price of four stock markets in China. In another study, Zhou et al. [159] implemented an LSTM and CNN ensemble, while Sonkavde et al. [45] used an ensemble of RF, XGBoost, and LSTM for TAINIWALCHM and AGROPHOS stock price forecasting. Among fuzzy applications in finance, research indicates that neuro-fuzzy models tend to perform better due to their ability to handle complex systems [160, 161].

Finally, regarding cryptocurrency forecasting, Greaves and Au [162] highlighted that ANNs outperformed linear regression, logistic regression, and SVM. Derbentsev et al. [163] showed the superiority of the binary autoregressive tree (BART) model over ARIMA and the autoregressive fractional integrated moving average (ARFIMA) model to predict Bitcoin, Ethereum, and Ripple prices. Aanandhi et al. [164] evaluated the performance of ARIMA and LSTM in the short-term prediction of Bitcoin prices and concluded that while ARIMA can capture the general trend, LSTM models outperform it in predicting both the direction and magnitude of price movements [165]. Other works have successfully applied LSTM for this task, including classical LSTM [166, 167, 168], LSTM combined with autoregressive characteristics [169], and LSTM combined with a gated recurrent unit (GRU) model [170]. RL is also employed to define investment strategies in cryptocurrencies. For instance, Jiang and Liang [171] implemented an approach combining RL and CNN, while Lee et al. [172] performed a similar study using inverse RL. Other research has explored deep RL for this purpose [173, 174]. Lucarelli and Borrotti [175] observed that double deep Q-learning networks trained with Sharpe ratio rewards for automated cryptocurrency trading outperformed traditional models in Bitcoin trading. The literature also indicates that RNNs are well-suited for this task [176, 177].

2.4 SUMMARY

This chapter provided a comprehensive literature review on the methods and concepts utilized in this work. The review began with an analysis of Mamdani and TSK models, emphasizing their respective advantages and limitations where applicable. It then covered GAs and ensemble models, which are central to the optimization and modeling strategies implemented in this work. Finally, the chapter reviewed the use of these techniques in the context of PV energy, finance, and cryptocurrency datasets. The

next chapter presents the theoretical background for the techniques applied in this study.

3 BACKGROUND KNOWLEDGE

This chapter provides a comprehensive overview of the fundamental concepts and formal methodologies of the machine learning techniques essential for this work, including Mamdani and TSK FIS, as well as GA and ensemble feature selection techniques.

3.1 FUZZY INFERENCE SYSTEMS

This section provides a detailed examination of FIS approaches, beginning with the Mamdani model and concluding with the TSK model.

3.1.1 MAMDANI FIS

The Mamdani fuzzy system is a rule-based model consisting of four main components: a fuzzifier, a rule base, an inference engine, and a defuzzifier, as illustrated in Figure 1. The following subsections detail each of these components, some of which are also shared with TSK models.

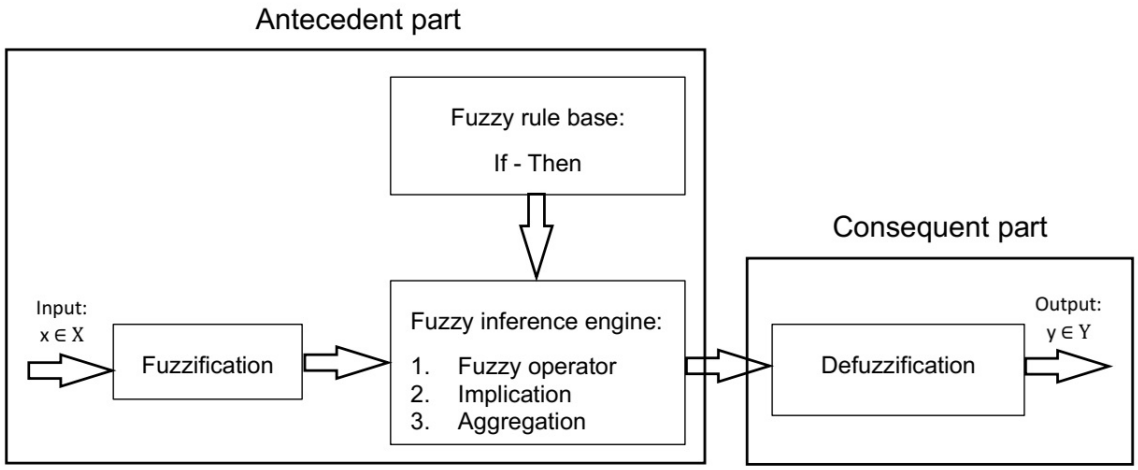


Figure 1 – Flowchart for the Mamdani FIS - Adapted from: [178]

3.1.1.1 FUZZIFICATION

Fuzzification involves mapping an input variable (x) to fuzzy sets (linguistic variables). Formally, given a universe of discourse X, a fuzzy set $\mathcal{A}$ is defined as follows:

Definition 1 *A fuzzy set $\mathcal{A}$ on a universe of discourse X is characterized by a membership function, $\mu_A : X \to [0, 1]$. Here, μ_A represents the degree of membership of an element $x \in X$ in the fuzzy set $\mathcal{A}$, where 0 denotes no membership and 1 denotes full membership. The formal representation of a fuzzy set is given by (3.1) [179].*

$$A = \{(x, \mu_A(x)) | x \in X\} \tag{3.1}$$

Common membership functions include Gaussian, triangular, and trapezoidal [180, 181, 182]. This study employs Gaussian membership functions, which is expressed as:

$$\mu_{\mathcal{A}}(x) = \exp\left[-\frac{1}{2}\left(\frac{x-v}{\sigma}\right)^2\right] \tag{3.2}$$

where v represents the mean of the fuzzy set and σ denotes the standard deviation. Gaussian membership functions are advantageous because they are smooth, non-zero across their entire domain, and effectively represent linguistic variables with precision and clarity [183]. However, future research may explore alternative membership function types for the proposed models.

3.1.1.2 FUZZY INFERENCE ENGINE

The fuzzy inference engine consists of three main components: the fuzzy operator, implication, and aggregation, as described below:

- **Fuzzy operator:** This step takes the results of the membership functions from the fuzzification process to compute the firing strength of a rule (w_i) [184]. Formally, $w_i(\mathbf{x}) = f(\mu_{\mathcal{A}_1}(x_1), \mu_{\mathcal{A}_2}(x_2), \ldots, \mu_{\mathcal{A}_p}(x_p))$, where $\mathbf{x} = [x_1, x_2, \ldots, x_p]^T$ is the input vector, p is the number of input attributes, and $\mu_{\mathcal{A}_j}(x_j)$ is the membership degree of the crisp input x_j in the fuzzy set $\mathcal{A}_j$, for the jth attribute. To combine the membership values from a rule's antecedent, intersection and union techniques are applied, commonly referred to as t-norm (triangular norm) and t-conorm (triangular conorm), respectively. Commonly used operators include minimum and product for intersection, and maximum for union, as given by (3.3), (3.4), and (3.5), respectively [185, 186].

$$\mu_{\mathcal{A}}(\mathbf{x}) = \min_{\forall j \in [1,2,\ldots,p]} \mu_{\mathcal{A}_j}(x_j) \tag{3.3}$$

$$\mu_{\mathcal{A}}(\mathbf{x}) = \prod_{j=1}^{p} \mu_{\mathcal{A}_j}(x_j) \tag{3.4}$$

$$\mu_{\mathcal{A}}(\mathbf{x}) = \max_{\forall j \in [1,2,\ldots,p]} \mu_{\mathcal{A}_j}(x_j) \tag{3.5}$$

- **Implication:** The membership value obtained in the previous step shapes the consequent membership function. Figure 2 depicts an example of implication. The

figure shows a rule that uses two fuzzy sets to model an input vector with two attributes and one fuzzy set to model the output. In the given example, the parameters of the Gaussian fuzzy sets, represented by the mean (v) and the standard deviation (σ), are $v = 2$ and $\sigma = 0.4$, for feature 1, $v = 7$ and $\sigma = 0.4$ for feature 2, and $v = 3$ and $\sigma = 0.4$ for the output. Given an input vector $\mathbf{x} = [2.3, 6.2]$ and using (3.2), the membership function values of 0.75 and 0.33 are obtained for the first and second attributes, respectively. Considering the minimum value as the fuzzy operator, the fuzzy set for the output is obtained, as shown in the figure. Fuzzy sets can be referred to as linguistic variables because they can be described as so, i.e., fuzzy set for the attribute 1 can represent the linguistic variable small, fuzzy set for attribute 2 as high, and fuzzy set for the target value as medium, for example.

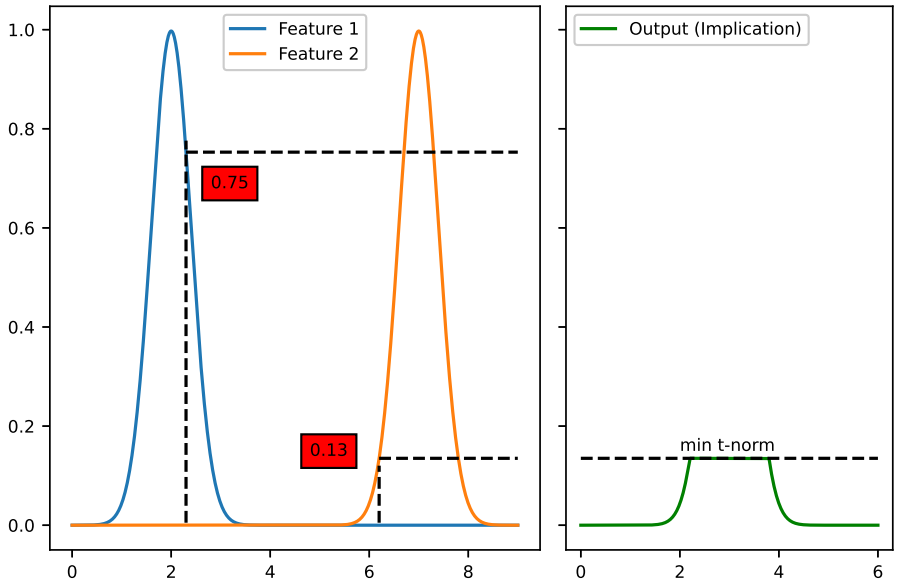


Figure 2 – Illustration of the implication step

- **Aggregation:** The final step in the fuzzy inference engine aggregates the fuzzy sets resulting from the implication process for all rules. Figure 3 shows an example of aggregation. A commonly used aggregation technique is the maximum operator, as given by (3.6), though other methods (e.g., summation and probabilistic aggregation) are also applicable [187, 188].

$$\mu_{aggregated}(\mathbf{x}) = \max\left\{\mu_{output_1}(\mathbf{x}), \mu_{output_2}(\mathbf{x}), \ldots, \mu_{output_R}(\mathbf{x})\right\} \tag{3.6}$$

where $\mu_{aggregated}(\mathbf{x})$ is the aggregated fuzzy set of an input x, max is the maximum operator, $\mu_{output_i}(\mathbf{x})$ is the output's fuzzy set resulting from the implication process, for $i = 1, 2, \ldots, R$ and R is the total number of rules.

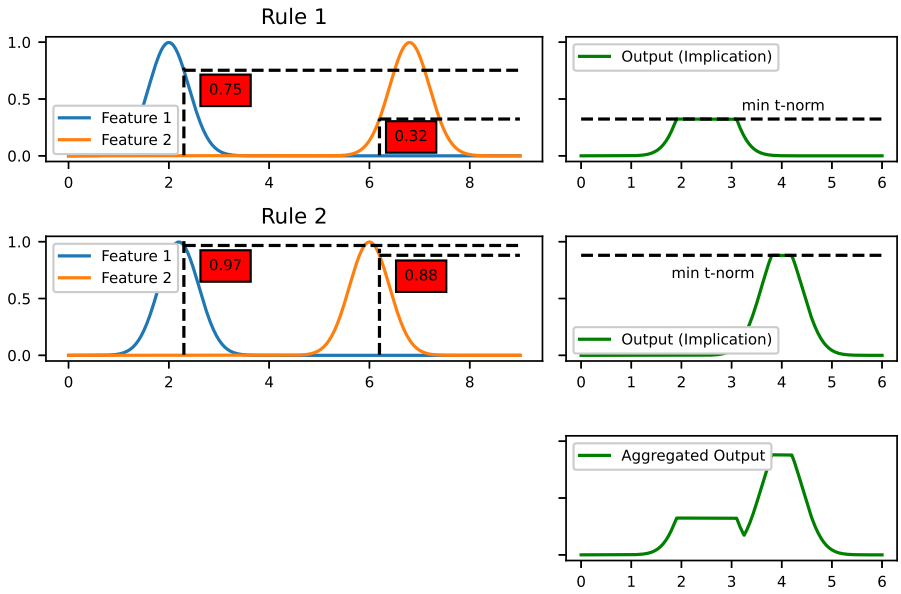


Figure 3 – Illustration of the aggregation step

3.1.1.3 DEFUZZIFICATION

Defuzzification converts the aggregated fuzzy set into a single crisp value [189]. Various defuzzification approaches exist, as discussed by Zimmermann [190] and Lee [191]. Chakraverty et al. [192] provide an overview of notable methods, including max-membership, centroid, weighted-average, and mean–max.

3.1.2 TAKAGI-SUGENO-KANG

The TSK model employs fuzzy rules and polynomial functions to model a nonlinear system through a collection of linear subsystems. TSK rules are expressed as follows:

$$R_i: \quad \textbf{IF} \quad \underbrace{\mathbf{x} \quad \text{is} \quad \mathcal{A}_i}_{\text{Antecedent}} \quad \textbf{THEN} \quad \underbrace{y_i = f(\mathbf{x}, \theta_i)}_{\text{Consequent}} \tag{3.7}$$

where $\mathbf{x} = [x_1, \ldots, x_p]^T \in I\!R^p$ is the input, p is the number of attributes in the input vector, $\mathcal{A}_i$ is the multidimensional antecedent (the combined fuzzy set region) for the ith rule, and y_i is the output of the ith rule calculated as a function of the input and the consequent parameters.

According to (3.7), TSK rules are divided into antecedent and consequent components. The antecedent component encompasses fuzzification and the calculation of the firing strength, while the consequent component employs polynomial functions to compute each rule's output. The TSK process is depicted in Figure 4.

The fuzzification process in TSK is identical to that in Mamdani systems. However, in the fuzzy inference stage, TSK only utilizes the fuzzy operator to calculate the firing strength. Another distinction between Mamdani and TSK models lies in how the consequent

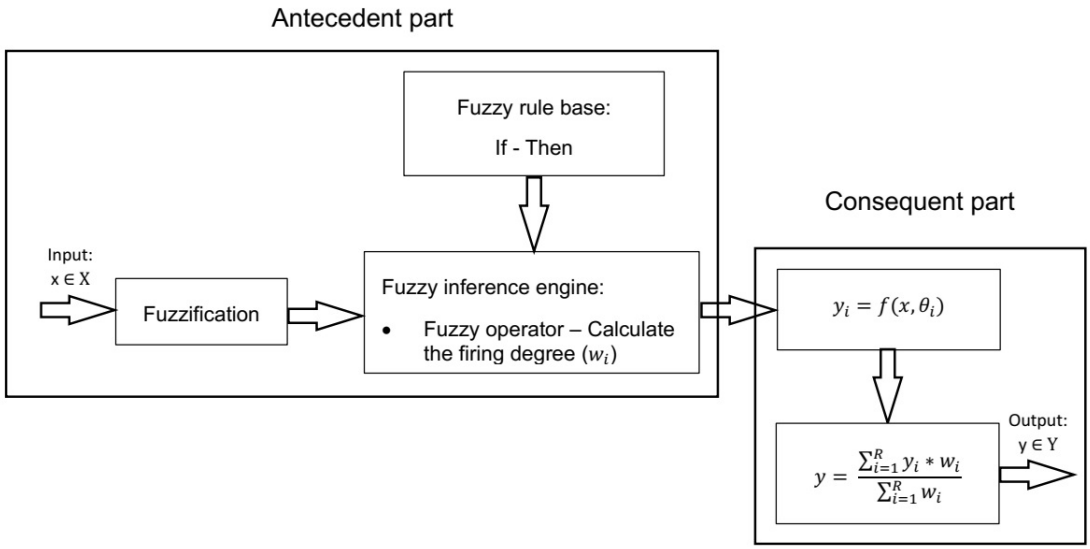


Figure 4 – Flowchart presenting the inference mechanism of TSK

part is handled. TSK computes the output of each rule using polynomial functions, as expressed by:

$$\hat{y}_i = \sum_{j=1}^{p+1} xe_j \theta_{i,j} = (\mathbf{xe})^T \theta_i \tag{3.8}$$

where $\hat{y}_i$ is the estimated output for the ith rule, for $i = 1, 2, \ldots, R$, $\mathbf{xe} = [1, \mathbf{x}^T]^T$ is a vector, and θ_i are the consequent parameters for ith rule.

Finally, the last step is to compute the output of the model as a weighted average of each rule's output, as follows:

$$\hat{y} = \sum_{i=1}^{R} w_i \hat{y}_i \tag{3.9}$$

where w_i is the normalized firing strength, is given by:

$$w_i = \frac{\tau_i}{\sum_{i=1}^{R} \tau_i} \tag{3.10}$$

TSK mechanisms are summarized in the following steps:

- **Fuzzification:** This step maps the input attributes to fuzzy sets using membership functions.
- **Firing Strength Calculation:** The firing strength of each rule is determined by applying a fuzzy operator (e.g., product or minimum) to the membership degrees of the input attributes.
- **Output Calculation:** Each rule's output is computed using a polynomial function of the input attributes and the corresponding consequent parameters.

- **Aggregation:** The final crisp output is obtained as a weighted average of the rules' outputs, where the weights are the normalized firing strengths.

This structure enables the TSK model to effectively handle nonlinear systems while maintaining interpretability. The use of polynomial functions in the consequent part enhances the model's precision and adaptability, making it suitable for a wide range of applications, including control systems, forecasting, and decision support systems.

3.2 FEATURE SELECTION

This section presents the theoretical background of the two feature selection techniques employed in this work.

3.2.1 GENETIC ALGORITHM

GA is one of the most widely recognized evolutionary algorithms [91]. Since its inception, GA has undergone significant development, with numerous variations proposed in the literature. This section focuses on the fundamental concepts of the classical GA. The GA process comprises four main components: initialization, selection, reproduction, and termination, as described bellow [193]:

- **Initialization:** This step involves randomly generating an initial population of potential solutions. These solutions, often referred to as individuals, constitute the starting point of the algorithm.
- **Selection:** In this step, the algorithm selects the most promising individuals based on a predefined fitness function. Various selection methods exist in the literature, including roulette wheel selection, linear ranking selection, exponential ranking selection, and tournament selection [194]. The choice of selection method significantly influences the performance of the algorithm.
- **Reproduction:** The selected individuals, termed parents, undergo genetic operations such as crossover (recombination) and mutation to produce offspring. This step is crucial for exploring the solution space and introducing new genetic material. These offspring form the population for the next generation, often replacing less fit individuals from the parent population. This process repeats iteratively until a termination criterion is met.
- **Termination:** The algorithm terminates when a specified criterion is satisfied. Common termination criteria include a fixed number of generations, a computational budget, or the achievement of a desired fitness level. For example, Wong et al. [195] proposed a novel termination criterion that reduces computational costs while maintaining solution quality.

Figure 5 presents a flowchart of the GA and its main components.

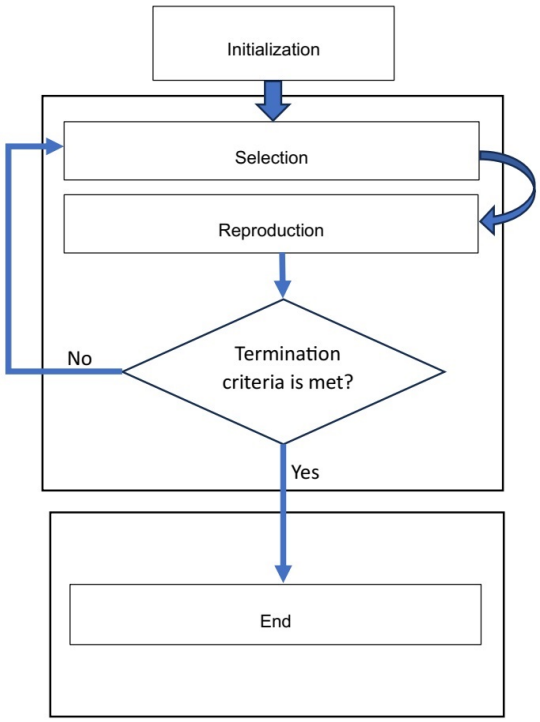


Figure 5 – GA flowchart

In addition to these components, GA requires defining several key hyperparameters, which significantly impact its performance:

- **Number of Generations:** Determines the maximum number of iterations the algorithm will perform.
- **Number of Parents Mating:** Specifies how many individuals from the current generation are selected to form a mating pool for reproduction.
- **Population Size:** Refers to the total number of individuals in the population at each iteration.

Proper tuning of these hyperparameters is essential for balancing exploration and exploitation, which directly affects the algorithm's efficiency and the quality of the final solution.

3.2.2 ENSEMBLE MODEL

Ensemble models are founded upon two fundamental pillars: the generation of a set of diverse learners and the combination of their outputs into a single, unified prediction. The formal definition of an ensemble model is presented below:

Definition 2 *Given a dataset and an input vector, an ensemble learning model ψ applies an aggregation function G to combine the outputs of Z individual learners to produce a single final output $\hat{y}$, given by:*

$$\hat{y} = G(f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_Z(\mathbf{x})) \tag{3.11}$$

The effectiveness of ensemble models is founded on two key principles: diversity and predictive performance.

- **Diversity:** The core idea of diversity is to produce individual learners with uncorrelated errors. By ensuring diversity among learners, ensemble methods reduce the risk of overfitting and improve generalization [196]. Techniques for inducing diversity include input manipulation, partitioning, output manipulation, and hybridization of ensemble methods [105].
- **Predictive Performance:** Each individual learner must achieve a predictive accuracy that is significantly better than random guessing. Ensuring high performance for individual models strengthens the overall reliability of the ensemble [197].

Once the diverse learners are generated, the next step is to aggregate their outputs. Zhou [198] discusses commonly used techniques for this purpose, including averaging, voting, and learning-based combination methods. Each aggregation approach is suited to specific problem types and datasets, providing flexibility in applying ensemble methods to various domains.

3.3 SUMMARY

This chapter has outlined the fundamental concepts of the machine learning techniques essential for this work. Initially, the two primary FIS, Mamdani and TSK, were discussed. The key distinction between these systems lies in the consequent component: while Mamdani models the output using fuzzy sets, TSK employs polynomial functions. Subsequently, the GA approach was reviewed. The main components of the GA include initialization, selection, reproduction, and termination. The GA is particularly suited for tasks where the size of the search space renders traditional optimization techniques impractical. Finally, the concept of ensemble models was introduced. Ensemble models generally offer improved performance with a reduced risk of overfitting. The next chapter introduces the proposed models.

4 NEW FUZZY INFERENCE SYSTEMS (NFISiS)

This chapter provides a comprehensive explanation of the proposed NFISiS, which consists of a collection of six new data-driven FIS models. First, the new data-driven mechanism for designing NMR and NTSK rules is introduced. Subsequently, the chapter explores the implementation of GA-based and ensemble-based approaches for feature selection, integrated with the new rule design techniques.

4.1 PROPOSED FIS DESIGN

The main steps to design the fuzzy rules for the proposed models are summarized as follows:

Step 1 - Define the Number of Rules: The user specifies the desired number of rules.

Step 2 - Define Intervals for Target Values: The model creates intervals for the target values based on specified criteria.

Step 3 - Define the Rules: Each interval from the previous step corresponds to a single rule.

Step 4 - Assign Samples to Rules: Each sample is assigned to a rule based on its corresponding output value.

Step 5 - Compute the Fuzzy Sets: The parameters of the fuzzy sets are computed using the assigned samples.

The primary objective is to generate rules based on the target value. In contrast to conventional TSK methods, which typically first define antecedent fuzzy sets and subsequently establish the rule base, NMR and NTSK reverses this procedure.

4.1.1 NEW MAMDANI REGRESSOR (NMR)

This subsection provides a detailed overview of NMR. It starts by presenting the fundamental concepts, followed by a description of the training and testing phases. Finally, a simple example is provided to illustrate the process.

4.1.1.1 AN OVERVIEW OF NMR

The training process of NMR rules is summarized in the five steps presented below:

Step 1 - Define the Number of Rules: The user specifies the desired number of rules ($R_{\max}$).

Step 2 - Partition the Output Space: The model creates equally spaced intervals across the range of the target variable.

Step 3 - Establish Consequent Rule Definitions: The intervals determine to which rule each sample is assigned by comparing the target value with each interval region.

Step 4 - Assign Samples to Rules: Each data sample is assigned to the rule corresponding to the interval of its target value.

Step 5 - Compute Fuzzy Set Parameters: The parameters for the antecedent and consequent membership functions are calculated from the partitioned data.

Figure 6 illustrates the steps involved in the training process of the NMR model.

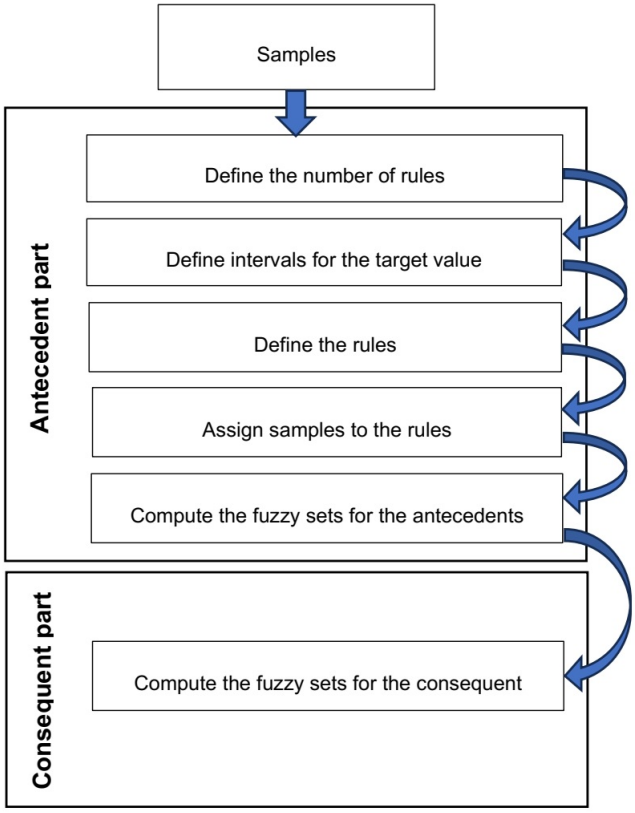


Figure 6 – Flowchart of the learning phase for NMR

4.1.1.2 DEFINING NMR RULES - TRAINING PHASE

NMR training phase begins with the user specifying the number of rules as $R_{\max}$, which is a key hyperparameter controlling model granularity. The algorithm then computes the size of each output interval (IS) based on the amplitude of the target variable's range as:

$$IS = \frac{\bar{y} - \underline{y}}{R_{\max}} \tag{4.1}$$

where $\bar{y}$ and $\underline{y}$ denote the maximum and minimum values of the target variable in the training data, respectively. Subsequently, IS is used to compute the interval for the

consequent of each rule, as follows:

$$Range_i = [\underline{y} + (i-1)IS, \underline{y} + (i)IS] \tag{4.2}$$

where $Range_i$ represents the interval of the consequent part of the ith rule and $i = 1, 2, \ldots, R_{\max}$.

Once the consequent intervals are defined, each training sample is assigned to its corresponding rule by comparing its target value with these intervals. The index of the rule assigned to the kth sample is determined by:

$$i_{S^k} = \begin{cases} \lfloor \frac{y^k - \underline{y}}{IS} \rfloor + 1, & \text{if } \; y^k < \bar{y} \\ R_{\max}, & \text{otherwise} \end{cases} \tag{4.3}$$

where i_{S^k} denotes the index of the rule to which the kth sample is assigned, and $\lfloor \rfloor$ represents the floor function.

The final step in defining the rules involves computing the fuzzy sets for the antecedent and consequent components. This work implements Gaussian fuzzy sets (see Equation (3.2)), which require only two parameters: the mean and the standard deviation. The complete training procedure is summarized in the pseudo-code presented in Algorithm 1.

Algorithm 1: NMR training procedure

Input: $\mathbf{x}, y$
Initialization: $R_{\max}$
Compute the interval size: Equation (4.1)
for $i = 1, 2, \ldots, R_{\max}$ **do**
    Compute the interval bounds: Equation (4.2)
for $k = 1, 2, \ldots, N-1$ **do**
    Determine rule index i_{S^k} for sample k: Equation (4.3)
Assign samples to rules based on i_{S^k}
for $i = 1, 2, \ldots, R_{\max}$ **do**
    Compute the parameters of the fuzzy sets

4.1.1.3 THE INFERENCE PROCESS OF NMR - TEST PHASE

During the inference phase, the NMR model first computes the normalized firing strength (w_i) for each rule as:

$$w_i = \frac{\prod_{j=1}^{p} \exp\left[-\frac{1}{2}\frac{\left(x_j^k - v_{i,j}\right)^2}{\sigma_{i,j}^2}\right]}{\sum_{i=1}^{R_{\max}} \left(\prod_{j=1}^{p} \exp\left[-\frac{1}{2}\frac{\left(x_j^k - v_{i,j}\right)^2}{\sigma_{i,j}^2}\right]\right)} \tag{4.4}$$

where $v_i = [v_{i,1}, \ldots, v_{i,p}]^T$ with $v_{i,j}$ consisting of the mean of the jth attribute for the ith rule, p is the dimension of the inputs (number of attributes for each sample), x_j^k is the kth input vector for the jth attribute, and $\sigma_i = [\sigma_{i,1}, \ldots, \sigma_{i,p}]^T$ with $\sigma_{i,j}$ consisting of the standard deviation of the jth attribute for the ith rule. It is simple to visualize that w_i is obtained from Equations (3.4) and (3.10).

Finally, defuzzification is performed to compute the model's output as follows:

$$\hat{y}^k = \sum_{i=1}^{R_{\max}} v_{i,output} \cdot w_i \tag{4.5}$$

where $v_{i,output}$ is mean of the consequent fuzzy set for the ith rule.

The complete inference procedure for the NMR model is summarized in the pseudo-code presented in Algorithm 2.

Algorithm 2: NMR inference procedure

Input: **x**
Output: $\hat{y}$
for $k = 1, 2, \ldots$ **do**
    Compute the normalized firing strength: Equation (4.4)
    Compute the output: Equations (4.5)

4.1.1.4 SIMPLE EXAMPLE

To illustrate the mechanism of the NMR model, a simple example using a synthetic dataset is provided in Table 1. The table comprises three columns: the first indicates the sample index; the second represents the number of auto insurance claims; and the third displays the total amount paid (in thousands). The objective is to use the number of claims as an input feature to predict the total amount paid.

Table 1 – Synthetic auto insurance dataset

Samples	Claims	Total amount
1	108	392.5
2	19	46.2
3	13	15.7
4	124	422.2
5	40	119.4

The first step is to define the number of rules. Subsequently, IS and $Range_i$ are computed using Equations (4.1) and (4.2), respectively. For example, with $R_{\max} = 2$, the interval size is calculated as $IS = (422.2 - 15.7)/2 = 203.25$. Accordingly, the range for the first rule is $Range_1 = [15.7 + 0 \cdot 203.25, 15.7 + 1 \cdot 203.25] = [15.7, 218.95]$, and for the second rule, it is $Range_2 = [15.7 + 1 \cdot 203.25, 15.7 + 2 \cdot 203.25] = [218.95, 422.2]$.

Consequently, the first rule contains samples whose output falls within $[15.7, 218.95]$, while the second rule contains samples falling within $[218.95, 422.2]$. Then, using Equation (4.3), samples 2, 3, and 5 are assigned to the first rule, and samples 1 and 4 are assigned to the second rule. Finally, the mean and standard deviation are computed using the clustered samples, resulting in the parameters presented in Table 2. With the rules defined, the model can perform inference on unseen data.

Table 2 – Parameters of the generated NMR rules

Rule	Claims	Total amount
1	24 ± 14.18	60.43 ± 53.29
2	116 ± 11.31	407.35 ± 21.00

The underlying principles of NMR can be extended to the New Mamdani Classifier (NMC). In this adaptation, instead of the user defining the number of rules, the number of rules is automatically set equal to the number of classes, with each class corresponding to a distinct rule. Although this work does not explore the NMC, the `nfisis` Python library includes an implementation of this model.

4.1.2 NEW TAKAGI-SUGENO-KANG (NTSK)

This subsection provides a detailed overview of NTSK. It starts by presenting the fundamental concepts, followed by a description of the training and testing phases. Finally, a simple example is provided to illustrate the process.

4.1.2.1 AN OVERVIEW OF NTSK

The NTSK model follows a process similar to that of NMR for the antecedent part, with slight variations. The primary distinction lies in the consequent part, where the model employs polynomial functions. The steps for defining the NTSK rules are as follows:

- **Step 1 - Specify the Number of Rules:** The user specifies the desired number of rules.
- **Step 2 - Compute Target Value Variations:** The first-order difference is calculated for consecutive target values.
- **Step 3 - Define Variation Intervals:** Equally spaced intervals are created based on the range of target value variations and $R_{\max}$.
- **Step 4 - Assign Samples to Rules:** Each data sample is assigned to a rule by comparing the sample's value variation with the intervals of each rule.

- **Step 5 - Compute Antecedent Parameters:** The parameters for the antecedent fuzzy sets are determined (e.g., mean and standard deviation for Gaussian membership functions).
- **Step 6 - Compute Consequent Parameters:** The consequent parameters (θ) are estimated using an adaptive filtering method.

The flowchart for the training phase of NTSK is presented in Figure 7. For the testing phase, the process resembles the classical TSK model (see Figure 4).

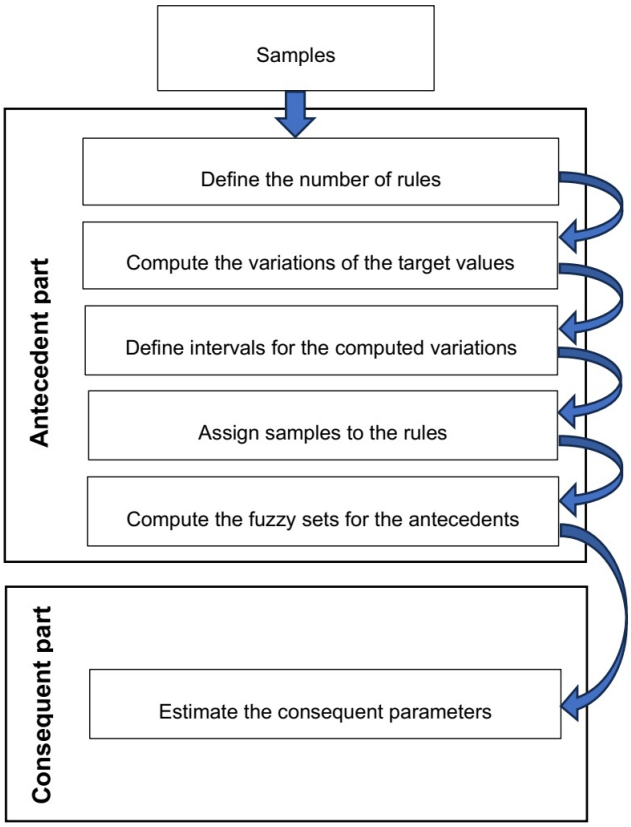


Figure 7 – Flowchart depicting NTSK training procedure

4.1.2.2 DEFINING NTSK RULES - TRAINING PHASE

The training of the NTSK model begins with the user specifying the maximum number of rules ($R_{\max}$). This hyperparameter provides direct control over the number of rules, a notable advantage over many data-driven FIS approaches. Next, the model analyzes the dynamics of the target variable by computing its first-order difference. This variation captures the local rate of change between consecutive samples and is calculated as:

$$tan(y^k) = \frac{y^{k+1} - y^k}{(k+1) - k} = y^{k+1} - y^k = \Delta y^k \tag{4.6}$$

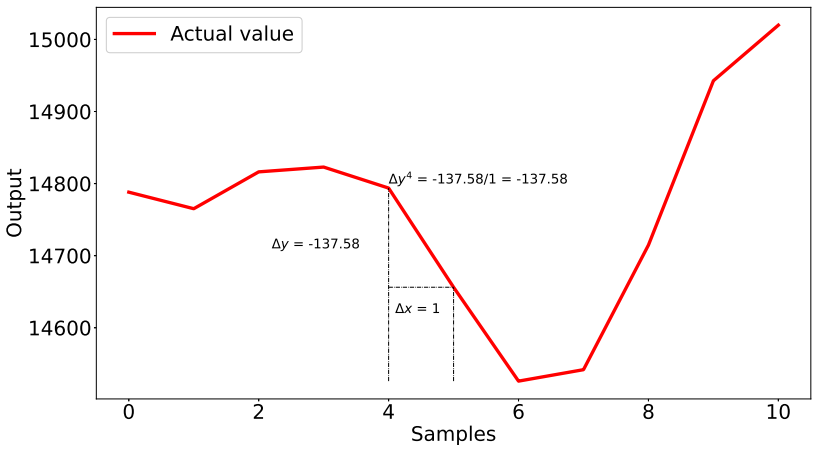


Figure 8 – Illustration of the calculation of Δy^k

where $tan(y^k)$ denotes the tangent of the kth sample, with $k = 1, \ldots, N-1$, and N representing the total number of samples. In the context of time series, the tangent is equivalent to the first-order difference, as $tan(y^k) = y^{k+1} - y^k = \Delta y^k$. For convenience, this work uses the terms first-order difference and tangent interchangeably, both denoted by Δy^k. The calculation is illustrated in Figure 8.

The primary motivation for computing the tangent is to capture local trends and rates of change in the target variable. After calculating this variation (Δy^k) for all applicable samples, the model partitions the range of these values into distinct intervals. This partitioning groups samples exhibiting similar variation. Each interval is defined by:

$$Range_i = [\underline{m}_\Delta + (i-1)IS, \underline{m}_\Delta + (i)IS] \tag{4.7}$$

where $Range_i$ represents the interval limits for the tangent of the ith rule, for $i = 1, 2, \ldots, R_{\max}$. Here, $R_{\max}$ denotes the specified number of rules, $\bar{m}_\Delta = \max(\Delta y^k)$ is the maximum tangent value, and $\underline{m}_\Delta = \min(\Delta y^k)$ is the minimum tangent value. Finally, the interval size (IS) is computed as:

$$IS = \frac{\bar{m}_\Delta - \underline{m}_\Delta}{R_{\max}} \tag{4.8}$$

Subsequently, the model assigns each sample to its corresponding rule according to the criterion:

$$i_{S^k} = \begin{cases} \lfloor \frac{\Delta y^k - \underline{m}_\Delta}{IS} \rfloor + 1, & \text{if } \Delta y^k < \bar{m}_\Delta \\ R_{\max}, & \text{otherwise} \end{cases} \tag{4.9}$$

where i_{S^k} is the rule that the kth sample will be included.

Next, the model computes the parameters of the fuzzy sets. Since this work utilizes Gaussian fuzzy sets, the mean and standard deviation are calculated as shown in Equations (4.10) and (4.11).

$$v_{i,j} = \bar{x}^l_{i,j} \tag{4.10}$$

where $v_{i,j}$ denotes the mean of the samples in the ith rule for the jth attribute, and $\bar{x}^l_{i,j}$ represents the lth sample assigned to the ith rule for the jth attribute, for $l = 1, 2, \ldots, z$, where z is the number of samples assigned to the ith rule.

$$\sigma_{i,j} = std(x^l_{i,j}) \tag{4.11}$$

where $\sigma_{i,j}$ denotes the standard deviation of the samples assigned to the ith rule for the jth attribute. It is calculated over the samples $x^l_{i,j}$ for $l = 1, 2, \ldots, z$, where z is the number of samples assigned to the ith rule.

Finally, the proposed model computes the consequent parameters. The RLS is an adaptive filter for parameter estimation that minimizes the total weighted squared error between the target value and the model's output. A significant advantage of the RLS algorithm over other methods, such as least mean squares (LMS), is the capacity to achieve good accuracy with faster convergence speed [199, 200, 201].

This work evaluates two RLS methods. The first one is the conventional one, discussed in [202], which is expressed as:

$$\begin{cases} K = \frac{P_i^{k-1}\mathbf{xe}^k}{\lambda + (\mathbf{xe}^k)^T P_i^{k-1}\mathbf{xe}^k} \\ P_i^k = \frac{1}{\lambda}\left[P_i^{k-1} - K(\mathbf{xe}^k)^T P_i^{k-1}\right] \\ \theta_i^k = \theta_i^{k-1} + P_i^k \mathbf{xe}^k \left(y^k - (\mathbf{xe}^k)^T \theta_i^{k-1}\right) \end{cases} \tag{4.12}$$

where P_i^{k-1} denotes the previously computed covariance matrix for the ith rule, P_i^k is the new covariance matrix for the ith rule, and K is an intermediate variable used to simplify the expression for P_i^k. Additionally, $\mathbf{xe}^k = [1, (\mathbf{x}^k)^T]^T$ is the extended input vector, θ_i^{k-1} represents the previously estimated parameters for the ith rule, θ_i^k is the current estimation of the consequent parameters, y^k is the current target value, and $\lambda \in [0, 1]$ is a parameter known as the forgetting factor. When $\lambda = 1$, all previous errors are given equal weight; conversely, for λ close to zero, past errors are less influential in updating the parameters.

A second RLS approach implemented is the wRLS, given in Equation (4.13), implemented in many rule-based eFSs [68, 69, 70, 72, 74]. In the wRLS, weights (w) are used to update the consequent parameters, in which the weights are the normalized firing strength. The firing strength will be computed for each rule when a new input vector is available. Then, rules with higher a normalized firing strength will be more updated, and rules with lower one will be less updated.

$$\begin{cases} P_i^k = P_i^{k-1} - \frac{w_i P_i^{k-1} \mathbf{x}\mathbf{e}^k (\mathbf{x}\mathbf{e}^k)^T P_i^{k-1}}{1+w_i (\mathbf{x}\mathbf{e}^k)^T P_i^{k-1} \mathbf{x}\mathbf{e}^k} \\ \theta_i^k = \theta_i^{k-1} + P_i^k \mathbf{x}\mathbf{e}^k w_i \left(y^k - (\mathbf{x}\mathbf{e}^k)^T \theta_i^{k-1}\right) \end{cases} \tag{4.13}$$

where w_i represents the normalized firing strength of the ith rule. A significant advantage of wRLS is that it does not require any hyperparameters. In contrast, the standard RLS approach requires the specification of λ.

In summary, the RLS approach estimates a single, global set of parameters (θ) for all rules, meaning model performance is decoupled from the number of rules, which primarily affects interpretability. In contrast, the wRLS approach estimates a unique set of parameters for each rule, making model performance directly dependent on the chosen value of $R_{\max}$.

Algorithm 3 details the training process for the NTSK model. In the first loop, the model computes the first-order difference of the target value for all consecutive training samples. Subsequently, the interval size is calculated. In the second loop, the model assigns a rule index to each sample. For example, if the user sets $R_{\max} = 4$, the model generates four rule indices, numbered from one to four. Consequently, for a given sample, if $i_{S^k} = 1$, the sample is assigned to the first rule; if $i_{S^k} = 2$, it is assigned to the second rule, and so on. In the third loop, the model computes the parameters of the Gaussian fuzzy sets for each attribute and rule based on the assigned samples. Finally, the model computes the consequent parameters using either the RLS or wRLS technique.

Algorithm 3: NTSK training procedure

Input: $\mathbf{x}, y$
Output: $\hat{y}$
Initialization: $R_{\max}$
for $k = 1, 2, \ldots, N-1$ **do**
    Compute the variation between two consecutive target values: Equation (4.6)
Compute the interval size: Equation (4.8)
for $k = 1, 2, \ldots, N-1$ **do**
    Compute i_{S^k}: Equation (4.9)
Cluster the samples based on i_{S^k}
for $i = 1, 2, \ldots, R_{\max}$ **do**
    Compute the parameters of the Gaussian fuzzy sets for each attribute: Equations (4.10) and (4.11)
for $k = 1, 2, \ldots, N$ **do**
    Compute the parameters of the consequent: Equations (4.12) for RLS or (4.13) for wRLS

4.1.2.3 THE INFERENCE PROCESS OF NTSK - TEST PHASE

During the inference phase, the trained NTSK model predicts the output for an unseen input vector ($\mathbf{x}^k$). The estimated output ($\hat{y}^k$) is calculated as the weighted sum of the consequent of all fuzzy rules:

$$\hat{y}^k = \sum_{i=1}^{R_{\max}} w_i(\mathbf{x}\mathbf{e}^k)^T\theta_i \tag{4.14}$$

where $\hat{y}^k$ is the estimated output.

Algorithm 4 presents the pseudo-code for the NTSK inference phase.

Algorithm 4: NTSK inference procedure

Input: $\mathbf{x}$
Output: $\hat{y}$
1 **for** $k = 1, 2, \ldots$ **do**
2 Compute the normalized firing strength: Equations (4.4)
3 Compute the model's output: Equations (4.14)

The NTSK model is not applicable to classification problems, as a classification version of NTSK would effectively reduce to the NMC. A formal demonstration is omitted, as this relationship is readily apparent. Consider the case where the target value is discrete (i.e., classes), and each rule corresponds to a single value rather than a range. In this scenario, there is no need for a polynomial function to estimate the output, as it is explicitly defined. The only requirement is to identify the rule with which the new input is most compatible.

4.2 PROPOSED FIS WITH FEATURE SELECTION

This section details the implementation of the two feature selection approaches, GA and ensemble, within the NMR and NTSK models.

4.2.1 GA IMPLEMENTATION FOR FEATURE SELECTION

The GA is employed as a wrapper method for feature selection, leading to the development of two new models: GEN-NMR and GEN-NTSK. This approach aims to identify the most informative subset of input features. In this framework, each candidate feature subset is encoded as a chromosome, represented by a binary vector with a length equal to the total number of input attributes. A value of '1' in the vector signifies that the corresponding feature is selected, while a '0' signifies its exclusion. A constraint is enforced to ensure that each chromosome contains at least one selected feature, thereby preventing empty feature sets.

The fitness of each chromosome is evaluated based on the performance of the corresponding fuzzy model (NTSK or NMR), which is trained and validated using only the feature subset encoded by that chromosome. Through the iterative application of standard evolutionary operators—selection, crossover, and mutation—the GA refines a population of chromosomes to find the feature combination that maximizes fitness (i.e., minimizes prediction error). The final output of this process is the optimal feature subset. The key hyperparameters governing the GA's operation, along with their default values, are summarized in Table 3.

Table 3 – GA hyperparameters and default values

Hyperparameter	**Description**	**Default Value**
num_generations	The total number of evolutionary generations the algorithm will run	10
num_parents_mating	The number of elite individuals selected to produce offspring for the next generation	5
sol_per_pop	The total number of solutions (chromosomes) maintained in the population for each generation	10
error_metric	The fitness function used to evaluate solutions. The GA's objective is to find a feature set that minimizes this error. Options include RMSE, NRMSE, NDEI, MAE, and MAPE	'RMSE'

Algorithm 5 outlines the steps involved in applying GA for feature selection.

The implementation of GA within the NFISiS model effectively handles high-dimensional datasets, achieving improved performance and interpretability. However, there remains a risk of converging to a local minimum, as the model may identify a subset that yields the lowest error within a given context but does not necessarily achieve optimal performance overall. To address this, an ensemble fuzzy approach can be applied to obtain better performance on average.

Algorithm 5: GA-based feature selection procedure

Input: $\mathbf{x}, y$
Output: attributes_vector
Internally separate the samples into train and test.
Initialize the vector of binary elements.
Generate a subset for each vector.
Call the fuzzy model and evaluate the fitness of the initial population.
for $generation = 1, 2, \ldots, n\_generations$ **do**
    Select the best individuals to constitute the parents.
    Apply reproduction to generate a new population.
    Call the fuzzy model and evaluate the fitness of the generated population.
Select the individual with the best fitness value as the winner.

4.2.2 ENSEMBLE FUZZY

As the final contribution, this work develops three distinct fuzzy ensemble models to enhance predictive accuracy and robustness: R-NTSK, R-NMR, and RF-NTSK. The efficacy of ensemble fuzzy methods is founded on two key principles: the diversity of the base learners and their individual predictive performance.

The R-NTSK and R-NMR models are homogeneous ensembles based on the Random Subspace method. For each ensemble, a total of $n\_estimators$ base models (NTSK or NMR) are trained. Each base model is trained on a different, randomly selected subset of the available input features. The final prediction of the ensemble is the average of the outputs from all $n\_estimators$ individual models.

The RF-NTSK model is a heterogeneous meta-ensemble that combines the predictions of two distinct models: an RF and the previously described R-NTSK ensemble. The final output is a weighted average of the predictions from these two models, where the weights are inversely proportional to each model's error on the training data. This allows the meta-ensemble to favor the more accurate predictor. This aggregation is formulated as follows:

$$\hat{y}^k = \hat{y}^k_{RF} \frac{\epsilon_{R-NTSK}}{\epsilon_{RF} + \epsilon_{R-NTSK}} + \hat{y}^k_{R-NTSK} \frac{\epsilon_{RF}}{\epsilon_{RF} + \epsilon_{R-NTSK}} \tag{4.15}$$

where ϵ_{RF} and ϵ_{R-NTSK} denote the estimated errors of the RF and R-NTSK models during the training phase, respectively, and $\hat{y}^k_{RF}$ and $\hat{y}^k_{R-NTSK}$ represent the outputs of the RF and R-NTSK models, respectively. As shown, the final prediction is a weighted average of the outputs from both models, where the weights are inversely proportional to the errors.

One of the key advantages of ensemble fuzzy models is their ability to mitigate overfitting by leveraging multiple estimators. While a single estimator is often susceptible to overfitting, the inclusion of multiple estimators introduces diversity, thereby reducing this risk. Consequently, ensemble fuzzy models exhibit improved generalization capability.

The procedure for constructing the random subspace ensembles is outlined in Algorithm 6.

Algorithm 6: Ensemble fuzzy training procedure

Input: $\mathbf{x}, y$
Output: $\hat{y}$
for $i = 1, 2, \ldots, n\_estimators$ **do**
  for $j = 1, 2, \ldots, n\_trials$ **do**
    Generate a subset of the original dataset.
    Separate 75% of the subset for train and the rest for test.
    Evaluate the fitness for each subset.
  Select the best model to compose the list of estimators.

4.3 SUMMARY

This chapter introduced the proposed NFISiS framework. The discussion commenced by presenting new data-driven algorithms for designing Mamdani and TSK rules, designated as NMR and NTSK, respectively. These models are designed to offer increased robustness, autonomy, simplicity, and interpretability. Subsequently, the chapter detailed the integration of the proposed models with the GA for feature selection, aiming to enhance both accuracy and interpretability. Finally, ensembles of the proposed fuzzy models were discussed as a strategy for enhancing performance and reducing the risk of overfitting. The next chapter presents the experimental setup employed in the simulations.

5 METHODOLOGY

This chapter presents the comprehensive experimental setup for the simulations, detailing the evaluation metrics, hardware and software environments, datasets, and their statistical properties.

5.1 EVALUATION INDICATORS

The model's performance is assessed using the normalized root-mean-square error (NRMSE), the non-dimensional index error (NDEI), and the mean absolute percentage error (MAPE), given by:

$$NRMSE = \frac{RMSE}{\bar{y} - \underline{y}} \tag{5.1}$$

$$NDEI = \frac{RMSE}{std([y^1, ..., y^N])} \tag{5.2}$$

$$MAPE = \frac{1}{N} \sum_{k=1}^{N} \frac{\|y^k - \hat{y}^k\|}{y^k} \tag{5.3}$$

respectively, where y^k is the kth actual value, $\hat{y}^k$ is the kth predicted value, $\bar{y}$ is the maximum value for y, $\underline{y}$ is the minimum value for y, N is the sample size, $std()$ is the standard deviation function, and RMSE is the root-mean-square error given by:

$$RMSE = \sqrt{\frac{1}{N} \sum_{k=1}^{N} (y^k - \hat{y}^k)^2} \tag{5.4}$$

Each metric provides a different perspective on the model's accuracy. RMSE is widely used because its quadratic scoring rule heavily penalizes large errors and is suitable for gradient-based optimization. To provide a scale-independent evaluation, two normalized versions of RMSE are reported. The NRMSE normalizes the error by the range of the target data, expressing it as a percentage, which is useful for comparing performance across datasets with different scales. The NDEI normalizes the RMSE by the standard deviation of the data, providing insight into whether the model's error is smaller than the data's natural variability. Finally, while the previous metrics focus on absolute error, MAPE is included to evaluate performance based on relative percentage error, which can be more intuitive. However, MAPE is unsuitable for time series containing zero values due to the division-by-zero issue. Furthermore, its normalization can make it highly sensitive to errors for target values that are close to zero.

The error results are reported using tables, with the results categorized into four groups. The first group comprises classical models, including a selection of models from

the `scikit-learn` library. The second group consists of DL models. The third group comprises the eFSs. Finally, the fourth group contains the models proposed in this thesis.

Furthermore, a novel metric, the Correct Percentage of Predicted Movements (CPPM), is introduced for financial and cryptocurrency datasets. It measures the proportion of correctly predicted upward or downward movements, thereby providing investors with a baseline for a model's directional accuracy. The CPPM is defined as follows:

$$CPPM = \frac{1}{N} \sum_{k=1}^{N} o^k \tag{5.5}$$

where N is the sample size and o^k is a binary indicator that equals 1 if the model correctly predicts the direction of movement for sample k, and 0 otherwise.

$$o^k = \begin{cases} 1, & \text{if } \Delta y^k \cdot \Delta \hat{y}^k > 0 \\ 0, & \text{otherwise} \end{cases} \tag{5.6}$$

To formally assess the statistical significance of the performance differences between models, the Harvey, Leybourne, and Newbold (HLN) or Modified Diebold-Mariano (MDM) test [203] is employed. The analysis was conducted using the `dieboldmariano` Python library, which implements squared error (i.e., $(y^k - \hat{y}^k)^2$) as its default loss function. The null hypothesis of the HLN test is that the two models being compared exhibit equal predictive accuracy. This hypothesis is rejected for p-values below a significance level of $\alpha = 0.05$. A rejection indicates that the observed difference in forecast accuracy between the two models is statistically significant. For rule-based models, the total number of final rules is also reported. The models' hyperparameters are optimized using grid search to minimize the error. The hyperparameter search spaces are shown in Appendix D. Furthermore, the selected hyperparameters are reported in Appendix E.

5.2 MODEL SELECTION AND VALIDATION

The following four-step methodology is proposed for selecting the optimal model configuration based on the experiments conducted in this research:

1. **Perform a Grid-Search:** Train and validated across a predefined range of number of rules (e.g., from 3 to 20 rules). This process generates the quantitative performance data required for empirical, data-driven model selection.

2. **Plot the Error Curve:** Plot the validation error as a function of the number of rules. This visualization aids in identifying the elbow point—the stage at which adding more rules yields diminishing returns in predictive accuracy.

3. **Conduct a Qualitative Analysis:** For the set of best-performing models identified near the elbow point, extract, tabulate and translate the fuzzy rules into a human-readable format using linguistic variables. For example:

   *"If Forex Volatility is **HIGH** and Stock Spillover is **LOW**, then Systemic Risk is **MEDIUM**."*

4. **Select the Optimal Model:** Select the final modelbased on a holistic evaluation of both quantitative and qualitative criteria. The optimal model is defined as the one offering the most compelling balance between strong predictive performance (low validation error) and a parsimonious, interpretable rule base that is logically defensible.

5.3 REPRODUCIBILITY

All experiments were conducted on a PC equipped with a 13th Gen Intel® Core™ i7-1360P processor (2.2 GHz) and 16 GB of RAM. The simulations were performed using Python 3.9, supported by the following libraries and frameworks:

- **Data Processing:** `Pandas` 2.1.4, `NumPy` 1.26.4, `SciPy` 1.12.0
- **Visualization:** `Matplotlib` 3.8.0
- **Statistical Tests:** `dieboldmariano` for the Diebold-Mariano test, and `statsmodels` 0.14.0 for the Augmented Dickey-Fuller (ADF) test
- **Time Series Datasets:** `yfinance` 0.2.41 for financial datasets and `python-binance` 1.0.19 for cryptocurrency data
- **Genetic Algorithm:** `PyGad` 3.3.1
- **Classic Machine Learning Models:** `scikit-learn` 1.6.1
- **Gradient Boosting Models:** `LightGBM` 4.3.0 and `XGBoost` 2.0.3
- **Support Vector Regression:** `lssvr`[1]
- **Neural Networks:** `Keras` 2.15.0 and `TensorFlow` 2.18.0
- **Evolving Fuzzy Systems:** `evolvingfuzzysystems`[2]
- **Proposed Models:** `nfisis`[3]

[1] `https://github.com/zealberth/lssvr.git`
[2] `https://pypi.org/project/evolvingfuzzysystems/`
[3] `https://pypi.org/project/nfisis/`

Appendix B presents a list of Python libraries developed and published during this doctoral research, which were implemented in this work. Furthermore, instructions for implementing the proposed models using `nfisis` are provided in Appendix C.

5.4 DATASETS

The benchmark series, renewable energy datasets, financial series, and cryptocurrency datasets used in this research are described in this section. Additionally, two statistical tests are employed to analyze the normality and stationarity of real-world time series: the Shapiro-Wilk test [204] and the Augmented Dickey-Fuller (ADF) test [205]. While the derivations of these tests are beyond the scope of this work, detailed explanations can be found in the referenced literature.

For the Shapiro-Wilk test, a p-value less than 0.05 rejects the null hypothesis, indicating that the series does not follow a normal distribution. Conversely, a p-value greater than 0.05 suggests that the series follows a normal distribution. Similarly, for the ADF test, a p-value below 0.05 supports stationarity, while a p-value greater than 0.05 indicates non-stationarity.

5.4.1 BENCHMARK SERIES

Three benchmark series are employed in this work: the Lorenz Attractor, the Mackey-Glass series, and a nonlinear dynamic system, all of which are widely used in the literature for validating new models. Lorenz [206] introduced a multivariate time series composed of three ordinary differential equations, known as the Lorenz Attractor. The system is defined by Equations (5.7), (5.8), and (5.9).

$$\frac{da}{dt} = \gamma(b - a) \tag{5.7}$$

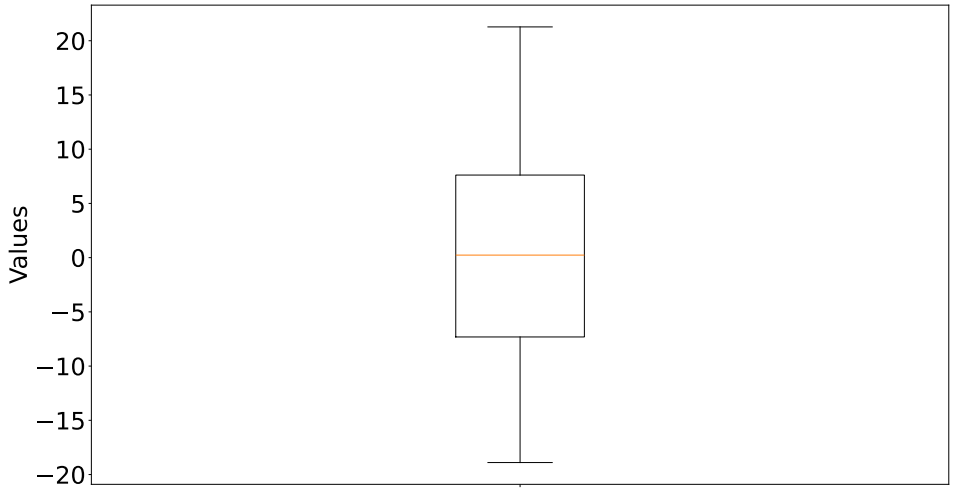
$$\frac{db}{dt} = a(\zeta - c) - b \tag{5.8}$$

$$\frac{dc}{dt} = ab - \beta c \tag{5.9}$$

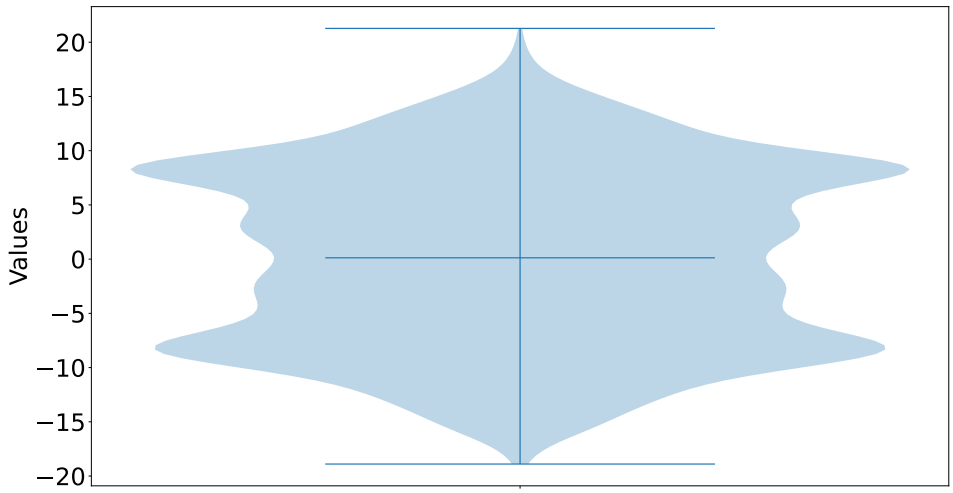
The parameters of the Lorenz system were set to $\gamma = 10$, $\beta = 2.667$, and $\zeta = 28$, with initial conditions $a(0) = 0$, $b(0) = 1$, and $c(0) = 1.05$, to induce chaotic behavior. A total of 10,000 data samples were generated. The objective is to predict a^{k+1} using the input vector $[a^k, b^k, c^k]$ for any step k. The first 8,000 data samples were used to train the models, while the final 2,000 samples were used for testing. This dataset is widely used to validate models on chaotic time series. Table 4 presents the correlation between the target

Table 4 – Correlation between the target and the attributes of Lorenz Attractor time series

Attribute 1	Attribute 2	Attribute 3
1.00	0.91	0.00



(a) Box plot for the Lorenz Attractor time series



(b) Violin plot for the Lorenz Attractor time series

Figure 9 – Box plot and violin plot for the Lorenz Attractor time series

value and the attributes for the series. Notably, the target value exhibits zero correlation with the last attribute. Figure 9 displays the box plot and the violin plot of the series.

The Mackey-Glass time series, a model of white blood cell production, was introduced by Mackey and Glass [207]. This series is governed by the following differential equation:

$$\frac{de}{dt} = \frac{0.2e(t-\eta)}{1+e^{10}(t-\eta)} - 0.1e(t-1) \tag{5.10}$$

where $e(0) = 1.2$ and $\eta = 17$. For $\eta \geq 17$ the series exhibits chaotic behavior [208].

Table 5 indicates that the target value exhibits the highest correlation with the second attribute. Figure 10 presents the box and violin plots for the series.

Table 5 – Correlation between the target and the attributes of Mackey-Glass time series

Attribute 1	Attribute 2	Attribute 3	Attribute 4
0.72	0.89	0.49	-0.15

The objective is to predict x^{k+85} using the input vector $[x^k, x^{k+6}, x^{k+12}, x^{k+18}]$ for any step k. The simulations employed 3,000 data samples for training ($k \in [201, 3200]$) and 500 data samples for testing ($k \in [5001, 5500]$).

Finally, the third benchmark is a nonlinear dynamic system [209, 210], governed by the following equation:

$$f^k = \frac{f^{k-1}f^{k-2}(f^{k-1}-0.5)}{1+(f^{k-1})^2+(f^{k-2})^2} - u^{k-1} \tag{5.11}$$

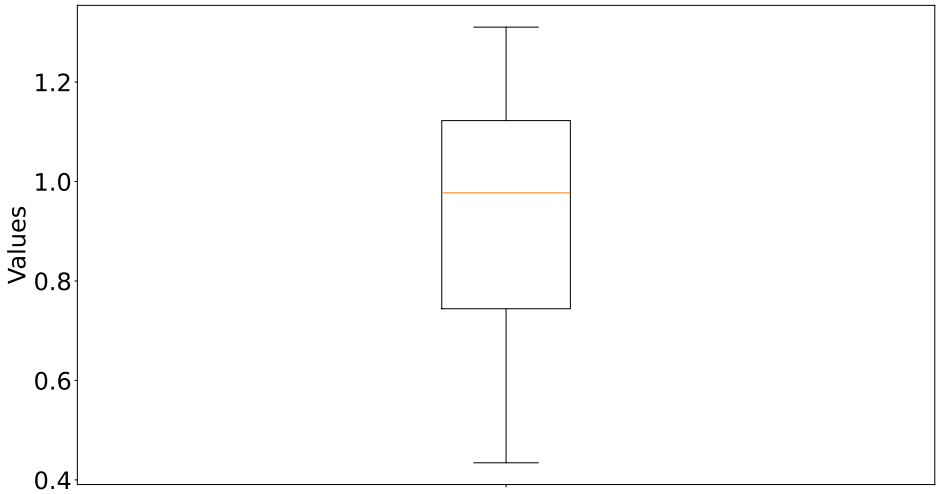
where $u^k = \sin(\frac{2\pi k}{25})$, and $f^0 = f^1 = 0$.

The input vector $[f^{k-2}, f^{k-1}, u^{k-1}]$ is used to predict f^k for any step k. The simulations were performed for $k \in [2, 5201]$, where the first 5,000 data samples were used for training and the final 200 data samples for testing. Table 6 presents the correlation between the target value and the attributes for the nonlinear time series. The target value is highly correlated with all attributes. Figure 11 displays the box and violin plots for the series.

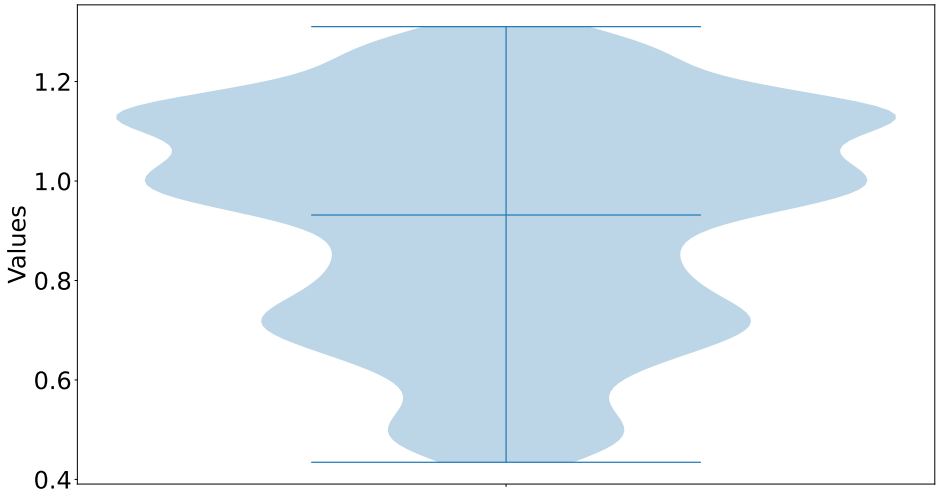
Table 6 – Correlation between the target and the attributes of Nonlinear time series

Attribute 1	Attribute 2	Attribute 3
0.87	0.97	0.99

Additionally, to assess the models' robustness, synthetic noise was added to the benchmark series. A corresponding noise signal was created by sampling from a Gaussian

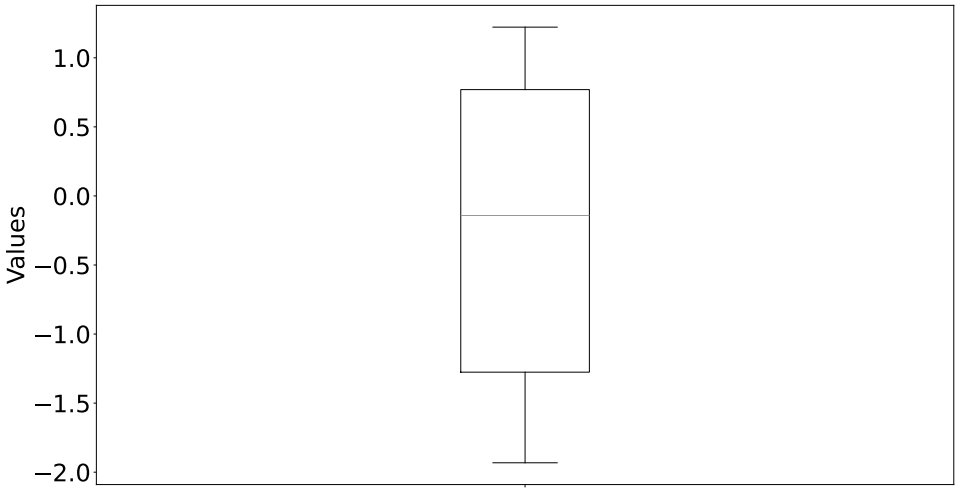


(a) Box plot for the Mackey-Glass time series

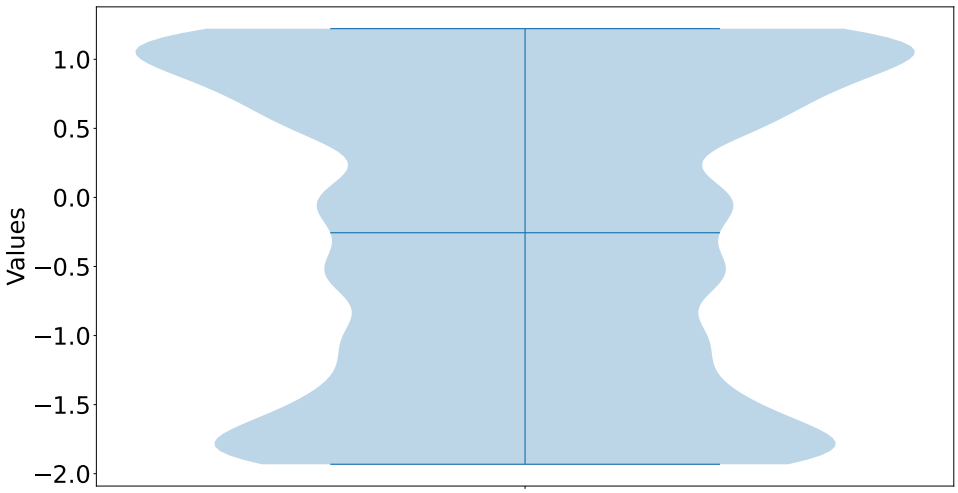


(b) Violin plot for the Mackey-Glass time series

Figure 10 – Box plot and violin plot for the Mackey-Glass time series



(a) Box plot for the Nonlinear time series



(b) Violin plot for the Nonlinear time series

Figure 11 – Box plot and violin plot for the Nonlinear time series

distribution. This constitutes additive white Gaussian noise with zero mean. The standard deviation of the noise (σ_{noise}) was set to 10% of the standard deviation of the original series ($\sigma_{S_{\text{original}}}$), as follows:

$$\sigma_{\text{noise}} = 0.1 \cdot \sigma_{S_{\text{original}}} \tag{5.12}$$

The final time series (S_{final}) was obtained by the element-wise addition of the original signal and the noise signal:

$$S_{\text{final}} = S_{\text{original}} + S_{\text{noise}}, \quad \text{where} \quad S_{\text{noise}} \sim \mathcal{N}(0, \sigma^2_{\text{noise}}) \tag{5.13}$$

5.4.2 PV ENERGY DATASETS

Finally, four PV energy datasets from two different power plants are utilized in the simulations. Two datasets originate from the Desert Knowledge Australia Solar Centre (DKASC) in Alice Springs, while the other two are from the Yulara Solar System[4]. The raw data were recorded at 5-minute intervals. Preprocessing was performed to remove null values and aggregate the data into daily values. The time period for all datasets spans from January 2021 to December 2022. Each dataset was divided chronologically into three distinct, non-overlapping subsets: 60% for training, 20% for validation, and 20% for testing. The training set is used to fit the model parameters, the validation set is used for hyperparameter tuning, and all final evaluation metrics reported in this study are computed on the unseen test set.

DKASC Alice Springs is a large-scale solar power station in the Alice Springs desert that houses PV technologies of various types, ages, brands, models, and configurations. Operating since 2008, the facility offers a vast database for researchers. This work aims to predict the daily power one step ahead using the following attributes as predictors: humidity [%], diffuse radiation [W/m^2], radiation [W/m^2], diffuse tilted [W/m^2], accumulated energy [kWh], rainfall [mm], wind direction [°], temperature [°C], global radiation [W/m^2], tilted [W/m^2], energy [kWh], power [kW], and current [A]. Table 7 presents the correlation between the target and the attributes for the Alice 1A and Alice 38 datasets. The results are listed in ascending order.

Furthermore, Figures 12a and 12b display the box plot and the violin plot for the target value of the Alice Springs datasets. The box plot indicates that the target value contains numerous outliers, particularly for values close to zero. Nonetheless, the violin plot demonstrates that the distribution approximates a normal distribution. A comparison of the two plots reveals how these outliers affect the overall shape of the distribution shown in the violin plot. More information about the datasets can be found on the official

[4] `https://dkasolarcentre.com.au/`

website: `https://dkasolarcentre.com.au/`. Since the plant employs solar panels with diverse characteristics, the panels are identified by specific numbers.

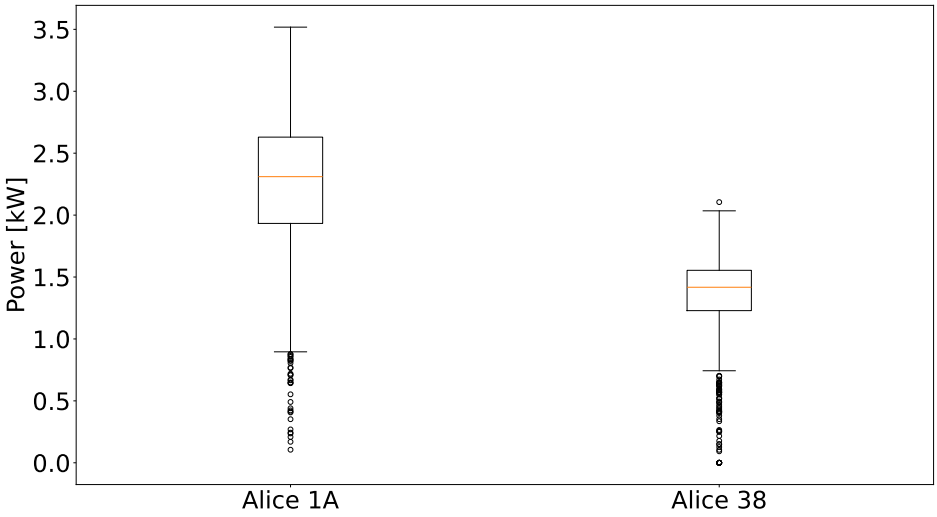
Table 7 – Correlation between the target and the attributes for Alice 1A and Alice 38

Attribute	Alice 1A	Alice 38
Humidity	-0.22	-0.18
Diffuse Radiation	-0.21	-0.22
Radiation Diffuse Tilted	-0.11	-0.13
Accumulated Energy	-0.08	-0.10
Rainfall	-0.01	-0.03
Wind Direction	0.13	0.09
Temperature	0.24	0.13
Energy	0.26	0.24
Radiation Global Tilted	0.38	0.24
Current	0.44	0.57
Power	0.55	0.57
Global Radiation	0.57	0.44

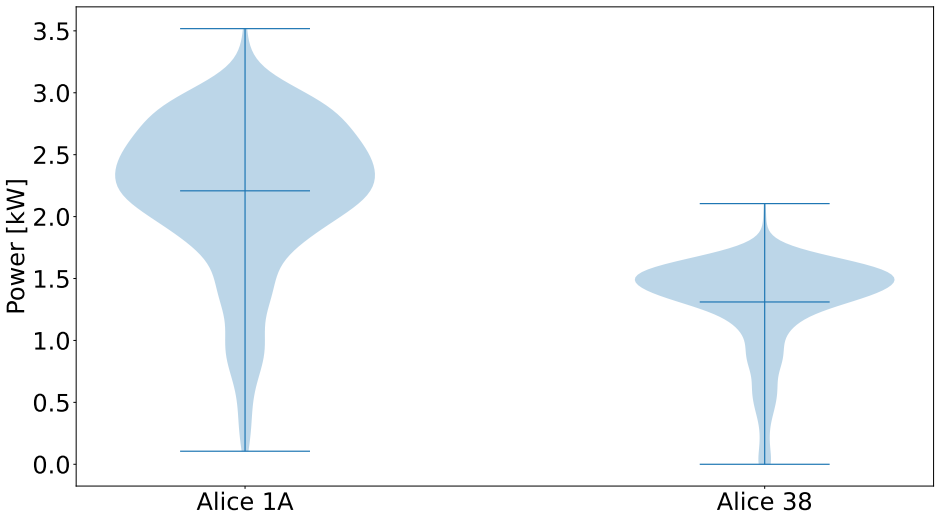
The Yulara Solar System, installed in 2014 near Yulara, operates a 1.8 MW solar photovoltaic plant. Two datasets from Yulara are utilized to predict the daily power one step ahead using the following predictors: air pressure, wind direction [°], rainfall [mm], hail, wind speed [m/s], accumulated energy [kWh], max wind speed [m/s], pyranometer, global radiation [W/m^2], temperature [°C], temperature probe 1 [°C], temperature probe 2 [°C], energy [kWh], power [kW], and current [A]. Further details regarding these attributes are available on the official website.

The two datasets selected for the simulations are Yulara 1 and Yulara 5. The raw data were recorded at 5-minute intervals. Preprocessing was performed to remove null values and aggregate the data into daily values. The datasets cover the period from January 2021 to December 2022. Table 8 presents the correlation between the target and the attributes for the Yulara 1 and Yulara 5 datasets. Notably, Yulara 1 exhibits stronger correlations with the attributes compared to Yulara 5. The results are listed in ascending order. Furthermore, Figures 13a and 13b display the box and violin plots for the target values of the Yulara datasets. The box plot reveals that only Yulara 5 contains outliers in the lower value range. In this case, drawing meaningful conclusions regarding the distribution solely from the violin plot is challenging.

Table 9 presents the characteristics of the PV datasets, while Table 10 details their statistical properties. The results suggest that only the Yulara 5 dataset follows a normal distribution, and that all datasets exhibit stationarity.



(a) Box plot for the target value of Alice 1A and Alice 38



(b) Violin plot for the target value of Alice 1A and Alice 38

Figure 12 – Box and violin plots for the Alice 1A and Alice 38 datasets

Table 8 – Correlation between Power One Step Ahead and Attributes for Yulara 1 and Yulara 5

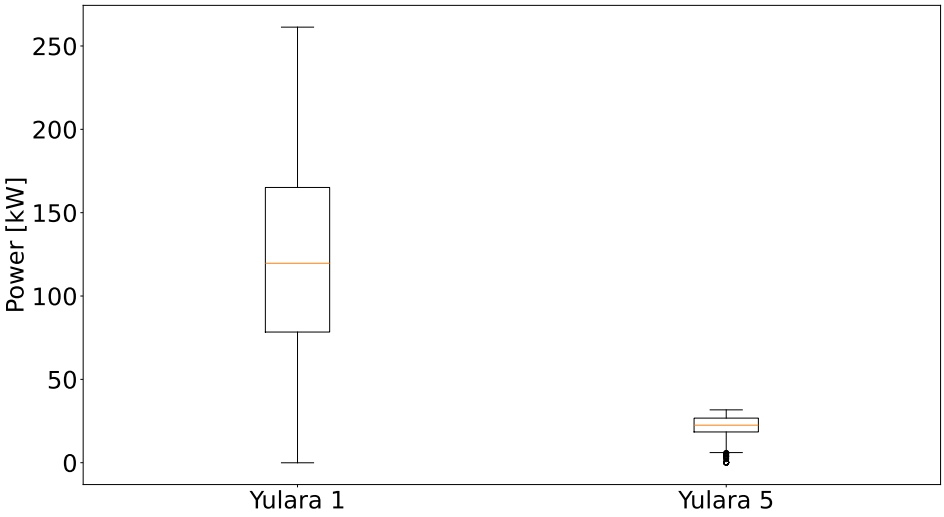
Attribute	Yulara 1	Yulara 5
Air Pressure	-0.43	-0.14
Wind Direction	-0.03	-0.03
Rainfall	0.09	0.02
Hail	0.09	0.07
Wind	0.10	0.07
Accumulated Energy	0.16	0.15
Max Wind Speed	0.19	0.18
Pyranometer	0.37	0.23
Global Radiation	0.50	0.26
Temperature	0.55	0.26
Temperature Probe 2	0.56	0.34
Temperature Probe 1	0.56	0.38
Energy	0.76	0.58
Power	0.76	0.58
Current	0.76	0.58

Table 9 – Characteristics of the PV datasets used in the simulations

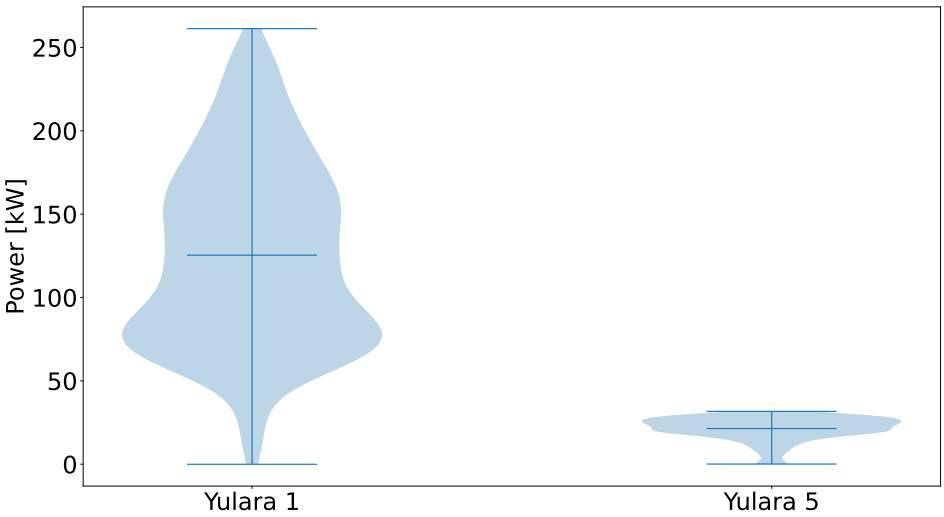
Characteristics	Alice 1A	Alice 38	Yulara 1	Yulara 5
Manufacturer	Trina	Q CELLS	-	-
Array Rating	10.5 kW	5.9 kW	1058.4 kW	105.9 kW
PV Technology	mono-Si	mono-Si	poly-Si	mono-Si
Array Structure Tracker	Dual Axis	Ground Mount	Ground Mount	Roof Mount
Installed	2009	2017	2016	2016

Table 10 – Statistical tests of the PV energy datasets

Test	Alice 1A	Alice 38	Yulara 1	Yulara 5
Shapiro-Wilk	$5.0 \cdot 10^{-17}$	$1.4 \cdot 10^{-27}$	$5.4 \cdot 10^{-11}$	0.37
ADF	$8.5 \cdot 10^{-8}$	$1.0 \cdot 10^{-20}$	$3.7 \cdot 10^{-20}$	$8.3 \cdot 10^{-4}$



(a) Box plot for the target value of Yulara 1 and Yulara 5



(b) Violin plot for the target value of Yulara 1 and Yulara 5

Figure 13 – Box and violin plots for the Yulara 1 and Yulara 5 datasets

5.4.3 FINANCIAL SERIES

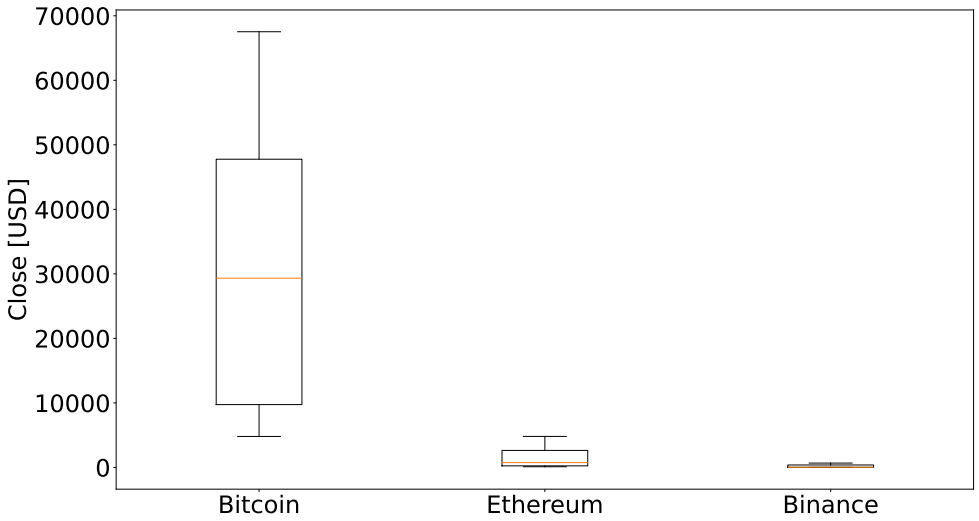
This work employs three well-known financial time series: the Standard & Poor's 500 (S&P 500), the National Association of Securities Dealers Automated Quotations (NASDAQ), and the Taiwan Stock Exchange Capitalization Weighted Stock Index (TAIEX). Financial time series are often characterized by significant non-stationary behavior, making them particularly challenging to model. The S&P 500 index comprises 500 large-cap stocks listed on major U.S. exchanges and is widely considered the most followed stock index, maintained by Dow Jones [211]. The NASDAQ, an American stock exchange founded in 1971 and privatized in 2006, is the second-largest stock exchange in the global market [212]. Finally, TAIEX, introduced in 1966, is the most widely quoted index of the Taiwan Stock Exchange Corporation (TSEC), established with a base value of 100 points [213].

Table 11 presents the correlation between the closing prices and various attributes of these financial series, with all attributes exhibiting high correlation with the target values. Additionally, Figures 14a and 14b illustrate the box and violin plots for the target values of the financial datasets. The box plot reveals no outliers in the series. Moreover, the TAIEX exhibits a bimodal distribution.

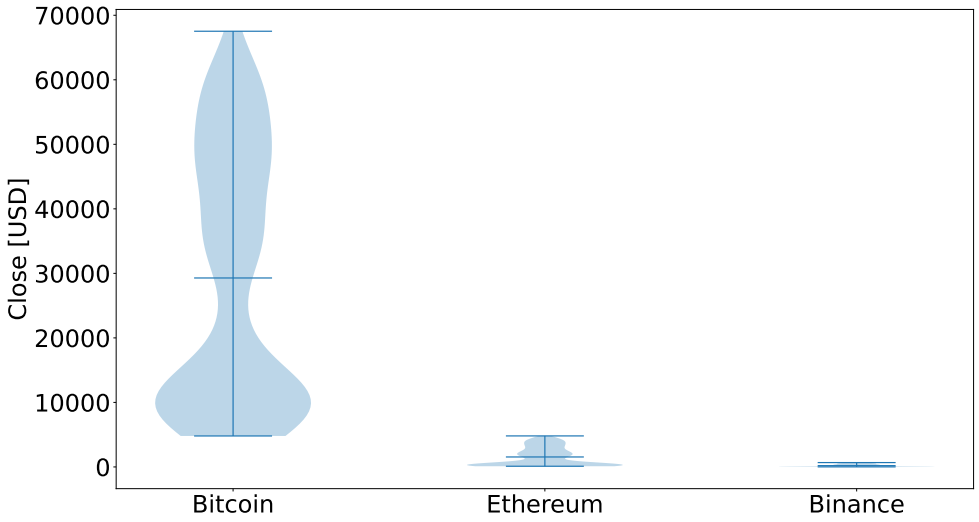
Table 11 – Correlation between the closing price and the attributes for finance series

Attribute	S&P 500	NASDAQ	TAIEX
Open	1.00	1.00	1.00
High	1.00	1.00	1.00
Low	1.00	1.00	1.00
Close	1.00	1.00	1.00

All simulations involve predicting the closing price one and five steps ahead, using the previous day's high, low, open, and close prices as predictors. The data spans the period from January 2020 to January 2022. A sliding window cross-validation approach is implemented. The outer loop employs ten folds to estimate the generalization error, while the inner loop utilizes a hold-out split on the training data for hyperparameter tuning. Specifically, the inner split allocates 80% of the data for training and 20% for validation. The final results are reported as the mean and standard deviation across the outer folds. Table 12 summarizes the statistical test results for the datasets. The findings indicate that the datasets do not follow a normal distribution and exhibit non-stationary behavior.



(a) Box plot for the target value of financial series



(b) Violin plot for the target value of financial series

Figure 14 – Box and violin plots for the financial series

Table 12 – Statistical tests of the financial datasets

Test	S&P 500	NASDAQ	TAIEX
Shapiro-Wilk	$5.5 \cdot 10^{-10}$	$2.3 \cdot 10^{-12}$	$2.9 \cdot 10^{-17}$
ADF	0.93	0.86	0.93

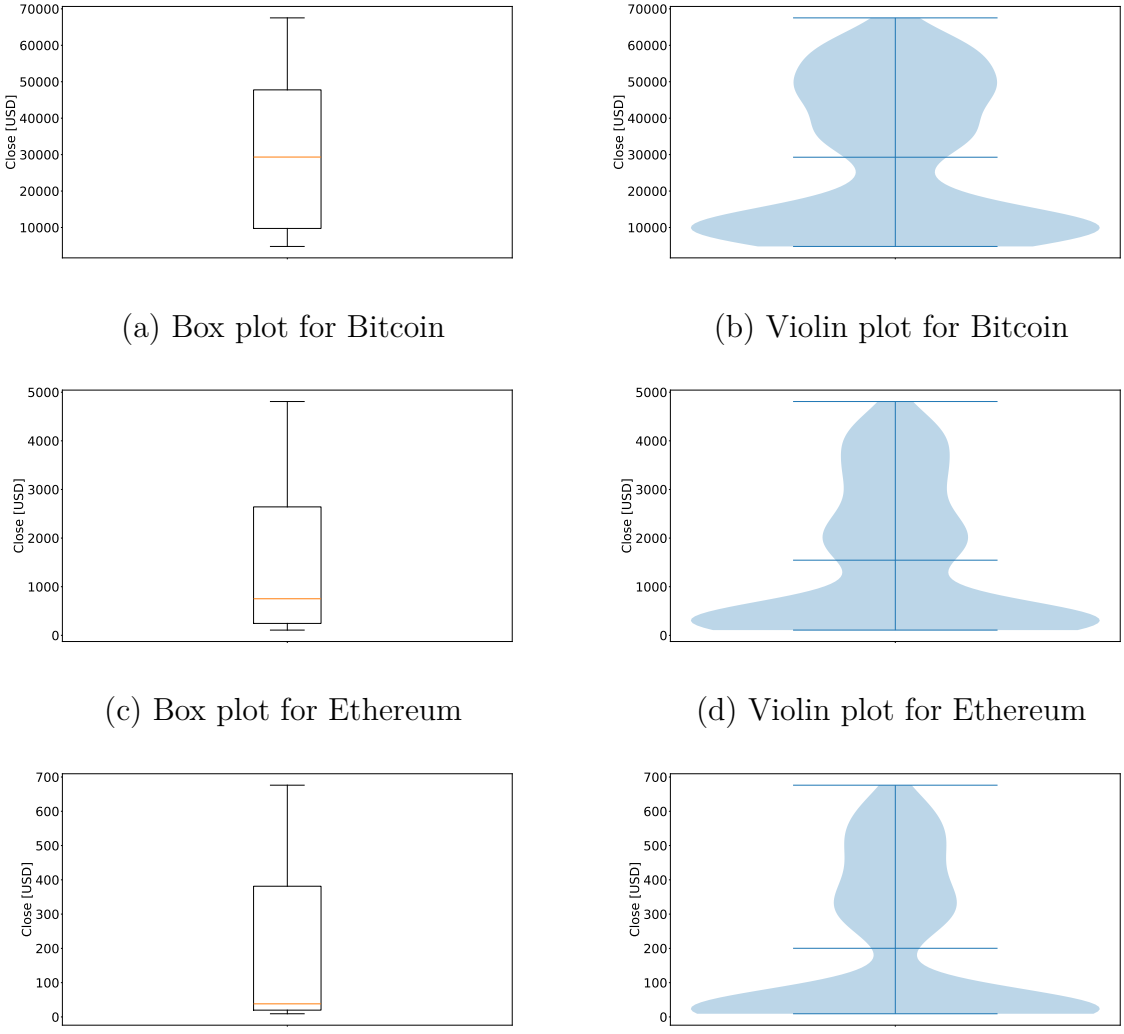
5.4.4 CRYPTOCURRENCIES

This study utilizes time series data from three major cryptocurrencies: Bitcoin (BTC), Ethereum (ETH), and Binance Coin (BNB). Bitcoin was officially launched on January 3, 2009, when the first block of its blockchain, the genesis block, was created [214]. Since then, it has experienced significant price appreciation. Figures 15a and 15b present the box and violin plots for Bitcoin, respectively; notably, the dataset contains no significant outliers. Subsequently, Ethereum was introduced by Vitalik Buterin in a 2013 white paper as a platform for building decentralized applications [215, 216]. Figures 15c and 15d display the box and violin plots for Ethereum. Finally, Binance Coin (BNB) is a prominent cryptocurrency first issued in July 2017 with a maximum supply of 200 million tokens [217]. Figures 15e and 15f show its box and violin plots. Table 13 details the correlation between the closing price and other attributes for each cryptocurrency. The target value exhibits high correlation with the open, high, low, and close prices, and moderate correlation with the quote asset volume, number of trades, and taker buy quote volume.

Table 13 – Correlation between the closing price and attributes for cryptocurrencies

Attribute	Bitcoin	Ethereum	Binance
Open	1.00	1.00	0.99
High	1.00	1.00	1.00
Low	1.00	1.00	0.99
Close	1.00	1.00	1.00
Volume	-0.09	-0.17	-0.18
Quote Asset Volume	0.70	0.65	0.60
Number of Trades	0.62	0.64	0.66
Taker Buy Base Volume	-0.08	-0.17	-0.17
Taker Buy Quote Volume	0.71	0.65	0.60

All simulations involve predicting the closing price one and five steps ahead, using the previous day's high, low, open, close, volume, quote asset volume, number of trades, taker buy base volume, and taker buy quote volume as predictors. The data spans the period from January 2020 to January 2022. A sliding window cross-validation approach is implemented. The outer loop employs ten folds to estimate the generalization error, while the inner loop utilizes a hold-out split on the training data for hyperparameter tuning. Specifically, the inner split allocates 80% of the data for training and 20% for validation. The final results are reported as the mean and standard deviation across the outer folds. Table 14 summarizes the statistical test results for the datasets. The findings indicate that the datasets do not follow a normal distribution and exhibit non-stationary behavior.



(a) Box plot for Bitcoin

(b) Violin plot for Bitcoin

(c) Box plot for Ethereum

(d) Violin plot for Ethereum

(e) Box plot for Binance

(f) Violin plot for Binance

Figure 15 – Box and violin plots for the target value of the cryptocurrency datasets

Table 14 – Statistical tests of the cryptocurrency datasets

Test	Bitcoin	Ethereum	Binance
Shapiro-Wilk	$1.0 \cdot 10^{-24}$	$4.3 \cdot 10^{-26}$	$1.2 \cdot 10^{-29}$
ADF	0.73	0.91	0.81

5.5 SUMMARY

This chapter outlined the metrics employed to assess model performance and introduced a novel metric specifically designed for evaluating financial and cryptocurrency datasets. It also highlighted the importance of selecting an appropriate error metric for the specific task. Details regarding the hardware, software, and libraries utilized were provided to ensure reproducibility. The proposed models have been made publicly available on PyPI, with a comprehensive description of the package provided to facilitate its application. Finally, the datasets used in the simulations were described in detail. Key statistics, such as the correlation between attributes and target values, were presented alongside box and violin plots. This analysis also included formal tests for the stationarity and normality of each time series. The next chapter presents the numerical simulations.

6 EXPERIMENTAL RESULTS AND DISCUSSION

This chapter provides an extensive presentation of the simulation results, followed by a thorough discussion of the number of rules, interpretability, computational performance, and stability analysis. Additionally, the observed limitations of the proposed approaches are addressed.

6.1 RESULTS

This section presents the simulation results, beginning with the benchmark series, followed by the solar energy and financial datasets, and concluding with the cryptocurrency datasets.

6.1.1 BENCHMARK SERIES

Table 15 presents the simulation results for the Lorenz Attractor. LS-SVM achieved the lowest errors among the classical models, while MLP performed best among the DL models. For the eFSs, eTS, Simpl_eTS, ePL, and ePL+ recorded the lowest errors; among the proposed fuzzy models, NTSK (RLS), NTSK (wRLS), GEN-NTSK (RLS), and GEN-NTSK (wRLS) yielded the best results. The eMG model produced the highest number of final rules. Table 16 displays the results of the MDM statistical test. A p-value lower than 0.05 indicates that the proposed model (column) statistically outperforms the compared model (row). The results indicate that the NTSK (RLS) and GEN-NTSK (RLS) models outperformed all other models. Furthermore, the NTSK (wRLS) and RF-NTSK models performed better than most classical and DL models. Figure 16 presents the best predictions from each model category for the Lorenz Attractor. All plotted models demonstrated high predictive accuracy, closely tracking the actual values.

Table 17 presents the simulation results for the Lorenz Attractor time series with a moderate level of noise. As expected, the errors substantially increased. Regarding the proposed models, the statistical test in Table 18 demonstrated that all models, except NMR and R-NMR, achieved lower errors than at least one of the compared models. Notably, the RF-NTSK outperformed 12 of the comparison models.

Table 15 – Simulations' results of Lorenz Attractor time series

Model	NRMSE	NDEI	MAPE	Rules
KNN [218]	0.00355	0.01641	0.04278	-
Regression Tree [219]	0.00349	0.01612	0.02535	-
Random Forest [112]	0.00167	0.00772	0.01262	-
SVM [220]	0.00148	0.00684	0.02107	-
LS-SVM [221]	**0.00063**	**0.00293**	**0.00532**	-
GBM [116]	0.00167	0.00772	0.01386	-
XGBoost [117]	0.00355	0.01640	0.03281	-
LGBM [118]	0.00243	0.01125	0.02212	-
MLP [222]	**0.00000**	**0.00000**	**0.00000**	-
CNN [223]	0.00146	0.00677	0.02083	-
RNN [224]	0.00020	0.00092	0.00215	-
LSTM [225]	0.00037	0.00170	0.00656	-
GRU [226]	0.00035	0.00163	0.00500	-
WaveNet [227]	0.00021	0.00097	0.00088	-
eTS [68]	**0.00000**	**0.00000**	**0.00000**	5
Simpl_eTS [69]	**0.00000**	0.00001	0.00002	2
exTS [70]	0.00007	0.00031	0.00050	5
ePL [72]	**0.00000**	**0.00000**	**0.00000**	1
eMG [74]	0.00040	0.00184	0.00505	54
ePL+ [73]	**0.00000**	**0.00000**	**0.00000**	1
ePL-KRLS-DISCO [75]	0.00035	0.00164	0.00208	20
NMR	0.01211	0.05600	0.15097	19
NTSK (RLS)	**0.00000**	**0.00000**	**0.00000**	1
NTSK (wRLS)	**0.00000**	0.00001	0.00001	1
GEN-NMR	0.01211	0.05600	0.15097	19
GEN-NTSK (RLS)	**0.00000**	**0.00000**	**0.00000**	1
GEN-NTSK (wRLS)	**0.00000**	0.00001	0.00001	1
R-NMR	0.02953	0.13657	0.62820	-
R-NTSK	0.00001	0.00005	0.00008	-
RF-NTSK	0.00001	0.00007	0.00011	-

Table 16 – Statistical test for Lorenz Attractor simulations

Model	NMR	NTSK (RLS)	NTSK (wRLS)	GEN-NMR	GEN-NTSK (RLS)	GEN-NTSK (wRLS)	R-NMR	R-NTSK	RF-NTSK
KNN [218]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
Regression Tree [219]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
Random Forest [112]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
SVM [220]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
LS-SVM [221]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
GBM [116]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
XGBoost [117]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
LGBM [118]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
MLP [222]	1.00	0.00	1.00	1.00	0.00	1.00	1.00	1.00	1.00
CNN [223]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
RNN [224]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
LSTM [225]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
GRU [226]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
WaveNet [227]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
eTS [68]	1.00	0.00	1.00	1.00	0.00	1.00	1.00	1.00	1.00
Simpl_eTS [69]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	1.00	1.00
exTS [70]	1.00	0.02	0.02	1.00	0.02	0.02	1.00	0.02	0.03
ePL [72]	1.00	0.00	1.00	1.00	0.00	1.00	1.00	1.00	1.00
eMG [74]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
ePL+ [73]	1.00	0.00	1.00	1.00	0.00	1.00	1.00	1.00	1.00
ePL-KRLS-DISCO [75]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00

Table 17 – Simulations' results of Lorenz Attractor time series with moderate level of noise

Model	NRMSE	NDEI	MAPE	Rules
KNN [218]	0.02493	0.12257	1.72540	-
Regression Tree [219]	0.02840	0.13961	1.52104	-
Random Forest [112]	0.02520	0.12386	**1.19619**	-
SVM [220]	**0.02410**	**0.11848**	2.77485	-
LS-SVM [221]	0.02633	0.12946	2.30759	-
GBM [116]	0.02531	0.12441	1.23998	-
XGBoost [117]	0.02557	0.12572	0.79261	-
LGBM [118]	0.02513	0.12352	1.68385	-
MLP [222]	0.02635	0.12955	**2.19797**	-
CNN [223]	0.02484	0.12213	2.28269	-
RNN [224]	0.02570	0.12634	2.47068	-
LSTM [225]	0.02656	0.13059	2.38655	-
GRU [226]	0.02569	0.12630	2.81280	-
WaveNet [227]	**0.02438**	**0.11983**	2.83930	-
eTS [68]	0.02633	0.12946	2.28601	1
Simpl_eTS [69]	0.02633	0.12946	2.28601	1
exTS [70]	0.02600	0.12780	2.31969	8
ePL [72]	0.02634	0.12948	2.24720	1
eMG [74]	**0.02396**	**0.11780**	2.65170	142
ePL+ [73]	0.02633	0.12946	**2.28017**	1
ePL-KRLS-DISCO [75]	0.02448	0.12034	2.80061	16
NMR	0.02724	0.13391	2.50302	14
NTSK (RLS)	0.02648	0.13019	2.13217	1
NTSK (wRLS)	0.02632	0.12940	2.28690	19
GEN-NMR	0.02674	0.13148	2.79764	15
GEN-NTSK (RLS)	0.02643	0.12991	2.09732	1
GEN-NTSK (wRLS)	0.02632	0.12940	2.28690	19
R-NMR	0.03527	0.17337	**1.33375**	-
R-NTSK	0.02633	0.12945	2.27748	-
RF-NTSK	**0.02505**	**0.12315**	1.71775	-

Table 18 – Statistical test for Lorenz Attractor simulation with moderate level of noise

Model	NMR	NTSK (RLS)	NTSK (wRLS)	GEN-NMR	GEN-NTSK (RLS)	GEN-NTSK (wRLS)	R-NMR	R-NTSK	RF-NTSK
KNN [218]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.70
Regression Tree [219]	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00
Random Forest [112]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.13
SVM [220]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
LS-SVM [221]	1.00	0.99	0.21	0.99	0.97	0.21	1.00	0.45	0.00
GBM [116]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.06
XGBoost [117]	1.00	1.00	0.99	1.00	1.00	0.99	1.00	1.00	0.00
LGBM [118]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.33
MLP [222]	1.00	1.00	0.10	0.99	1.00	0.10	1.00	0.19	0.00
CNN [223]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.82
RNN [224]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
LSTM [225]	1.00	0.17	0.00	0.91	0.06	0.00	1.00	0.01	0.00
GRU [226]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
WaveNet [227]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
eTS [68]	1.00	0.99	0.00	1.00	0.97	0.00	1.00	0.32	0.00
Simpl_eTS [69]	1.00	0.99	0.00	1.00	0.97	0.00	1.00	0.32	0.00
exTS [70]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
ePL [72]	1.00	0.99	0.04	0.99	0.99	0.04	1.00	0.23	0.00
eMG [74]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
ePL+ [73]	1.00	0.99	0.00	1.00	0.97	0.00	1.00	0.09	0.00
ePL-KRLS-DISCO [75]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99

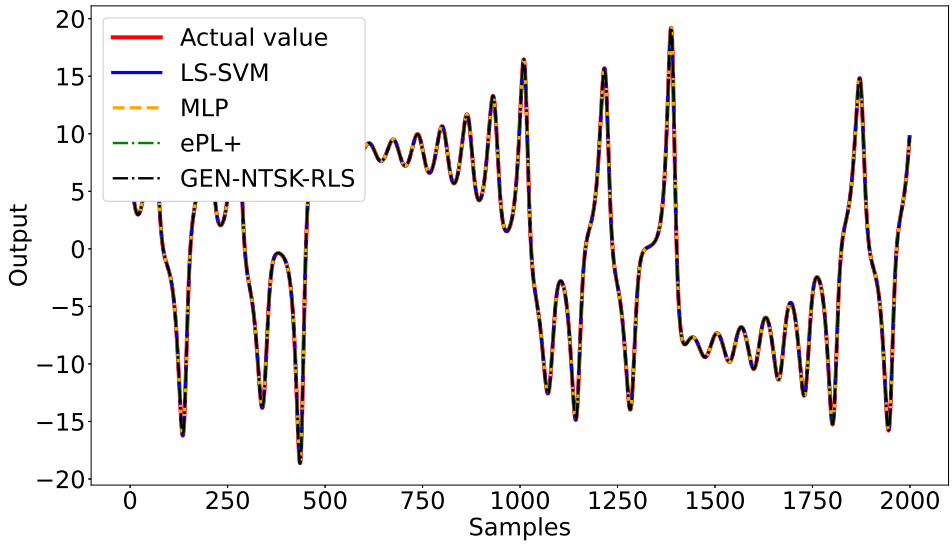


Figure 16 – Prediction performance on the Lorenz Attractor time series

Table 19 presents the simulation results for the Mackey-Glass time series. KNN obtained the lowest errors among the classical models, while LSTM performed best among the DL models. ePL-KRLS-DISCO recorded the lowest errors among all models, and RF-NTSK yielded the best results among the proposed fuzzy models. Notably, a simpler model like KNN can outperform more complex models, such as DL architectures. This can be attributed to the inherent complexity of DL, where optimizing numerous hyperparameters—such as the learning rate, number of layers and neurons, and activation functions makes finding a global minimum challenging. In some cases, simpler, well-suited models can be more effective. Among the rule-based models, most exhibited higher errors, with the exceptions being eMG, ePL-KRLS-DISCO, and the proposed RF-NTSK. The eMG model also generated 186 final rules, the highest of any model. Table 20 shows the statistical test results for the proposed models. Contrary to the Lorenz results, most of the proposed models did not show statistically superior performance. The main exception is RF-NTSK, which was statistically superior to several classical, DL, and eFS models, including SVM, XGBoost, and MLP. Figure 17 presents the best predictions from each model category for the Mackey-Glass time series. The results suggest that models capable of capturing long-term dependencies, such as ePL-KRLS-DISCO and certain ensemble approaches, are well-suited for predicting this type of chaotic series.

Table 21 presents the simulation results for the Mackey-Glass time series with a moderate level of noise. The best-performing model for each category remained the same; however, KNN and ePL-KRLS-DISCO exhibited substantially increased errors. Regarding the proposed models, the statistical test in Table 22 demonstrated that only the RF-NTSK achieved superior performance, outperforming 10 of the comparison models.

Table 19 – Simulations' results of the Mackey-Glass time series

Model	NRMSE	NDEI	MAPE	Rules
KNN [218]	**0.00737**	**0.02873**	**0.00447**	-
Regression Tree [219]	0.02054	0.08007	0.00985	-
Random Forest [112]	0.01064	0.04146	0.00666	-
SVM [220]	0.06987	0.27232	0.06244	-
LS-SVM [221]	0.09983	0.38908	0.08763	-
GBM [116]	0.00870	0.03390	0.00526	-
XGBoost [117]	0.06875	0.26795	0.05901	-
LGBM [118]	0.01263	0.04924	0.00906	-
MLP [222]	0.04539	0.17692	0.03147	-
CNN [223]	0.02210	0.08614	0.01712	-
RNN [224]	0.01872	0.07296	0.01432	-
LSTM [225]	**0.01237**	**0.04820**	**0.00977**	-
GRU [226]	0.01279	0.04983	0.01019	-
WaveNet [227]	0.01695	0.06607	0.01356	-
eTS [68]	0.09082	0.35399	0.07706	6
Simpl_eTS [69]	0.09270	0.36129	0.07888	13
exTS [70]	0.08789	0.34257	0.07305	7
ePL [72]	0.09779	0.38113	0.08506	1
eMG [74]	0.01279	0.04984	0.00985	186
ePL+ [73]	0.09134	0.35601	0.07836	6
ePL-KRLS-DISCO [75]	**0.00341**	**0.01328**	**0.00216**	33
NMR	0.09857	0.38417	0.08280	10
NTSK (RLS)	0.09955	0.38798	0.08515	1
NTSK (wRLS)	0.09026	0.35180	0.07901	18
GEN-NMR	0.09944	0.38759	0.08506	9
GEN-NTSK (RLS)	0.12141	0.47320	0.10543	1
GEN-NTSK (wRLS)	0.09032	0.35204	0.07905	19
R-NMR	0.10307	0.40174	0.08705	-
R-NTSK	0.09679	0.37725	0.08500	-
RF-NTSK	**0.01566**	**0.06105**	**0.01280**	-

Table 20 – Statistical MDM test for Mackey-Glass time series

Model	NMR	NTSK (RLS)	NTSK (wRLS)	GEN-NMR	GEN-NTSK (RLS)	GEN-NTSK (wRLS)	R-NMR	R-NTSK	RF-NTSK
KNN [218]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Regression Tree [219]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.07
Random Forest [112]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
SVM [220]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
LS-SVM [221]	0.22	0.44	0.00	0.38	1.00	0.00	0.98	0.00	0.00
GBM [116]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
XGBoost [117]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
LGBM [118]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
MLP [222]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
CNN [223]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
RNN [224]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
LSTM [225]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
GRU [226]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
WaveNet [227]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.08
eTS [68]	1.00	1.00	0.39	1.00	1.00	0.40	1.00	1.00	0.00
Simpl_eTS [69]	1.00	1.00	0.07	1.00	1.00	0.08	1.00	1.00	0.00
exTS [70]	1.00	1.00	0.87	1.00	1.00	0.87	1.00	1.00	0.00
ePL [72]	0.69	0.82	0.00	0.95	1.00	0.00	1.00	0.07	0.00
eMG [74]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
ePL+ [73]	1.00	1.00	0.30	1.00	1.00	0.31	1.00	1.00	0.00
ePL-KRLS-DISCO [75]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table 21 – Simulations' results of the Mackey-Glass time series with moderate level of noise

Model	NRMSE	NDEI	MAPE	Rules
KNN [218]	**0.06438**	**0.27324**	**0.05420**	-
Regression Tree [219]	0.07927	0.33642	0.06429	-
Random Forest [112]	0.06451	0.27378	0.05519	-
SVM [220]	0.06938	0.29443	0.06271	-
LS-SVM [221]	0.09208	0.39078	0.08817	-
GBM [116]	0.06476	0.27485	0.05453	-
XGBoost [117]	0.08201	0.34804	0.07688	-
LGBM [118]	0.06653	0.28234	0.05581	-
MLP [222]	0.07225	0.30661	0.06327	-
CNN [223]	0.08156	0.34614	0.07399	-
RNN [224]	0.06728	0.28554	0.05689	-
LSTM [225]	**0.05954**	**0.25270**	**0.04987**	-
GRU [226]	0.06569	0.27877	0.05777	-
WaveNet [227]	0.06922	0.29378	0.05859	-
eTS [68]	0.09199	0.39040	0.08796	1
Simpl_eTS [69]	0.09199	0.39040	0.08796	1
exTS [70]	0.08541	0.36246	0.08002	5
ePL [72]	0.09016	0.38264	0.08577	1
eMG [74]	0.07821	0.33191	0.06940	97
ePL+ [73]	0.09190	0.39002	0.08788	1
ePL-KRLS-DISCO [75]	**0.06558**	**0.27833**	**0.05792**	25
NMR	0.09323	0.39567	0.08661	14
NTSK (RLS)	0.09108	0.38655	0.08462	1
NTSK (wRLS)	0.09091	0.38583	0.08640	19
GEN-NMR	0.09465	0.40170	0.08858	17
GEN-NTSK (RLS)	0.09532	0.40452	0.08797	1
GEN-NTSK (wRLS)	0.09582	0.40664	0.09042	19
R-NMR	0.09615	0.40806	0.08951	-
R-NTSK	0.09169	0.38912	0.08704	-
RF-NTSK	**0.07008**	**0.29741**	**0.06532**	-

Table 22 – Statistical MDM test for Mackey-Glass time series with moderate level of noise

Model	NMR	NTSK (RLS)	NTSK (wRLS)	GEN-NMR	GEN-NTSK (RLS)	GEN-NTSK (wRLS)	R-NMR	R-NTSK	RF-NTSK
KNN [218]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Regression Tree [219]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
Random Forest [112]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
SVM [220]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.65
LS-SVM [221]	0.78	0.29	0.01	0.99	0.92	1.00	0.99	0.21	0.00
GBM [116]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
XGBoost [117]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
LGBM [118]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.98
MLP [222]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.09
CNN [223]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
RNN [224]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.91
LSTM [225]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
GRU [226]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99
WaveNet [227]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.69
eTS [68]	0.80	0.31	0.01	1.00	0.93	1.00	0.99	0.26	0.00
Simpl_eTS [69]	0.80	0.31	0.01	1.00	0.93	1.00	0.99	0.26	0.00
exTS [70]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
ePL [72]	0.98	0.71	0.96	1.00	0.99	1.00	1.00	1.00	0.00
eMG [74]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
ePL+ [73]	0.82	0.33	0.02	1.00	0.94	1.00	0.99	0.32	0.00
ePL-KRLS-DISCO [75]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

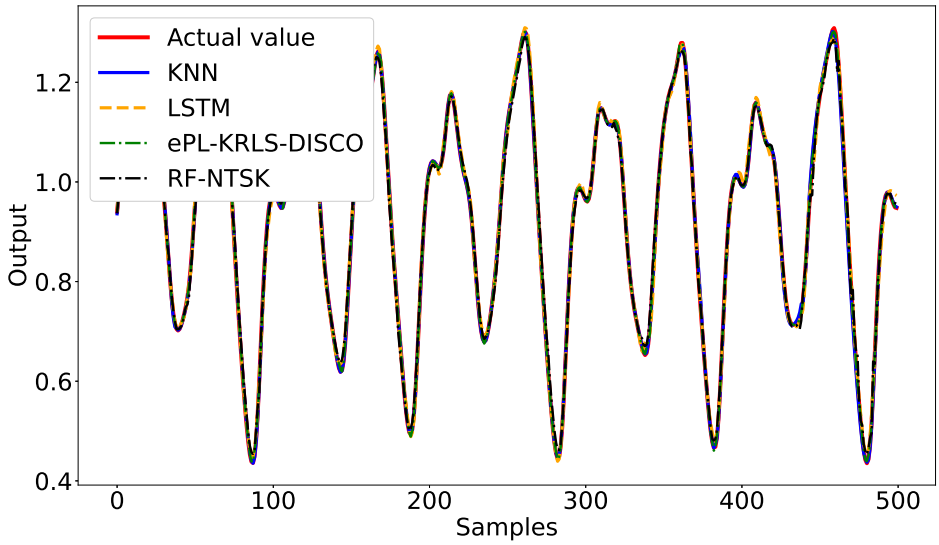


Figure 17 – Prediction performance on Mackey-Glass time series

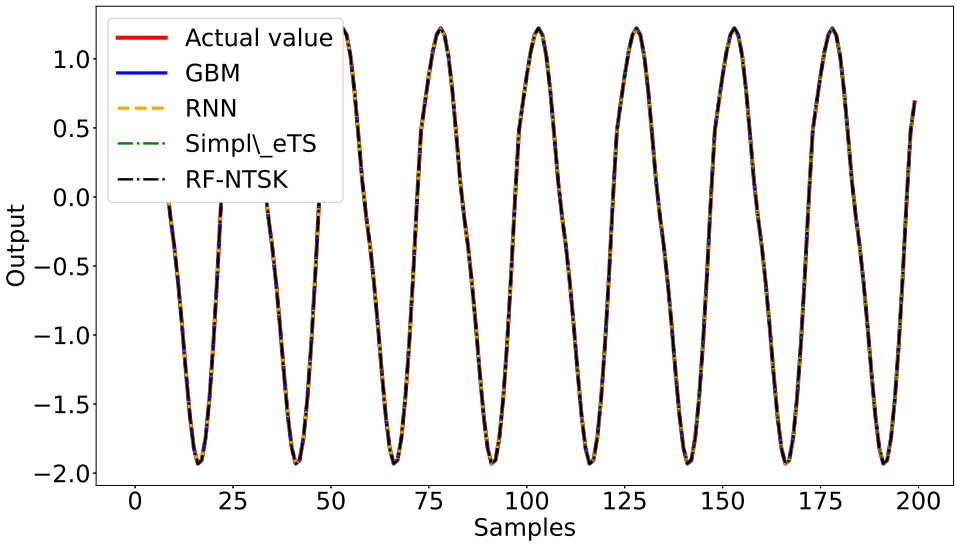


Figure 18 – Prediction performance on the nonlinear time series

Table 23 presents the simulation results for the nonlinear time series. KNN, RT, RF, and GBM obtained the lowest errors among the classical models, while RNN performed best among the DL models. For the eFSs, Simpl_eTS and ePL-KRLS-DISCO recorded the lowest errors, and RF-NTSK yielded the best results among the proposed models. Within the proposed group, the NTSK models outperformed the NMR models, which recorded the highest errors. The eMG model generated the highest number of final rules. Table 24 confirms that RF-NTSK yields better predictions than all models except KNN, RT, RF, and LGBM. Figure 18 presents the best predictions from each model category for the nonlinear time series.

Table 23 – Simulations' results of the nonlinear time series

Model	NRMSE	NDEI	MAPE	Rules
KNN [218]	**0.00000**	**0.00000**	**0.00000**	-
Regression Tree [219]	**0.00000**	**0.00000**	**0.00000**	-
Random Forest [112]	**0.00000**	**0.00000**	**0.00000**	-
SVM [220]	0.02038	0.05866	0.34041	-
LS-SVM [221]	0.01539	0.04430	0.41073	-
GBM [116]	**0.00000**	**0.00000**	**0.00000**	-
XGBoost [117]	0.00412	0.01186	0.04387	-
LGBM [118]	0.00560	0.01612	0.01342	-
MLP [222]	0.00110	0.00317	0.00404	-
CNN [223]	0.00100	0.00288	0.00450	-
RNN [224]	**0.00087**	**0.00252**	**0.00273**	-
LSTM [225]	0.00137	0.00396	0.00701	-
GRU [226]	0.00280	0.00805	0.03260	-
WaveNet [227]	0.00092	0.00264	0.00574	-
eTS [68]	0.00203	0.00585	0.05778	5
Simpl_eTS [69]	**0.00046**	**0.00134**	0.00246	21
exTS [70]	0.00522	0.01501	0.10631	3
ePL [72]	0.02399	0.06905	0.20790	1
eMG [74]	0.00523	0.01505	0.00786	41
ePL+ [73]	0.02673	0.07692	0.35664	2
ePL-KRLS-DISCO [75]	0.00180	0.00517	**0.00147**	33
NMR	0.03385	0.09741	0.23639	19
NTSK (RLS)	0.02466	0.07098	0.37729	1
NTSK (wRLS)	0.02083	0.05993	0.25286	17
GEN-NMR	0.02815	0.08100	0.23091	19
GEN-NTSK (RLS)	0.02466	0.07098	0.37729	1
GEN-NTSK (wRLS)	0.02083	0.05993	0.25286	17
R-NMR	0.05409	0.15567	1.06570	-
R-NTSK	0.02219	0.06386	0.33846	-
RF-NTSK	**0.00000**	**0.00000**	**0.00000**	-

Table 24 – Statistical MDM test for nonlinear time series

Model	NMR	NTSK (RLS)	NTSK (wRLS)	GEN-NMR	GEN-NTSK (RLS)	GEN-NTSK (wRLS)	R-NMR	R-NTSK	RF-NTSK
KNN [218]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Regression Tree [219]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Random Forest [112]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
SVM [220]	1.00	1.00	0.66	1.00	1.00	0.66	1.00	0.94	0.00
LS-SVM [221]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
GBM [116]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
XGBoost [117]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
LGBM [118]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.09
MLP [222]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
CNN [223]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
RNN [224]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
LSTM [225]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
GRU [226]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
WaveNet [227]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
eTS [68]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
Simpl_eTS [69]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
exTS [70]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
ePL [72]	1.00	0.70	0.05	0.96	0.70	0.05	1.00	0.13	0.00
eMG [74]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
ePL+ [73]	1.00	0.02	0.00	0.85	0.02	0.00	1.00	0.00	0.00
ePL-KRLS-DISCO [75]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00

Table 25 presents the simulation results for the nonlinear time series with a moderate level of noise. In this scenario, the best-performing models for each category differed significantly from the noise-free case. Random Forest (RF) yielded the lowest errors, whereas the proposed models, with the exception of RF-NTSK, exhibited higher errors. The statistical test results in Table 26 indicated that only RF-NTSK achieved statistically superior performance, outperforming 6 of the comparison models.

Table 25 – Simulations' results of the nonlinear time series with moderate level of noise

Model	NRMSE	NDEI	MAPE	Rules
KNN [218]	0.03177	0.10330	0.22483	-
Regression Tree [219]	0.03422	0.11126	0.25777	-
Random Forest [112]	**0.03078**	**0.10005**	**0.22680**	-
SVM [220]	0.03207	0.10427	0.24195	-
LS-SVM [221]	0.04614	0.15000	0.39798	-
GBM [116]	0.03231	0.10502	0.23946	-
XGBoost [117]	0.03359	0.10920	0.24271	-
LGBM [118]	0.03162	0.10280	0.24375	-
MLP [222]	0.03498	0.11371	0.29191	-
CNN [223]	0.03419	0.11115	0.30793	-
RNN [224]	0.03376	0.10974	**0.28021**	-
LSTM [225]	0.03360	0.10922	0.31463	-
GRU [226]	0.03486	0.11333	0.28948	-
WaveNet [227]	**0.03350**	**0.10889**	0.28508	-
eTS [68]	0.04595	0.14937	0.38946	1
Simpl_eTS [69]	0.04595	0.14937	0.38946	1
exTS [70]	0.04038	0.13126	0.31380	2
ePL [72]	0.04577	0.14879	0.38709	2
eMG [74]	0.03433	0.11160	**0.26104**	20
ePL+ [73]	0.04576	0.14876	0.38684	1
ePL-KRLS-DISCO [75]	**0.03362**	**0.10930**	0.28295	13
NMR	0.05172	0.16814	0.40322	18
NTSK (RLS)	0.04614	0.15000	0.39385	1
NTSK (wRLS)	0.04453	0.14476	0.36534	16
GEN-NMR	0.05093	0.16558	0.43517	19
GEN-NTSK (RLS)	0.05953	0.19353	0.56624	1
GEN-NTSK (wRLS)	0.05899	0.19179	0.54555	5
R-NMR	0.05955	0.19358	0.60643	-
R-NTSK	0.05164	0.16789	0.48268	-
RF-NTSK	**0.03452**	**0.11223**	**0.28578**	-

Table 26 – Statistical MDM test for nonlinear time series with moderate level of noise

Model	NMR	NTSK (RLS)	NTSK (wRLS)	GEN-NMR	GEN-NTSK (RLS)	GEN-NTSK (wRLS)	R-NMR	R-NTSK	RF-NTSK
KNN [218]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99
Regression Tree [219]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.57
Random Forest [112]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
SVM [220]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.98
LS-SVM [221]	1.00	0.50	0.00	0.99	1.00	1.00	1.00	1.00	0.00
GBM [116]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.96
XGBoost [117]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.81
LGBM [118]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99
MLP [222]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.35
CNN [223]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.60
RNN [224]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.73
LSTM [225]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.76
GRU [226]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.40
WaveNet [227]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.81
eTS [68]	1.00	0.70	0.00	0.99	1.00	1.00	1.00	1.00	0.00
Simpl_eTS [69]	1.00	0.70	0.00	0.99	1.00	1.00	1.00	1.00	0.00
exTS [70]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
ePL [72]	1.00	0.84	0.00	0.99	1.00	1.00	1.00	1.00	0.00
eMG [74]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.55
ePL+ [73]	1.00	0.83	0.00	0.99	1.00	1.00	1.00	1.00	0.00
ePL-KRLS-DISCO [75]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.75

6.1.2 RENEWABLE ENERGY

Table 27 presents the simulation results for the Alice 1A time series. SVM and LS-SVM achieved the lowest errors among the classical models, while LSTM performed best among the DL models. For the eFSs, eTS and ePL+ recorded the lowest errors, and GEN-NTSK (wRLS) and R-NTSK yielded the best results among the proposed fuzzy models. Furthermore, eTS obtained the lowest NRMSE and NDEI across all simulations, and ePL+ achieved the lowest MAPE, highlighting the efficacy of eFSs on real-world datasets.

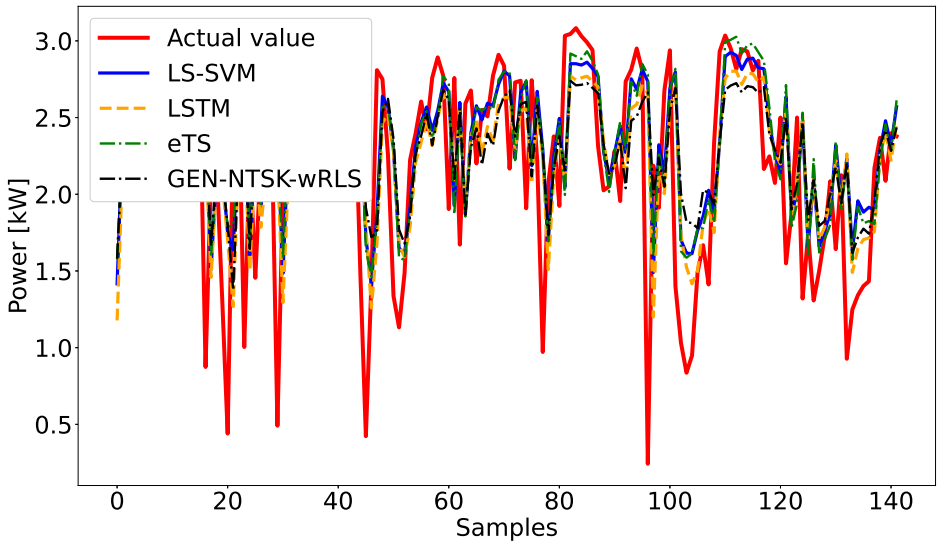


Figure 19 – Prediction performance on Alice 1A dataset

Although the eFS models achieved the best overall metrics (eTS for NRMSE/NDEI and ePL+ for MAPE), the efficacy of the proposed models is validated by the statistical

tests in Table 28. This analysis confirms that GEN-NTSK (wRLS) demonstrated a statistically significant advantage over eight of the twenty-one comparison models, with R-NTSK outperforming seven. Regarding model complexity, ePL, NTSK (RLS), and GEN-NTSK (RLS) generated the fewest rules, whereas eMG produced the highest number. A qualitative analysis of the predictions, illustrated in Figure 19, reveals a common challenge across all models: a difficulty in accurately forecasting extreme points. This suggests that the presence of outliers in the time series may limit the predictive performance of even the best-performing methods.

Table 27 – Simulations' results for the Alice 1A dataset

Model	NRMSE	NDEI	MAPE	Rules
KNN [218]	0.22656	1.00709	0.40166	-
Regression Tree [219]	0.21325	0.94790	0.37171	-
Random Forest [112]	0.21480	0.95481	0.36574	-
SVM [220]	0.20714	0.92075	**0.35383**	-
LS-SVM [221]	**0.20669**	**0.91876**	0.35509	-
GBM [116]	0.21849	0.97119	0.36887	-
XGBoost [117]	0.21538	0.95736	0.37573	-
LGBM [118]	0.21320	0.94771	0.37708	-
MLP [222]	0.26037	1.15734	0.46606	-
CNN [223]	0.22639	1.00634	0.40184	-
RNN [224]	0.20881	0.92818	0.36124	-
LSTM [225]	**0.20776**	**0.92350**	**0.35657**	-
GRU [226]	0.26014	1.15636	0.39765	-
WaveNet [227]	0.25401	1.12910	0.45572	-
eTS [68]	**0.20639**	**0.91742**	0.35508	4
Simpl_eTS [69]	0.33692	1.49764	0.43992	56
exTS [70]	0.20861	0.92729	0.36044	3
ePL [72]	0.20953	0.93136	0.35760	1
eMG [74]	0.40435	1.79734	0.56272	140
ePL+ [73]	0.20757	0.92266	**0.34947**	3
ePL-KRLS-DISCO [75]	0.29263	1.30077	0.46094	33
NMR	0.24337	1.08178	0.38902	16
NTSK (RLS)	0.24953	1.10918	0.43855	1
NTSK (wRLS)	0.20957	0.93154	0.35129	4
GEN-NMR	0.23421	1.04108	0.38867	17
GEN-NTSK (RLS)	0.22072	0.98112	0.36194	1
GEN-NTSK (wRLS)	**0.20711**	**0.92061**	0.36308	19
R-NMR	0.23220	1.03216	0.36129	-
R-NTSK	0.21431	0.95261	**0.35784**	-
RF-NTSK	0.21177	0.94132	0.35908	-

Table 28 – Statistical MDM test for Alice 1A

Model	NMR	NTSK (RLS)	NTSK (wRLS)	GEN-NMR	GEN-NTSK (RLS)	GEN-NTSK (wRLS)	R-NMR	R-NTSK	RF-NTSK
KNN [218]	0.85	1.00	0.07	0.70	0.27	0.00	0.65	0.10	0.05
Regression Tree [219]	0.99	1.00	0.29	0.95	0.82	0.14	0.97	0.56	0.37
Random Forest [112]	0.98	1.00	0.18	0.93	0.80	0.07	0.96	0.47	0.14
SVM [220]	0.99	1.00	0.74	0.98	0.99	0.50	1.00	0.99	0.94
LS-SVM [221]	0.99	1.00	0.74	0.98	1.00	0.55	1.00	0.99	0.97
GBM [116]	0.96	1.00	0.14	0.88	0.60	0.06	0.88	0.31	0.13
XGBoost [117]	0.97	1.00	0.19	0.92	0.77	0.04	0.93	0.43	0.18
LGBM [118]	0.99	1.00	0.30	0.96	0.83	0.05	0.95	0.57	0.37
MLP [222]	0.15	0.18	0.00	0.04	0.00	0.00	0.05	0.00	0.00
CNN [223]	0.97	0.95	0.06	0.86	0.33	0.01	0.67	0.14	0.09
RNN [224]	1.00	1.00	0.55	0.99	0.96	0.37	1.00	0.87	0.69
LSTM [225]	1.00	1.00	0.67	0.98	0.99	0.44	1.00	0.98	0.82
GRU [226]	0.00	0.30	0.00	0.00	0.01	0.00	0.04	0.00	0.00
WaveNet [227]	0.24	0.37	0.00	0.07	0.02	0.00	0.12	0.00	0.00
eTS [68]	0.99	1.00	0.75	0.98	0.99	0.57	1.00	0.97	0.96
Simpl_eTS [69]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
exTS [70]	0.99	1.00	0.57	0.97	0.97	0.37	0.98	0.88	0.81
ePL [72]	0.99	1.00	0.50	0.96	0.99	0.28	0.99	0.94	0.78
eMG [74]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ePL+ [73]	0.99	1.00	0.74	0.98	0.99	0.46	1.00	0.98	0.91
ePL-KRLS-DISCO [75]	0.03	0.03	0.00	0.01	0.00	0.00	0.00	0.00	0.00

The simulation results for the Alice 38 time series, presented in Table 29, underscore the strong performance of the proposed ensemble models. Within the fuzzy model category, R-NTSK and RF-NTSK delivered the top results. Their performance was highly competitive against the best models from other classes, including LS-SVM, CNN, GRU, eTS, and exTS. The statistical analysis in Table 30 further substantiates these findings, confirming that R-NTSK held a statistically significant advantage over eight of the comparison models. Regarding model complexity, ePL, NTSK (RLS), and GEN-NTSK (RLS) were the most parsimonious, generating the fewest rules, whereas eMG produced the most complex model. A visual inspection of the best predictions from each model class, presented in Figure 20, provides a qualitative comparison of their forecasting behavior in tracking the series' volatility.

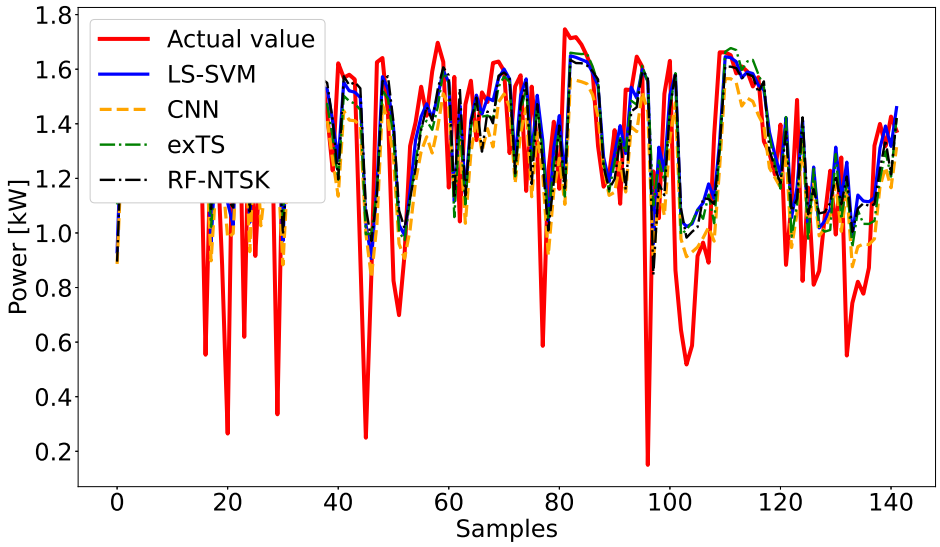


Figure 20 – Prediction performance on Alice 38 dataset

Table 29 – Simulations' results for the Alice 38 dataset

Model	NRMSE	NDEI	MAPE	Rules
KNN [218]	0.23499	1.04722	0.37156	-
Regression Tree [219]	0.21404	0.95385	0.34513	-
Random Forest [112]	0.21363	0.95204	0.34780	-
SVM [220]	0.21480	0.95723	0.34494	-
LS-SVM [221]	**0.20813**	**0.92755**	**0.33040**	-
GBM [116]	0.25371	1.13065	0.37142	-
XGBoost [117]	0.22014	0.98103	0.34765	-
LGBM [118]	0.21548	0.96026	0.34166	-
MLP [222]	0.21231	0.94616	0.34199	-
CNN [223]	**0.20700**	**0.92250**	0.32452	-
RNN [224]	0.21020	0.93677	0.33125	-
LSTM [225]	0.20858	0.92953	0.33252	-
GRU [226]	0.21089	0.93981	**0.32443**	-
WaveNet [227]	0.25446	1.13400	0.42215	-
eTS [68]	0.21114	0.94093	**0.32321**	3
Simpl_eTS [69]	0.22844	1.01805	0.34967	36
exTS [70]	**0.20443**	**0.91105**	0.33023	3
ePL [72]	0.58321	2.59907	0.68757	1
eMG [74]	0.28917	1.28866	0.42192	138
ePL+ [73]	0.25132	1.11999	0.35191	11
ePL-KRLS-DISCO [75]	0.26151	1.16542	0.36657	33
NMR	0.24519	1.09269	0.39498	9
NTSK (RLS)	0.25395	1.13171	0.41610	1
NTSK (wRLS)	0.23656	1.05424	0.35931	11
GEN-NMR	0.26173	1.16638	0.35348	13
GEN-NTSK (RLS)	0.21581	0.96176	0.34244	1
GEN-NTSK (wRLS)	0.25224	1.12411	0.36122	13
R-NMR	0.21897	0.97582	0.33655	-
R-NTSK	0.21150	0.94252	**0.33032**	-
RF-NTSK	**0.21087**	**0.93974**	0.33716	-

Table 30 – Statistical MDM test for Alice 38

Model	NMR	NTSK (RLS)	NTSK (wRLS)	GEN-NMR	GEN-NTSK (RLS)	GEN-NTSK (wRLS)	R-NMR	R-NTSK	RF-NTSK
KNN [218]	0.87	0.99	0.55	0.91	0.01	0.87	0.09	0.01	0.00
Regression Tree [219]	1.00	1.00	0.98	1.00	0.64	0.99	0.72	0.30	0.17
Random Forest [112]	1.00	1.00	0.99	1.00	0.69	1.00	0.74	0.30	0.10
SVM [220]	1.00	1.00	0.98	1.00	0.60	1.00	0.70	0.24	0.17
LS-SVM [221]	1.00	1.00	1.00	1.00	0.99	1.00	0.94	0.90	0.88
GBM [116]	0.30	0.51	0.11	0.69	0.00	0.47	0.01	0.00	0.00
XGBoost [117]	0.99	1.00	0.97	1.00	0.24	0.99	0.44	0.04	0.02
LGBM [118]	1.00	1.00	0.98	1.00	0.53	1.00	0.67	0.17	0.10
MLP [222]	1.00	1.00	0.99	1.00	0.72	1.00	0.88	0.44	0.39
CNN [223]	1.00	1.00	1.00	1.00	0.87	1.00	0.97	0.80	0.76
RNN [224]	1.00	1.00	0.99	1.00	0.78	1.00	0.97	0.60	0.55
LSTM [225]	1.00	1.00	1.00	1.00	0.86	1.00	0.96	0.72	0.68
GRU [226]	1.00	1.00	0.99	1.00	0.71	1.00	0.89	0.54	0.50
WaveNet [227]	0.12	0.48	0.13	0.63	0.00	0.45	0.01	0.00	0.00
eTS [68]	1.00	1.00	0.99	1.00	0.76	1.00	0.84	0.52	0.48
Simpl_eTS [69]	0.93	0.95	0.78	0.99	0.15	0.99	0.06	0.04	0.05
exTS [70]	1.00	1.00	1.00	1.00	0.99	1.00	0.96	0.97	0.98
ePL [72]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
eMG [74]	0.02	0.06	0.01	0.13	0.00	0.06	0.00	0.00	0.00
ePL+ [73]	0.40	0.54	0.22	0.69	0.08	0.52	0.06	0.05	0.05
ePL-KRLS-DISCO [75]	0.18	0.35	0.06	0.51	0.00	0.30	0.00	0.00	0.00

Table 31 presents the simulation results for the Yulara 1 time series. RF achieved the lowest errors among the classical models, while CNN performed best among the DL models. For the eFSs, eTS and exTS recorded the lowest errors; among the proposed fuzzy models, GEN-NMR and RF-NTSK yielded the best results. Notably, CNN achieved the lowest errors overall. The eMG model generated 189 final rules, the highest number among all models. In contrast, ePL, ePL+, NTSK (RLS), and GEN-NTSK (RLS) generated only a single final rule. Table 32 displays the statistical test results for the proposed models. The results indicate the superior performance of RF-NTSK over many of the comparison models. Figure 21 presents the best predictions from each model category for the Yulara 1 time series.

Table 31 – Simulations' results for the Yulara 1 dataset

Model	NRMSE	NDEI	MAPE	Rules
KNN [218]	0.21399	1.02255	0.39364	-
Regression Tree [219]	0.21037	1.00523	0.28632	-
Random Forest [112]	**0.17592**	**0.84064**	**0.27256**	-
SVM [220]	0.21055	1.00610	0.34657	-
LS-SVM [221]	0.21319	1.01869	0.36743	-
GBM [116]	0.18105	0.86514	0.29454	-
XGBoost [117]	0.20440	0.97670	0.30765	-
LGBM [118]	0.17896	0.85515	0.28816	-
MLP [222]	0.17511	0.83676	0.26667	-
CNN [223]	**0.16568**	**0.79168**	**0.25176**	-
RNN [224]	0.22641	1.08191	0.29806	-
LSTM [225]	0.24267	1.15959	0.29554	-
GRU [226]	0.22945	1.09639	0.29685	-
WaveNet [227]	0.17783	0.84973	0.27339	-
eTS [68]	0.18750	0.89595	**0.25613**	4
Simpl_eTS [69]	0.20622	0.98540	0.28118	45
exTS [70]	**0.16995**	**0.81211**	0.25916	5
ePL [72]	0.22578	1.07888	0.39487	1
eMG [74]	0.28605	1.36686	0.51978	176
ePL+ [73]	0.21503	1.02751	0.36951	1
ePL-KRLS-DISCO [75]	0.21213	1.01365	0.36978	20
NMR	0.27135	1.29663	0.38104	8
NTSK (RLS)	0.20606	0.98463	0.34618	1
NTSK (wRLS)	0.22261	1.06370	0.38723	2
GEN-NMR	0.19319	0.92315	**0.26033**	19
GEN-NTSK (RLS)	0.26105	1.24740	0.45110	1
GEN-NTSK (wRLS)	0.21812	1.04229	0.37640	19
R-NMR	0.26611	1.27157	0.36162	-
R-NTSK	0.20941	1.00066	0.35864	-
RF-NTSK	**0.18351**	**0.87687**	0.30299	-

Table 32 – Statistical MDM test for Yulara 1

Model	NMR	NTSK (RLS)	NTSK (wRLS)	GEN-NMR	GEN-NTSK (RLS)	GEN-NTSK (wRLS)	R-NMR	R-NTSK	RF-NTSK
KNN [218]	1.00	0.21	0.79	0.06	1.00	0.65	1.00	0.33	0.00
Regression Tree [219]	1.00	0.36	0.81	0.01	1.00	0.71	1.00	0.47	0.01
Random Forest [112]	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00	0.98
SVM [220]	1.00	0.25	1.00	0.06	1.00	1.00	1.00	0.29	0.00
LS-SVM [221]	1.00	0.12	1.00	0.05	1.00	1.00	1.00	0.00	0.00
GBM [116]	1.00	1.00	1.00	0.91	1.00	1.00	1.00	1.00	0.63
XGBoost [117]	1.00	0.56	0.90	0.10	1.00	0.84	1.00	0.65	0.02
LGBM [118]	1.00	1.00	1.00	0.97	1.00	1.00	1.00	1.00	0.76
MLP [222]	1.00	1.00	1.00	0.98	1.00	1.00	1.00	1.00	0.94
CNN [223]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
RNN [224]	0.99	0.11	0.42	0.01	0.96	0.33	0.99	0.17	0.00
LSTM [225]	0.93	0.02	0.16	0.00	0.81	0.11	0.93	0.04	0.00
GRU [226]	0.99	0.08	0.36	0.01	0.94	0.27	0.99	0.13	0.00
WaveNet [227]	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00	0.93
eTS [68]	1.00	0.94	0.99	0.75	1.00	0.98	1.00	0.95	0.36
Simpl_eTS [69]	1.00	0.50	0.84	0.15	1.00	0.77	1.00	0.58	0.05
exTS [70]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99
ePL [72]	0.99	0.00	0.01	0.01	1.00	0.00	0.98	0.00	0.00
eMG [74]	0.25	0.00	0.00	0.00	0.04	0.00	0.18	0.00	0.00
ePL+ [73]	1.00	0.03	1.00	0.04	1.00	0.85	1.00	0.00	0.00
ePL-KRLS-DISCO [75]	1.00	0.24	0.86	0.06	1.00	0.74	1.00	0.37	0.00

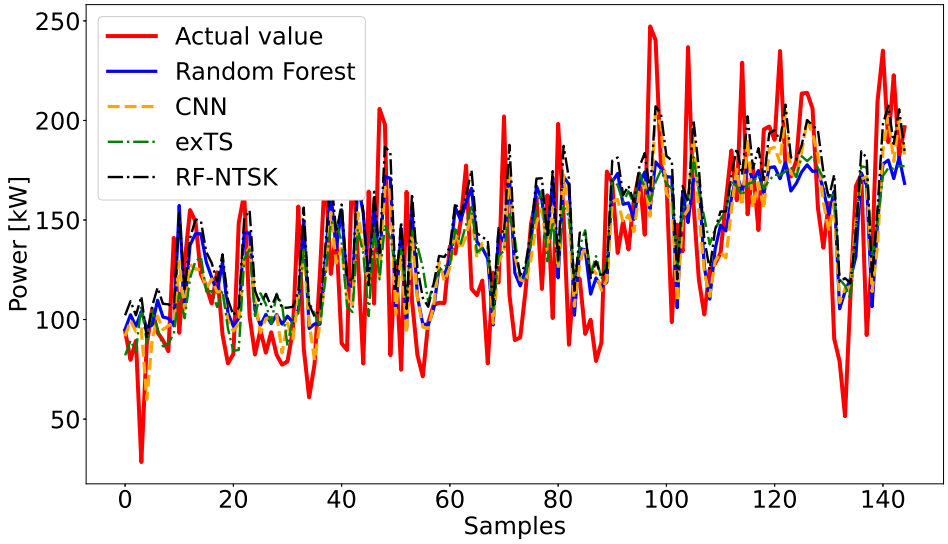


Figure 21 – Prediction performance on Yulara 1 dataset

Table 33 presents the simulation results for the Yulara 5 time series. LS-SVM achieved the lowest errors among the classical models, while CNN and GRU performed best among the DL models. For the eFSs, ePL-KRLS-DISCO recorded the lowest errors, and GEN-NTSK (wRLS) and R-NTSK yielded the best results among the proposed models. Overall, ePL-KRLS-DISCO achieved the lowest NRMSE and NDEI, whereas R-NTSK achieved the lowest MAPE. regarding complexity, Simpl_eTS generated 431 final rules, the highest number among all models, followed by eMG with 372. Table 34 displays the statistical test results for the proposed models, supporting the superior performance of GEN-NTSK (wRLS). Figure 22 presents the best predictions from each model category for the Yulara 5 time series.

Table 33 – Simulations' results for the Yulara 5 dataset

Model	NRMSE	NDEI	MAPE	Rules
KNN [218]	0.23408	1.10290	0.33711	-
Regression Tree [219]	0.21492	1.01262	0.29218	-
Random Forest [112]	0.21306	1.00388	0.28702	-
SVM [220]	0.22354	1.05322	0.29446	-
LS-SVM [221]	**0.20783**	**0.97920**	**0.28330**	-
GBM [116]	0.21769	1.02570	0.28887	-
XGBoost [117]	0.22848	1.07649	0.30705	-
LGBM [118]	0.21589	1.01717	0.28938	-
MLP [222]	0.21564	1.01601	0.30108	-
CNN [223]	**0.21186**	**0.99821**	0.28915	-
RNN [224]	0.29585	1.39393	0.37098	-
LSTM [225]	0.30403	1.43246	0.37513	-
GRU [226]	0.22070	1.03985	**0.28804**	-
WaveNet [227]	0.21745	1.02455	0.29647	-
eTS [68]	0.21658	1.02046	0.28506	3
Simpl_eTS [69]	0.22494	1.05983	0.29851	431
exTS [70]	0.20669	0.97386	0.28372	3
ePL [72]	0.20992	0.98907	0.29031	1
eMG [74]	0.32813	1.54603	0.41719	372
ePL+ [73]	0.20917	0.98555	0.28964	4
ePL-KRLS-DISCO [75]	**0.20501**	**0.96595**	**0.28255**	20
NMR	0.32884	1.54935	0.39332	2
NTSK (RLS)	0.22291	1.05029	0.31533	1
NTSK (wRLS)	0.21398	1.00818	0.28825	5
GEN-NMR	0.29698	1.39928	0.36355	15
GEN-NTSK (RLS)	0.22080	1.04035	0.29857	1
GEN-NTSK (wRLS)	**0.20625**	**0.97176**	0.28503	15
R-NMR	0.36554	1.72230	0.42459	-
R-NTSK	0.20632	0.97209	**0.27714**	-
RF-NTSK	0.20749	0.97761	0.27978	-

Table 34 – Statistical MDM test for Yulara 5

Model	NMR	NTSK (RLS)	NTSK (wRLS)	GEN-NMR	GEN-NTSK (RLS)	GEN-NTSK (wRLS)	R-NMR	R-NTSK	RF-NTSK
KNN [218]	1.00	0.04	0.02	1.00	0.04	0.00	1.00	0.00	0.00
Regression Tree [219]	1.00	0.79	0.43	1.00	0.81	0.08	1.00	0.03	0.02
Random Forest [112]	1.00	0.82	0.57	1.00	0.88	0.13	1.00	0.07	0.01
SVM [220]	1.00	0.47	0.05	1.00	0.27	0.00	1.00	0.00	0.00
LS-SVM [221]	1.00	0.98	0.93	1.00	1.00	0.26	1.00	0.23	0.46
GBM [116]	1.00	0.68	0.27	1.00	0.67	0.05	1.00	0.02	0.00
XGBoost [117]	1.00	0.28	0.03	1.00	0.14	0.00	1.00	0.00	0.00
LGBM [118]	1.00	0.73	0.38	1.00	0.74	0.08	1.00	0.05	0.02
MLP [222]	1.00	0.72	0.42	1.00	0.70	0.12	1.00	0.13	0.13
CNN [223]	1.00	0.85	0.66	1.00	0.92	0.14	1.00	0.11	0.08
RNN [224]	1.00	0.00	0.00	0.59	0.00	0.00	1.00	0.00	0.00
LSTM [225]	1.00	0.00	0.00	0.08	0.00	0.00	1.00	0.00	0.00
GRU [226]	1.00	0.58	0.09	1.00	0.51	0.01	1.00	0.00	0.00
WaveNet [227]	1.00	0.68	0.28	1.00	0.68	0.03	1.00	0.03	0.02
eTS [68]	1.00	0.73	0.29	1.00	0.77	0.03	1.00	0.01	0.04
Simpl_eTS [69]	1.00	0.43	0.11	1.00	0.32	0.02	1.00	0.01	0.02
exTS [70]	1.00	0.99	0.97	1.00	1.00	0.39	1.00	0.44	0.58
ePL [72]	1.00	0.94	0.82	1.00	0.97	0.09	1.00	0.15	0.27
eMG [74]	0.51	0.00	0.00	0.14	0.00	0.00	0.91	0.00	0.00
ePL+ [73]	1.00	0.95	0.84	1.00	0.96	0.22	1.00	0.27	0.37
ePL-KRLS-DISCO [75]	1.00	0.97	0.92	1.00	0.99	0.58	1.00	0.60	0.69

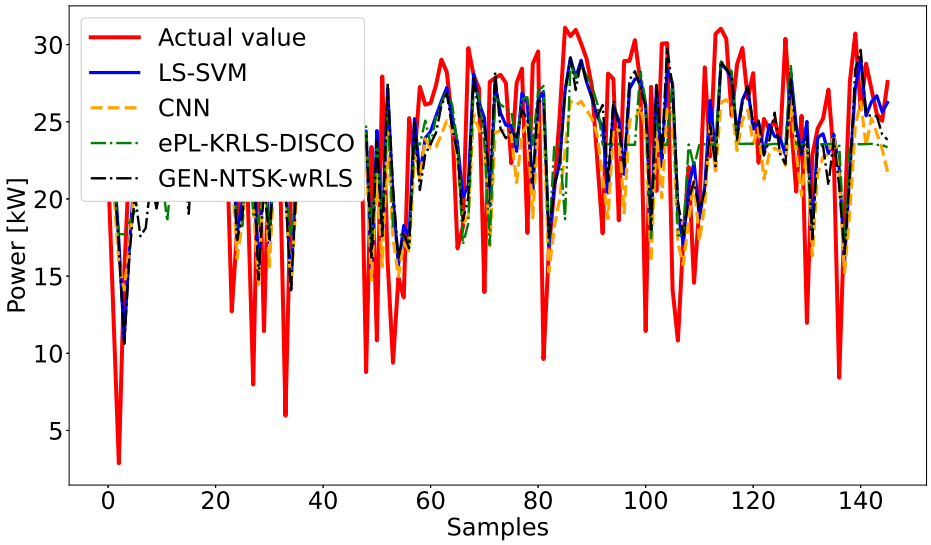


Figure 22 – Prediction performance on Yulara 5 dataset

6.1.3 FINANCIAL SERIES

Table 35 presents the simulation results for predicting the S&P 500 one step ahead. Among the classical models, KNN and XGBoost achieved the lowest mean errors, while MLP, CNN, and WaveNet exhibited the lowest errors among the DL models. For the eFSs, eTS and ePL performed best; among the proposed fuzzy systems, NTSK (RLS), NTSK (wRLS), GEN-NTSK (RLS), GEN-NTSK (wRLS), and R-NTSK yielded the lowest errors. Notably, Simpl_eTS produced the highest mean and standard deviation for the number of rules. In contrast, several models performed poorly, including SVM (among classical models) and the recurrent neural networks (RNN, LSTM, GRU). In terms of directional accuracy, the eMG model achieved the highest mean CPPM. The statistical test results in Table 36 confirm that the proposed NTSK-based models were statistically superior to many of the benchmark approaches.

Table 35 – Simulations' results of the S&P 500 series (horizon = 1)

Model	NRMSE	NDEI	MAPE	CPPM (%)	Rules
KNN [218]	**0.31 (0.15)**	**1.23 (0.55)**	**0.02 (0.01)**	44.55 (6.76)	-
Regression Tree [219]	0.33 (0.13)	1.31 (0.47)	**0.02 (0.01)**	44.55 (6.36)	-
Random Forest [112]	0.74 (0.41)	3.03 (1.88)	0.05 (0.02)	42.27 (5.20)	-
SVM [220]	2.21 (1.43)	9.01 (6.19)	0.16 (0.06)	50.91 (7.89)	-
LS-SVM [221]	1.59 (0.82)	6.51 (3.78)	0.12 (0.03)	51.14 (8.15)	-
GBM [116]	0.43 (0.19)	1.73 (0.75)	0.03 (0.01)	45.68 (6.62)	-
XGBoost [117]	0.35 (0.15)	1.38 (0.57)	**0.02 (0.01)**	43.41 (5.79)	-
LGBM [118]	1.60 (0.97)	6.54 (4.33)	0.11 (0.04)	43.18 (4.55)	-
MLP [222]	**0.13 (0.02)**	**0.52 (0.13)**	**0.01 (0.01)**	51.14 (8.15)	-
CNN [223]	0.13 (0.03)	0.53 (0.13)	**0.01 (0.01)**	50.23 (8.29)	-
RNN [224]	2.13 (1.36)	8.68 (5.94)	0.14 (0.06)	41.82 (5.40)	-
LSTM [225]	1.52 (1.32)	6.26 (5.80)	0.11 (0.07)	46.36 (6.83)	-
GRU [226]	3.05 (1.28)	12.32 (5.18)	0.30 (0.19)	45.00 (7.46)	-
WaveNet [227]	0.14 (0.03)	0.56 (0.14)	**0.01 (0.01)**	51.14 (9.44)	-
eTS [68]	**0.13 (0.03)**	**0.54 (0.13)**	0.01 (0.01)	49.32 (8.07)	4 (0)
Simpl_eTS [69]	0.16 (0.04)	0.64 (0.19)	0.01 (0.01)	51.36 (7.89)	56 (24)
exTS [70]	0.15 (0.06)	0.63 (0.25)	0.01 (0.02)	51.36 (7.27)	4 (0)
ePL [72]	**0.13 (0.03)**	**0.54 (0.13)**	**0.01 (0.00)**	49.77 (9.12)	1 (0)
eMG [74]	0.25 (0.11)	1.01 (0.41)	0.02 (0.02)	53.18 (8.64)	12 (2)
ePL+ [73]	0.96 (2.38)	3.52 (8.42)	0.06 (0.15)	49.32 (6.97)	3 (1)
ePL-KRLS-DISCO [75]	0.42 (0.29)	1.66 (1.03)	0.03 (0.02)	46.59 (7.28)	32 (4)
NMR	0.96 (0.35)	3.89 (1.54)	0.06 (0.02)	47.05 (7.47)	6 (0)
NTSK (RLS)	**0.13 (0.03)**	**0.52 (0.13)**	**0.01 (0.00)**	47.73 (7.19)	1 (0)
NTSK (wRLS)	**0.13 (0.03)**	0.54 (0.12)	**0.01 (0.00)**	49.55 (8.79)	1 (0)
GEN-NMR	1.18 (0.91)	4.64 (3.31)	0.08 (0.06)	49.09 (8.14)	9 (0)
GEN-NTSK (RLS)	0.14 (0.03)	0.57 (0.17)	**0.01 (0.00)**	47.05 (10.01)	1 (0)
GEN-NTSK (wRLS)	0.14 (0.03)	0.56 (0.13)	0.01 (0.01)	49.09 (9.38)	1 (0)
R-NMR	0.93 (0.34)	3.81 (1.60)	0.06 (0.01)	50.91 (8.64)	-
R-NTSK	**0.13 (0.03)**	0.53 (0.12)	0.01 (0.01)	49.32 (8.20)	-
RF-NTSK	0.16 (0.04)	0.67 (0.18)	0.01 (0.01)	49.32 (8.20)	-

Table 36 – Statistical MDM test for S&P 500 (horizon = 1)

Model	NMR	NTSK (RLS)	NTSK (wRLS)	GEN-NMR	GEN-NTSK (RLS)	GEN-NTSK (wRLS)	R-NMR	R-NTSK	RF-NTSK
KNN [218]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
Regression Tree [219]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
Random Forest [112]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
SVM [220]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LS-SVM [221]	0.00	0.00	0.00	0.78	0.00	0.00	0.00	0.00	0.00
GBM [116]	0.00	0.00	0.00	0.66	0.00	0.00	0.00	0.00	0.00
XGBoost [117]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
LGBM [118]	0.00	0.00	0.00	0.61	0.00	0.00	0.00	0.00	0.00
MLP [222]	1.00	0.07	0.26	1.00	0.20	1.00	1.00	0.46	1.00
CNN [223]	1.00	0.08	0.25	1.00	0.20	0.96	1.00	0.43	1.00
RNN [224]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LSTM [225]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
GRU [226]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
WaveNet [227]	1.00	0.01	0.03	1.00	0.03	0.35	1.00	0.05	0.94
eTS [68]	1.00	0.00	0.01	1.00	0.01	1.00	1.00	0.01	1.00
Simpl_eTS [69]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.01
exTS [70]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
ePL [72]	1.00	0.02	0.57	1.00	0.38	0.99	1.00	0.82	1.00
eMG [74]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
ePL+ [73]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ePL-KRLS-DISCO [75]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00

Table 37 presents the simulation results for predicting the S&P 500 five steps ahead. KNN and RF obtained the lowest mean errors among the classical models, while MLP, CNN, and WaveNet performed best among the DL models. For the eFSs, eTS, Simpl_eTS, and exTS recorded the lowest errors; among the proposed fuzzy models, NTSK (RLS) and GEN-NTSK (RLS) yielded the best results. Simpl_eTS exhibited the highest mean and standard deviation for the number of rules. Conversely, SVM, LS-SVM, LightGBM, RNN, LSTM, ePL+, NMR, and GEN-NMR exhibited higher errors. GEN-NMR achieved the highest mean CPPM. Table 38 confirms that the NTSK models achieved lower errors than many of the comparison models. Figure 23 shows the predictions for the S&P 500 five steps ahead. It is worth noting that the curve for RF is non-smooth, resembling a step line chart.

Table 37 – Simulations' results of the S&P 500 series (horizon = 5)

Model	NRMSE	NDEI	MAPE	CPPM (%)	Rules
KNN [218]	0.54 (0.25)	**2.14 (0.92)**	0.04 (0.03)	46.36 (6.03)	-
Regression Tree [219]	0.58 (0.31)	2.29 (1.15)	0.05 (0.03)	48.41 (6.27)	-
Random Forest [112]	**0.54 (0.20)**	2.15 (0.74)	**0.04 (0.02)**	47.73 (5.18)	-
SVM [220]	2.20 (1.40)	8.97 (6.08)	0.16 (0.06)	49.09 (6.52)	-
LS-SVM [221]	1.60 (0.81)	6.53 (3.74)	0.12 (0.03)	48.86 (6.45)	-
GBM [116]	0.55 (0.24)	2.21 (0.91)	0.04 (0.03)	46.82 (6.36)	-
XGBoost [117]	0.58 (0.26)	2.31 (0.94)	0.05 (0.03)	46.36 (5.94)	-
LGBM [118]	1.64 (0.93)	6.67 (4.19)	0.12 (0.03)	43.18 (4.31)	-
MLP [222]	0.26 (0.07)	1.06 (0.31)	**0.02 (0.02)**	48.86 (6.45)	-
CNN [223]	0.25 (0.05)	1.00 (0.21)	**0.02 (0.02)**	47.73 (5.93)	-
RNN [224]	2.16 (1.36)	8.80 (5.99)	0.15 (0.06)	42.50 (5.09)	-
LSTM [225]	1.06 (0.82)	4.38 (3.68)	0.07 (0.04)	41.82 (5.40)	-
GRU [226]	0.49 (0.32)	1.94 (1.20)	0.04 (0.03)	47.73 (9.59)	-
WaveNet [227]	**0.24 (0.04)**	**0.95 (0.17)**	**0.02 (0.01)**	47.95 (5.79)	-
eTS [68]	**0.29 (0.09)**	**1.18 (0.39)**	**0.03 (0.03)**	49.09 (5.77)	5 (1)
Simpl_eTS [69]	0.37 (0.18)	1.51 (0.69)	**0.03 (0.03)**	47.73 (4.87)	56 (38)
exTS [70]	0.29 (0.09)	1.17 (0.38)	**0.03 (0.03)**	50.45 (6.17)	5 (0)
ePL [72]	0.44 (0.17)	1.78 (0.66)	0.04 (0.02)	51.36 (7.55)	1 (0)
eMG [74]	0.53 (0.28)	2.12 (0.98)	0.04 (0.03)	51.14 (6.68)	20 (5)
ePL+ [73]	1.19 (2.61)	4.54 (9.60)	0.08 (0.14)	50.68 (5.75)	1 (0)
ePL-KRLS-DISCO [75]	0.61 (0.39)	2.40 (1.41)	0.05 (0.03)	50.23 (3.86)	31 (5)
NMR	1.20 (0.51)	4.86 (2.25)	0.09 (0.01)	52.95 (6.02)	3 (0)
NTSK (RLS)	**0.27 (0.08)**	**1.10 (0.28)**	**0.02 (0.02)**	49.32 (4.88)	1 (0)
NTSK (wRLS)	0.29 (0.08)	1.17 (0.34)	0.03 (0.03)	48.41 (4.07)	6 (0)
GEN-NMR	1.60 (0.99)	6.43 (3.85)	0.12 (0.08)	54.55 (6.10)	15 (1)
GEN-NTSK (RLS)	0.28 (0.07)	1.13 (0.28)	**0.02 (0.02)**	50.45 (4.64)	1 (0)
GEN-NTSK (wRLS)	0.30 (0.10)	1.21 (0.43)	0.03 (0.03)	48.64 (4.45)	7 (0)
R-NMR	0.86 (0.32)	3.49 (1.51)	0.06 (0.01)	51.59 (6.59)	-
R-NTSK	0.30 (0.08)	1.21 (0.35)	0.03 (0.03)	48.86 (4.69)	-
RF-NTSK	0.34 (0.10)	1.38 (0.39)	0.03 (0.03)	48.64 (5.86)	-

Table 38 – Statistical MDM test for S&P 500 (horizon = 5)

Model	NMR	NTSK (RLS)	NTSK (wRLS)	GEN-NMR	GEN-NTSK (RLS)	GEN-NTSK (wRLS)	R-NMR	R-NTSK	RF-NTSK
KNN [218]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
Regression Tree [219]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
Random Forest [112]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
SVM [220]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LS-SVM [221]	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
GBM [116]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
XGBoost [117]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
LGBM [118]	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
MLP [222]	1.00	0.01	1.00	1.00	0.25	1.00	1.00	1.00	1.00
CNN [223]	1.00	0.96	1.00	1.00	1.00	1.00	1.00	1.00	1.00
RNN [224]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LSTM [225]	0.96	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
GRU [226]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
WaveNet [227]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
eTS [68]	1.00	0.00	0.00	1.00	0.00	1.00	1.00	0.02	0.81
Simpl_eTS [69]	1.00	0.00	0.00	1.00	0.00	0.11	1.00	0.00	0.00
exTS [70]	1.00	0.00	0.00	1.00	0.00	0.99	1.00	0.34	1.00
ePL [72]	1.00	0.00	0.00	1.00	0.00	0.01	1.00	0.00	0.00
eMG [74]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
ePL+ [73]	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00
ePL-KRLS-DISCO [75]	1.00	0.00	0.00	1.00	0.00	0.00	0.75	0.00	0.00

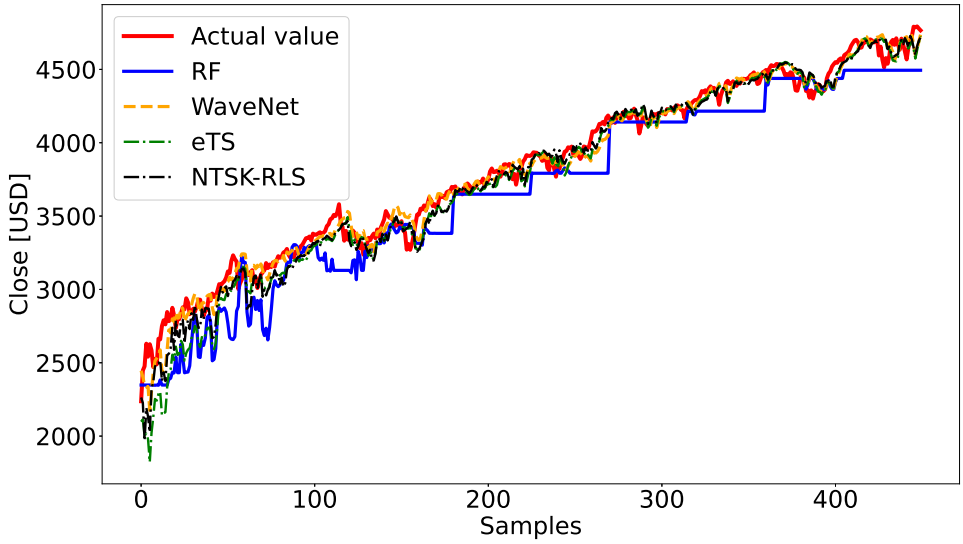


Figure 23 – Prediction performance on S&P 500 dataset (horizon = 5)

Table 39 presents the simulation results for predicting the NASDAQ one step ahead. RT and RF obtained the lowest mean errors among the classical models, while MLP performed best among the DL models. For the eFSs, eTS recorded the lowest errors; among the proposed fuzzy models, NTSK (RLS), NTSK (wRLS), GEN-NTSK (RLS), GEN-NTSK (wRLS), and R-NTSK yielded the lowest errors. Simpl_eTS exhibited the highest mean and standard deviation for the number of rules. Conversely, SVM, RNN, LSTM, and NMR exhibited higher errors. GEN-NMR achieved the best mean CPPM. Table 40 confirms that the NTSK-based models, especially R-NTSK, achieved lower errors than many of the comparison models.

Table 39 – Simulations' results of the NASDAQ series (horizon = 1)

Model	NRMSE	NDEI	MAPE	CPPM (%)	Rules
KNN [218]	0.34 (0.21)	1.33 (0.88)	0.03 (0.02)	45.68 (9.06)	-
Regression Tree [219]	**0.34 (0.19)**	**1.33 (0.81)**	**0.03 (0.01)**	48.86 (9.27)	-
Random Forest [112]	0.34 (0.21)	1.33 (0.92)	**0.03 (0.01)**	45.91 (8.31)	-
SVM [220]	2.07 (1.13)	8.13 (4.73)	0.19 (0.05)	45.00 (7.66)	-
LS-SVM [221]	1.68 (0.86)	6.59 (3.66)	0.15 (0.04)	48.64 (7.89)	-
GBM [116]	1.38 (0.76)	5.43 (3.21)	0.12 (0.03)	46.36 (6.36)	-
XGBoost [117]	0.35 (0.19)	1.39 (0.83)	0.03 (0.02)	44.55 (5.68)	-
LGBM [118]	0.42 (0.21)	1.65 (0.88)	0.04 (0.02)	42.05 (8.27)	-
MLP [222]	**0.12 (0.03)**	**0.48 (0.14)**	**0.01 (0.00)**	48.41 (9.54)	-
CNN [223]	0.15 (0.02)	0.59 (0.11)	0.01 (0.01)	47.05 (7.94)	-
RNN [224]	2.13 (1.16)	8.34 (4.84)	0.19 (0.05)	43.18 (4.55)	-
LSTM [225]	1.55 (1.18)	6.14 (5.02)	0.15 (0.08)	46.14 (8.32)	-
GRU [226]	1.01 (0.60)	3.95 (2.51)	0.09 (0.04)	40.91 (6.74)	-
WaveNet [227]	0.14 (0.03)	0.54 (0.14)	0.01 (0.01)	48.18 (5.64)	-
eTS [68]	**0.12 (0.02)**	**0.48 (0.11)**	**0.01 (0.00)**	48.18 (8.79)	4 (0)
Simpl_eTS [69]	0.15 (0.03)	0.57 (0.14)	0.01 (0.01)	49.09 (9.97)	47 (24)
exTS [70]	0.13 (0.03)	0.51 (0.11)	0.01 (0.01)	49.09 (9.71)	4 (0)
ePL [72]	0.30 (0.40)	1.15 (1.52)	0.05 (0.11)	49.09 (8.39)	1 (0)
eMG [74]	0.36 (0.24)	1.43 (1.05)	0.03 (0.02)	48.18 (5.06)	5 (1)
ePL+ [73]	0.16 (0.05)	0.64 (0.22)	0.02 (0.01)	46.59 (6.20)	3 (1)
ePL-KRLS-DISCO [75]	0.98 (0.51)	3.75 (1.88)	0.06 (0.04)	49.09 (9.44)	32 (3)
NMR	1.36 (0.66)	5.34 (2.79)	0.12 (0.03)	51.14 (8.27)	2 (0)
NTSK (RLS)	**0.13 (0.03)**	0.52 (0.14)	0.01 (0.01)	47.27 (7.32)	1 (0)
NTSK (wRLS)	**0.13 (0.03)**	**0.49 (0.11)**	**0.01 (0.00)**	47.95 (8.71)	14 (1)
GEN-NMR	0.91 (0.48)	3.58 (2.02)	0.08 (0.03)	51.36 (8.21)	3 (0)
GEN-NTSK (RLS)	0.14 (0.03)	0.55 (0.15)	**0.01 (0.00)**	44.77 (5.84)	1 (0)
GEN-NTSK (wRLS)	**0.13 (0.03)**	0.51 (0.12)	**0.01 (0.00)**	48.86 (7.69)	16 (1)
R-NMR	0.80 (0.47)	3.10 (1.86)	0.07 (0.04)	47.27 (5.73)	-
R-NTSK	**0.13 (0.03)**	**0.49 (0.11)**	**0.01 (0.00)**	47.50 (8.16)	-
RF-NTSK	0.17 (0.06)	0.66 (0.26)	0.02 (0.01)	49.09 (7.20)	-

Table 40 – Statistical MDM test for NASDAQ (horizon = 1)

Model	NMR	NTSK (RLS)	NTSK (wRLS)	GEN-NMR	GEN-NTSK (RLS)	GEN-NTSK (wRLS)	R-NMR	R-NTSK	RF-NTSK
KNN [218]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
Regression Tree [219]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
Random Forest [112]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
SVM [220]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LS-SVM [221]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
GBM [116]	0.45	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
XGBoost [117]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
LGBM [118]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
MLP [222]	1.00	1.00	0.97	1.00	1.00	1.00	1.00	0.99	1.00
CNN [223]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.84
RNN [224]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LSTM [225]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
GRU [226]	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
WaveNet [227]	1.00	0.09	0.00	1.00	0.50	0.01	1.00	0.00	1.00
eTS [68]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Simpl_eTS [69]	1.00	0.00	0.00	1.00	0.04	0.00	1.00	0.00	1.00
exTS [70]	1.00	0.47	0.00	1.00	0.98	0.03	1.00	0.00	1.00
ePL [72]	1.00	0.00	0.00	0.50	0.00	0.00	0.52	0.00	0.00
eMG [74]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
ePL+ [73]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.07
ePL-KRLS-DISCO [75]	0.94	0.00	0.00	0.04	0.00	0.00	0.04	0.00	0.00

Table 41 presents the simulations' results for the NASDAQ to predict five steps ahead. KNN, XGBoost, and LGBM obtained the lowest mean errors among the classical models, MLP, CNN, and WaveNet the lowest among DL models, eTS and exTS the lowest for the eFSs, and NTSK (RLS), NTSK (wRLS), GEN-NTSK (RLS), GEN-NTSK (wRLS) and R-NTSK, the lowest errors for the proposed fuzzy models. Simpl_eTS had the highest number of rules. SVM, RNN, GRU, and ePL-KRLS-DISCO performed higher errors. NMR performed the best mean for CPPM. Table 42 supports that the NTSK models perform lower errors than many of the compared models. Figure 24 shows the predictions for NASDAQ five steps ahead.

Table 41 – Simulations' results of the NASDAQ series (horizon = 5)

Model	NRMSE	NDEI	MAPE	CPPM (%)	Rules
KNN [218]	**0.50 (0.28)**	**1.94 (1.17)**	**0.05 (0.02)**	47.95 (7.29)	-
Regression Tree [219]	0.57 (0.29)	2.24 (1.28)	0.05 (0.03)	47.05 (5.75)	-
Random Forest [112]	0.52 (0.26)	2.02 (1.10)	0.05 (0.03)	46.36 (7.06)	-
SVM [220]	2.05 (1.11)	8.04 (4.65)	0.19 (0.04)	52.05 (3.44)	-
LS-SVM [221]	0.81 (0.31)	3.18 (1.37)	0.08 (0.02)	51.36 (4.79)	-
GBM [116]	0.52 (0.23)	2.05 (0.94)	0.05 (0.03)	50.91 (7.41)	-
XGBoost [117]	0.53 (0.22)	2.09 (0.95)	**0.05 (0.02)**	45.91 (4.75)	-
LGBM [118]	0.51 (0.26)	2.00 (1.10)	**0.05 (0.02)**	42.05 (9.05)	-
MLP [222]	0.26 (0.06)	1.01 (0.26)	**0.02 (0.01)**	52.05 (3.44)	-
CNN [223]	0.26 (0.06)	1.00 (0.29)	**0.02 (0.01)**	51.59 (4.55)	-
RNN [224]	2.10 (1.16)	8.24 (4.83)	0.19 (0.05)	40.91 (6.74)	-
LSTM [225]	0.78 (0.44)	3.01 (1.67)	0.09 (0.06)	51.59 (6.43)	-
GRU [226]	4.00 (1.98)	15.69 (8.33)	0.37 (0.06)	40.91 (6.74)	-
WaveNet [227]	**0.24 (0.06)**	**0.95 (0.26)**	**0.02 (0.01)**	52.05 (4.71)	-
eTS [68]	**0.30 (0.09)**	**1.15 (0.30)**	**0.03 (0.02)**	52.05 (5.33)	4 (0)
Simpl_eTS [69]	0.36 (0.11)	1.40 (0.46)	0.04 (0.02)	51.14 (3.97)	72 (51)
exTS [70]	0.27 (0.05)	1.05 (0.22)	**0.03 (0.02)**	51.14 (4.22)	3 (0)
ePL [72]	0.34 (0.05)	1.34 (0.24)	0.04 (0.03)	51.14 (4.57)	1 (0)
eMG [74]	0.51 (0.26)	1.99 (1.01)	0.05 (0.04)	47.73 (5.93)	8 (2)
ePL+ [73]	0.78 (1.24)	2.97 (4.62)	0.08 (0.14)	51.59 (6.75)	1 (0)
ePL-KRLS-DISCO [75]	3.83 (3.03)	14.67 (10.87)	0.26 (0.29)	50.00 (8.86)	20 (0)
NMR	0.84 (0.43)	3.27 (1.74)	0.08 (0.03)	52.95 (6.97)	4 (0)
NTSK (RLS)	0.30 (0.07)	1.16 (0.28)	**0.03 (0.02)**	51.59 (7.05)	1 (0)
NTSK (wRLS)	**0.28 (0.06)**	**1.09 (0.21)**	**0.03 (0.02)**	51.59 (4.55)	8 (0)
GEN-NMR	0.83 (0.60)	3.20 (2.28)	0.08 (0.06)	49.55 (6.88)	15 (1)
GEN-NTSK (RLS)	0.29 (0.06)	1.13 (0.26)	**0.03 (0.02)**	50.00 (5.57)	1 (0)
GEN-NTSK (wRLS)	0.28 (0.05)	1.09 (0.21)	**0.03 (0.02)**	52.50 (4.60)	1 (0)
R-NMR	0.81 (0.44)	3.12 (1.72)	0.07 (0.04)	47.27 (6.25)	-
R-NTSK	0.29 (0.06)	1.11 (0.21)	**0.03 (0.02)**	51.82 (4.64)	-
RF-NTSK	0.34 (0.10)	1.33 (0.41)	0.04 (0.02)	52.27 (5.47)	-

Table 42 – Statistical MDM test for NASDAQ (horizon = 5)

Model	NMR	NTSK (RLS)	NTSK (wRLS)	GEN-NMR	GEN-NTSK (RLS)	GEN-NTSK (wRLS)	R-NMR	R-NTSK	RF-NTSK
KNN [218]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
Regression Tree [219]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
Random Forest [112]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
SVM [220]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LS-SVM [221]	1.00	0.00	0.00	1.00	0.00	0.00	0.99	0.00	0.00
GBM [116]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
XGBoost [117]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
LGBM [118]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
MLP [222]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
CNN [223]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
RNN [224]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LSTM [225]	0.00	0.00	0.00	0.96	0.00	0.00	0.00	0.00	0.00
GRU [226]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
WaveNet [227]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
eTS [68]	1.00	0.00	0.00	1.00	0.01	0.00	1.00	0.00	0.98
Simpl_eTS [69]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
exTS [70]	1.00	0.96	0.69	1.00	1.00	0.63	1.00	0.93	1.00
ePL [72]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.02
eMG [74]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
ePL+ [73]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ePL-KRLS-DISCO [75]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

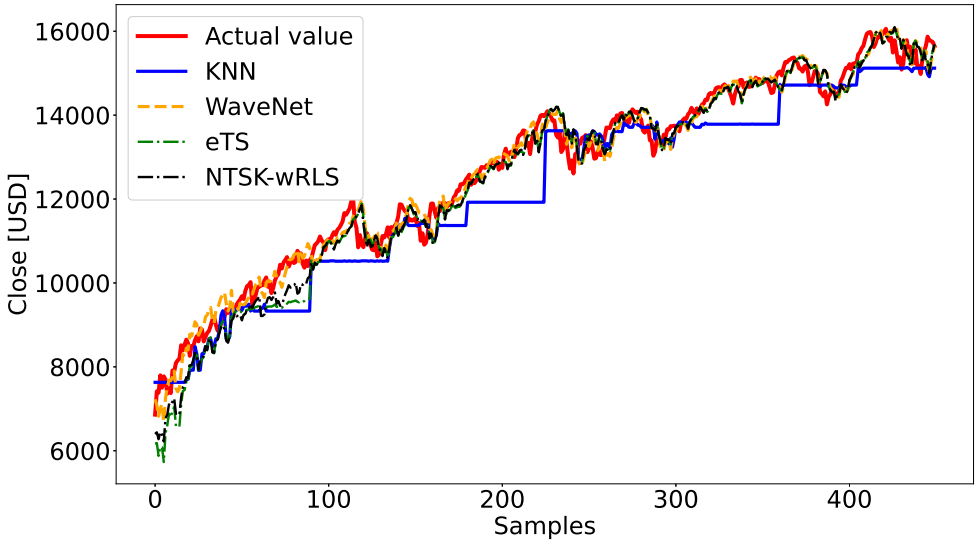


Figure 24 – Prediction performance on NASDAQ dataset (horizon = 5)

Table 43 presents the simulation results for predicting the TAIEX one step ahead. KNN, RT, and XGBoost obtained the lowest mean errors among the classical models, while MLP performed best among the DL models. For the eFSs, eTS, Simpl_eTS, exTS, and ePL recorded the lowest errors; among the proposed models, NTSK (RLS), NTSK (wRLS), GEN-NTSK (RLS), GEN-NTSK (wRLS), R-NTSK, and RF-NTSK yielded the lowest errors. Simpl_eTS generated the highest number of rules. Conversely, SVM, GBM, LGBM, RNN, LSTM, and ePL+ exhibited higher errors. R-NMR achieved the best mean CPPM. Table 44 confirms that the NTSK models achieved lower errors than many of the comparison models.

Table 43 – Simulations' results of the TAIEX (horizon = 1)

Model	NRMSE	NDEI	MAPE	CPPM (%)	Rules
KNN [218]	**0.27 (0.15)**	1.00 (0.51)	0.03 (0.02)	49.77 (10.51)	-
Regression Tree [219]	**0.27 (0.15)**	**0.97 (0.50)**	**0.02 (0.02)**	50.00 (8.96)	-
Random Forest [112]	0.28 (0.15)	1.02 (0.49)	0.03 (0.02)	52.79 (11.11)	-
SVM [220]	1.93 (0.99)	7.34 (3.86)	0.18 (0.07)	50.70 (5.68)	-
LS-SVM [221]	0.30 (0.12)	1.15 (0.52)	0.03 (0.01)	51.63 (5.96)	-
GBM [116]	1.15 (0.51)	4.39 (2.10)	0.11 (0.04)	50.70 (9.46)	-
XGBoost [117]	**0.27 (0.15)**	1.00 (0.49)	0.03 (0.02)	50.23 (8.33)	-
LGBM [118]	1.28 (0.52)	4.88 (2.22)	0.12 (0.04)	45.35 (8.84)	-
MLP [222]	**0.11 (0.04)**	**0.43 (0.18)**	**0.01 (0.00)**	50.93 (6.10)	-
CNN [223]	0.17 (0.05)	0.64 (0.23)	0.02 (0.01)	52.33 (7.37)	-
RNN [224]	1.77 (0.86)	6.77 (3.54)	0.16 (0.07)	43.72 (8.24)	-
LSTM [225]	3.27 (1.44)	12.56 (6.30)	0.31 (0.08)	43.72 (8.24)	-
GRU [226]	0.60 (0.51)	2.30 (2.04)	0.06 (0.05)	44.88 (8.45)	-
WaveNet [227]	0.17 (0.07)	0.64 (0.32)	0.02 (0.01)	52.56 (7.73)	-
eTS [68]	**0.11 (0.04)**	**0.43 (0.18)**	**0.01 (0.01)**	51.63 (8.24)	5 (0)
Simpl_eTS [69]	0.14 (0.04)	0.53 (0.17)	**0.01 (0.01)**	52.33 (5.81)	81 (70)
exTS [70]	0.13 (0.06)	0.50 (0.24)	**0.01 (0.01)**	53.02 (9.11)	4 (0)
ePL [72]	0.13 (0.05)	0.51 (0.23)	**0.01 (0.01)**	51.63 (7.11)	1 (0)
eMG [74]	0.27 (0.27)	0.97 (0.85)	0.03 (0.04)	52.79 (10.09)	7 (3)
ePL+ [73]	1.06 (1.73)	4.11 (6.69)	0.12 (0.19)	52.33 (7.22)	1 (0)
ePL-KRLS-DISCO [75]	0.61 (0.65)	2.15 (2.04)	0.04 (0.05)	50.23 (5.81)	13 (7)
NMR	0.96 (0.44)	3.67 (1.88)	0.09 (0.04)	53.02 (8.44)	2 (0)
NTSK (RLS)	**0.11 (0.04)**	0.44 (0.17)	**0.01 (0.01)**	53.02 (7.34)	1 (0)
NTSK (wRLS)	**0.11 (0.04)**	0.44 (0.18)	**0.01 (0.01)**	51.63 (8.56)	4 (0)
GEN-NMR	0.71 (0.36)	2.71 (1.48)	0.07 (0.04)	54.65 (9.60)	3 (0)
GEN-NTSK (RLS)	0.12 (0.04)	0.48 (0.21)	**0.01 (0.01)**	54.42 (9.08)	1 (0)
GEN-NTSK (wRLS)	**0.11 (0.04)**	**0.43 (0.18)**	**0.01 (0.01)**	50.93 (8.67)	3 (0)
R-NMR	0.91 (0.91)	3.58 (4.14)	0.07 (0.05)	54.88 (9.08)	-
R-NTSK	**0.11 (0.04)**	**0.43 (0.18)**	**0.01 (0.01)**	52.79 (7.50)	-
RF-NTSK	0.15 (0.05)	0.57 (0.18)	**0.01 (0.01)**	52.79 (6.74)	-

Table 44 – Statistical MDM test for TAIEX (horizon = 1)

Model	NMR	NTSK (RLS)	NTSK (wRLS)	GEN-NMR	GEN-NTSK (RLS)	GEN-NTSK (wRLS)	R-NMR	R-NTSK	RF-NTSK
KNN [218]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
Regression Tree [219]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
Random Forest [112]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
SVM [220]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LS-SVM [221]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
GBM [116]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
XGBoost [117]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
LGBM [118]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
MLP [222]	1.00	0.99	1.00	1.00	1.00	0.98	1.00	0.98	1.00
CNN [223]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.03
RNN [224]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LSTM [225]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
GRU [226]	1.00	0.00	0.00	0.01	0.00	0.00	0.54	0.00	0.00
WaveNet [227]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.16
eTS [68]	1.00	0.96	1.00	1.00	1.00	0.99	1.00	0.90	1.00
Simpl_eTS [69]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	1.00
exTS [70]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.99
ePL [72]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	1.00
eMG [74]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
ePL+ [73]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ePL-KRLS-DISCO [75]	0.36	0.03	0.03	0.17	0.03	0.03	0.24	0.03	0.03

Table 45 presents the simulation results for predicting the TAIEX five steps ahead. KNN, RT, RF, LS-SVM, and XGBoost obtained the lowest mean errors among the classical models, while MLP, CNN, and WaveNet performed best among the DL models. For the eFSs, exTS recorded the lowest errors; among the proposed models, GEN-NTSK (RLS) and GEN-NTSK (wRLS) yielded the lowest errors. Simpl_eTS generated the highest number of rules. Conversely, SVM, LightGBM, and RNN exhibited higher errors. The ePL-KRLS-DISCO model achieved the best mean CPPM. Table 46 confirms that the NTSK models achieved lower errors than many of the comparison models. Figure 25 shows the predictions for the TAIEX five steps ahead.

Table 45 – Simulations' results of the TAIEX (horizon = 5)

Model	NRMSE	NDEI	MAPE	CPPM (%)	Rules
KNN [218]	0.47 (0.23)	1.77 (0.94)	**0.05 (0.04)**	46.67 (8.40)	-
Regression Tree [219]	0.52 (0.28)	1.94 (1.14)	**0.05 (0.04)**	45.48 (5.88)	-
Random Forest [112]	**0.45 (0.23)**	**1.70 (0.94)**	**0.05 (0.04)**	47.38 (5.68)	-
SVM [220]	1.88 (0.96)	7.12 (3.73)	0.18 (0.07)	48.33 (5.54)	-
LS-SVM [221]	1.23 (0.45)	4.67 (1.95)	0.12 (0.05)	47.62 (9.93)	-
GBM [116]	0.85 (0.23)	3.22 (0.98)	0.08 (0.02)	45.95 (6.39)	-
XGBoost [117]	0.48 (0.26)	1.82 (1.06)	**0.05 (0.04)**	46.67 (8.19)	-
LGBM [118]	1.18 (0.48)	4.49 (2.02)	0.11 (0.04)	45.24 (8.72)	-
MLP [222]	0.23 (0.06)	0.89 (0.29)	**0.02 (0.01)**	48.33 (5.54)	-
CNN [223]	0.27 (0.09)	1.02 (0.43)	**0.02 (0.01)**	48.33 (4.52)	-
RNN [224]	1.73 (0.82)	6.58 (3.39)	0.17 (0.07)	44.05 (8.40)	-
LSTM [225]	0.43 (0.27)	1.61 (1.07)	0.04 (0.03)	43.33 (7.05)	-
GRU [226]	0.59 (0.38)	2.29 (1.71)	0.05 (0.03)	44.05 (8.40)	-
WaveNet [227]	**0.22 (0.08)**	**0.86 (0.38)**	**0.02 (0.01)**	48.33 (4.13)	-
eTS [68]	0.26 (0.07)	0.99 (0.34)	**0.02 (0.01)**	50.48 (6.10)	5 (0)
Simpl_eTS [69]	0.28 (0.09)	1.08 (0.41)	0.03 (0.01)	48.57 (6.58)	55 (38)
exTS [70]	**0.25 (0.06)**	**0.96 (0.26)**	**0.02 (0.01)**	50.24 (6.34)	3 (0)
ePL [72]	0.34 (0.14)	1.31 (0.60)	0.03 (0.02)	47.86 (6.25)	1 (0)
eMG [74]	0.51 (0.33)	1.92 (1.30)	0.05 (0.04)	50.24 (4.82)	13 (6)
ePL+ [73]	0.33 (0.15)	1.28 (0.67)	0.03 (0.02)	49.76 (5.05)	3 (1)
ePL-KRLS-DISCO [75]	0.68 (0.27)	2.53 (0.94)	0.06 (0.04)	54.05 (4.65)	30 (5)
NMR	0.69 (0.22)	2.62 (0.96)	0.07 (0.03)	52.38 (6.73)	4 (0)
NTSK (RLS)	0.27 (0.08)	1.02 (0.38)	0.02 (0.01)	51.90 (5.91)	1 (0)
NTSK (wRLS)	0.26 (0.06)	0.98 (0.31)	0.02 (0.01)	50.71 (6.21)	13 (0)
GEN-NMR	0.79 (0.35)	2.98 (1.37)	0.07 (0.04)	51.19 (8.13)	5 (0)
GEN-NTSK (RLS)	0.26 (0.08)	1.01 (0.40)	**0.02 (0.00)**	48.33 (4.40)	1 (0)
GEN-NTSK (wRLS)	**0.25 (0.06)**	**0.96 (0.29)**	0.02 (0.01)	51.67 (7.15)	13 (0)
R-NMR	0.75 (0.31)	2.79 (1.09)	0.07 (0.04)	48.81 (5.76)	-
R-NTSK	0.26 (0.08)	1.01 (0.36)	0.02 (0.01)	50.24 (6.25)	-
RF-NTSK	0.33 (0.12)	1.24 (0.54)	0.03 (0.02)	51.19 (6.14)	-

Table 46 – Statistical MDM test for TAIEX (horizon = 5)

Model	NMR	NTSK (RLS)	NTSK (wRLS)	GEN-NMR	GEN-NTSK (RLS)	GEN-NTSK (wRLS)	R-NMR	R-NTSK	RF-NTSK
KNN [218]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
Regression Tree [219]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
Random Forest [112]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
SVM [220]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LS-SVM [221]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
GBM [116]	0.00	0.00	0.00	0.36	0.00	0.00	0.37	0.00	0.00
XGBoost [117]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
LGBM [118]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
MLP [222]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
CNN [223]	1.00	0.53	0.19	1.00	0.19	0.10	1.00	0.48	1.00
RNN [224]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LSTM [225]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
GRU [226]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
WaveNet [227]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
eTS [68]	1.00	0.97	0.71	1.00	0.74	0.33	1.00	0.99	1.00
Simpl_eTS [69]	1.00	0.03	0.00	1.00	0.00	0.00	1.00	0.00	1.00
exTS [70]	1.00	0.75	0.24	1.00	0.24	0.07	1.00	0.74	1.00
ePL [72]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.01
eMG [74]	0.99	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
ePL+ [73]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.03
ePL-KRLS-DISCO [75]	0.06	0.00	0.00	0.91	0.00	0.00	0.93	0.00	0.00

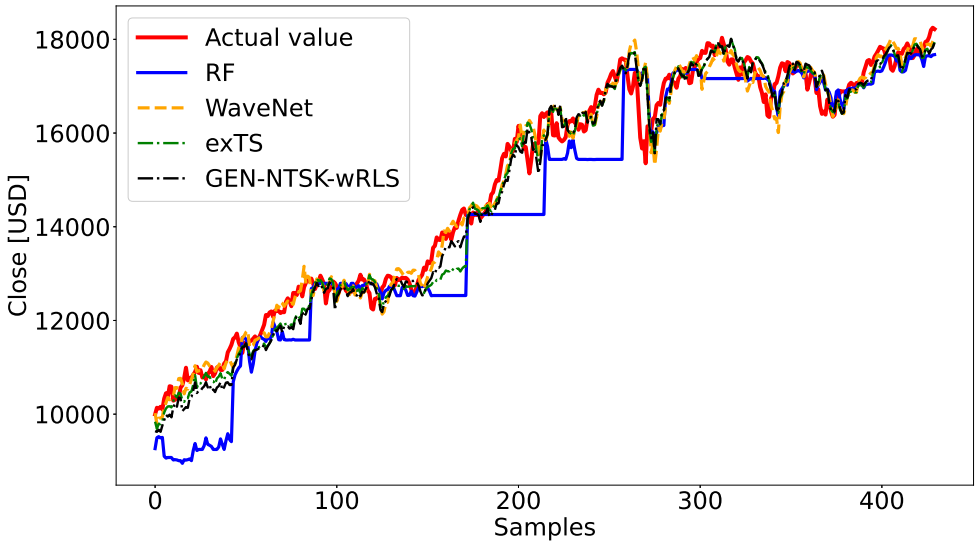


Figure 25 – Prediction performance on TAIEX dataset (horizon = 5)

6.1.4 CRYPTOCURRENCIES

Table 47 presents the simulation results for predicting Bitcoin one step ahead. XGBoost obtained the lowest mean errors among the classical models, while WaveNet performed best among the DL models. For the eFSs, eTS recorded the lowest errors, and R-NTSK yielded the best results among the proposed models. Notably, the proposed R-NTSK achieved the lowest errors among all simulations. Conversely, SVM, RNN, GRU, and ePL exhibited higher errors. KNN achieved the best mean for CPPM. Table 48 confirms that the NTSK-based models achieved lower errors than many of the comparison models.

Table 47 – Simulations' results of Bitcoin (horizon = 1)

Model	NRMSE	NDEI	MAPE	CPPM (%)	Rules
KNN [218]	0.32 (0.22)	1.21 (0.89)	0.13 (0.11)	51.69 (5.25)	-
Regression Tree [219]	0.35 (0.23)	1.37 (0.96)	0.14 (0.10)	46.00 (6.05)	-
Random Forest [112]	0.38 (0.23)	1.41 (0.94)	0.15 (0.10)	48.77 (2.29)	-
SVM [220]	1.11 (0.50)	4.09 (1.90)	0.51 (0.26)	51.08 (6.15)	-
LS-SVM [221]	0.67 (0.33)	2.60 (1.56)	0.29 (0.17)	47.85 (7.68)	-
GBM [116]	0.70 (0.29)	2.65 (1.27)	0.30 (0.15)	47.85 (3.86)	-
XGBoost [117]	**0.28 (0.20)**	**1.06 (0.79)**	**0.11 (0.10)**	51.23 (6.67)	-
LGBM [118]	0.71 (0.29)	2.65 (1.24)	0.31 (0.15)	46.31 (6.02)	-
MLP [222]	0.20 (0.18)	0.86 (0.95)	0.06 (0.04)	46.31 (6.76)	-
CNN [223]	0.23 (0.20)	0.92 (0.86)	0.08 (0.06)	47.69 (6.34)	-
RNN [224]	1.05 (0.53)	3.92 (2.04)	0.45 (0.24)	46.62 (6.23)	-
LSTM [225]	0.46 (0.34)	1.81 (1.50)	0.18 (0.17)	42.46 (4.47)	-
GRU [226]	1.72 (0.87)	6.96 (5.24)	0.68 (0.13)	46.31 (6.17)	-
WaveNet [227]	**0.15 (0.11)**	**0.59 (0.49)**	**0.05 (0.04)**	47.23 (5.80)	-
eTS [68]	**0.11 (0.03)**	**0.43 (0.21)**	**0.04 (0.01)**	45.38 (7.70)	4 (0)
Simpl_eTS [69]	0.15 (0.07)	0.61 (0.39)	0.06 (0.03)	48.77 (7.05)	33 (19)
exTS [70]	0.13 (0.05)	0.50 (0.27)	0.05 (0.03)	47.38 (7.87)	7 (3)
ePL [72]	1.10 (1.00)	3.97 (3.60)	0.52 (0.48)	46.62 (10.30)	2 (0)
eMG [74]	0.34 (0.21)	1.30 (0.86)	0.13 (0.10)	49.38 (4.54)	48 (37)
ePL+ [73]	0.18 (0.12)	0.68 (0.43)	0.07 (0.07)	46.62 (6.78)	5 (3)
ePL-KRLS-DISCO [75]	0.80 (1.24)	2.80 (4.00)	0.30 (0.38)	48.46 (8.06)	15 (7)
NMR	0.55 (0.54)	2.11 (2.25)	0.20 (0.19)	45.38 (6.75)	18 (2)
NTSK (RLS)	0.10 (0.04)	0.39 (0.19)	0.04 (0.02)	46.62 (9.36)	1 (0)
NTSK (wRLS)	0.09 (0.03)	0.37 (0.17)	0.03 (0.02)	48.00 (9.40)	1 (0)
GEN-NMR	0.77 (0.39)	3.03 (1.88)	0.31 (0.16)	49.85 (4.36)	18 (2)
GEN-NTSK (RLS)	0.10 (0.03)	0.39 (0.17)	0.04 (0.02)	43.69 (9.13)	1 (0)
GEN-NTSK (wRLS)	0.09 (0.03)	0.37 (0.17)	0.03 (0.02)	45.85 (10.15)	1 (0)
R-NMR	0.50 (0.48)	1.89 (1.95)	0.18 (0.17)	46.46 (5.54)	-
R-NTSK	**0.09 (0.02)**	**0.35 (0.16)**	**0.03 (0.01)**	46.15 (7.72)	-
RF-NTSK	0.13 (0.05)	0.48 (0.22)	0.05 (0.02)	46.46 (7.96)	-

Table 48 – Statistical MDM test for Bitcoin (horizon = 1)

Model	NMR	NTSK (RLS)	NTSK (wRLS)	GEN-NMR	GEN-NTSK (RLS)	GEN-NTSK (wRLS)	R-NMR	R-NTSK	RF-NTSK
KNN [218]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
Regression Tree [219]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
Random Forest [112]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
SVM [220]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LS-SVM [221]	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
GBM [116]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
XGBoost [117]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
LGBM [118]	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00
MLP [222]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	1.00
CNN [223]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.97
RNN [224]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LSTM [225]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
GRU [226]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
WaveNet [227]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	1.00
eTS [68]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	1.00
Simpl_eTS [69]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.97
exTS [70]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	1.00
ePL [72]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
eMG [74]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
ePL+ [73]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
ePL-KRLS-DISCO [75]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

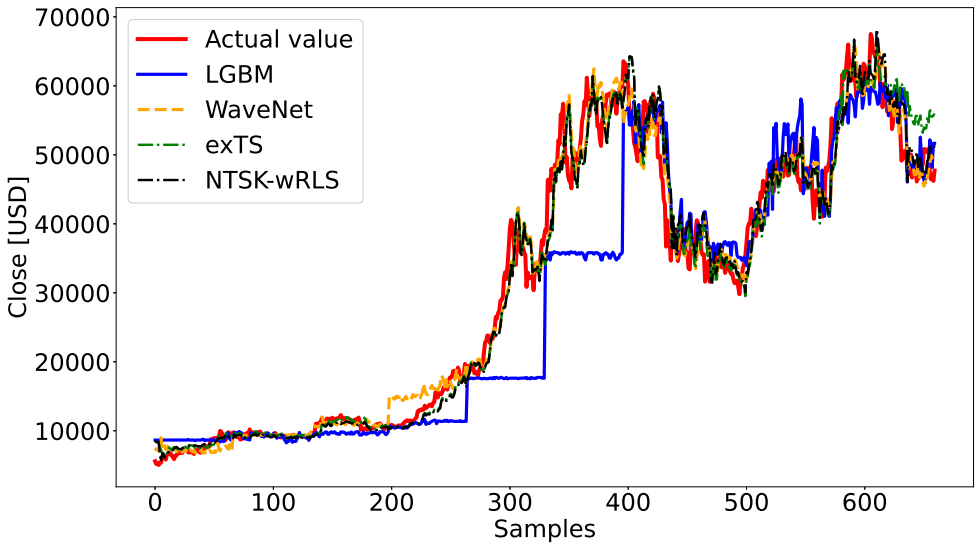


Figure 26 – Prediction performance on Bitcoin dataset (horizon = 5)

Table 49 presents the simulation results for predicting Bitcoin five steps ahead. XGBoost and LGBM obtained the lowest mean errors among the classical models, while WaveNet performed best among the DL models. For the eFSs, exTS recorded the lowest errors; among the proposed models, NTSK (wRLS), GEN-NTSK (RLS), and R-NTSK yielded the lowest errors. Simpl_eTS generated the highest number of rules. Conversely, SVM, GRU, and ePL-KRLS-DISCO exhibited higher errors. The ePL+ approach obtained higher errors, it achieved the best mean CPPM. Table 50 confirms that the NTSK-based models achieved lower errors than many of the comparison models. Figure 26 shows the predictions for Bitcoin five steps ahead.

Table 49 – Simulations' results of Bitcoin (horizon = 5)

Model	NRMSE	NDEI	MAPE	CPPM (%)	Rules
KNN [218]	0.41 (0.22)	1.57 (0.96)	0.16 (0.11)	48.62 (5.29)	-
Regression Tree [219]	0.43 (0.19)	1.62 (0.84)	0.17 (0.09)	46.77 (5.85)	-
Random Forest [112]	0.44 (0.22)	1.64 (0.91)	0.18 (0.11)	50.31 (5.29)	-
SVM [220]	1.10 (0.50)	4.05 (1.91)	0.51 (0.26)	52.00 (7.37)	-
LS-SVM [221]	0.96 (0.36)	3.61 (1.59)	0.42 (0.18)	51.23 (6.98)	-
GBM [116]	0.42 (0.23)	1.61 (0.97)	0.17 (0.11)	45.54 (4.57)	-
XGBoost [117]	0.40 (0.18)	1.55 (0.84)	**0.16 (0.09)**	49.23 (6.31)	-
LGBM [118]	**0.39 (0.21)**	**1.50 (0.88)**	0.16 (0.11)	50.15 (7.12)	-
MLP [222]	0.25 (0.13)	1.00 (0.69)	0.09 (0.04)	50.00 (5.93)	-
CNN [223]	0.24 (0.11)	0.94 (0.51)	0.09 (0.04)	52.31 (6.42)	-
RNN [224]	0.96 (0.39)	3.63 (1.69)	0.42 (0.20)	46.62 (6.00)	-
LSTM [225]	0.96 (0.39)	3.61 (1.68)	0.42 (0.20)	46.31 (6.17)	-
GRU [226]	1.25 (0.74)	5.06 (4.28)	0.47 (0.14)	45.38 (6.35)	-
WaveNet [227]	**0.20 (0.05)**	**0.74 (0.23)**	**0.08 (0.05)**	50.00 (7.67)	-
eTS [68]	0.24 (0.08)	0.92 (0.48)	0.09 (0.03)	50.77 (5.46)	4 (0)
Simpl_eTS [69]	0.34 (0.18)	1.35 (0.86)	0.13 (0.06)	48.31 (4.78)	31 (18)
exTS [70]	**0.19 (0.04)**	**0.74 (0.23)**	**0.07 (0.03)**	49.69 (6.27)	3 (0)
ePL [72]	0.55 (0.46)	2.17 (1.91)	0.25 (0.29)	49.85 (6.35)	1 (0)
eMG [74]	0.40 (0.15)	1.54 (0.72)	0.15 (0.06)	49.38 (6.05)	21 (16)
ePL+ [73]	0.42 (0.31)	1.64 (1.36)	0.16 (0.13)	53.54 (4.00)	2 (1)
ePL-KRLS-DISCO [75]	1.05 (1.01)	3.85 (3.24)	0.38 (0.48)	47.23 (3.77)	25 (9)
NMR	0.86 (0.34)	3.29 (1.60)	0.34 (0.14)	51.54 (5.34)	2 (0)
NTSK (RLS)	0.21 (0.03)	0.79 (0.23)	0.08 (0.03)	52.00 (5.45)	1 (0)
NTSK (wRLS)	**0.19 (0.04)**	0.74 (0.29)	**0.07 (0.02)**	50.62 (4.79)	10 (2)
GEN-NMR	0.71 (0.50)	2.70 (2.08)	0.26 (0.16)	51.54 (6.46)	18 (3)
GEN-NTSK (RLS)	**0.19 (0.04)**	0.74 (0.25)	0.07 (0.03)	50.62 (4.94)	1 (0)
GEN-NTSK (wRLS)	0.27 (0.12)	0.99 (0.41)	0.12 (0.10)	50.62 (5.26)	5 (0)
R-NMR	0.56 (0.46)	2.17 (1.92)	0.20 (0.14)	50.15 (5.02)	-
R-NTSK	0.19 (0.04)	**0.73 (0.27)**	0.07 (0.03)	50.15 (4.36)	-
RF-NTSK	0.23 (0.06)	0.89 (0.34)	0.09 (0.04)	51.23 (6.71)	-

Table 50 – Statistical MDM test for Bitcoin (horizon = 5)

Model	NMR	NTSK (RLS)	NTSK (wRLS)	GEN-NMR	GEN-NTSK (RLS)	GEN-NTSK (wRLS)	R-NMR	R-NTSK	RF-NTSK
KNN [218]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
Regression Tree [219]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
Random Forest [112]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
SVM [220]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LS-SVM [221]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
GBM [116]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
XGBoost [117]	1.00	0.00	0.00	1.00	0.00	0.02	1.00	0.00	0.00
LGBM [118]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
MLP [222]	1.00	1.00	0.18	1.00	0.62	1.00	1.00	0.09	1.00
CNN [223]	1.00	1.00	0.20	1.00	0.58	1.00	1.00	0.12	1.00
RNN [224]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LSTM [225]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
GRU [226]	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
WaveNet [227]	1.00	1.00	0.77	1.00	0.94	1.00	1.00	0.70	1.00
eTS [68]	1.00	0.06	0.00	1.00	0.00	1.00	1.00	0.00	0.39
Simpl_eTS [69]	1.00	0.00	0.00	1.00	0.00	1.00	1.00	0.00	0.00
exTS [70]	1.00	0.99	0.00	1.00	0.00	1.00	1.00	0.00	1.00
ePL [72]	1.00	0.00	0.00	1.00	0.00	0.00	0.05	0.00	0.00
eMG [74]	1.00	0.00	0.00	1.00	0.00	0.80	1.00	0.00	0.00
ePL+ [73]	1.00	0.00	0.00	1.00	0.00	0.00	0.04	0.00	0.00
ePL-KRLS-DISCO [75]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 51 presents the simulation results for predicting Ethereum one step ahead. RF and GBM obtained the lowest mean errors among the classical models, while CNN and WaveNet performed best among the DL models. For the eFSs, eTS recorded the lowest errors; among all models, NTSK (wRLS), GEN-NTSK (wRLS), and R-NTSK yielded the lowest errors. Simpl_eTS generated the highest number of rules. Conversely, SVM, RNN, LSTM, ePL, and ePL-KRLS-DISCO exhibited higher errors. Furthermore, ePL-KRLS-DISCO achieved the best mean CPPM. Table 52 confirms that R-NTSK achieved lower errors than all of the comparison models.

Table 51 – Simulations' results of Ethereum (horizon = 1)

Model	NRMSE	NDEI	MAPE	CPPM (%)	Rules
KNN [218]	0.44 (0.17)	1.77 (0.86)	0.21 (0.13)	49.08 (5.44)	-
Regression Tree [219]	0.37 (0.14)	1.51 (0.77)	0.16 (0.09)	47.23 (6.60)	-
Random Forest [112]	0.30 (0.17)	1.20 (0.80)	**0.13 (0.11)**	49.08 (5.94)	-
SVM [220]	1.34 (0.81)	5.28 (3.23)	0.64 (0.26)	48.92 (7.53)	-
LS-SVM [221]	0.85 (0.35)	3.38 (1.54)	0.43 (0.17)	49.38 (8.01)	-
GBM [116]	**0.29 (0.16)**	**1.17 (0.77)**	**0.13 (0.11)**	50.46 (4.66)	-
XGBoost [117]	0.31 (0.14)	1.24 (0.69)	0.14 (0.10)	49.38 (4.59)	-
LGBM [118]	0.36 (0.18)	1.44 (0.85)	0.16 (0.12)	47.85 (5.98)	-
MLP [222]	0.29 (0.26)	1.28 (1.35)	0.13 (0.13)	48.46 (5.30)	-
CNN [223]	**0.21 (0.18)**	**0.87 (0.90)**	0.09 (0.09)	47.08 (4.47)	-
RNN [224]	1.11 (0.65)	4.42 (2.66)	0.52 (0.21)	46.31 (7.30)	-
LSTM [225]	1.46 (0.57)	5.94 (2.91)	0.70 (0.15)	45.85 (8.50)	-
GRU [226]	0.47 (0.26)	1.93 (1.32)	0.24 (0.20)	49.08 (6.13)	-
WaveNet [227]	0.22 (0.14)	0.90 (0.71)	**0.09 (0.07)**	48.46 (5.11)	-
eTS [68]	**0.18 (0.10)**	**0.74 (0.54)**	**0.07 (0.03)**	48.46 (6.57)	6 (0)
Simpl_eTS [69]	0.37 (0.23)	1.51 (1.15)	0.19 (0.14)	49.08 (6.09)	46 (5)
exTS [70]	0.21 (0.16)	0.88 (0.81)	0.09 (0.07)	47.69 (6.19)	5 (0)
ePL [72]	1.67 (2.39)	6.39 (9.33)	0.67 (0.88)	45.38 (5.64)	2 (1)
eMG [74]	0.33 (0.19)	1.27 (0.75)	0.14 (0.11)	48.15 (8.14)	41 (36)
ePL+ [73]	0.37 (0.48)	1.47 (1.95)	0.14 (0.13)	47.38 (7.08)	1 (0)
ePL-KRLS-DISCO [75]	1.90 (3.57)	6.62 (11.69)	0.51 (0.72)	51.23 (6.00)	8 (7)
NMR	0.54 (0.31)	2.15 (1.39)	0.24 (0.18)	47.23 (7.76)	10 (0)
NTSK (RLS)	0.15 (0.07)	0.58 (0.23)	0.06 (0.04)	46.15 (4.67)	1 (0)
NTSK (wRLS)	**0.10 (0.02)**	**0.42 (0.14)**	**0.04 (0.01)**	47.38 (5.00)	10 (1)
GEN-NMR	0.63 (0.44)	2.55 (1.96)	0.30 (0.19)	49.23 (6.23)	3 (0)
GEN-NTSK (RLS)	0.20 (0.17)	0.85 (0.90)	0.07 (0.04)	46.62 (6.98)	1 (0)
GEN-NTSK (wRLS)	0.11 (0.03)	0.44 (0.16)	**0.04 (0.01)**	47.23 (5.29)	3 (0)
R-NMR	0.40 (0.26)	1.62 (1.21)	0.18 (0.16)	48.31 (5.72)	-
R-NTSK	0.11 (0.03)	0.44 (0.17)	**0.04 (0.01)**	46.15 (5.28)	-
RF-NTSK	0.14 (0.04)	0.57 (0.27)	0.06 (0.03)	46.62 (5.76)	-

Table 52 – Statistical MDM test for Ethereum (horizon = 1)

Model	NMR	NTSK (RLS)	NTSK (wRLS)	GEN-NMR	GEN-NTSK (RLS)	GEN-NTSK (wRLS)	R-NMR	R-NTSK	RF-NTSK
KNN [218]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
Regression Tree [219]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
Random Forest [112]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
SVM [220]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LS-SVM [221]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
GBM [116]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
XGBoost [117]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
LGBM [118]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
MLP [222]	1.00	0.03	0.00	1.00	0.06	0.00	1.00	0.00	0.84
CNN [223]	1.00	0.91	0.00	1.00	0.97	0.00	1.00	0.00	1.00
RNN [224]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LSTM [225]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
GRU [226]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
WaveNet [227]	1.00	0.38	0.00	1.00	0.60	0.00	1.00	0.00	0.98
eTS [68]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.01
Simpl_eTS [69]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
exTS [70]	1.00	1.00	0.00	1.00	1.00	0.00	1.00	0.00	1.00
ePL [72]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
eMG [74]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
ePL+ [73]	0.69	0.00	0.00	1.00	0.00	0.00	0.02	0.00	0.00
ePL-KRLS-DISCO [75]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

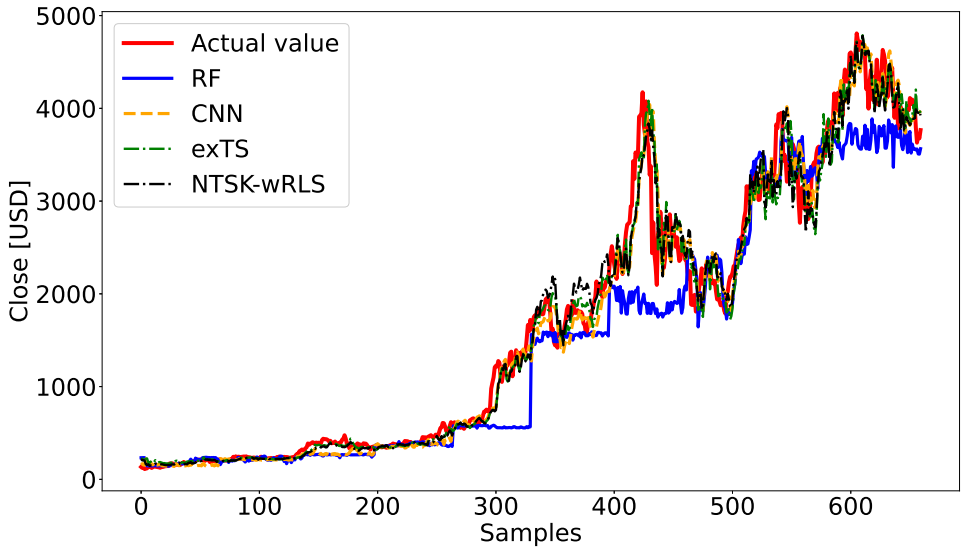


Figure 27 – Prediction performance on Ethereum dataset (horizon = 5)

Table 53 presents the simulation results for predicting Ethereum five steps ahead. RF obtained the lowest mean errors among the classical models, while CNN performed best among the DL models. For the eFSs, exTS recorded the lowest errors; among the proposed models, NTSK (RLS) and NTSK (wRLS) yielded the lowest errors. Simpl_eTS generated the highest number of rules. Conversely, SVM, RNN, LSTM, and ePL exhibited higher errors. Furthermore, ePL and GEN-NTSK (wRLS) achieved the best mean for CPPM. Table 54 confirms that the NTSK models achieved lower errors than many of the comparison models. Figure 27 shows the predictions for Ethereum five steps ahead.

Table 53 – Simulations' results of Ethereum (horizon = 5)

Model	NRMSE	NDEI	MAPE	CPPM (%)	Rules
KNN [218]	0.49 (0.18)	1.95 (0.87)	0.24 (0.14)	50.00 (8.23)	-
Regression Tree [219]	0.46 (0.11)	1.81 (0.62)	0.19 (0.07)	46.92 (4.97)	-
Random Forest [112]	**0.40 (0.12)**	**1.62 (0.67)**	**0.18 (0.09)**	50.00 (4.83)	-
SVM [220]	1.34 (0.80)	5.26 (3.22)	0.64 (0.25)	50.00 (6.43)	-
LS-SVM [221]	0.44 (0.23)	1.80 (1.18)	0.22 (0.14)	48.00 (7.31)	-
GBM [116]	0.73 (0.34)	2.90 (1.46)	0.34 (0.13)	45.85 (7.62)	-
XGBoost [117]	0.44 (0.13)	1.80 (0.78)	0.19 (0.08)	47.23 (8.40)	-
LGBM [118]	0.43 (0.16)	1.71 (0.76)	0.20 (0.11)	46.62 (5.06)	-
MLP [222]	0.31 (0.19)	1.30 (1.05)	0.14 (0.10)	48.92 (6.87)	-
CNN [223]	**0.28 (0.12)**	**1.15 (0.64)**	**0.12 (0.06)**	50.46 (7.93)	-
RNN [224]	1.18 (0.68)	4.66 (2.77)	0.55 (0.22)	45.85 (8.16)	-
LSTM [225]	1.01 (0.50)	4.14 (2.34)	0.48 (0.21)	46.77 (6.24)	-
GRU [226]	0.78 (0.30)	3.21 (1.67)	0.36 (0.14)	49.69 (8.48)	-
WaveNet [227]	0.28 (0.16)	1.19 (0.87)	0.12 (0.07)	50.31 (7.15)	-
eTS [68]	0.35 (0.19)	1.40 (0.84)	0.16 (0.11)	49.85 (7.76)	5 (0)
Simpl_eTS [69]	0.41 (0.20)	1.68 (1.02)	0.21 (0.13)	47.54 (7.55)	44 (5)
exTS [70]	**0.23 (0.07)**	**0.90 (0.29)**	**0.10 (0.05)**	49.54 (4.12)	3 (0)
ePL [72]	4.75 (5.46)	18.55 (21.58)	1.94 (1.86)	50.77 (5.92)	3 (2)
eMG [74]	0.50 (0.22)	2.03 (1.16)	0.23 (0.13)	49.08 (6.79)	12 (9)
ePL+ [73]	0.33 (0.13)	1.33 (0.75)	0.15 (0.08)	50.00 (7.29)	2 (0)
ePL-KRLS-DISCO [75]	0.63 (0.49)	2.43 (1.94)	0.26 (0.19)	50.15 (6.53)	8 (7)
NMR	0.52 (0.24)	2.09 (1.17)	0.24 (0.15)	46.15 (5.06)	3 (0)
NTSK (RLS)	0.24 (0.06)	0.94 (0.31)	**0.10 (0.03)**	49.54 (3.82)	1 (0)
NTSK (wRLS)	**0.23 (0.06)**	**0.94 (0.33)**	**0.10 (0.03)**	50.31 (6.74)	6 (0)
GEN-NMR	0.52 (0.24)	2.10 (1.15)	0.25 (0.15)	46.62 (5.55)	3 (0)
GEN-NTSK (RLS)	0.24 (0.06)	0.96 (0.33)	0.10 (0.04)	49.08 (5.48)	1 (0)
GEN-NTSK (wRLS)	0.31 (0.21)	1.24 (0.86)	0.13 (0.08)	50.77 (4.67)	6 (0)
R-NMR	0.49 (0.24)	1.94 (1.08)	0.23 (0.14)	50.00 (7.02)	-
R-NTSK	0.24 (0.06)	0.94 (0.33)	0.10 (0.04)	49.85 (6.96)	-
RF-NTSK	0.25 (0.06)	1.00 (0.37)	0.11 (0.04)	48.92 (6.52)	-

Table 54 – Statistical MDM test for Ethereum (horizon = 5)

Model	NMR	NTSK (RLS)	NTSK (wRLS)	GEN-NMR	GEN-NTSK (RLS)	GEN-NTSK (wRLS)	R-NMR	R-NTSK	RF-NTSK
KNN [218]	1.00	0.00	0.00	1.00	0.00	0.31	1.00	0.00	0.00
Regression Tree [219]	1.00	0.00	0.00	1.00	0.00	0.42	1.00	0.00	0.00
Random Forest [112]	1.00	0.00	0.00	1.00	0.00	0.66	1.00	0.00	0.00
SVM [220]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LS-SVM [221]	1.00	0.00	0.00	1.00	0.00	1.00	1.00	0.00	0.00
GBM [116]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
XGBoost [117]	1.00	0.00	0.00	1.00	0.00	0.48	1.00	0.00	0.00
LGBM [118]	1.00	0.00	0.00	1.00	0.00	0.19	1.00	0.00	0.00
MLP [222]	1.00	0.60	0.13	1.00	0.67	1.00	1.00	0.01	0.57
CNN [223]	1.00	0.95	0.53	1.00	0.93	1.00	1.00	0.37	0.84
RNN [224]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LSTM [225]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
GRU [226]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
WaveNet [227]	1.00	0.90	0.39	1.00	0.89	1.00	1.00	0.17	0.79
eTS [68]	1.00	0.17	0.04	1.00	0.14	1.00	1.00	0.05	0.06
Simpl_eTS [69]	1.00	0.00	0.00	1.00	0.00	1.00	1.00	0.00	0.00
exTS [70]	1.00	1.00	0.54	1.00	0.96	1.00	1.00	0.36	0.83
ePL [72]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
eMG [74]	1.00	0.00	0.00	1.00	0.00	0.02	1.00	0.00	0.00
ePL+ [73]	1.00	0.00	0.00	1.00	0.00	1.00	1.00	0.00	0.00
ePL-KRLS-DISCO [75]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 55 – Simulations' results of the Binance coin (horizon = 1)

Model	NRMSE	NDEI	MAPE	CPPM (%)	Rules
KNN [218]	0.42 (0.22)	1.88 (1.12)	0.24 (0.22)	49.23 (5.42)	-
Regression Tree [219]	0.34 (0.13)	1.44 (0.52)	0.18 (0.18)	43.38 (6.45)	-
Random Forest [112]	**0.23 (0.10)**	**1.04 (0.61)**	**0.15 (0.19)**	46.31 (5.82)	-
SVM [220]	1.24 (0.95)	5.10 (3.57)	0.62 (0.33)	51.23 (5.38)	-
LS-SVM [221]	1.02 (0.65)	4.22 (2.51)	0.52 (0.26)	51.54 (6.78)	-
GBM [116]	0.26 (0.10)	1.16 (0.52)	**0.15 (0.19)**	49.38 (4.69)	-
XGBoost [117]	0.28 (0.11)	1.23 (0.64)	0.16 (0.20)	46.00 (4.74)	-
LGBM [118]	0.27 (0.12)	1.22 (0.67)	0.17 (0.21)	47.54 (4.64)	-
MLP [222]	0.35 (0.32)	1.60 (1.69)	0.16 (0.13)	47.08 (6.16)	-
CNN [223]	0.31 (0.35)	1.44 (1.87)	0.13 (0.11)	46.92 (5.11)	-
RNN [224]	0.49 (0.34)	2.25 (1.87)	0.28 (0.25)	48.00 (4.66)	-
LSTM [225]	0.39 (0.37)	1.87 (2.02)	0.23 (0.26)	45.85 (3.49)	-
GRU [226]	0.42 (0.36)	1.97 (1.97)	0.23 (0.21)	46.92 (4.83)	-
WaveNet [227]	**0.19 (0.11)**	**0.84 (0.59)**	**0.08 (0.04)**	47.54 (5.17)	-
eTS [68]	0.36 (0.32)	1.61 (1.50)	0.17 (0.14)	50.15 (6.39)	5 (0)
Simpl_eTS [69]	0.40 (0.32)	1.85 (1.65)	0.22 (0.20)	50.77 (5.01)	44 (7)
exTS [70]	0.31 (0.23)	1.30 (0.93)	0.14 (0.12)	48.92 (6.45)	6 (3)
ePL [72]	6.98 (10.78)	26.30 (40.31)	2.77 (3.61)	45.08 (3.37)	4 (4)
eMG [74]	0.31 (0.15)	1.38 (0.79)	0.15 (0.11)	44.46 (5.65)	20 (20)
ePL+ [73]	**0.13 (0.04)**	**0.53 (0.14)**	**0.06 (0.03)**	45.08 (2.92)	1 (0)
ePL-KRLS-DISCO [75]	0.17 (0.05)	0.74 (0.24)	0.08 (0.07)	49.85 (4.92)	3 (3)
NMR	0.27 (0.12)	1.18 (0.52)	0.17 (0.20)	48.00 (6.03)	14 (1)
NTSK (RLS)	**0.11 (0.03)**	0.46 (0.12)	**0.05 (0.02)**	45.38 (3.53)	1 (0)
NTSK (wRLS)	0.15 (0.08)	0.65 (0.42)	0.06 (0.03)	47.23 (4.62)	5 (0)
GEN-NMR	0.31 (0.11)	1.38 (0.63)	0.19 (0.22)	45.54 (5.43)	3 (0)
GEN-NTSK (RLS)	**0.11 (0.03)**	**0.46 (0.11)**	0.05 (0.03)	45.54 (4.57)	1 (0)
GEN-NTSK (wRLS)	0.15 (0.08)	0.62 (0.35)	0.06 (0.04)	46.62 (5.20)	6 (0)
R-NMR	0.25 (0.10)	1.11 (0.57)	0.16 (0.19)	46.31 (3.92)	-
R-NTSK	0.16 (0.12)	0.69 (0.47)	0.07 (0.06)	46.62 (4.13)	-
RF-NTSK	0.17 (0.07)	0.74 (0.38)	0.09 (0.08)	48.31 (2.85)	-

Table 55 presents the simulation results for predicting Binance Coin one step ahead. RF and GBM obtained the lowest mean errors among the classical models, while WaveNet performed best among the DL models. For the eFSs, ePL+ recorded the lowest errors; among the proposed models, NTSK (RLS) and GEN-NTSK (RLS) yielded the lowest errors. Simpl_eTS generated the highest number of rules. Conversely, SVM and ePL exhibited higher errors. Furthermore, LS-SVM achieved the best mean for CPPM. Table 56 confirms that the NTSK-based models achieved lower errors than many of the comparison models.

Table 56 – Statistical MDM test for Binance (horizon = 1)

Model	NMR	NTSK (RLS)	NTSK (wRLS)	GEN-NMR	GEN-NTSK (RLS)	GEN-NTSK (wRLS)	R-NMR	R-NTSK	RF-NTSK
KNN [218]	0.00	0.00	0.00	0.52	0.00	0.00	0.18	0.00	0.00
Regression Tree [219]	0.96	0.00	0.00	1.00	0.00	0.00	0.99	0.00	0.00
Random Forest [112]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
SVM [220]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LS-SVM [221]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
GBM [116]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
XGBoost [117]	0.82	0.00	0.00	1.00	0.00	0.00	0.99	0.00	0.00
LGBM [118]	0.00	0.00	0.00	0.25	0.00	0.00	0.01	0.00	0.00
MLP [222]	1.00	0.11	0.00	1.00	0.54	0.01	1.00	0.00	1.00
CNN [223]	1.00	0.88	0.47	1.00	0.99	0.65	1.00	0.32	1.00
RNN [224]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LSTM [225]	0.04	0.00	0.00	1.00	0.00	0.00	0.74	0.00	0.00
GRU [226]	0.02	0.00	0.00	0.63	0.00	0.00	0.20	0.00	0.00
WaveNet [227]	1.00	0.54	0.13	1.00	0.90	0.22	1.00	0.10	1.00
eTS [68]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	1.00
Simpl_eTS [69]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
exTS [70]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.02
ePL [72]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
eMG [74]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
ePL+ [73]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.98
ePL-KRLS-DISCO [75]	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00

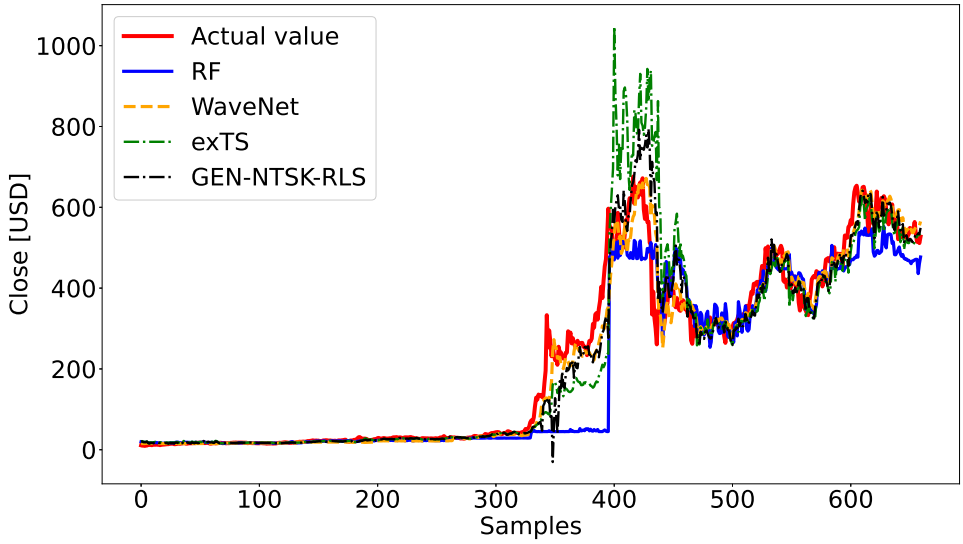


Figure 28 – Prediction performance on Binance dataset (horizon = 5)

Table 57 presents the simulation results for predicting Binance Coin five steps ahead. RF and LGBM obtained the lowest errors among the classical models, while WaveNet performed best among the DL models. For the eFSs, exTS recorded the lowest errors, and GEN-NTSK (RLS) yielded the lowest errors among all models. Simpl_eTS generated the highest number of rules. Conversely, ePL and ePL-KRLS-DISCO exhibited higher errors. Furthermore, ePL+ achieved the best mean for CPPM. Table 58 confirms that the NTSK models achieved lower errors than many of the comparison models. Figure 28 illustrates the predictions for Binance Coin five steps ahead.

Table 57 – Simulations' results of the Binance coin (horizon = 5)

Model	NRMSE	NDEI	MAPE	CPPM (%)	Rules
KNN [218]	0.54 (0.24)	2.33 (1.08)	0.26 (0.21)	47.85 (5.03)	-
Regression Tree [219]	0.50 (0.20)	2.13 (0.81)	0.25 (0.19)	46.46 (4.23)	-
Random Forest [112]	**0.38 (0.17)**	**1.68 (0.94)**	**0.21 (0.20)**	48.31 (4.42)	-
SVM [220]	0.77 (0.34)	3.33 (1.71)	0.43 (0.23)	49.38 (5.52)	-
LS-SVM [221]	0.69 (0.30)	2.98 (1.58)	0.38 (0.22)	48.92 (5.32)	-
GBM [116]	0.42 (0.16)	1.81 (0.84)	0.22 (0.20)	50.46 (4.23)	-
XGBoost [117]	0.45 (0.20)	1.94 (1.02)	0.22 (0.20)	46.31 (6.29)	-
LGBM [118]	0.40 (0.15)	1.76 (0.85)	**0.21 (0.20)**	50.46 (5.75)	-
MLP [222]	0.39 (0.33)	1.79 (1.82)	0.18 (0.12)	49.54 (6.80)	-
CNN [223]	0.33 (0.27)	1.52 (1.47)	0.15 (0.09)	50.15 (7.79)	-
RNN [224]	0.71 (0.68)	3.05 (2.78)	0.34 (0.26)	47.54 (6.29)	-
LSTM [225]	0.42 (0.27)	1.95 (1.56)	0.24 (0.22)	50.92 (6.93)	-
GRU [226]	0.45 (0.33)	2.10 (1.87)	0.25 (0.21)	48.77 (4.57)	-
WaveNet [227]	**0.31 (0.23)**	**1.40 (1.26)**	**0.13 (0.07)**	48.77 (6.81)	-
eTS [68]	0.42 (0.26)	1.82 (1.23)	0.19 (0.13)	47.85 (6.76)	4 (0)
Simpl_eTS [69]	0.47 (0.32)	2.12 (1.72)	0.25 (0.18)	49.85 (6.28)	29 (3)
exTS [70]	**0.33 (0.16)**	**1.37 (0.54)**	**0.18 (0.16)**	48.31 (6.35)	5 (2)
ePL [72]	5.54 (7.91)	20.91 (29.55)	2.36 (2.76)	47.54 (6.32)	3 (3)
eMG [74]	0.48 (0.19)	2.08 (0.91)	0.25 (0.19)	49.08 (6.96)	19 (20)
ePL+ [73]	0.47 (0.25)	1.94 (0.95)	0.23 (0.16)	52.15 (6.05)	2 (1)
ePL-KRLS-DISCO [75]	1.77 (1.44)	7.69 (6.78)	0.92 (1.39)	49.85 (6.08)	10 (11)
NMR	0.44 (0.25)	1.87 (0.97)	0.22 (0.18)	48.31 (3.90)	10 (0)
NTSK (RLS)	0.41 (0.17)	1.80 (0.93)	0.24 (0.27)	45.54 (5.76)	1 (0)
NTSK (wRLS)	0.29 (0.14)	1.23 (0.59)	0.14 (0.09)	48.77 (7.12)	11 (1)
GEN-NMR	0.41 (0.21)	1.77 (0.84)	0.24 (0.22)	48.46 (5.56)	9 (0)
GEN-NTSK (RLS)	**0.27 (0.10)**	**1.13 (0.39)**	**0.13 (0.09)**	48.31 (3.96)	1 (0)
GEN-NTSK (wRLS)	0.30 (0.15)	1.29 (0.67)	0.14 (0.09)	49.38 (5.30)	12 (1)
R-NMR	0.36 (0.14)	1.53 (0.52)	0.21 (0.20)	50.62 (4.38)	-
R-NTSK	0.30 (0.15)	1.27 (0.64)	0.14 (0.09)	47.85 (7.52)	-
RF-NTSK	0.31 (0.14)	1.35 (0.68)	0.16 (0.11)	48.31 (7.16)	-

Table 58 – Statistical MDM test for Binance (horizon = 5)

Model	NMR	NTSK (RLS)	NTSK (wRLS)	GEN-NMR	GEN-NTSK (RLS)	GEN-NTSK (wRLS)	R-NMR	R-NTSK	RF-NTSK
KNN [218]	0.12	0.98	0.00	0.00	0.00	0.00	1.00	0.00	0.00
Regression Tree [219]	0.00	0.93	0.00	0.00	0.00	0.00	0.91	0.00	0.00
Random Forest [112]	0.87	0.99	0.00	0.36	0.00	0.00	1.00	0.00	0.00
SVM [220]	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LS-SVM [221]	0.00	0.16	0.00	0.00	0.00	0.00	0.00	0.00	0.00
GBM [116]	0.28	0.99	0.00	0.02	0.00	0.00	1.00	0.00	0.00
XGBoost [117]	0.03	0.98	0.00	0.00	0.00	0.00	1.00	0.00	0.00
LGBM [118]	0.17	0.98	0.00	0.02	0.00	0.00	1.00	0.00	0.00
MLP [222]	1.00	1.00	0.98	1.00	1.00	0.99	1.00	0.96	1.00
CNN [223]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
RNN [224]	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LSTM [225]	0.97	0.99	0.00	0.70	0.00	0.00	1.00	0.00	0.00
GRU [226]	0.95	0.99	0.00	0.60	0.00	0.00	1.00	0.00	0.00
WaveNet [227]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
eTS [68]	1.00	1.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
Simpl_eTS [69]	1.00	1.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
exTS [70]	0.44	0.98	0.00	0.26	0.00	0.00	1.00	0.00	0.00
ePL [72]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
eMG [74]	0.15	0.98	0.00	0.01	0.00	0.00	1.00	0.00	0.00
ePL+ [73]	0.01	0.94	0.00	0.00	0.00	0.00	0.96	0.00	0.00
ePL-KRLS-DISCO [75]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

6.2 RULES, ERRORS, AND INTERPRETABILITY

When evaluating the impact of the number of rules on model performance, it is observed that for RLS-based models, the number of rules does not influence the error metric, as each rule shares identical consequent parameters. Since λ is the sole hyperparameter influencing the consequent parameters and remains constant across all rules, the results are invariant to changes in $R_{\max}$. Consequently, only λ requires tuning to minimize error, while $R_{\max}$ can be adjusted to optimize interpretability.

In contrast, for models based on NMR and NTSK (wRLS), $R_{\max}$ directly affects predictive performance. Therefore, $R_{\max}$ must be tuned to achieve the desired balance within the accuracy-interpretability trade-off. Figure 29 illustrates the NDEI as a function of $R_{\max}$ for the NMR model. For the NASDAQ dataset, the NDEI appears to reach its minimum at approximately five rules, beyond which it begins to increase. For the remaining datasets, the minimum is achieved with approximately three rules, after which it stabilizes. Furthermore, the standard deviation tends to increase alongside the number of rules. The results suggest that NMR-based models achieve lower errors within the range $R_{\max} \in [3, 5]$. For higher values, the increased complexity yields no performance benefit.

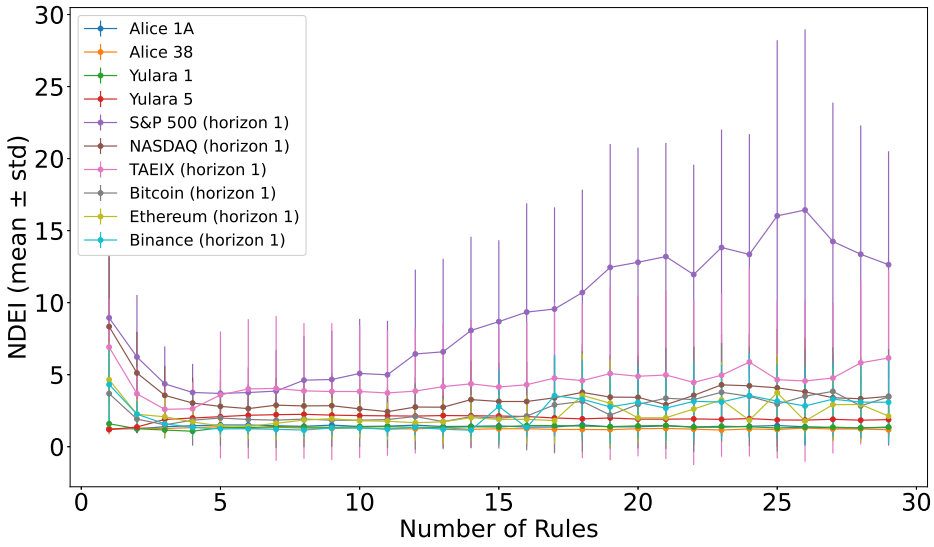
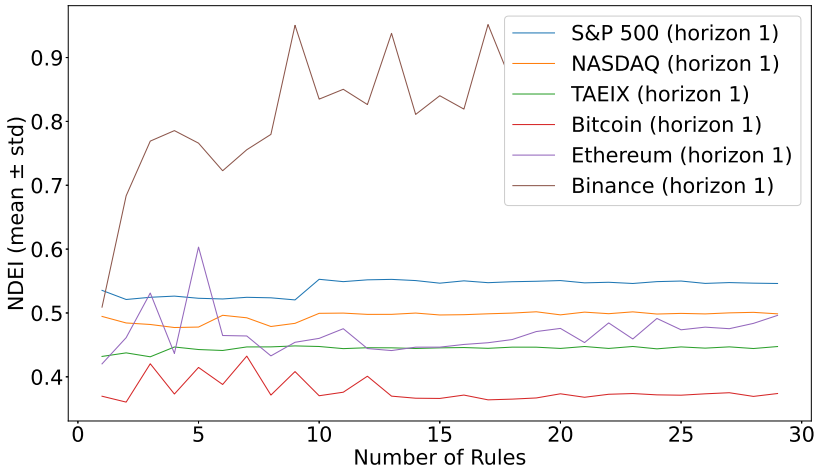
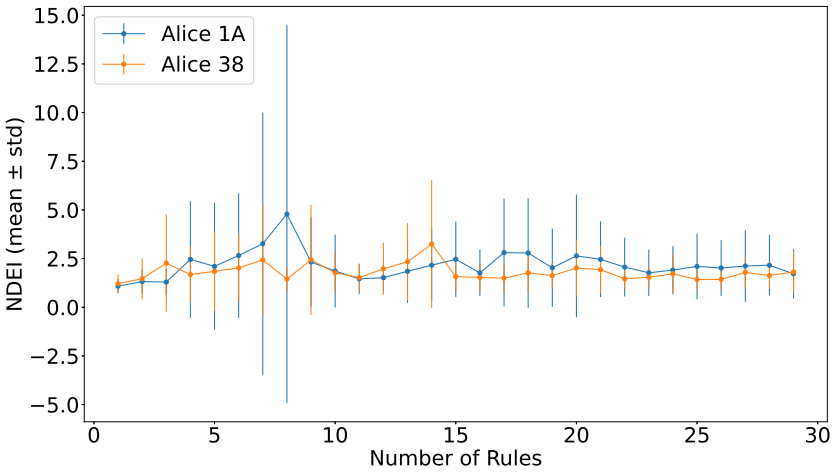


Figure 29 – NDEI as a function of $R_{\max}$ for NMR

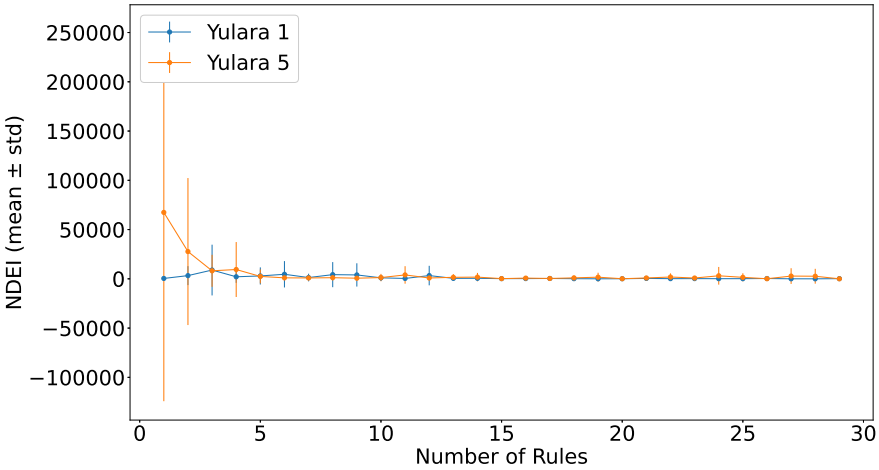
Figure 30 illustrates how the NDEI varies with the number of rules for the NTSK (wRLS) model when applied to the datasets. The plots indicate that there is no clear, consistent pattern for the errors as a function of $R_{\max}$. For example, for the S&P 500, the NDEI remains approximately constant regarding $R_{\max}$. Conversely, for the Alice datasets (Figure 30b), the errors remain stable for up to 15 rules, while for the Yulara datasets (Figure 30c), the error converges after approximately three rules.



(a) All models but solar energy datasets



(b) Alice datasets



(c) Yulara datasets

Figure 30 – NDEI as a function of $R_{\max}$ for NTSK (wRLS)

Table 59 presents the rule base generated by the NMR model when configured with four rules. The values denote the normalized mean and standard deviation of the fuzzy sets. However, as this numerical representation is not intuitively interpreted, linguistic variables are employed to provide a qualitative description, as shown in Table 60. For example, the first rule can be interpreted as follows: *IF the Open, High, Low, and Close prices are all "Very Low", THEN the Close price one step ahead will also be "Very Low".*

Table 59 – NMR rule base for the S&P 500 dataset ($R_{\max} = 4$)

Rule	Open	High	Low	Close	Next Close
1	0.16 (0.08)	0.17 (0.07)	0.18 (0.09)	0.18 (0.08)	2687.28 (197.81)
2	0.38 (0.07)	0.39 (0.06)	0.40 (0.07)	0.40 (0.06)	3251.66 (159.87)
3	0.61 (0.07)	0.61 (0.07)	0.62 (0.07)	0.62 (0.07)	3823.05 (165.25)
4	0.85 (0.07)	0.85 (0.07)	0.85 (0.07)	0.86 (0.07)	4430.70 (177.85)

Table 60 – Linguistic interpretation of the NMR rule base for the S&P 500

Rule	Open	High	Low	Close	Next Close
1	Very Low	Very Low	Very Low	Very Low	Very Low
2	Low	Low	Medium	Medium	Low
3	High	High	High	High	High
4	Very High	Very High	Very High	Very High	Very High

The same interpretability principle applies to the NTSK-based models, as presented in Table 61. The Open, High, Low, and Close columns display the mean and standard deviation of the antecedent fuzzy sets. These numerical values can be replaced by linguistic variables. The *Next Close* column indicates the expected variation of the closing price for each rule. This feature constitutes the primary advantage of the NTSK model regarding interpretability. Unlike models that utilize abstract polynomial parameters in the consequent, NTSK provides the expected variation of the target value directly. For instance, considering Table 61, if a new sample is most compatible with the first rule, the *Next Close* value is expected to range between -324.89 and -186.07. Therefore, if the current value is $25,000$, the next value is predicted to fall between $24,675.11$ and $24,813.93$.

Table 61 – NTSK (wRLS) rule base for the S&P 500 dataset ($R_{\max} = 4$)

Rule	Open	High	Low	Close	Next Close
1	0.41 (0.34)	0.41 (0.34)	0.40 (0.35)	0.41 (0.35)	[-324.89, -186.07]
2	0.50 (0.28)	0.50 (0.27)	0.50 (0.27)	0.50 (0.27)	[-186.07, -47.26]
3	0.60 (0.23)	0.60 (0.23)	0.61 (0.22)	0.61 (0.23)	[-47.26, 91.56]
4	0.25 (0.27)	0.27 (0.26)	0.25 (0.26)	0.27 (0.26)	[91.56, 230.38]

Furthermore, the proposed genetic-based models enhance interpretability through attribute selection. For instance, the solar energy datasets contain numerous attributes, which can complicate the visual representation of the rules. The proposed approaches simplify this by selecting key attributes, thereby facilitating human comprehension. Table 62 displays the rule base generated by GEN-NMR, while Table 63 provides the corresponding linguistic interpretation. Notably, the model selected only two attributes out of twelve. Additionally, Figure 64 presents the four rules generated by GEN-NTSK (wRLS), where the model selected six attributes out of twelve.

Table 62 – GEN-NMR rule base for the Alice 1A dataset ($R_{\max} = 4$)

Rule	Wind Direction [°]	Energy [kWh]	Next Power [kW]
1	28.32 (16.20)	39.73 (16.45)	0.70 (0.42)
2	29.09 (16.09)	41.44 (14.90)	1.45 (0.25)
3	30.00 (15.04)	54.13 (39.23)	2.27 (0.23)
4	35.15 (15.86)	66.19 (25.16)	2.87 (0.15)

Table 63 – Linguistic interpretation of the GEN-NMR rule base for Alice 1A

Rule	Wind Direction	Energy	Next Power
1	Very Low	Very Low	Very Low
2	Low	Low	Low
3	High	High	High
4	Very High	Very High	Very High

As previously stated, the model selected six attributes from the twelve available. A key observation is that the model did not simply select the six most correlated attributes. Indeed, while the three most correlated attributes (current, power, and global radiation) were selected, the genetic algorithm also chose rainfall, an attribute with near-zero correlation, and diffuse radiation, a negatively correlated attribute. This suggests an important advantage: genetic-based approaches can uncover underlying relationships between the target value and the attributes that may not be readily apparent from simple correlation analysis.

Table 64 – Rules of GEN-NTSK (wRLS) for Alice 1A with four rules

Rule	Diffuse Radiation $[W/m^2]$	Rainfall $[mm]$	Wind Direction $[°]$	Current $[A]$	Power $[kW]$	Global Radiation $[W/m^2]$	Next Power $[kW]$
1	57.67 (27.35)	0.07 (0.20)	24.22 (18.12)	5.04 (0.85)	2.48 (0.49)	301.66 (69.41)	[-2.56, -1.37]
2	61.40 (37.21)	0.56 (2.41)	31.20 (15.20)	4.48 (1.07)	2.18 (0.64)	264.34 (88.11)	[-1.37, -0.18]
3	54.35 (35.91)	0.46 (2.45)	31.33 (15.58)	4.54 (1.11)	2.24 (0.57)	270.44 (78.31)	[-0.18, 1.01]
4	79.98 (34.09)	0.96 (3.43)	27.92 (15.82)	3.82 (1.33)	1.79 (0.76)	218.76 (95.23)	[1.01, 2.20]

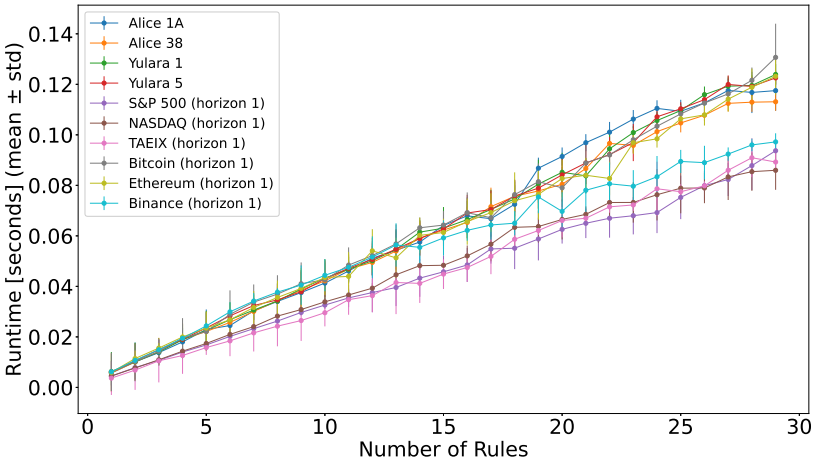
6.3 COMPUTATIONAL PERFORMANCE

Faster computational speed and practical usability increase the applicability and acceptability of models in real-world applications [228]. In this context, Figure 31 illustrates how runtime varies with the number of rules for NMR and NTSK. The plots indicate that runtime increases linearly with the number of rules for all models; however, the fastest model is NTSK (RLS). This occurs because, in this model, the parameters of the consequent component are identical for all rules. Consequently, regardless of the number of rules, the RLS-based model has only one set of consequent parameters to update. Conversely, NMR is slower than NTSK (RLS) because it must compute the fuzzy sets of the consequent component for each rule, yet it remains faster than NTSK (wRLS), which recursively updates the parameters of a polynomial function for every rule.

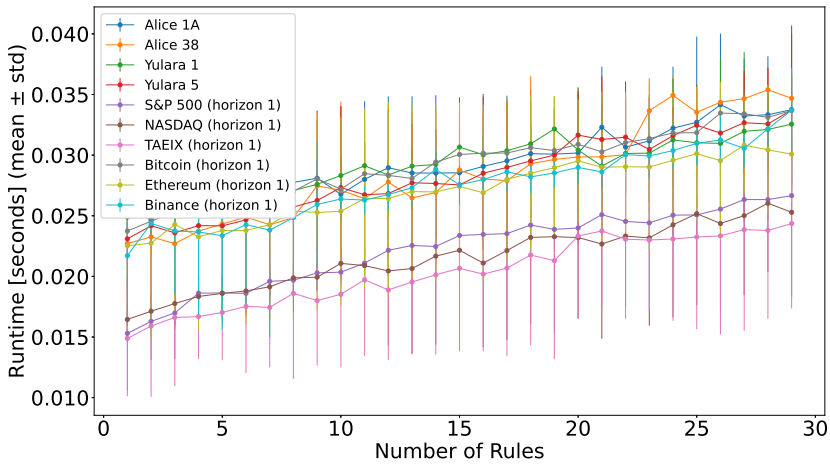
Further analysis of the plots reveals that the PV datasets require more processing time due to the higher dimensionality of their input vectors. Table 65 details the runtime increase per additional rule across all datasets. The computational cost of adding a new rule in NTSK (RLS) is, on average, approximately 12 times lower than that of NMR and 32 times lower than that of NTSK (wRLS).

Table 65 – Runtime increase per additional rule (in seconds)

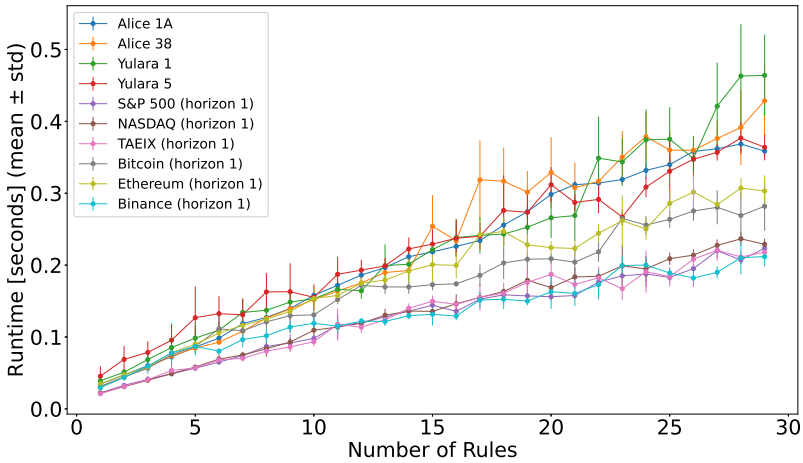
Dataset	NMR	NTSK (RLS)	NTSK (wRLS)
Alice 1A	0.0038	0.0003	0.0112
Alice 38	0.0037	0.0004	0.0137
Yulara 1	0.0040	0.0003	0.0146
Yulara 5	0.0040	0.0004	0.0110
S&P 500 (horizon = 1)	0.0030	0.0004	0.0069
NASDAQ (horizon = 1)	0.0028	0.0003	0.0071
TAIEX (horizon = 1)	0.0029	0.0003	0.0068
BITCOIN (horizon = 1)	0.0043	0.0003	0.0086
ETHEREUM (horizon = 1)	0.0040	0.0003	0.0092
BINANCE (horizon = 1)	0.0031	0.0004	0.0063
Mean	0.0036	0.0003	0.0095



(a) NMR

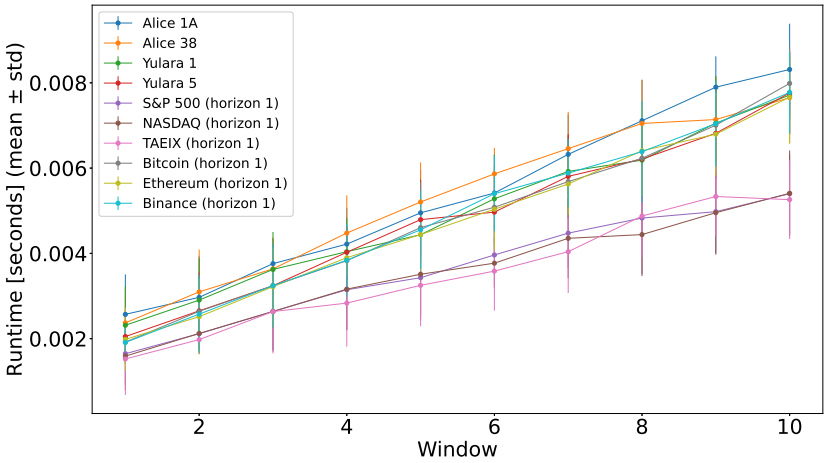


(b) NTSK (RLS)

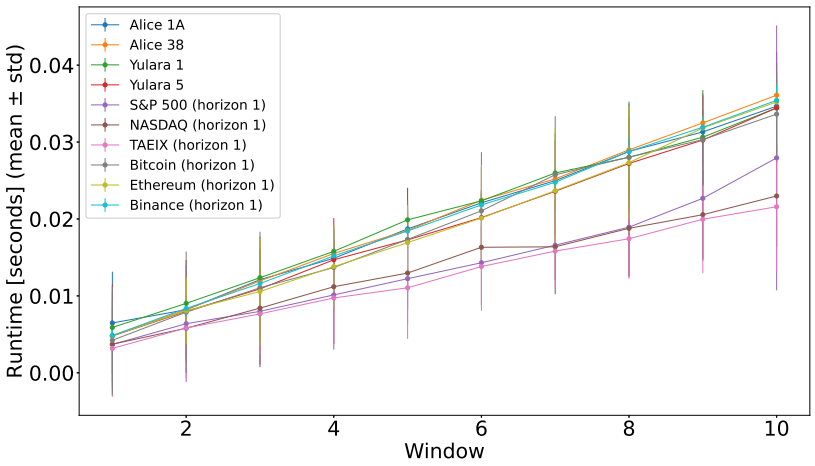


(c) NTSK (wRLS)

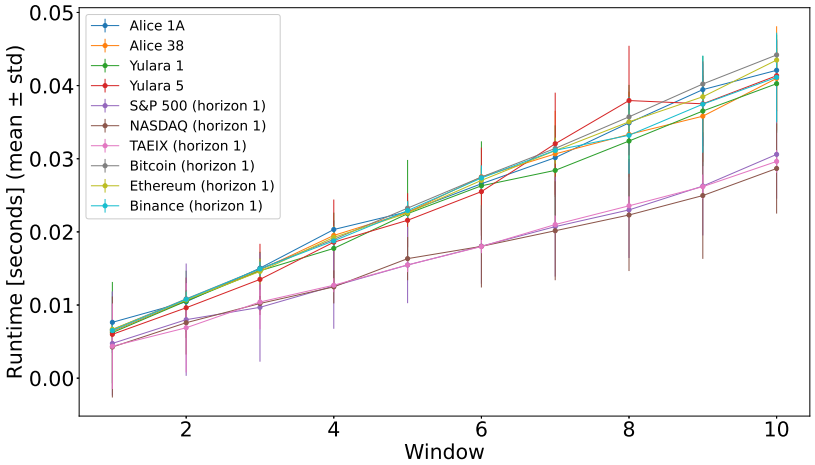
Figure 31 – Runtime as a function of $R_{\max}$



(a) NMR



(b) NTSK (RLS)



(c) NTSK (wRLS)

Figure 32 – Runtime as a function of window size with fixed $R_{\max}$

Conversely, Figure 32 suggests a linear variation in the models' runtime as a function of window size (i.e., partitions of the total dataset), where Window 1 represents 10% of the total samples, Window 2 represents 20%, and so forth. Although the results suggest that the runtime of NMR, NTSK (RLS), and NTSK (wRLS) varies linearly with both the number of rules and the number of samples, further formal investigation is required to validate the computational complexity of the proposed models.

6.4 MONTE CARLO SIMULATION FOR STABILITY ANALYSIS

To evaluate the robustness and stability of the proposed models, a Monte Carlo simulation was conducted. The results are summarized in Table 66, which details the mean and standard deviation of the NRMSE, NDEI, and MAPE error metrics for each model across the four PV energy datasets. The standard deviation serves as a direct indicator of model stability, where a lower value signifies greater robustness. The primary observation from this analysis is the superior stability of R-NMR and R-NTSK compared to the GA-based models. The R-NTSK model, in particular, demonstrates exceptional stability, frequently achieving a standard deviation of 0.00 (e.g., on Alice 1A, Alice 38, and Yulara 5). This indicates that its performance is highly consistent across different simulation runs. The R-NMR model also performs well, with low standard deviations, such as an NRMSE of 0.22 ± 0.01 on Alice 1A. Conversely, the GA-based models exhibit significant performance variance. The most unstable results were observed for GEN-NTSK (RLS) on the Yulara 1 dataset and GEN-NMR on the Yulara 5 dataset. This variability suggests that these models are highly sensitive to their initialization or data partitions.

Table 66 – Monte Carlo simulation for stability analysis

Model	NRMSE	NDEI	MAPE
Alice 1A			
GEN-NMR	0.24 ± 0.02	1.05 ± 0.09	0.41 ± 0.05
GEN-NTSK (RLS)	0.22 ± 0.01	0.98 ± 0.04	0.37 ± 0.02
GEN-NTSK (wRLS)	0.22 ± 0.01	0.96 ± 0.04	0.36 ± 0.01
R-NMR	0.22 ± 0.01	1.00 ± 0.03	0.39 ± 0.01
R-NTSK	0.21 ± 0.00	0.92 ± 0.00	0.36 ± 0.00
Alice 38			
GEN-NMR	0.22 ± 0.02	0.99 ± 0.09	0.35 ± 0.04
GEN-NTSK (RLS)	0.23 ± 0.02	1.01 ± 0.10	0.35 ± 0.05
GEN-NTSK (wRLS)	0.21 ± 0.01	0.94 ± 0.04	0.33 ± 0.01
R-NMR	0.21 ± 0.00	0.95 ± 0.01	0.35 ± 0.01
R-NTSK	0.21 ± 0.00	0.92 ± 0.00	0.33 ± 0.00
Yulara 1			
GEN-NMR	0.17 ± 0.03	0.83 ± 0.16	0.24 ± 0.08
GEN-NTSK (RLS)	0.38 ± 0.20	1.83 ± 0.96	0.62 ± 0.35
GEN-NTSK (wRLS)	0.23 ± 0.03	1.13 ± 0.12	0.43 ± 0.06
R-NMR	0.26 ± 0.01	1.24 ± 0.06	0.30 ± 0.02
R-NTSK	0.15 ± 0.01	0.73 ± 0.04	0.26 ± 0.02
Yulara 5			
GEN-NMR	0.24 ± 0.12	1.12 ± 0.54	0.26 ± 0.14
GEN-NTSK (RLS)	0.25 ± 0.07	1.15 ± 0.34	0.29 ± 0.08
GEN-NTSK (wRLS)	0.13 ± 0.04	0.61 ± 0.20	0.15 ± 0.05
R-NMR	0.19 ± 0.01	0.88 ± 0.03	0.22 ± 0.02
R-NTSK	0.12 ± 0.00	0.57 ± 0.02	0.15 ± 0.01

6.5 DISCUSSION

Regarding the proposed models, RF-NTSK demonstrated lower errors for the Mackey-Glass and nonlinear datasets, indicating its suitability for handling both short- and long-term time series. Conversely, for the Lorenz Attractor dataset, all models, except those based on NMR, achieved lower errors. Notably, for the benchmark datasets with a moderate level of noise, the best-performing proposed model performed worse than the best models from other classes. This suggests that the proposed models may not be suitable for handling noisy datasets in isolation. However, for the renewable energy datasets, the NTSK model, when combined with the feature selection approach, achieved superior performance compared to the other proposed models. This suggests that feature selection is effective in addressing the challenges posed by noisy datasets. Finally, for the financial and cryptocurrency datasets, NTSK consistently produced the lowest errors across all simulations.

For the eFSs, the ePL-KRLS-DISCO model achieved the lowest errors for the

Mackey-Glass and nonlinear datasets, indicating its effectiveness in handling datasets with reduced uncertainty. However, for real-world datasets, the eTS and exTS models outperformed the other eFS models. Notably, Simpl_eTS generated the highest number of rules in many simulations. Regarding DL models, CNN exhibited superior performance for the renewable energy datasets, achieving the lowest errors. However, DL models demonstrated high error variability for the financial and cryptocurrency datasets, indicating instability. Finally, among the classical models, LS-SVM achieved the lowest errors for the renewable energy datasets. Interestingly, in some simulations, KNN also achieved the lowest errors among the classical models, despite its simplicity.

It is noteworthy that none of the models were able to increase the CPPM significantly above 50%, indicating that the models correctly predicted the trend only slightly more than 50% of the time. Values close to 50% suggest that trend predictions are nearly random, akin to making decisions by tossing a coin. Therefore, developing techniques to enhance the predictability of trends is essential to support decision-making in trading.

Another important factor examined in this work is the number of rules in the fuzzy systems. The graphs comparing $R_{\max}$ and error for NMR suggest that the lowest errors are achieved at about three to five rules. On the other hand, for NTSK (RLS) and NTSK (wRLS), the results indicate that there is no clear correlation between $R_{\max}$ and the errors, implying that the choice of $R_{\max}$ is primarily heuristic. Regarding computational cost, the graphs reveal that runtime scales linearly with $R_{\max}$ for NMR and NTSK (wRLS), whereas for NTSK (RLS), runtime remains largely unaffected by $R_{\max}$. Furthermore, for all three models, runtime varies linearly with the number of input samples. Consequently, a key objective is to determine the number of rules that optimally balances interpretability, runtime, and error minimization. The runtime was not analyzed for approaches involving feature selection, as it varies depending on multiple factors.

Furthermore, there is no single theoretical formula for determining the number of rules. The process is largely empirical and heuristic, guided by a fundamental trade-off. Too few rules may create an overly simplistic model that cannot capture the complex dynamics of the system. Conversely, too many rules can lead to overfitting and create a black box that is as difficult to interpret as a complex neural network, defeating the purpose of using an interpretable model.

A particularly noteworthy finding is the strong performance of the RLS-based models, which were often highly effective with just a single fuzzy rule. Since the RLS filter estimates a single, global parameter vector for the consequent, a one-rule configuration effectively collapses the fuzzy model into a global linear model. The excellent results from this single-rule model suggest that for these specific datasets, a well-tuned linear model provides a very strong baseline and is sufficient to capture a significant portion of the underlying system dynamics.

However, the true value of the fuzzy models is demonstrated by the wRLS-based approaches. These models, which use distinct local parameters for each fuzzy rule, consistently outperformed their RLS counterparts. This indicates that while a global linear model is effective, the fuzzy partitioning of the input space allows the wRLS models to capture additional nonlinearities and local behaviors that the global model misses, leading to statistically significant improvements in accuracy. Therefore, the framework not only produces a high-performing model but also provides insight into the degree of nonlinearity present in the data.

Beyond predictive accuracy, the proposed models offers critical advantages in interpretability and transparency. The ability for a user to directly specify the number of rules provides explicit control over the accuracy-interpretability trade-off. This benefit is amplified by the GA-based feature selection, as exemplified by the GEN-NTSK (wRLS) model applied to the Alice 1A dataset. The GA selected an optimal subset of 6 out of the original 12 attributes, which delivered two simultaneous benefits:

- **Improved Interpretability:** The resulting rule base is far simpler and easier to analyze, as seen in Table 64. The antecedent of each rule is defined by only six features instead of twelve, focusing the analysis on the most influential variables. For greater transparency, the numerical fuzzy sets in the table could be translated into intuitive linguistic terms (e.g., “low”, “medium”, “high”).
- **Improved Accuracy:** The model’s performance was enhanced by focusing only on the most relevant features. This demonstrates the GA’s capacity to uncover complex correlations and filter out noisy or redundant attributes, leading to a more robust and effective model.

In summary, the simplicity of the rule induction mechanism, combined with the optimization from feature selection and the robustness added by ensembling, results in a framework that is both powerful and practical for real-world applications.

While the proposed models demonstrated strong performance, some limitations were observed:

- The equally spaced regions for the consequent component are uniform for all rules, disregarding data density.
- It is assumed that the relationship between the input vector and the target value is one-to-one. Consequently, in cases where it is many-to-one or one-to-many, the models may perform worse.

- The GA-based approaches search for a subset of features that improves predictive performance; however, this can lead to overfitting (i.e., a feature subset that performs well on a specific range but does not generalize well).
- The ensemble fuzzy approaches suffer from reduced interpretability.

6.6 SUMMARY

This chapter presented a detailed account of the simulations, reporting the obtained error metrics and the number of generated rules. Subsequently, key aspects of the proposed models, such as their rule bases, interpretability, and computational performance, were examined. Finally, a critical discussion of the results was provided. The next chapter presents the conclusions of this work.

7 CONCLUSIONS

This work introduces a new framework of fuzzy inference systems (NFISiS) for time series forecasting, comprising a series of new data-driven fuzzy models. NFISiS includes a novel data-driven approach to design Mamdani and TSK fuzzy rules with direct control over the number of rules, termed the New Mamdani Regressor (NMR) and New Takagi-Sugeno-Kang (NTSK). The novel mechanism for designing Mamdani and TSK rules offers reduced complexity, higher autonomy, improved accuracy, and fewer hyperparameters. In general, the NTSK model outperformed the NMR model regarding error metrics.

Furthermore, two adaptive filter approaches for defining the consequent parameters of NTSK were evaluated: RLS and wRLS. The RLS approach estimates a single, global parameter vector for the consequent, which effectively collapses the fuzzy model into a global linear model, thereby rendering RLS faster than wRLS. While the results suggest that for specific datasets, a well-tuned RLS provides a very strong baseline and is sufficient to capture a significant portion of the underlying system dynamics, the true value of the NTSK is demonstrated by the wRLS-based approaches. These are able to capture additional nonlinearities and local behaviors that the global model misses, leading to statistically significant improvements in accuracy.

Additionally, two feature selection methods were implemented in the proposed fuzzy models, GA and ensemble, aiming to enhance the models' ability to handle large datasets, optimize performance, increase interpretability, and avoid overfitting. In general, the implementation of the GA increased interpretability and reduced both errors and complexity by selecting only the most relevant set of features for a specific dataset. On the other hand, the fuzzy ensemble avoids the risk of overfitting.

The models were applied to several time series, including benchmark, solar energy, financial, and cryptocurrency datasets. Their performance was evaluated in terms of error metrics and the number of rules, with further analysis conducted on rule bases, interpretability, and computational performance. Regarding computational performance, the plots reveal that runtime scales linearly with $R_{\max}$ for NMR and NTSK (wRLS), whereas the runtime of NTSK (RLS) is largely unaffected by $R_{\max}$. Furthermore, for all three models, runtime varies linearly with the number of input samples. Additionally, the graphs comparing $R_{\max}$ and error for the NMR model suggest that the lowest errors are achieved with approximately three to five rules. In contrast, for NTSK (RLS) and NTSK (wRLS), the results indicate that there is no clear correlation between $R_{\max}$ and the errors, implying that the choice of $R_{\max}$ is primarily heuristic. Another aspect evaluated concerns the stability of the stochastic models; the results indicate that the fuzzy ensembles are more stable than the GA-based models.

The benchmark simulations indicate that NTSK-based models perform better when predicting complex datasets over short time horizons; however, their performance diminishes over long time horizons. Furthermore, results from the financial and cryptocurrency datasets demonstrate the models' suitability for non-stationary data. Conversely, although NMR-based models typically do not perform as well as NTSK on non-stationary datasets, they achieved better results on the energy datasets, supporting their suitability for data populated by outliers. The implementation of GA for feature selection proved effective for improving model performance and interpretability by producing results with fewer attributes. One of the main advantages of NTSK compared to the conventional TSK is its increased interpretability, as the polynomial functions of the consequent can be replaced by the expected variation of the target value within the rule-based structure. On the other hand, the fuzzy ensembles achieved the lowest errors in many of the simulations, suggesting the models perform well in a broader range of datasets.

While classical models obtained good results in straightforward simulations, their performance decreased significantly in more complex scenarios. Regarding DL models, they performed well across all benchmark series; however, their varied results on real-world datasets limited a conclusive analysis. As for eFSs, they generally performed similarly to the proposed models. However, eFSs present several drawbacks, such as a larger number of hyperparameters, limited interpretability, a lack of control over the number of rules, and greater complexity. Since the proposed models overcome these shortcomings, they are preferable to eFSs.

While the proposed models demonstrated strong performance, it is important to acknowledge their limitations. First, the equally spaced regions for the consequent component are uniform for all rules, disregarding the underlying data density. Second, the models assume a one-to-one relationship between the input vector and the target value, which may reduce efficacy on problems with more complex many-to-one mappings. Third, the GA-based feature selection approaches, while powerful, can be prone to overfitting, potentially finding a feature subset that does not generalize well. Finally, the ensemble fuzzy approaches, although highly accurate, achieve this performance at the expense of the core interpretability for which fuzzy systems are known.

These limitations give rise to several promising avenues for future research. To address architectural constraints, one direction involves exploring Type-2 fuzzy systems to better manage uncertainty and alternative membership functions to capture complex relationships. This also includes integrating advanced algorithms such as Kernel Recursive Least Squares (KRLS) for more robust parameter estimation and investigating non-uniform consequent partitions that respect data density. Another avenue concerns the learning paradigm. Future work could focus on developing online learning mechanisms to adapt to data streams and creating methods for the automatic optimization of fuzzy set parameters. To combat overfitting and the loss of interpretability, research is needed on regularization

techniques for GA-based selection and methods for rule-base merging or pruning in ensembles. Finally, the scope of evaluation could be broadened. A formal computational complexity analysis is warranted, and testing the frameworks on a more diverse range of high-dimensionality datasets is essential for validating their scalability and generalization capabilities.

REFERENCES

[1] A. Shrestha, A. Mahmood, Review of deep learning algorithms and architectures, IEEE Access 7 (2019) 53040–53065. doi:https://doi.org/10.1109/ACCESS.2019.2912200.

[2] X. Wang, Y. Zhao, F. Pourpanah, Recent advances in deep learning, International Journal of Machine Learning and Cybernetics 11 (2020) 747–750. doi:https://doi.org/10.1007/s13042-020-01096-5.

[3] A. Rai, Explainable AI: From black box to glass box, Journal of the Academy of Marketing Science 48 (2020) 137–141. doi:https://doi.org/10.1007/s11747-019-00710-5.

[4] L. A. Zadeh, Fuzzy sets, Information and Control 8 (3) (1965) 338–353. doi:https://doi.org/10.1016/S0019-9958(65)90241-X.

[5] G. Bojadziev, M. Bojadziev, Fuzzy logic for business, finance, and management, Vol. 23, World Scientific, 2007.

[6] I. Özkal, I. A. Özkan, F. Başçiftçi, Metaverse token price forecasting using artificial neural networks (anns) and adaptive neural fuzzy inference system (anfis), Neural Computing and Applications 36 (7) (2024) 3267–3290. doi:https://doi.org/10.1007/s00521-023-09228-y.

[7] K. G. Nalbant, A methodology for personnel selection in business development: An interval type 2-based fuzzy DEMATEL-ANP approach, Heliyon 10 (1) (2024). doi:https://doi.org/10.1016/j.heliyon.2023.e23698.

[8] M. Nasehi, Advanced applications of fuzzy systems and computational intelligence in decision-support systems, Soft Computing Fusion with Applications 1 (4) (2024) 220–228. doi:https://doi.org/10.22105/scfa.v1i4.60.

[9] K. Sareen, B. K. Panigrahi, T. Shikhola, A short-term solar irradiance forecasting modelling approach based on three decomposition algorithms and adaptive neuro-fuzzy inference system, Expert Systems with Applications 231 (2023) 120770. doi:https://doi.org/10.1016/j.eswa.2023.120770.

[10] M. I. Guerra, F. M. de Araújo, J. T. de Carvalho Neto, R. G. Vieira, Survey on adaptative neural fuzzy inference system (ANFIS) architecture applied to photovoltaic systems, Energy Systems 15 (2) (2024) 505–541. doi:https://doi.org/10.1007/s12667-022-00513-8.

[11] P. S. Szczepaniak, P. J. Lisboa, Fuzzy systems in medicine, Vol. 41, Physica, 2012.

[12] A. Y. Abdalla, T. Y. Abdalla, A. M. Chyaid, Internet of things based fuzzy systems for medical applications: A review, IEEE Access (2024). doi:https://doi.org/10.1109/ACCESS.2024.3487812.

[13] R.-E. Precup, H. Hellendoorn, A survey on industrial applications of fuzzy control, Computers in Industry 62 (3) (2011) 213–226. doi:https://doi.org/10.1016/j.compind.2010.10.001.

[14] W. Emam, J. Ahmmad, T. Mahmood, U. ur Rehman, S. Yin, Classification of artificial intelligence tools for civil engineering under the notion of complex fuzzy rough frank aggregation operators, Scientific Reports 14 (1) (2024) 11892. doi:https://doi.org/10.1038/s41598-024-60561-1.

[15] M. Komiyama, K. Yoshimoto, M. Sisido, K. Ariga, Chemistry can make strict and fuzzy controls for bio-systems: DNA nanoarchitectonics and cell-macromolecular nanoarchitectonics, Bulletin of the Chemical Society of Japan 90 (9) (2017) 967–1004. doi:https://doi.org/10.1246/bcsj.20170156.

[16] A. K. Shaout, Z. Brauchler, Fuzzy battery manager: Charging and balancing rechargeable battery cells with fuzzy logic, Electronics 14 (7) (2025) 1470. doi:https://doi.org/10.3390/electronics14071470.

[17] N. L. M. Jailani, J. K. Dhanasegaran, G. Alkawsi, A. A. Alkahtani, C. C. Phing, Y. Baashar, L. F. Capretz, A. Q. Al-Shetwi, S. K. Tiong, Investigating the Power of LSTM-Based Models in Solar Energy Forecasting, Processes 11 (5) (2023) 1382. doi:https://doi.org/10.3390/pr11051382.

[18] P. Melin, O. Mendoza, O. Castillo, Face recognition with an improved interval type-2 fuzzy logic Sugeno integral and modular neural networks, IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans 41 (5) (2011) 1001–1012. doi:https://doi.org/10.1109/TSMCA.2010.2104318.

[19] A. Lemos, W. Caminhas, F. Gomide, Adaptive fault detection and diagnosis using an evolving fuzzy classifier, Information Sciences 220 (2013) 64–85. doi:https://doi.org/10.1016/j.ins.2011.08.030.

[20] E. H. Mamdani, Application of fuzzy algorithms for control of simple dynamic plant, in: Proceedings of the Institution of Electrical Engineers, Vol. 121, IET, 1974, pp. 1585–1588. doi:https://doi.org/10.1049/piee.1974.0328.

[21] T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, IEEE Transactions on Systems, Man, and Cybernetics SMC-15 (1) (1985) 116–132. doi:https://doi.org/10.1109/TSMC.1985.6313399.

[22] M. Sugeno, G. Kang, Structure identification of fuzzy model, Fuzzy Sets and Systems 28 (1) (1988) 15–33. doi:http://dx.doi.org/10.1016/0165-0114(88)90113-3.

[23] V. Ojha, A. Abraham, V. Snášel, Heuristic design of fuzzy inference systems: A review of three decades of research, Engineering Applications of Artificial Intelligence 85 (2019) 845–864. doi:https://doi.org/10.1016/j.engappai.2019.08.010.

[24] T. Miller, Explanation in artificial intelligence: Insights from the social sciences, Artificial Intelligence 267 (2019) 1–38. doi:https://doi.org/10.1016/j.artint.2018.07.007.

[25] P. Linardatos, V. Papastefanopoulos, S. Kotsiantis, Explainable AI: A review of machine learning interpretability methods, Entropy 23 (1) (2020) 18. doi:https://doi.org/10.3390/e23010018.

[26] D. Kukolj, E. Levi, Identification of complex systems based on neural and Takagi-Sugeno fuzzy model, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 34 (1) (2004) 272–282. doi:https://doi.org/10.1109/TSMCB.2003.811119.

[27] T. A. Johansen, R. Shorten, R. Murray-Smith, On the interpretation and identification of dynamic Takagi-Sugeno fuzzy models, IEEE Transactions on Fuzzy Systems 8 (3) (2000) 297–313. doi:https://doi.org/10.1109/91.855918.

[28] B. Zhang, Y. C. Shin, A data-driven approach of Takagi-Sugeno fuzzy control of unknown nonlinear systems, Applied Sciences 11 (1) (2020) 62. doi:https://doi.org/10.3390/app11010062.

[29] H. Han, H. Han, D. Zhao, X. Gao, Y. Yang, Takagi–Sugeno Fuzzy Realization of Stability Performance-Based Fault-Tolerant Control for Nonlinear Systems, IEEE Transactions on Fuzzy Systems 30 (10) (2022) 4249–4261. doi:https://doi.org/10.1109/TFUZZ.2022.3146979.

[30] K. S. T. R. Alves, C. D. de Jesus, E. P. de Aguiar, A new Takagi–Sugeno–Kang model for time series forecasting, Engineering Applications of Artificial Intelligence 133 (2024) 108155. doi:https://doi.org/10.1016/j.engappai.2024.108155.

[31] K. Sa Teles Rocha Alves, R. Ballini, E. Pestana de Aguiar, Financial Series Forecasting: A New Fuzzy Inference System for Crisp Values and Interval-Valued Predictions, Computational Economics (2024) 1–49doi:https://doi.org/10.1007/s10614-024-10670-w.

[32] R. Zebari, A. Abdulazeez, D. Zeebaree, D. Zebari, J. Saeed, A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction, Journal of Applied Science and Technology Trends 1 (1) (2020) 56–70. doi:https://doi.org/10.38094/jastt1224.

[33] A. S. Eesa, Z. Orman, A. M. A. Brifcani, A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems, Expert Systems with Applications 42 (5) (2015) 2670–2679. doi:https://doi.org/10.1016/j.eswa.2014.11.009.

[34] Y. Mao, Y. Yang, A wrapper feature subset selection method based on randomized search and multilayer structure, BioMed Research International 2019 (1) (2019) 9864213. doi:https://doi.org/10.1155/2019/9864213.

[35] A. Sohail, Genetic algorithms in the fields of artificial intelligence and data sciences, Annals of Data Science 10 (4) (2023) 1007–1018. doi:https://doi.org/10.1007/s40745-021-00354-9.

[36] R. Forsati, A. Moayedikia, R. Jensen, M. Shamsfard, M. R. Meybodi, Enriched ant colony optimization and its application in feature selection, Neurocomputing 142 (2014) 354–371. doi:https://doi.org/10.1016/j.neucom.2014.03.053.

[37] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, Journal of Machine Learning Research 3 (Mar) (2003) 1157–1182.

[38] H.-H. Hsu, C.-W. Hsieh, M.-D. Lu, Hybrid feature selection by combining filters and wrappers, Expert Systems with Applications 38 (7) (2011) 8144–8150. doi:https://doi.org/10.1016/j.eswa.2010.12.156.

[39] V. Bolón-Canedo, A. Alonso-Betanzos, Ensembles for feature selection: A review and future trends, Information Fusion 52 (2019) 1–12. doi:https://doi.org/10.1016/j.inffus.2018.11.008.

[40] X. Dong, Z. Yu, W. Cao, Y. Shi, Q. Ma, A survey on ensemble learning, Frontiers of Computer Science 14 (2020) 241–258. doi:https://doi.org/10.1007/s11704-019-8208-z.

[41] A. Alcañiz, D. Grzebyk, H. Ziar, O. Isabella, Trends and gaps in photovoltaic power forecasting with machine learning, Energy Reports 9 (2023) 447–471. doi:https://doi.org/10.1016/j.egyr.2022.11.208.

[42] G. Notton, M.-L. Nivet, C. Voyant, C. Paoli, C. Darras, F. Motte, A. Fouilloy, Intermittent and stochastic character of renewable energy sources: Consequences, cost of intermittence and benefit of forecasting, Renewable and Sustainable Energy Reviews 87 (2018) 96–105. doi:https://doi.org/10.1016/j.rser.2018.02.007.

[43] M. J. Mayer, G. Gróf, Extensive comparison of physical models for photovoltaic power forecasting, Applied Energy 283 (2021) 116239. doi:https://doi.org/10.1016/j.apenergy.2020.116239.

[44] J. Mann, J. N. Kutz, Dynamic mode decomposition for financial trading strategies, Quantitative Finance 16 (11) (2016) 1643–1655. doi:https://doi.org/10.1080/14697688.2016.1170194.

[45] G. Sonkavde, D. S. Dharrao, A. M. Bongale, S. T. Deokate, D. Doreswamy, S. K. Bhat, Forecasting stock market prices using machine learning and deep learning models: A systematic review, performance analysis and discussion of implications, International Journal of Financial Studies 11 (3) (2023) 94. doi:https://doi.org/10.3390/ijfs11030094.

[46] D. Anderson, L. Hall, MR. FIS: Mamdani rule style fuzzy inference system, in: IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 99CH37028), Vol. 5, IEEE, 1999, pp. 238–243. doi:https://doi.org/10.1109/ICSMC.1999.815554.

[47] J.-C. Duan, F.-L. Chung, A Mamdani type multistage fuzzy neural network model, in: 1998 IEEE International Conference on Fuzzy Systems Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98CH36228), Vol. 2, IEEE, 1998, pp. 1253–1258. doi:https://doi.org/10.1109/FUZZY.1998.686298.

[48] H. Ying, TITO Mamdani fuzzy PI/PD Controllersas nonlinear, variable gain PI/PD controllers, International Journal of Fuzzy Systems 2 (3) (2000) 192–197.

[49] O. Uncu, K. Kilic, L. Turksen, A new fuzzy inference approach based on Mamdani inference using discrete type 2 fuzzy sets, in: 2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583), Vol. 3, IEEE, 2004, pp. 2272–2277. doi:https://doi.org/10.1109/ICSMC.2004.1400667.

[50] U. Kaymak, W.-M. Van Den Bergh, J. van den Berg, A fuzzy additive reasoning scheme for probabilistic Mamdani fuzzy systems, in: The 12th IEEE International Conference on Fuzzy Systems, 2003. FUZZ'03., Vol. 1, IEEE, 2003, pp. 331–336. doi:https://doi.org/10.1109/FUZZ.2003.1209384.

[51] Y. Chai, L. Jia, Z. Zhang, Mamdani model based adaptive neural fuzzy inference system and its application, International Journal of Computer and Information Engineering 3 (3) (2009) 663–670.

[52] W. L. Tung, C. Quek, A Mamdani-Takagi-Sugeno based linguistic neural-fuzzy inference system for improved interpretability-accuracy representation, in: 2009 IEEE International Conference on Fuzzy Systems, IEEE, 2009, pp. 367–372. doi:https://doi.org/10.1109/FUZZY.2009.5277194.

[53] M. J. Gacto, M. Galende, R. Alcalá, F. Herrera, Metsk-hde: A multiobjective evolutionary algorithm to learn accurate TSK-fuzzy systems in high-dimensional and large-scale regression problems, Information Sciences 276 (2014) 63–79. doi:https://doi.org/10.1016/j.ins.2014.02.047.

[54] J. Gou, F. Hou, W. Chen, C. Wang, W. Luo, Improving Wang–Mendel method performance in fuzzy rules generation using the fuzzy C-means clustering algorithm, Neurocomputing 151 (2015) 1293–1304. doi:https://doi.org/10.1016/j.neucom.2014.10.077.

[55] M. Asadi, Optimized Mamdani fuzzy models for predicting the strength of intact rocks and anisotropic rock masses, Journal of Rock Mechanics and Geotechnical Engineering 8 (2) (2016) 218–224. doi:https://doi.org/10.1016/j.jrmge.2015.11.005.

[56] D. Pekaslan, C. Wagner, J. M. Garibaldi, L. G. Marin, D. Sáez, Uncertainty-aware forecasting of renewable energy sources, in: 2020 IEEE International Conference on Big Data and Smart Computing (BigComp), IEEE, 2020, pp. 240–246. doi:https://doi.org/10.1109/BigComp48618.2020.00-68.

[57] M. A. Kacimi, O. Guenounou, L. Brikh, F. Yahiaoui, N. Hadid, New mixed-coding PSO algorithm for a self-adaptive and automatic learning of Mamdani fuzzy rules, Engineering Applications of Artificial Intelligence 89 (2020) 103417. doi:https://doi.org/10.1016/j.engappai.2019.103417.

[58] T. Zhang, C. Wagner, Learning causal fuzzy logic rules by leveraging markov blankets, in: 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE, 2021, pp. 2794–2799. doi:https://doi.org/10.1109/SMC52423.2021.9659303.

[59] R. Navarro-Almanza, M. A. Sanchez, J. R. Castro, O. Mendoza, G. Licea, Interpretable Mamdani neuro-fuzzy model through context awareness and linguistic adaptation, Expert Systems with Applications 189 (2022) 116098. doi:https://doi.org/10.1016/j.eswa.2021.116098.

[60] Y. Zhang, H. Qu, W. Wang, J. Zhao, et al., A novel fuzzy time series forecasting model based on multiple linear regression and time series clustering, Mathematical Problems in Engineering 2020 (2020). doi:https://doi.org/10.1155/2020/9546792.

[61] C. Fantuzzi, R. Rovatti, On the approximation capabilities of the homogeneous Takagi-Sugeno model, in: Proceedings of IEEE 5th International Fuzzy Systems, Vol. 2, IEEE, 1996, pp. 1067–1072. doi:https://doi.org/10.1109/FUZZY.1996.552326.

[62] J. Joh, Y.-H. Chen, R. Langari, On the stability issues of linear Takagi-Sugeno fuzzy models, IEEE Transactions on Fuzzy Systems 6 (3) (1998) 402–410. doi:https://doi.org/10.1109/91.705508.

[63] A.-T. Nguyen, T. Taniguchi, L. Eciolaza, V. Campos, R. Palhares, M. Sugeno, Fuzzy control systems: Past, present and future, IEEE Computational Intelligence Magazine 14 (1) (2019) 56–68. doi:https://doi.org/10.1109/MCI.2018.2881644.

[64] H. Ying, General SISO Takagi-Sugeno fuzzy systems with linear rule consequent are universal approximators, IEEE Transactions on Fuzzy Systems 6 (4) (1998) 582–587. doi:https://doi.org/10.1109/91.728456.

[65] J.-S. Jang, ANFIS: adaptive-network-based fuzzy inference system, IEEE Transactions on Systems, Man, and Cybernetics 23 (3) (1993) 665–685. doi:https://doi.org/10.1109/21.256541.

[66] N. Kasabov, et al., Evolving fuzzy neural networks-algorithms, applications and biological motivation, Methodologies for the Conception, Design and Application of Soft Computing, World Scientific 1 (1998) 271–274.

[67] N. K. Kasabov, Q. Song, DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction, IEEE Transactions on Fuzzy Systems 10 (2) (2002) 144–154. doi:https://doi.org/10.1109/91.995117.

[68] P. P. Angelov, D. P. Filev, An approach to online identification of Takagi-Sugeno fuzzy models, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 34 (1) (2004) 484–498. doi:https://doi.org/10.1109/TSMCB.2003.817053.

[69] P. Angelov, D. Filev, Simpl_eTS: A simplified method for learning evolving Takagi-Sugeno fuzzy models, in: The 14th IEEE International Conference on Fuzzy Systems, 2005. FUZZ'05., IEEE, 2005, pp. 1068–1073. doi:https://doi.org/10.1109/FUZZY.2005.1452543.

[70] P. Angelov, X. Zhou, Evolving fuzzy systems from data streams in real-time, in: 2006 International Symposium on Evolving Fuzzy Systems, IEEE, 2006, pp. 29–35. doi:https://doi.org/10.1109/ISEFS.2006.251157.

[71] E. D. Lughofer, FLEXFIS: A robust incremental learning approach for evolving Takagi–Sugeno fuzzy models, IEEE Transactions on Fuzzy Systems 16 (6) (2008) 1393–1410. doi:https://doi.org/10.1109/TFUZZ.2008.925908.

[72] E. Lima, M. Hell, R. Ballini, F. Gomide, Evolving fuzzy modeling using participatory learning, Evolving Intelligent Systems: Methodology and Applications (2010) 67–86doi:https://doi.org/10.1002/9780470569962.ch4.

[73] L. Maciel, F. Gomide, R. Ballini, An enhanced approach for evolving participatory learning fuzzy modeling, in: 2012 IEEE Conference on Evolving and Adaptive Intelligent Systems, IEEE, 2012, pp. 23–28. doi:https://doi.org/10.1109/EAIS.2012.6232799.

[74] A. Lemos, W. Caminhas, F. Gomide, Multivariable Gaussian evolving fuzzy modeling system, IEEE Transactions on Fuzzy Systems 19 (1) (2010) 91–104. doi:https://doi.org/10.1109/TFUZZ.2010.2087381.

[75] K. S. T. R. Alves, E. P. de Aguiar, A novel rule-based evolving fuzzy system applied to the thermal modeling of power transformers, Applied Soft Computing 112 (2021) 107764. doi:https://doi.org/10.1016/j.asoc.2021.107764.

[76] H. Vernieuwe, B. De Baets, N. E. Verhoest, Comparison of clustering algorithms in the identification of Takagi–Sugeno models: A hydrological case study, Fuzzy Sets and Systems 157 (21) (2006) 2876–2896. doi:https://doi.org/10.1016/j.fss.2006.04.007.

[77] C. Kung, J. Su, Affine Takagi-Sugeno fuzzy modelling algorithm by fuzzy c-regression models clustering with a novel cluster validity criterion, IET Control Theory & Applications 1 (5) (2007) 1255–1265. doi:https://doi.org/10.1049/iet-cta:20060415.

[78] H. Du, N. Zhang, Application of evolving Takagi–Sugeno fuzzy model to nonlinear system identification, Applied Soft Computing 8 (1) (2008) 676–686. doi:https://doi.org/10.1016/j.asoc.2007.05.006.

[79] B. Rezaee, M. F. Zarandi, Data-driven fuzzy modeling for Takagi–Sugeno–Kang fuzzy system, Information Sciences 180 (2) (2010) 241–255. doi:https://doi.org/10.1016/j.ins.2009.08.021.

[80] J.-X. Xu, Z.-Q. Guo, T. H. Lee, Design and implementation of a Takagi–Sugeno-type fuzzy logic controller on a two-wheeled mobile robot, IEEE Transactions on Industrial Electronics 60 (12) (2012) 5717–5728.

[81] N. J. Cheung, X.-M. Ding, H.-B. Shen, OptiFel: A convergent heterogeneous particle swarm optimization algorithm for Takagi–Sugeno fuzzy modeling, IEEE Transactions on Fuzzy Systems 22 (4) (2013) 919–933. doi:https://doi.org/10.1109/TFUZZ.2013.2278972.

[82] R.-E. Precup, M.-C. Sabau, E. M. Petriu, Nature-inspired optimal tuning of input membership functions of Takagi-Sugeno-Kang fuzzy models for anti-lock braking systems, Applied Soft Computing 27 (2015) 575–589. doi:https://doi.org/10.1016/j.asoc.2014.07.004.

[83] S. Vrkalovic, T.-A. Teban, I.-D. Borlea, Stable Takagi-Sugeno fuzzy control designed by optimization, International Journal of Artificial Intelligence 15 (2) (2017) 17–29.

[84] S.-H. Tsai, Y.-W. Chen, A novel identification method for Takagi–Sugeno fuzzy model, Fuzzy Sets and Systems 338 (2018) 117–135. doi:https://doi.org/10.1016/j.fss.2017.10.012.

[85] X. Lai, L. Zhang, X. Liu, Takagi-Sugeno modeling of incomplete data for missing value imputation with the use of alternate learning, IEEE Access 8 (2020) 83633–83644. doi:https://doi.org/10.1109/ACCESS.2020.2991669.

[86] E. Zander, B. van Oostendorp, B. Bede, Reinforcement learning with Takagi-Sugeno-Kang fuzzy systems, Complex Engineering Systems 3 (2) (2023). doi:https://doi.org/10.20517/ces.2023.11.

[87] H. Fang, Y. Tu, S. He, H. Wang, C. Sun, S. S. Cheng, Self-learning Takagi-Sugeno fuzzy control with application to semi-car active suspension model, IEEE Transactions on Fuzzy Systems (2023). doi:https://doi.org/10.1109/TFUZZ.2023.3290041.

[88] Y. Zhang, G. Wang, T. Zhou, G. Ren, S. Lam, W. Ding, J. Cai, Deep Reconciled & Self-Paced TSK Fuzzy System Ensemble for Imbalanced Data Classification: Architecture, Interpretability and Theory., IEEE Transactions on Fuzzy Systems (2024). doi:https://doi.org/10.1109/TFUZZ.2024.3442821.

[89] J. H. Holland, Genetic algorithms and the optimal allocation of trials, SIAM Journal on Computing 2 (2) (1973) 88–105. doi:https://doi.org/10.1137/0202009.

[90] J. H. Holland, Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence, MIT Press, 1992. doi:https://doi.org/10.7551/mitpress/1090.001.0001.

[91] A. E. Eiben, J. E. Smith, Introduction to evolutionary computing, Springer, 2015. doi:https://doi.org/10.1007/978-3-662-44874-8.

[92] X. Shi, L. Wan, H. Lee, X. Yang, L. Wang, Y. Liang, An improved genetic algorithm with variable population-size and a PSO-GA based hybrid evolutionary algorithm, in: Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 03EX693), Vol. 3, IEEE, 2003, pp. 1735–1740. doi:https://doi.org/10.1109/ICMLC.2003.1259777.

[93] C.-F. Juang, A hybrid of genetic algorithm and particle swarm optimization for recurrent network design, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 34 (2) (2004) 997–1006. doi:https://doi.org/10.1109/TSMCB.2003.818557.

[94] L. Wang, A hybrid genetic algorithm–neural network strategy for simulation optimization, Applied Mathematics and Computation 170 (2) (2005) 1329–1343. doi:https://doi.org/10.1016/j.amc.2005.01.024.

[95] R. H. Sheikh, M. M. Raghuwanshi, A. N. Jaiswal, Genetic algorithm based clustering: a survey, in: 2008 First International Conference on Emerging Trends in Engineering and Technology, IEEE, 2008, pp. 314–319. doi:https://doi.org/10.1109/ICETET.2008.48.

[96] I. Syarif, A. Prugel-Bennett, G. Wills, SVM parameter optimization using grid search and genetic algorithm to improve classification performance, TELKOMNIKA (Telecommunication Computing Electronics and Control) 14 (4) (2016) 1502–1509. doi:http://doi.org/10.12928/telkomnika.v14i4.3956.

[97] Y. Ding, X. Fu, Kernel-based fuzzy c-means clustering algorithm based on genetic algorithm, Neurocomputing 188 (2016) 233–238. doi:https://doi.org/10.1016/j.neucom.2015.01.106.

[98] A. Sehgal, H. La, S. Louis, H. Nguyen, Deep reinforcement learning using genetic algorithm for parameter optimization, in: 2019 Third IEEE International Conference on Robotic Computing (IRC), IEEE, 2019, pp. 596–601. doi:https://doi.org/10.1109/IRC.2019.00121.

[99] M. Zivkovic, V. K, N. Bacanin, A. Djordjevic, M. Antonijevic, I. Strumberger, T. A. Rashid, Hybrid genetic algorithm and machine learning method for covid-19 cases prediction, in: Proceedings of International Conference on Sustainable Expert Systems: ICSES 2020, Springer, 2021, pp. 169–184. doi:https://doi.org/10.1007/978-981-33-4355-9_14.

[100] C. Gao, W. Bao, S. Wang, J. Zheng, L. Wang, Y. Ren, L. Jiao, J. Wang, X. Wang, DockingGA: enhancing targeted molecule generation using transformer neural network and genetic algorithm with docking simulation, Briefings in Functional Genomics (2024) elae011doi:https://doi.org/10.1093/bfgp/elae011.

[101] N. Maleki, Y. Zeinali, S. T. A. Niaki, A k-NN method for lung cancer prognosis with the use of a genetic algorithm for feature selection, Expert Systems with Applications 164 (2021) 113981. doi:https://doi.org/10.1016/j.eswa.2020.113981.

[102] R. Guha, M. Ghosh, S. Kapri, S. Shaw, S. Mutsuddi, V. Bhateja, R. Sarkar, Deluge based genetic algorithm for feature selection, Evolutionary Intelligence 14 (2021) 357–367. doi:https://doi.org/10.1007/s12065-019-00218-5.

[103] Z. Halim, M. N. Yousaf, M. Waqas, M. Sulaiman, G. Abbas, M. Hussain, I. Ahmad, M. Hanif, An effective genetic algorithm-based feature selection method for intrusion detection systems, Computers & Security 110 (2021) 102448. doi:https://doi.org/10.1016/j.cose.2021.102448.

[104] M. G. Altarabichi, S. Nowaczyk, S. Pashami, P. S. Mashhadi, Fast Genetic Algorithm for feature selection — A qualitative approximation approach, Expert Systems with Applications 211 (2023) 118528. doi:https://doi.org/10.1016/j.eswa.2022.118528.

[105] O. Sagi, L. Rokach, Ensemble learning: A survey, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 8 (4) (2018) e1249. doi:https://doi.org/10.1002/widm.1249.

[106] Y. Freund, R. E. Schapire, A desicion-theoretic generalization of on-line learning and an application to boosting, in: European Conference on Computational Learning Theory, Springer, 1995, pp. 23–37. doi:https://doi.org/10.1007/3-540-59119-2_166.

[107] C. Cortes, M. Mohri, U. Syed, Deep boosting, in: International Conference on Machine Learning, PMLR, 2014, pp. 1179–1187.

[108] G. Rätsch, T. Onoda, K.-R. Müller, Soft margins for adaboost, Machine Learning 42 (2001) 287–320. doi:https://doi.org/10.1023/A:1007618119488.

[109] A. Vezhnevets, V. Vezhnevets, Modest adaboost-teaching adaboost to generalize better, in: Graphicon, Vol. 12(5), Citeseer, 2005, pp. 987–997.

[110] S. Avidan, SpatialBoost: Adding Spatial Reasoning to AdaBoost, in: Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006, Proceedings, Part IV 9, Springer, 2006, pp. 386–396. doi:https://doi.org/10.1007/11744085_30.

[111] I. Palit, C. K. Reddy, Scalable and parallel boosting with mapreduce, IEEE Transactions on Knowledge and Data Engineering 24 (10) (2011) 1904–1916. doi:https://doi.org/10.1109/TKDE.2011.208.

[112] T. K. Ho, Random decision forests, in: Proceedings of 3rd International Conference on Document Analysis and Recognition, Vol. 1, IEEE, 1995, pp. 278–282. doi:https://doi.org/10.1109/ICDAR.1995.598994.

[113] L. Breiman, Random forests, Machine Learning 45 (2001) 5–32. doi:https://doi.org/10.1023/A:1010933404324.

[114] T. Han, D. Jiang, Q. Zhao, L. Wang, K. Yin, Comparison of random forest, artificial neural networks and support vector machine for intelligent diagnosis of rotating machinery, Transactions of the Institute of Measurement and Control 40 (8) (2018) 2681–2693. doi:https://doi.org/10.1177/0142331217708242.

[115] P. Geurts, D. Ernst, L. Wehenkel, Extremely randomized trees, Machine Learning 63 (2006) 3–42. doi:https://doi.org/10.1007/s10994-006-6226-1.

[116] J. H. Friedman, Greedy function approximation: a gradient boosting machine, Annals of Statistics (2001) 1189–1232doi:https://doi.org/10.1214/aos/1013203451.

[117] T. Chen, C. Guestrin, XGBoost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794. doi:https://doi.org/10.1145/2939672.2939785.

[118] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T.-Y. Liu, LightGBM: A highly efficient gradient boosting decision tree, Advances in Neural Information Processing Systems 30 (2017). doi:https://doi.org/10.1007/s11280-022-01033-2.

[119] L. Wang, S. Mao, B. Wilamowski, Short-term load forecasting with LSTM based ensemble learning, in: 2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), IEEE, 2019, pp. 793–800. doi:https://doi.org/10.1109/iThings/GreenCom/CPSCom/SmartData.2019.00145.

[120] H. Wang, M. Li, X. Yue, Inclstm: incremental ensemble LSTM model towards time series data, Computers & Electrical Engineering 92 (2021) 107156. doi:https://doi.org/10.1016/j.compeleceng.2021.107156.

[121] K. V. Kumar, R. Anitha, A novel ensemble model by combining LSTM, BiLSTM, and facebook prophet algorithm to forecast stock prices, in: 2022 Third International Conference on Intelligent Computing Instrumentation and Control Technologies (ICICICT), IEEE, 2022, pp. 1044–1047. doi:https://doi.org/10.1109/ICICICT54557.2022.9917634.

[122] L. Mochurad, A. Dereviannyi, An ensemble approach integrating LSTM and ARIMA models for enhanced financial market predictions, Royal Society Open Science 11 (9) (2024) 240699. doi:https://doi.org/10.1098/rsos.240699.

[123] Z. Ahmad, V. Shanmugasundaram, Biju, R. Khan, Improving LSTM forecasting through ensemble learning: a comparative analysis of various models, International Journal of Information Technology 16 (8) (2024) 5113–5131. doi:https://doi.org/10.1007/s41870-024-02157-6.

[124] M. Farsi, Application of ensemble RNN deep neural network to the fall detection through IoT environment, Alexandria Engineering Journal 60 (1) (2021) 199–211. doi:https://doi.org/10.1016/j.aej.2020.06.056.

[125] E. Borandag, Software fault prediction using an RNN-based deep learning approach and ensemble machine learning techniques, Applied Sciences 13 (3) (2023) 1639. doi:https://doi.org/10.3390/app13031639.

[126] K. Zhang, N. Chen, J. Liu, M. Beer, A GRU-based ensemble learning method for time-variant uncertain structural response analysis, Computer Methods in Applied Mechanics and Engineering 391 (2022) 114516. doi:https://doi.org/10.1016/j.cma.2021.114516.

[127] M. R. Ahmed, S. Islam, A. M. Islam, S. Shatabda, An ensemble 1D-CNN-LSTM-GRU model with data augmentation for speech emotion recognition, Expert Systems with Applications 218 (2023) 119633. doi:https://doi.org/10.1016/j.eswa.2023.119633.

[128] H. Hua, M. Liu, Y. Li, S. Deng, Q. Wang, An ensemble framework for short-term load forecasting based on parallel CNN and GRU with improved resnet, Electric Power Systems Research 216 (2023) 109057. doi:https://doi.org/10.1016/j.epsr.2022.109057.

[129] C. Wan, J. Zhao, Y. Song, Z. Xu, J. Lin, Z. Hu, Photovoltaic and solar power forecasting for smart grid energy management, CSEE Journal of Power and Energy Systems 1 (4) (2015) 38–46. doi:https://doi.org/10.17775/CSEEJPES.2015.00046.

[130] F. Barbieri, S. Rajakaruna, A. Ghosh, Very short-term photovoltaic power forecasting with cloud modeling: A review, Renewable and Sustainable Energy Reviews 75 (2017) 242–263. doi:https://doi.org/10.1016/j.rser.2016.10.068.

[131] U. K. Das, K. S. Tey, M. Seyedmahmoudian, S. Mekhilef, M. Y. I. Idris, W. Van Deventer, B. Horan, A. Stojcevski, Forecasting of photovoltaic power generation and model optimization: A review, Renewable and Sustainable Energy Reviews 81 (2018) 912–928. doi:https://doi.org/10.1016/j.rser.2017.08.017.

[132] M. Munsif, M. Ullah, U. Fath, S. U. Khan, N. Khan, S. W. Baik, CT-NET: A Novel Convolutional Transformer-Based Network for Short-Term Solar Energy Forecasting Using Climatic Information., Computer Systems Science & Engineering 47 (2) (2023). doi:https://doi.org/10.32604/csse.2023.038514d.

[133] A. Khandakar, M. EH Chowdhury, M. Khoda Kazi, K. Benhmed, F. Touati, M. Al-Hitmi, A. Jr SP Gonzales, Machine learning based photovoltaics (PV) power prediction using different environmental parameters of Qatar, Energies 12 (14) (2019) 2782. doi:https://doi.org/10.3390/en12142782.

[134] I. El Haji, M. Megrini, M. Kchikach, S. Sahbani, A. Gaga, A. El Hasnaoui, Performance optimization of symmetrical multi-level boost converter using hybrid MPPT-ANN for solar energy applications, Results in Engineering 26 (2025) 104729. doi:https://doi.org/10.1016/j.rineng.2025.104729.

[135] D. O'Leary, J. Kubby, et al., Feature selection and ANN solar power prediction, Journal of Renewable Energy 2017 (2017). doi:https://doi.org/10.1155/2017/2437387.

[136] W. Du, X. Liu, W. Wu, Relieff-ann synergistic regression model for solar microgrid power generation prediction, in: 2025 International Conference on Power Electronics and Electric Drives (PEED), IEEE, 2025, pp. 51–55. doi:https://doi.org/10.1109/PEED63748.2025.00018.

[137] N. Sharma, V. Puri, S. Mahajan, L. Abualigah, R. A. Zitar, A. H. Gandomi, Solar power forecasting beneath diverse weather conditions using GD and LM-artificial neural networks, Scientific Reports 13 (1) (2023) 8517. doi:https://doi.org/10.1038/s41598-023-35457-1.

[138] H.-T. Yang, C.-M. Huang, Y.-C. Huang, Y.-S. Pai, A weather-based hybrid method for 1-day ahead hourly forecasting of PV power output, IEEE Transactions on Sustainable Energy 5 (3) (2014) 917–926. doi:https://doi.org/10.1109/TSTE.2014.2313600.

[139] S. Preda, S.-V. Oprea, A. Bâra, A. Belciu, PV forecasting using support vector machine learning in a big data analytics context, Symmetry 10 (12) (2018) 748. doi:https://doi.org/10.3390/sym10120748.

[140] Y. Li, Y. He, Y. Su, L. Shu, Forecasting the daily power output of a grid-connected photovoltaic system based on multivariate adaptive regression splines, Applied Energy 180 (2016) 392–401. doi:https://doi.org/10.1016/j.apenergy.2016.07.052.

[141] A. T. Eseye, J. Zhang, D. Zheng, Short-term photovoltaic solar power forecasting using a hybrid Wavelet-PSO-SVM model based on SCADA and meteorological information, Renewable Energy 118 (2018) 357–367. doi:https://doi.org/10.1016/j.renene.2017.11.011.

[142] W. VanDeventer, E. Jamei, G. S. Thirunavukkarasu, M. Seyedmahmoudian, T. K. Soon, B. Horan, S. Mekhilef, A. Stojcevski, Short-term PV power forecasting using hybrid GASVM technique, Renewable Energy 140 (2019) 367–379. doi:https://doi.org/10.1016/j.renene.2019.02.087.

[143] M. Pan, C. Li, R. Gao, Y. Huang, H. You, T. Gu, F. Qin, Photovoltaic power forecasting based on a support vector machine with improved ant colony optimization, Journal of Cleaner Production 277 (2020) 123948. doi:https://doi.org/10.1016/j.jclepro.2020.123948.

[144] F. Liu, R. Li, Y. Li, R. Yan, T. Saha, Takagi–Sugeno fuzzy model-based approach considering multiple weather factors for the photovoltaic power short-term forecasting, IET Renewable Power Generation 11 (10) (2017) 1281–1287. doi:https://doi.org/10.1049/iet-rpg.2016.1036.

[145] L. Liu, F. Liu, Y. Zheng, A novel ultra-short-term PV power forecasting method based on DBN-based Takagi-Sugeno fuzzy model, Energies 14 (20) (2021) 6447. doi:https://doi.org/10.3390/en14206447.

[146] P. Vijayalakshmi, S. Buvaneswari, M. Harikumaran, A. Sasikala, S. Prabhu, N. Jayanthi, Comparative analysis of ANN and ANFIS models for solar energy prediction: Advancing forecasting accuracy in photovoltaic systems, AIP Conference Proceedings 3231 (1) (2024) 040008. doi:https://doi.org/10.1063/5.0235865.

[147] T. Salameh, M. M. Farag, A.-K. Hamid, M. Hussein, Adaptive neuro-fuzzy inference system for accurate power forecasting for on-grid photovoltaic systems: A case study in Sharjah, UAE, Energy Conversion and Management: X 26 (2025) 100958. doi:https://doi.org/10.1016/j.ecmx.2025.100958.

[148] R. Jalalifar, M. R. Delavar, S. F. Ghaderi, A novel approach for photovoltaic plant site selection in megacities utilizing power load forecasting and fuzzy inference system, Renewable Energy 243 (2025) 122527. doi:https://doi.org/10.1016/j.renene.2025.122527.

[149] M. Ghofrani, M. Ghayekhloo, R. Azimi, A novel soft computing framework for solar radiation forecasting, Applied Soft Computing 48 (2016) 207–216. doi:https://doi.org/10.1016/j.asoc.2016.07.022.

[150] E. W. Law, A. A. Prasad, M. Kay, R. A. Taylor, Direct normal irradiance forecasting and its application to concentrated solar thermal output forecasting–A review, Solar Energy 108 (2014) 287–307. doi:https://doi.org/10.1016/j.solener.2014.07.008.

[151] A. W. Li, G. S. Bastos, Stock market forecasting using deep learning and technical analysis: a systematic review, IEEE Access 8 (2020) 185232–185242. doi:https://doi.org/10.1109/ACCESS.2020.3030226.

[152] M. Nabipour, P. Nayyeri, H. Jabani, A. Mosavi, E. Salwana, Deep learning for stock market prediction, Entropy 22 (8) (2020) 840. doi:https://doi.org/10.3390/e22080840.

[153] W. Jiang, Applications of deep learning in stock market prediction: recent progress, Expert Systems with Applications 184 (2021) 115537. doi:https://doi.org/10.1016/j.eswa.2021.115537.

[154] J. Qiu, B. Wang, C. Zhou, Forecasting stock prices with long-short term memory neural network based on attention mechanism, PloS One 15 (1) (2020) e0227222. doi:https://doi.org/10.1371/journal.pone.0227222.

[155] S. Banik, N. Sharma, M. Mangla, S. N. Mohanty, S. Shitharth, LSTM based decision support system for swing trading in stock market, Knowledge-Based Systems 239 (2022) 107994. doi:https://doi.org/10.1016/j.knosys.2021.107994.

[156] M. Ballings, D. Van den Poel, N. Hespeels, R. Gryp, Evaluating multiple classifiers for stock price direction prediction, Expert Systems with Applications 42 (20) (2015) 7046–7056. doi:https://doi.org/10.1016/j.eswa.2015.05.013.

[157] Z. Zhu, K. He, Prediction of Amazon's Stock Price Based on ARIMA, XGBoost, and LSTM Models, Proceedings of Business and Economic Studies 5 (5) (2022) 127–136. doi:https://doi.org/10.26689/pbes.v5i5.4432.

[158] Y. Xu, C. Yang, S. Peng, Y. Nojima, A hybrid two-stage financial stock forecasting algorithm based on clustering and ensemble learning, Applied Intelligence 50 (2020) 3852–3867. doi:https://doi.org/10.1007/s10489-020-01766-5.

[159] X. Zhou, Z. Pan, G. Hu, S. Tang, C. Zhao, Stock market prediction on high-frequency data using generative adversarial nets., Mathematical Problems in Engineering (2018). doi:https://doi.org/10.1155/2018/4907423.

[160] S. Rajab, V. Sharma, A review on the applications of neuro-fuzzy systems in business, Artificial Intelligence Review 49 (2018) 481–510. doi:https://doi.org/10.1007/s10462-016-9536-0.

[161] M. Sanchez-Roger, M. D. Oliver-Alfonso, C. Sanchís-Pedregosa, Fuzzy logic and its uses in finance: a systematic review exploring its potential to deal with banking crises, Mathematics 7 (11) (2019) 1091. doi:https://doi.org/10.3390/math7111091.

[162] A. Greaves, B. Au, Using the Bitcoin transaction graph to predict the price of Bitcoin, No Data 8 (2015) 416–443.

[163] V. Derbentsev, N. Datsenko, O. Stepanenko, V. Bezkorovainyi, Forecasting cryptocurrency prices time series using machine learning approach, in: SHS Web of Conferences, Vol. 65, EDP Sciences, 2019, p. 02001. doi:https://doi.org/10.1051/shsconf/20196502001.

[164] S. Aanandhi, S. Akhilaa, V. Vardarajan, M. Sathiyanarayanan, et al., Cryptocurrency price prediction using time series forecasting (ARIMA), in: 2021 4th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), IEEE, 2021, pp. 598–602. doi:https://doi.org/10.1109/ISRITI54043.2021.9702842.

[165] J. Wu, X. Zhang, F. Huang, H. Zhou, R. Chandra, Review of deep learning models for crypto price prediction: implementation and evaluation, arXiv preprint arXiv:2405.11431 (2024).

[166] B. Ly, D. Timaul, A. Lukanan, J. Lau, E. Steinmetz, Applying deep learning to better predict cryptocurrency trends, in: Midwest Instruction and Computing Symposium, 2018.

[167] M. Saad, J. Choi, D. Nyang, J. Kim, A. Mohaisen, Toward characterizing blockchain-based cryptocurrencies for highly accurate predictions, IEEE Systems Journal 14 (1) (2019) 321–332. doi:https://doi.org/10.1109/JSYST.2019.2927707.

[168] S. Lahmiri, S. Bekiros, Cryptocurrency forecasting with deep learning chaotic neural networks, Chaos, Solitons & Fractals 118 (2019) 35–40. doi:https://doi.org/10.1016/j.chaos.2018.11.014.

[169] C.-H. Wu, C.-C. Lu, Y.-F. Ma, R.-S. Lu, A new forecasting framework for Bitcoin price with LSTM, in: 2018 IEEE International Conference on Data Mining Workshops (ICDMW), IEEE, 2018, pp. 168–175. doi:https://doi.org/10.1109/ICDMW.2018.00032.

[170] M. M. Patel, S. Tanwar, R. Gupta, N. Kumar, A deep learning-based cryptocurrency price prediction scheme for financial institutions, Journal of Information Security and Applications 55 (2020) 102583. doi:https://doi.org/10.1016/j.jisa.2020.102583.

[171] Z. Jiang, J. Liang, Cryptocurrency portfolio management with deep reinforcement learning, in: 2017 Intelligent Systems Conference (IntelliSys), IEEE, 2017, pp. 905–913. doi:https://doi.org/10.1109/IntelliSys.2017.8324237.

[172] K. Lee, S. Ulkuatam, P. Beling, W. Scherer, Generating synthetic Bitcoin transactions and predicting market price movement via inverse reinforcement learning and agent-based modeling, Journal of Artificial Societies and Social Simulation 21 (3) (2018). doi:https://doi.org/10.18564/jasss.3733.

[173] C. Betancourt, W.-H. Chen, Deep reinforcement learning for portfolio management of markets with a dynamic number of assets, Expert Systems with Applications 164 (2021) 114002. doi:https://doi.org/10.1016/j.eswa.2020.114002.

[174] M. Schnaubelt, Deep reinforcement learning for the optimal placement of cryptocurrency limit orders, European Journal of Operational Research 296 (3) (2022) 993–1006. doi:https://doi.org/10.1016/j.ejor.2021.04.050.

[175] G. Lucarelli, M. Borrotti, A deep reinforcement learning approach for automated cryptocurrency trading, in: Artificial Intelligence Applications and Innovations: 15th IFIP WG 12.5 International Conference, AIAI 2019, Hersonissos, Crete, Greece, May 24–26, 2019, Proceedings 15, Springer, 2019, pp. 247–258. doi:https://doi.org/10.1007/978-3-030-19823-7_20.

[176] S. Marne, S. Churi, D. Correia, J. Gomes, Predicting Price of Cryptocurrency–A Deep Learning Approach, NTASU-9 (3) (2020) 140–157.

[177] V. D'Amato, S. Levantesi, G. Piscopo, Deep learning in predicting cryptocurrency volatility, Physica A: Statistical Mechanics and its Applications 596 (2022) 127158. doi:https://doi.org/10.1016/j.physa.2022.127158.

[178] S. N. Putri, D. Saputro, Construction fuzzy logic with curve shoulder in inference system Mamdani, in: Journal of Physics: Conference Series, Vol. 1776(1), IOP Publishing, 2021, p. 012060. doi:https://doi.org/10.1088/1742-6596/1776/1/012060.

[179] J. M. Mendel, Uncertain rule-based fuzzy systems, Introduction and New Directions 684 (2017). doi:https://doi.org/10.1007/978-3-319-51370-6.

[180] J. Zhao, B. K. Bose, Evaluation of membership functions for fuzzy logic controlled induction motor drive, in: IEEE 2002 28th Annual Conference of the Industrial Electronics Society. IECON 02, Vol. 1, IEEE, 2002, pp. 229–234. doi:https://doi.org/10.1109/IECON.2002.1187512.

[181] C. Radhika, R. Parvathi, Intuitionistic fuzzification functions, Global Journal of Pure and Applied Mathematics 12 (2) (2016) 1211–1227.

[182] A. Jain, A. Sharma, Membership function formulation methods for fuzzy logic systems: A comprehensive review, Journal of Critical Reviews 7 (19) (2020) 8717–8733.

[183] S. Azimi, H. Miar-Naimi, Designing programmable current-mode Gaussian and bell-shaped membership function, Analog Integrated Circuits and Signal Processing 102 (2) (2020) 323–330. doi:https://doi.org/10.1007/s10470-019-01567-y.

[184] E. Ontiveros-Robles, P. Melin, O. Castillo, Relevance of polynomial order in Takagi-Sugeno fuzzy inference systems applied in diagnosis problems, in: 2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), IEEE, 2019, pp. 1–6. doi:https://doi.org/10.1109/FUZZ-IEEE.2019.8859028.

[185] F. Farahbod, M. Eftekhari, Comparison of different t-norm operators in classification problems, arXiv preprint arXiv:1208.1955 (2012). doi:https://doi.org/10.48550/arXiv.1208.1955.

[186] H.-J. Zimmermann, Fuzzy set theory, Wiley Interdisciplinary Reviews: Computational Statistics 2 (3) (2010) 317–332. doi:https://doi.org/10.1002/wics.82.

[187] F. Afrinaldi, H.-C. Zhang, A fuzzy logic based aggregation method for life cycle impact assessment, Journal of Cleaner Production 67 (2014) 159–172. doi:https://doi.org/10.1016/j.jclepro.2013.12.010.

[188] M. Kalinic, J. M. Krisp, Fuzzy inference approach in traffic congestion detection, Annals of GIS 25 (4) (2019) 329–336. doi:https://doi.org/10.1080/19475683.2019.1675760.

[189] C.-C. Lee, Fuzzy logic in control systems: fuzzy logic controller. i, IEEE Transactions on Systems, Man, and Cybernetics 20 (2) (1990) 404–418. doi:https://doi.org/10.1109/21.52551.

[190] H.-J. Zimmermann, Fuzzy set theory—and its applications, Springer Science & Business Media, 2011. doi:https://doi.org/10.1007/978-94-010-0646-0.

[191] K. H. Lee, First course on fuzzy theory and applications, Vol. 27, Springer Science & Business Media, 2004. doi:https://doi.org/10.1007/3-540-32366-X.

[192] S. Chakraverty, D. M. Sahoo, N. R. Mahato, S. Chakraverty, D. M. Sahoo, N. R. Mahato, Defuzzification, Concepts of Soft Computing: Fuzzy and ANN with Programming (2019) 117–127doi:https://doi.org/10.1007/978-981-13-7430-2_7.

[193] M. Kumar, D. M. Husain, N. Upreti, D. Gupta, Genetic algorithm: Review and application, Available at SSRN 3529843 (2010). doi:https://dx.doi.org/10.2139/ssrn.3529843.

[194] A. Shukla, H. M. Pandey, D. Mehrotra, Comparative review of selection techniques in genetic algorithm, in: 2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE), IEEE, 2015, pp. 515–519. doi:https://doi.org/10.1109/ABLAZE.2015.7154916.

[195] J. Y. Wong, S. Sharma, G. Rangaiah, Design of shell-and-tube heat exchangers for multiple objectives using elitist non-dominated sorting genetic algorithm with termination criteria, Applied Thermal Engineering 93 (2016) 888–899. doi:https://doi.org/10.1016/j.applthermaleng.2015.10.055.

[196] H. Deng, G. Runger, E. Tuv, M. Vladimir, A time series forest for classification and feature extraction, Information Sciences 239 (2013) 142–153. doi:https://doi.org/10.1016/j.ins.2013.02.030.

[197] Q. Dai, R. Ye, Z. Liu, Considering diversity and accuracy simultaneously for ensemble pruning, Applied Soft Computing 58 (2017) 75–91. doi:https://doi.org/10.1016/j.asoc.2017.04.058.

[198] Z.-H. Zhou, Ensemble methods: foundations and algorithms, CRC Press, 2012.

[199] S. Bannour, M. R. Azimi-Sadjadi, Principal component extraction using recursive least squares learning, IEEE Transactions on Neural Networks 6 (2) (1995) 457–469. doi:https://doi.org/10.1109/72.363480.

[200] M. H. Hayes, Statistical digital signal processing and modeling, John Wiley & Sons, 1996.

[201] S. S. Haykin, Adaptive filter theory, Pearson Education India, 2002.

[202] P. S. Diniz, et al., Adaptive filtering, Vol. 4, Springer, 1997. doi:https://doi.org/10.1007/978-3-030-29057-3.

[203] D. Harvey, S. Leybourne, P. Newbold, Testing the equality of prediction mean squared errors, International Journal of Forecasting 13 (2) (1997) 281–291. doi:https://doi.org/10.1016/S0169-2070(96)00719-4.

[204] S. S. Shapiro, M. B. Wilk, An analysis of variance test for normality (complete samples), Biometrika 52 (3/4) (1965) 591–611. doi:https://doi.org/10.2307/2333709.

[205] D. A. Dickey, W. A. Fuller, Likelihood ratio statistics for autoregressive time series with a unit root, Econometrica: Journal of the Econometric Society (1981) 1057–1072 doi:https://doi.org/10.2307/1912517.

[206] E. N. Lorenz, Deterministic nonperiodic flow, Journal of Atmospheric Sciences 20 (2) (1963) 130–141. doi:https://doi.org/10.1175/1520-0469(1963)020\%3C0130:DNF\%3E2.0.CO;2.

[207] M. C. Mackey, L. Glass, Oscillation and chaos in physiological control systems, Science 197 (4300) (1977) 287–289. doi:https://doi.org/10.1126/science.267326.

[208] J. D. Farmer, Chaotic attractors of an infinite-dimensional dynamical system, Physica D: Nonlinear Phenomena 4 (3) (1982) 366–393.

[209] L.-X. Wang, J. M. Mendel, Back-propagation fuzzy system as nonlinear dynamic system identifiers, in: [1992 Proceedings] IEEE International Conference on Fuzzy Systems, IEEE, 1992, pp. 1409–1418. doi:https://doi.org/10.1109/FUZZY.1992.258711.

[210] L. Wang, J. Yen, Extracting fuzzy rules for system modeling using a hybrid of genetic algorithms and Kalman filter, Fuzzy Sets and Systems 101 (3) (1999) 353–362. doi:https://doi.org/10.1016/S0165-0114(97)00098-5.

[211] M. Shu, R. Song, W. Zhu, The 'COVID'crash of the 2020 US stock market, The North American Journal of Economics and Finance 58 (2021) 101497. doi:https://doi.org/10.1016/j.najef.2021.101497.

[212] Y. F. Bueso, M. Tangney, Synthetic biology in the driving seat of the bioeconomy, Trends in Biotechnology 35 (5) (2017) 373–378. doi:https://doi.org/10.1016/j.tibtech.2017.02.002.

[213] S.-H. Liao, H.-h. Ho, H.-w. Lin, Mining stock category association and cluster on Taiwan stock market, Expert Systems with Applications 35 (1-2) (2008) 19–29. doi:https://doi.org/10.1016/j.eswa.2007.06.001.

[214] Z. Eksi, D. Schreitl, Closing a Bitcoin Trade Optimally under Partial Information: Performance Assessment of a Stochastic Disorder Model, Mathematics 10 (1) (2022) 157. doi:https://doi.org/10.3390/math10010157.

[215] N. Atzei, M. Bartoletti, T. Cimoli, A survey of attacks on Ethereum smart contracts (sok), in: Principles of Security and Trust: 6th International Conference, POST 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings 6, Springer, 2017, pp. 164–186. doi:https://doi.org/10.1007/978-3-662-54455-6_8.

[216] A. Dutta, L. C. Voumik, A. Ramamoorthy, S. Ray, A. Raihan, Predicting Cryptocurrency Fraud Using ChaosNet: The Ethereum Manifestation, Journal of Risk and Financial Management 16 (4) (2023) 216. doi:https://doi.org/10.3390/jrfm16040216.

[217] S. K. Mallick, Causal relationship between Crypto currencies: An Analytical Study between Bitcoin and Binance Coin, The Journal of Contemporary Issues in Business and Government 26 (2) (2020) 2171–2181.

[218] E. Fix, J. Hodges, Discriminatory analysis, nonparametric discrimination, Tech. rep., Consistency Properties. Technical Report 4, USAF School of Aviation Medicine, Randolph Field. (1951).

[219] L. Breiman, J. H. Friedman, R. A. Olshen, C. J. Stone, Classification and regression trees, Chapman & Hall / CRC, 1984.

[220] C. Cortes, V. Vapnik, Support-vector networks, Machine Learning 20 (1995) 273–297. doi:https://doi.org/10.1007/BF00994018.

[221] J. A. Suykens, J. Vandewalle, Least squares support vector machine classifiers, Neural Processing Letters 9 (1999) 293–300. doi:https://doi.org/10.1023/A:1018628609742.

[222] F. Rosenblatt, The perceptron: a probabilistic model for information storage and organization in the brain., Psychological Review 65 (6) (1958) 386. doi:https://psycnet.apa.org/doi/10.1037/h0042519.

[223] K. Fukushima, Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position, Biological Cybernetics 36 (4) (1980) 193–202. doi:https://doi.org/10.1007/BF00344251.

[224] J. J. Hopfield, Neural networks and physical systems with emergent collective computational abilities., Proceedings of the National Academy of Sciences 79 (8) (1982) 2554–2558. doi:https://doi.org/10.1073/pnas.79.8.2554.

[225] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Computation 9 (8) (1997) 1735–1780. doi:https://doi.org/10.1162/neco.1997.9.8.1735.

[226] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, arXiv preprint arXiv:1412.3555 (2014). doi:https://doi.org/10.48550/arXiv.1412.3555.

[227] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu, Wavenet: A generative model for raw audio, arXiv preprint arXiv:1609.03499 (2016). doi:https://doi.org/10.48550/arXiv.1609.03499.

[228] D. Leite, I. Škrjanc, F. Gomide, An overview on evolving systems and learning from stream data, Evolving Systems 11 (2) (2020) 181–198. doi:https://doi.org/10.1007/s12530-020-09334-5.

APPENDIX A – LIST OF PUBLICATIONS

This appendix presents the list of publications resulting from the Ph.D. program.

The list of papers published in journals is as follows:

- **K. S. T. R. Alves**, R. Ballini, E. P. de Aguiar, Financial Series Forecasting: A New Fuzzy Inference System for Crisp Values and Interval-Valued Predictions, Computational Economics. Doi: https://doi.org/10.1007/s10614-024-10670-w
- **K. S. T. R. Alves**, C. D. de Jesus, E. P. de Aguiar, A New Takagi-Sugeno-Kang Model for Time Series Forecasting, Engineering Applications of Artificial Intelligence 133 (2024) 108155. Doi: https://doi.org/10.1016/j.engappai.2024.108155.

The list of conference papers published is as follows:

- **K. S. T. R. Alves**, R. Ballini, E. P. de Aguiar, A New Fuzzy Inference System Applied to Time Series Forecasting, in Anais do XVI Congresso Brasileiro de Inteligência Computacional (CBIC 2023), SBIC, pp. 1-6. Doi: http://dx.doi.org/10.21528/CBIC2023-003
- V. H. S. Pereira, **K. S. T. R. Alves**, E. P. de Aguiar, A Machine Learning approach to predict Length of Stay of vehicles in an inbound logistics operation, in Anais do XVI Congresso Brasileiro de Inteligência Computacional (CBIC 2023), SBIC, pp. 1-6. Doi: http://dx.doi.org/10.21528/CBIC2023-005
- **K. S. T. R. Alves**, D. Pekaslan, C. Wagner, E. P. de Aguiar, A New evolving Fuzzy System with Mechanisms to Deal With Uncertainties in Times Series Forecasting, in 2022 IEEE Latin American Conference on Computational Intelligence (LA-CCI), IEEE, pp. 1-6. Doi: https://doi.org/10.1109/LA-CCI54402.2022.9981311
- E. S. O. Marques, **K. S. T. R. Alves**, D. Pekaslan, E. P. de Aguiar, Kernel Evolving Participatory Fuzzy Modeling for Time Series Forecasting: New Perspectives Based on Distance Measures, in: 2022 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), IEEE, pp. 1-8. Doi: https://doi.org/10.1109/FUZZ-IEEE55066.2022.9882602
- E. S. O. Marques, **K. S. T. R. Alves**, D. Pekaslan, E. P. de Aguiar, Kernel evolving participatory fuzzy modeling for time series forecasting: New perspectives based on similarity measures, in: 2022 IEEE International Conference on Evolving and Adaptive Intelligent Systems (EAIS), IEEE, pp. 1-8. Doi: https://doi.org/10.1109/EAIS51927.2022.9787687

APPENDIX B – LIST OF PYTHON LIBRARIES

This appendix lists the open-source Python libraries developed and published during the Ph.D. program.

- **evolvingfuzzysystems:** `https://pypi.org/project/evolvingfuzzysystems/`
- **expandingnet:** `https://pypi.org/project/expandingnet/`
- **handsonmachinelearn:** `https://pypi.org/project/handsonmachinelearn/`
- **krls:** `https://pypi.org/project/krls/`
- **nfisis:** `https://pypi.org/project/nfisis/`

APPENDIX C – HOW TO IMPLEMENT THE PROPOSED MODELS

The models proposed in this work can be obtained via pip by typing: `pip install nfisis`. After installing the package, the models can be imported using the following commands:

- **New Mamdani Classifier:** `from nfisis.fuzzy import NewMamdaniClassifier as NMC`
- **New Mamdani Regressor:** `from nfisis.fuzzy import NewMamdaniRegressor as NMR`
- **New Takagi-Sugeno-Kang:** `from nfisis.fuzzy import NTSK`
- **GEN-NMR:** `from nfisis.genetic import GEN_NMR`
- **GEN-NTSK:** `from nfisis.genetic import GEN_NTSK`
- **R-NMR:** `from nfisis.ensemble import R_NMR`
- **R-NTSK:** `from nfisis.ensemble import R_NTSK`

Once the libraries have been imported, the methods `fit` and `predict` can be used, as exemplified in Algorithm 7.

Algorithm 7: Training and testing NTSK

```
1 from nfisis.fuzzy import NTSK
2 model = NTSK()
3 model.fit(X_train, y_train)
4 y_pred = model.predict(y_test)
```

For `nfisis.fuzzy` and `nfisis.genetic`, the user can invoke the methods `model.show_rules(` to display the generated rules and `model.plot_hist()` to plot the histogram of the fuzzy sets.

Finally, the list of hyperparameters for each model is detailed below:

- **New Mamdani Classifier:**

- **fuzzy_operator: {"prod", "max", "min", "minmax", "equal"}, default="minmax"** Mathematical function used in fuzzy logic to manipulate fuzzy sets and perform operations like intersection, union, and complement.

- **ponder: bool, default=True** If True, the number of samples in a rule is used to weight its firing strength.

- **New Mamdani Regressor:**

- **rules: int, default=5** The number of rules that the model will create.

- **fuzzy_operator: {"prod", "max", "min", "minmax", "equal"}, default="prod"** Mathematical function used in fuzzy logic to manipulate fuzzy sets and perform operations like intersection, union, and complement.

- **ponder: bool, default=True** If True, the number of samples in a rule is used to weight its firing strength.

- **NTSK:**

- **rules: int, default=5** The number of rules that the model will create.

- **lambda1: float in the range [0,1], default=1** The forgetting factor for the adaptive filter (adaptive_filter). Only valid for adaptive_filter = "RLS".

- **adaptive_filter: {"RLS", "wRLS"}, default="RLS"** Used to estimate the consequent parameters.

- **fuzzy_operator: {"prod", "max", "min", "minmax", "equal"}, default="prod"** Used in fuzzy logic to manipulate fuzzy sets and perform operations like intersection, union, and complement.

- **ponder: bool, default=True** If True, the number of samples in a rule is used to weight its firing strength.

- **omega: int, default=1000** Used to initialize the covariance matrix of the adaptive filtering approaches.

- **GEN-NMR:**

- **rules: int, default=5** The number of rules that the model will create.

- **fuzzy_operator: {"prod", "max", "min", "minmax", "equal"}, default="prod"** Used in fuzzy logic to manipulate fuzzy sets and perform operations like intersection, union, and complement.

- **ponder: bool, default=True** If True, the number of samples in a rule is used to weight its firing strength.

- **num_generations: int, default=10.** The total number of iterations the algorithm runs.

- **num_parents_mating: int, default=5** Specifies how many individuals (solutions) from the current population are selected to produce offspring for the next generation.

- **sol_per_population: int, default=10** Refers to the total number of solutions (individuals) in a population at any given time.

- **error_metric: {"RMSE", "NRMSE", "NDEI", "MAE", "MAPE", "CPPM"}, default="RMSE".** Defines which error metric will be implemented to evaluate the best solutions.

- **print_information: bool, default=False** Determines whether information about the training phase will be printed.

- **parallel_processing: int or None, default=None** Enables parallel processing via PyGAD[1].

• **GEN-NTSK:**

- **rules: int, default=5** The number of rules that the model will create.

- **lambda1: float in the range [0,1], default=1** Forgetting factor for the adaptive filter. Only valid for adaptive_filter = "RLS".

- **adaptive_filter: {"RLS", "wRLS"}, default="wRLS"** Used to estimate the consequent parameters.

- **fuzzy_operator: {"prod", "max", "min", "minmax", "equal"}, default="prod"** Used in fuzzy logic to manipulate fuzzy sets and perform operations like intersection, union, and complement.

- **ponder: bool, default=True** If True, the number of samples in a rule is used to weight its firing strength.

- **omega: int, default=1000** Used to initialize the covariance matrix of the adaptive filtering approaches.

- **num_generations: int, default=10.** The total number of iterations the algorithm runs.

- **num_parents_mating: int, default=5** Specifies how many individuals from the current population are selected to produce offspring.

[1] If None (Default), no parallel processing is applied. It accepts a list/tuple of 2 elements: 1) 'process' or 'thread', 2) The number of processes/threads. For example, parallel_processing=['process', 10] applies 10 processes. If a positive integer is assigned, it is used as the number of threads. Source: `https://pygad.readthedocs.io/en/latest/pygad.html`

- **sol_per_population: int, default=10** The total number of solutions in a population at any given time.

- **error_metric: {"RMSE", "NRMSE", "NDEI", "MAE", "MAPE", "CPPM"}, default="RMSE".** Defines which error metric will be implemented to evaluate the best solutions.

- **print_information: bool, default=False** Determines whether information about the training phase will be printed.

- **parallel_processing: int or None, default=None** Enables parallel processing via PyGAD.

• **R-NMR:**

- **n_estimators: int, default=100** The number of fuzzy models that will be created.

- **n_trials: int, default=5** The number of fuzzy models that will be created with different hyperparameters per iteration.

- **combination: {"mean", "median", "weighted_average"}, default="mean"** Used to combine the output of each individual estimator.

- **error_metric: {"RMSE", "NRMSE", "NDEI", "MAE", "MAPE", "CPPM"}, default="RMSE".** Defines which error metric will be implemented to evaluate the best solutions.

- **print_information: bool, default=False** Determines whether information about the training phase will be printed.

- **parallel_processing: int, default=0** Implements multiprocessing to execute n worker processes in parallel using CPU cores. If $n = 0$, the code runs sequentially. If $n = -1$, it uses all available cores. If used, the main code must be isolated using `if __name__ == '__main__':`.

• **R-NTSK:**

- **n_estimators: int, default=100** The number of fuzzy models that will be created.

- **n_trials: int, default=5** The number of fuzzy models that will be created with different hyperparameters per iteration.

- **combination: {"mean", "median", "weighted_average"}, default="mean"** Used to combine the output of each individual estimator.

- **error_metric: {"RMSE", "NRMSE", "NDEI", "MAE", "MAPE", "CPPM"}, default="RMSE".** Defines which error metric will be implemented to evaluate the best solutions.

- **print_information: bool, default=False** Determines whether information about the training phase will be printed.

- **parallel_processing: int, default=0** Implements multiprocessing to execute n worker processes in parallel using CPU cores. If $n = 0$, the code runs sequentially. If $n = -1$, it uses all available cores. If used, the main code must be isolated using `if __name__ == '__main__'`:.

APPENDIX D – HYPERPARAMETER SEARCH SPACES

Table 67 – Hyperparameter search spaces for model tuning

Model	hyperparameters
KNN [218]	n_neighbors: [2, 3, 5, 11]
Regression Tree [219]	max_depth: [2, 4, 8, 12, 16, 20], max_features: [2, 4, 6, 8, 10]
Random Forest [112]	max_depth: [2, 4, 8, 12, 16, 20], max_features: [2, 4, 6, 8, 10], n_estimators: [50, 100, 150, 200]
SVM [220]	C: [0.01,0.1,1,10], gamma: [0.01,0.5,1,10,50], kernel: [linear, rbf, sigmoid]
LS-SVM [221]	kernel: [linear]
GBM [116]	learning_rate: [0.01, 0.05, 0.1], max_depth: [2, 4, 8, 12, 16, 20], max_features: [2, 4, 6, 8, 10], n_estimators: [50, 100, 150, 200]
XGBoost [117]	eta: [0.3, 0.4, 0.5], eval_metric: [rmse], gamma: [0.3, 0.4, 0.5], max_depth: [2, 4, 8, 12, 16, 20], min_child_weight: [2, 5], n_estimators: [50, 100, 150, 200]
LGBM [118]	learning_rate: [0.01, 0.05, 0.1, 0.5], max_depth: [2, 4, 8, 12, 16, 20], max_features: [2, 4, 6, 8, 10], n_estimators: [50, 100, 150, 200]
MLP [222]	n_hidden: [0,1,2,3], n_neurons: Integer in [1, 99], activation: [elu, exponential, gelu, hard_sigmoid, linear, relu, selu, sigmoid, softplus, softsign, swish, tanh], learning_rate: Log-uniform dist. in $[10^{-5}, 0.5]$
CNN [223]	n_hidden: [0,1,2,3], n_neurons: Integer in [1, 99], learning_rate: Log-uniform dist. in $[10^{-5}, 0.5]$
RNN [224]	n_hidden: [1,2,3,4,5], n_neurons: Integer in [1, 99], learning_rate: Log-uniform dist. in $[10^{-5}, 0.5]$
LSTM [225]	n_neurons: Integer in [1, 99], n_lstm_hidden: [1,2,3,4,5], neurons_dense: Integer in [1, 99], dropout_rate: [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9], n_dense_hidden: [0,1,2,3,4], learning_rate: Log-uniform dist. in $[10^{-5}, 0.5]$
GRU [226]	filters: [2,4,8,16,32,64], kernel_size: [1,2,3,4,5], strides: [1,2,3,4,5], n_neurons: Integer in [1, 99], n_gru_hidden: [1,2,3,4,5], neurons_dense: Integer in [1, 99], dropout_rate: [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9], n_dense_hidden: [0,1,2,3,4], learning_rate: Log-uniform dist. in $[10^{-5}, 0.5]$
WaveNet [227]	dilation_rate: [(1),(1,2),(1,2,4),(1,2,4,8),(1,2,4,8,16),(1,2,4,8,16,32)], repeat: [1,2], learning_rate: Log-uniform dist. in $[10^{-5}, 0.5]$
eTS [68]	omega: [50, 100, 250, 500, 1000, 10000], r: [0.1, 0.3, 0.5, 0.7, 5, 10, 50]
Simpl_eTS [69]	omega: [50, 250, 500, 750, 1000], r: [0.1, 0.3, 0.5, 0.7]
exTS [70]	mu: [0.1, 0.3, 0.5, 0.7], omega: [50, 250, 750, 1000]
ePL [72]	alpha: [0.001, 0.01, 0.1, 0.9], beta: [0.001, 0.005, 0.01, 0.1, 0.2], lambda1: [0.001, 0.1], r: [0.1, 0.25, 0.5, 0.75], s: [100, 10000]
eMG [74]	alpha: [0.001, 0.01], lambda1: [0.1, 0.5], omega: $[10^4]$, sigma: [0.001, 0.003], w: [10, 50]
ePL+ [73]	alpha: [0.001, 0.01, 0.1], beta: [0.01, 0.1, 0.25], e_utility: [0.03, 0.05], lambda1: [1e-7, 0.001], omega: [100, 10000], pi: [0.3, 0.5], sigma: [0.5, 1, 10, 50]
ePL-KRLS-DISCO [75]	alpha: [0.05, 0.1], beta: [0.01, 0.1, 0.25], e_utility: [0.03, 0.05], lambda1: [1e-7,1e-3], sigma: [0.5, 1, 10, 50]
NMR	fuzzy_operator: [prod,min,max,minmax], rules: Integer in [1, 19]
NTSK (RLS)	adaptive_filter: [RLS], fuzzy_operator: [prod,min,max,minmax], lambda1: [0.95,0.96,0.97,0.98,0.99], rules: [1]
NTSK (wRLS)	adaptive_filter: [wRLS], fuzzy_operator: [prod,min,max,minmax], rules: Integer in [1, 19]
GEN-NMR	error_metric: [RMSE,MAE,CPPM], fuzzy_operator: [prod,min,max,minmax], num_generations: [10], num_parents_mating: [5], parallel_processing: [10], rules: [1, 3, 5, 7, 9, 11, 13, 15, 17, 19], sol_per_pop: [10]
GEN-NTSK (RLS)	error_metric: [RMSE,MAE,CPPM], adaptive_filter: [RLS], fuzzy_operator: [prod,min,max,minmax], num_generations: [10], num_parents_mating: [5], parallel_processing: [10], lambda1: [0.95,0.96,0.97,0.98,0.99], rules: [1], sol_per_pop: [10]
GEN-NTSK (wRLS)	error_metric: [RMSE,MAE,CPPM], adaptive_filter: [wRLS], fuzzy_operator: [prod,min,max,minmax], num_generations: [10], num_parents_mating: [5], parallel_processing: [10], rules: [1, 3, 5, 7, 9, 11, 13, 15, 17, 19], sol_per_pop: [10]
R-NMR	combination: [mean,median,weighted_average], n_estimators: [50]
R-NTSK	combination: [mean,median,weighted_average], n_estimators: [50]

APPENDIX E – MODELS' HYPERPARAMETERS

Table 68 – Models' hyperparameters for Lorenz Attractor time series

Model	hyperparameters
KNN [218]	n_neighbors: 2
Regression Tree [219]	max_depth: 20, max_features: 8
Random Forest [112]	max_depth: 20, max_features: 2, n_estimators: 200
SVM [220]	C: 10, gamma: 0.5, kernel: rbf
LS-SVM [221]	kernel: linear
GBM [116]	learning_rate: 0.05, max_depth: 12, max_features: 2, n_estimators: 200
XGBoost [117]	eta: 0.3, eval_metric: rmse, gamma: 0.3, max_depth: 12, min_child_weight: 2, n_estimators: 50
LGBM [118]	learning_rate: 0.5, max_depth: 16, max_features: 2, n_estimators: 200, verbosity: -1
MLP [222]	n_hidden: 0, n_neurons: 0, activation: relu, learning_rate: 0.474, input_shape: [3]
CNN [223]	n_hidden: 3, n_neurons: 57, learning_rate: 0.005
RNN [224]	n_hidden: 1, n_neurons: 53, learning_rate: 0.00058
LSTM [225]	n_neurons: 92, n_lstm_hidden: 2, neurons_dense: 1, dropout_rate: 0, n_dense_hidden: 2, learning_rate: 0.00043
GRU [226]	filters: 4, kernel_size: 3, strides: 4, n_neurons: 28, n_gru_hidden: 2, neurons_dense: 1, dropout_rate: 0, n_dense_hidden: 1, learning_rate: 0.00061
WaveNet [227]	dilation_rate: (1, 2, 4), repeat: 1, learning_rate: 0.00028
eTS [68]	omega: 10000, r: 0.1
Simpl_eTS [69]	omega: 1000, r: 0.7
exTS [70]	mu: 0.3, omega: 1000
ePL [72]	alpha: 0.9, beta: 0.01, lambda1: 0.001, r: 0.75, s: 10000
eMG [74]	alpha: 0.001, lambda1: 0.1, omega: 10000, sigma: 0.003, w: 10
ePL+ [73]	alpha: 0.01, beta: 0.25, e_utility: 0.03, lambda1: 0.25, omega: 10000, pi: 0.3, sigma: 0.1
ePL-KRLS-DISCO [75]	alpha: 0.1, beta: 0.25, e_utility: 0.05, lambda1: 1e-07, sigma: 10
NMR	fuzzy_operator: min, rules: 19
NTSK (RLS)	adaptive_filter: RLS, fuzzy_operator: prod, lambda1: 0.97, rules: 1
NTSK (wRLS)	adaptive_filter: wRLS, fuzzy_operator: prod, rules: 1
GEN-NMR	error_metric: RMSE, fuzzy_operator: min, num_generations: 10, num_parents_mating: 5, parallel_processing: 10, rules: 19, sol_per_pop: 10
GEN-NTSK (RLS)	adaptive_filter: RLS, error_metric: RMSE, fuzzy_operator: prod, lambda1: 0.99, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 1, sol_per_pop: 5
GEN-NTSK (wRLS)	adaptive_filter: wRLS, error_metric: RMSE, fuzzy_operator: max, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 19, sol_per_pop: 5
R-NMR	combination: median, n_estimators: 50
R-NTSK	combination: median, n_estimators: 50

Table 69 – Models' hyperparameters for Lorenz Attractor time series with moderate level of noise

Model	hyperparameters
KNN [218]	n_neighbors: 11
Regression Tree [219]	max_depth: 8, max_features: 8
Random Forest [112]	max_depth: 12, max_features: 2, n_estimators: 150
SVM [220]	C: 10, gamma: 10, kernel: rbf
LS-SVM [221]	kernel: linear
GBM [116]	learning_rate: 0.05, max_depth: 8, max_features: 2, n_estimators: 100
XGBoost [117]	eta: 0.3, eval_metric: rmse, gamma: 0.5, max_depth: 8, min_child_weight: 5, n_estimators: 200
LGBM [118]	learning_rate: 0.1, max_depth: 20, max_features: 2, n_estimators: 150, verbosity: -1
MLP [222]	n_hidden: 0, n_neurons: 0, activation: relu, learning_rate: 0.474, input_shape: [3]
CNN [223]	n_hidden: 3, n_neurons: 57, learning_rate: 0.005
RNN [224]	n_hidden: 1, n_neurons: 53, learning_rate: 0.00058
LSTM [225]	n_neurons: 92, n_lstm_hidden: 2, neurons_dense: 1, dropout_rate: 0, n_dense_hidden: 2, learning_rate: 0.00043
GRU [226]	filters: 4, kernel_size: 3, strides: 4, n_neurons: 28, n_gru_hidden: 2, neurons_dense: 1, dropout_rate: 0, n_dense_hidden: 1, learning_rate: 0.00061
WaveNet [227]	dilation_rate: (1, 2, 4), repeat: 1, learning_rate: 0.00028
eTS [68]	omega: 50, r: 0.1
Simpl_eTS [69]	omega: 50, r: 0.1
exTS [70]	mu: 0.7, omega: 1000
ePL [72]	alpha: 0.01, beta: 0.2, lambda1: 0.001, omega: 100, sigma: 0.1
eMG [74]	alpha: 0.001, lambda1: 0.5, omega: 10000, sigma: 0.003, w: 10
ePL+ [73]	alpha: 0.001, beta: 0.01, e_utility: 0.03, lambda1: 0.25, omega: 100, pi: 0.3, sigma: 0.1
ePL-KRLS-DISCO [75]	alpha: 0.1, beta: 0.1, e_utility: 0.05, lambda1: 0.001, sigma: 1
NMR	fuzzy_operator: prod, rules: 14
NTSK (RLS)	adaptive_filter: RLS, fuzzy_operator: prod, lambda1: 0.99, rules: 1
NTSK (wRLS)	adaptive_filter: wRLS, fuzzy_operator: prod, rules: 19
GEN-NMR	error_metric: RMSE, fuzzy_operator: prod, num_generations: 10, num_parents_mating: 5, parallel_processing: 10, rules: 15, sol_per_pop: 10
GEN-NTSK (RLS)	adaptive_filter: RLS, error_metric: RMSE, fuzzy_operator: prod, lambda1: 0.99, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 1, sol_per_pop: 5
GEN-NTSK (wRLS)	adaptive_filter: wRLS, error_metric: RMSE, fuzzy_operator: prod, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 19, sol_per_pop: 5
R-NMR	combination: weighted_average, n_estimators: 50
R-NTSK	combination: median, n_estimators: 50

Table 70 – Models' hyperparameters for Mackey-Glass time series

Model	hyperparameters
KNN [218]	n_neighbors: 2
Regression Tree [219]	max_depth: 16, max_features: 10
Random Forest [112]	max_depth: 20, max_features: 8, n_estimators: 150
SVM [220]	C: 10, gamma: 10, kernel: rbf
LS-SVM [221]	kernel: linear
GBM [116]	learning_rate: 0.05, max_depth: 12, max_features: 2, n_estimators: 150
XGBoost [117]	eta: 0.5, eval_metric: rmse, gamma: 0.3, max_depth: 8, min_child_weight: 2, n_estimators: 50
LGBM [118]	learning_rate: 0.1, max_depth: 20, max_features: 2, n_estimators: 200, verbosity: -1
MLP [222]	n_hidden: 2, n_neurons: 77, activation: relu, learning_rate: 0.145, input_shape: [4]
CNN [223]	n_hidden: 3, n_neurons: 97, learning_rate: 0.006
RNN [224]	n_hidden: 5, n_neurons: 21, learning_rate: 0.003
LSTM [225]	n_neurons: 83, n_lstm_hidden: 4, neurons_dense: 1, dropout_rate: 0, n_dense_hidden: 2, learning_rate: 0.006
GRU [226]	filters: 4, kernel_size: 2, strides: 2, n_neurons: 76, n_gru_hidden: 4, neurons_dense: 0, dropout_rate: 0, n_dense_hidden: 0, learning_rate: 0.016
WaveNet [227]	dilation_rate: (1, 2, 4), repeat: 2, learning_rate: 0.007
eTS [68]	omega: 10000, r: 0.1
Simpl_eTS [69]	omega: 1000, r: 0.1
exTS [70]	mu: 0.1, omega: 1000
ePL [72]	alpha: 0.001, beta: 0.2, lambda1: 0.1, r: 0.25, s: 10000
eMG [74]	alpha: 0.001, lambda1: 0.5, omega: 10000, sigma: 0.001, w: 10
ePL+ [73]	alpha: 0.001, beta: 0.1, e_utility: 0.05, lambda1: 0.75, omega: 100, pi: 0.5, sigma: 0.1
ePL-KRLS-DISCO [75]	alpha: 0.1, beta: 0.01, e_utility: 0.03, lambda1: 1e-07, sigma: 0.5
NMR	fuzzy_operator: prod, rules: 10
NTSK (RLS)	adaptive_filter: RLS, fuzzy_operator: prod, lambda1: 0.99, rules: 1
NTSK (wRLS)	adaptive_filter: wRLS, fuzzy_operator: prod, rules: 18
GEN-NMR	error_metric: CPPM, fuzzy_operator: minmax, num_generations: 10, num_parents_mating: 5, parallel_processing: 10, rules: 9, sol_per_pop: 10
GEN-NTSK (RLS)	adaptive_filter: RLS, error_metric: CPPM, fuzzy_operator: prod, lambda1: 0.97, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 1, sol_per_pop: 5
GEN-NTSK (wRLS)	adaptive_filter: wRLS, error_metric: MAE, fuzzy_operator: prod, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 17, sol_per_pop: 5
R-NMR	combination: median, n_estimators: 50
R-NTSK	combination: weighted_average, n_estimators: 50

Table 71 – Models' hyperparameters for Mackey-Glass time series with moderate level of noise

Model	hyperparameters
KNN [218]	n_neighbors: 11
Regression Tree [219]	max_depth: 8, max_features: 4
Random Forest [112]	max_depth: 16, max_features: 2, n_estimators: 200
SVM [220]	C: 1, gamma: 50, kernel: rbf
LS-SVM [221]	kernel: linear
GBM [116]	learning_rate: 0.05, max_depth: 8, max_features: 2, n_estimators: 100
XGBoost [117]	eta: 0.3, eval_metric: rmse, gamma: 0.3, max_depth: 8, min_child_weight: 2, n_estimators: 50
LGBM [118]	learning_rate: 0.1, max_depth: 20, max_features: 2, n_estimators: 100, verbosity: -1
MLP [222]	n_hidden: 2, n_neurons: 77, activation: relu, learning_rate: 0.145, input_shape: [4]
CNN [223]	n_hidden: 3, n_neurons: 97, learning_rate: 0.006
RNN [224]	n_hidden: 5, n_neurons: 21, learning_rate: 0.003
LSTM [225]	n_neurons: 83, n_lstm_hidden: 4, neurons_dense: 1, dropout_rate: 0, n_dense_hidden: 2, learning_rate: 0.006
GRU [226]	filters: 4, kernel_size: 2, strides: 2, n_neurons: 76, n_gru_hidden: 4, neurons_dense: 0, dropout_rate: 0, n_dense_hidden: 0, learning_rate: 0.016
WaveNet [227]	dilation_rate: (1, 2, 4), repeat: 2, learning_rate: 0.007
eTS [68]	omega: 10000, r: 0.1
Simpl_eTS [69]	omega: 1000, r: 0.1
exTS [70]	mu: 0.3, omega: 1000
ePL [72]	alpha: 0.001, beta: 0.2, lambda1: 0.001, omega: 100, sigma: 0.1
eMG [74]	alpha: 0.001, lambda1: 0.5, omega: 10000, sigma: 0.003, w: 10
ePL+ [73]	alpha: 0.001, beta: 0.01, e_utility: 0.03, lambda1: 0.25, omega: 100, pi: 0.3, sigma: 0.1
ePL-KRLS-DISCO [75]	alpha: 0.05, beta: 0.1, e_utility: 0.03, lambda1: 0.001, sigma: 0.5
NMR	fuzzy_operator: prod, rules: 14
NTSK (RLS)	adaptive_filter: RLS, fuzzy_operator: prod, lambda1: 0.99, rules: 1
NTSK (wRLS)	adaptive_filter: wRLS, fuzzy_operator: prod, rules: 19
GEN-NMR	error_metric: CPPM, fuzzy_operator: minmax, num_generations: 10, num_parents_mating: 5, parallel_processing: 10, rules: 17, sol_per_pop: 10
GEN-NTSK (RLS)	adaptive_filter: RLS, error_metric: CPPM, fuzzy_operator: prod, lambda1: 0.99, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 1, sol_per_pop: 5
GEN-NTSK (wRLS)	adaptive_filter: wRLS, error_metric: MAE, fuzzy_operator: prod, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 19, sol_per_pop: 5
R-NMR	combination: median, n_estimators: 50
R-NTSK	combination: median, n_estimators: 50

Table 72 – Models' hyperparameters for Nonlinear time series

Model	hyperparameters
KNN [218]	n_neighbors: 11
Regression Tree [219]	max_depth: 12, max_features: 8
Random Forest [112]	max_depth: 12, max_features: 2, n_estimators: 50
SVM [220]	C: 10, gamma: 0.01, kernel: rbf
LS-SVM [221]	kernel: linear
GBM [116]	learning_rate: 0.1, max_depth: 8, max_features: 8, n_estimators: 200
XGBoost [117]	eta: 0.5, eval_metric: rmse, gamma: 0.3, max_depth: 4, min_child_weight: 2, n_estimators: 50
LGBM [118]	learning_rate: 0.5, max_depth: 20, max_features: 2, n_estimators: 200, verbosity: -1
MLP [222]	n_hidden: 2, n_neurons: 87, activation: relu, learning_rate: 0.063, input_shape: [3]
CNN [223]	n_hidden: 2, n_neurons: 52, learning_rate: 0.002
RNN [224]	n_hidden: 4, n_neurons: 78, learning_rate: 0.00039
LSTM [225]	n_neurons: 28, n_lstm_hidden: 2, neurons_dense: 0, dropout_rate: 0, n_dense_hidden: 0, learning_rate: 0.024
GRU [226]	filters: 32, kernel_size: 2, strides: 1, n_neurons: 36, n_gru_hidden: 4, neurons_dense: 0, dropout_rate: 0, n_dense_hidden: 0, learning_rate: 0.016
WaveNet [227]	dilation_rate: (1, 2, 4, 8, 16), repeat: 2, learning_rate: 0.002
eTS [68]	omega: 10000, r: 0.1
Simpl_eTS [69]	omega: 1000, r: 0.1
exTS [70]	mu: 0.1, omega: 50
ePL [72]	alpha: 0.9, beta: 0.005, lambda1: 0.001, r: 0.1, s: 10000
eMG [74]	alpha: 0.01, lambda1: 0.5, omega: 10000, sigma: 0.001, w: 10
ePL+ [73]	alpha: 0.1, beta: 0.25, e_utility: 0.03, lambda1: 0.25, omega: 100, pi: 0.3, sigma: 0.1
ePL-KRLS-DISCO [75]	alpha: 0.1, beta: 0.1, e_utility: 0.03, lambda1: 1e-07, sigma: 1
NMR	fuzzy_operator: min, rules: 19
NTSK (RLS)	adaptive_filter: RLS, fuzzy_operator: prod, lambda1: 0.99, rules: 1
NTSK (wRLS)	adaptive_filter: wRLS, fuzzy_operator: min, rules: 17
GEN-NMR	error_metric: RMSE, fuzzy_operator: min, num_generations: 10, num_parents_mating: 5, parallel_processing: 10, rules: 19, sol_per_pop: 10
GEN-NTSK (RLS)	adaptive_filter: RLS, error_metric: RMSE, fuzzy_operator: prod, lambda1: 0.99, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 1, sol_per_pop: 5
GEN-NTSK (wRLS)	adaptive_filter: wRLS, error_metric: RMSE, fuzzy_operator: min, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 17, sol_per_pop: 5
R-NMR	combination: mean, n_estimators: 50
R-NTSK	combination: median, n_estimators: 50

Table 73 – Models' hyperparameters for Nonlinear time series with moderate level of noise

Model	hyperparameters
KNN [218]	n_neighbors: 11
Regression Tree [219]	max_depth: 8, max_features: 8
Random Forest [112]	max_depth: 8, max_features: 2, n_estimators: 150
SVM [220]	C: 10, gamma: 50, kernel: rbf
LS-SVM [221]	kernel: linear
GBM [116]	learning_rate: 0.05, max_depth: 8, max_features: 2, n_estimators: 100
XGBoost [117]	eta: 0.3, eval_metric: rmse, gamma: 0.3, max_depth: 8, min_child_weight: 5, n_estimators: 50
LGBM [118]	learning_rate: 0.05, max_depth: 20, max_features: 2, n_estimators: 150, verbosity: -1
MLP [222]	n_hidden: 2, n_neurons: 87, activation: relu, learning_rate: 0.063, input_shape: [3]
CNN [223]	n_hidden: 2, n_neurons: 52, learning_rate: 0.002
RNN [224]	n_hidden: 4, n_neurons: 78, learning_rate: 0.00039
LSTM [225]	n_neurons: 28, n_lstm_hidden: 2, neurons_dense: 0, dropout_rate: 0, n_dense_hidden: 0, learning_rate: 0.024
GRU [226]	filters: 32, kernel_size: 2, strides: 1, n_neurons: 36, n_gru_hidden: 4, neurons_dense: 0, dropout_rate: 0, n_dense_hidden: 0, learning_rate: 0.016
WaveNet [227]	dilation_rate: (1, 2, 4, 8, 16), repeat: 2, learning_rate: 0.002
eTS [68]	omega: 50, r: 0.1
Simpl_eTS [69]	omega: 50, r: 0.1
exTS [70]	mu: 0.1, omega: 1000
ePL [72]	alpha: 0.001, beta: 0.2, lambda1: 0.1, omega: 10000, sigma: 0.25
eMG [74]	alpha: 0.001, lambda1: 0.1, omega: 10000, sigma: 0.003, w: 50
ePL+ [73]	alpha: 0.1, beta: 0.01, e_utility: 0.03, lambda1: 0.25, omega: 100, pi: 0.3, sigma: 0.25
ePL-KRLS-DISCO [75]	alpha: 0.05, beta: 0.25, e_utility: 0.05, lambda1: 0.001, sigma: 0.5
NMR	fuzzy_operator: min, rules: 18
NTSK (RLS)	adaptive_filter: RLS, fuzzy_operator: prod, lambda1: 0.99, rules: 1
NTSK (wRLS)	adaptive_filter: wRLS, fuzzy_operator: prod, rules: 16
GEN-NMR	error_metric: RMSE, fuzzy_operator: min, num_generations: 10, num_parents_mating: 5, parallel_processing: 10, rules: 19, sol_per_pop: 10
GEN-NTSK (RLS)	adaptive_filter: RLS, error_metric: RMSE, fuzzy_operator: prod, lambda1: 0.99, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 1, sol_per_pop: 5
GEN-NTSK (wRLS)	adaptive_filter: wRLS, error_metric: RMSE, fuzzy_operator: prod, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 5, sol_per_pop: 5
R-NMR	combination: weighted_average, n_estimators: 50
R-NTSK	combination: median, n_estimators: 50

Table 74 – Models' hyperparameters for Alice 1A

Model	hyperparameters
KNN [218]	n_neighbors: 11
Regression Tree [219]	max_depth: 2, max_features: 6
Random Forest [112]	max_depth: 4, max_features: 4, n_estimators: 100
SVM [220]	C: 10, gamma: 0.01, kernel: linear
LS-SVM [221]	kernel: linear
GBM [116]	learning_rate: 0.1, max_depth: 12, max_features: 2, n_estimators: 200
XGBoost [117]	eta: 0.3, eval_metric: rmse, gamma: 0.4, max_depth: 2, min_child_weight: 2, n_estimators: 50
LGBM [118]	learning_rate: 0.01, max_depth: 2, max_features: 2, n_estimators: 200, verbosity: -1
MLP [222]	n_hidden: 1, n_neurons: 38, activation: tanh, learning_rate: 0.005, input_shape: 12
CNN [223]	n_hidden: 0, n_neurons: 0, learning_rate: 0.00068
RNN [224]	n_hidden: 1, n_neurons: 3, learning_rate: 0.02
LSTM [225]	n_neurons: 45, n_lstm_hidden: 1, neurons_dense: 1, dropout_rate: 0, n_dense_hidden: 0, learning_rate: 0.01
GRU [226]	filters: 64, kernel_size: 1, strides: 5, n_neurons: 56, n_gru_hidden: 5, neurons_dense: 1, dropout_rate: 0, n_dense_hidden: 2, learning_rate: 0.00091
WaveNet [227]	dilation_rate: (1, 2, 4), repeat: 1, learning_rate: 0.00026
eTS [68]	omega: 10000, r: 10
Simpl_eTS [69]	omega: 50, r: 0.1
exTS [70]	mu: 0.1, omega: 1000
ePL [72]	alpha: 0.1, beta: 0.1, lambda1: 0.001, r: 0.1, s: 100
eMG [74]	alpha: 0.01, lambda1: 0.1, omega: 10000, sigma: 0.003, w: 10
ePL+ [73]	alpha: 0.1, beta: 0.01, e_utility: 0.05, lambda1: 0.75, omega: 10000, pi: 0.3, sigma: 0.1
ePL-KRLS-DISCO [75]	alpha: 0.1, beta: 0.25, e_utility: 0.03, lambda1: 0.001, sigma: 10
NMR	fuzzy_operator: max, rules: 16
NTSK (RLS)	adaptive_filter: RLS, fuzzy_operator: prod, lambda1: 0.99, rules: 1
NTSK (wRLS)	adaptive_filter: wRLS, fuzzy_operator: minmax, rules: 4
GEN-NMR	error_metric: CPPM, fuzzy_operator: prod, num_generations: 10, num_parents_mating: 5, parallel_processing: 10, rules: 17, sol_per_pop: 10
GEN-NTSK (RLS)	adaptive_filter: RLS, error_metric: MAE, fuzzy_operator: prod, lambda1: 0.97, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 1, sol_per_pop: 5
GEN-NTSK (wRLS)	adaptive_filter: wRLS, error_metric: CPPM, fuzzy_operator: prod, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 3, sol_per_pop: 5
R-NMR	combination: median, n_estimators: 50
R-NTSK	combination: median, n_estimators: 50

Table 75 – Models' hyperparameters for Alice 38

Model	hyperparameters
KNN [218]	n_neighbors: 3
Regression Tree [219]	max_depth: 2, max_features: 8
Random Forest [112]	max_depth: 2, max_features: 6, n_estimators: 50
SVM [220]	C: 10, gamma: 0.01, kernel: rbf
LS-SVM [221]	kernel: linear
GBM [116]	learning_rate: 0.05, max_depth: 20, max_features: 4, n_estimators: 150
XGBoost [117]	eta: 0.4, eval_metric: rmse, gamma: 0.4, max_depth: 2, min_child_weight: 2, n_estimators: 100
LGBM [118]	learning_rate: 0.1, max_depth: 2, max_features: 2, n_estimators: 50, verbosity: -1
MLP [222]	n_hidden: 1, n_neurons: 38, activation: tanh, learning_rate: 0.005, input_shape: 12
CNN [223]	n_hidden: 0, n_neurons: 0, learning_rate: 0.033
RNN [224]	n_hidden: 1, n_neurons: 6, learning_rate: 0.08
LSTM [225]	n_neurons: 58, n_lstm_hidden: 1, neurons_dense: 1, dropout_rate: 0, n_dense_hidden: 3, learning_rate: 0.03
GRU [226]	filters: 64, kernel_size: 1, strides: 5, n_neurons: 72, n_gru_hidden: 4, neurons_dense: 0, dropout_rate: 0, n_dense_hidden: 0, learning_rate: 0.00068
WaveNet [227]	dilation_rate: (1, 2, 4, 8, 16), repeat: 1, learning_rate: 0.00026
eTS [68]	omega: 50, r: 0.1
Simpl_eTS [69]	omega: 250, r: 0.1
exTS [70]	mu: 0.5, omega: 1000
ePL [72]	alpha: 0.9, beta: 0.01, lambda1: 0.001, r: 0.1, s: 10000
eMG [74]	alpha: 0.001, lambda1: 0.1, omega: 10000, sigma: 0.003, w: 10
ePL+ [73]	alpha: 0.001, beta: 0.01, e_utility: 0.05, lambda1: 0.75, omega: 100, pi: 0.5, sigma: 0.5
ePL-KRLS-DISCO [75]	alpha: 0.05, beta: 0.01, e_utility: 0.03, lambda1: 0.001, sigma: 1
NMR	fuzzy_operator: minmax, rules: 9
NTSK (RLS)	adaptive_filter: RLS, fuzzy_operator: prod, lambda1: 0.99, rules: 1
NTSK (wRLS)	adaptive_filter: wRLS, fuzzy_operator: prod, rules: 11
GEN-NMR	error_metric: CPPM, fuzzy_operator: min, num_generations: 10, num_parents_mating: 5, parallel_processing: 10, rules: 13, sol_per_pop: 10
GEN-NTSK (RLS)	adaptive_filter: RLS, error_metric: RMSE, fuzzy_operator: min, lambda1: 0.97, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 1, sol_per_pop: 5
GEN-NTSK (wRLS)	adaptive_filter: wRLS, error_metric: MAE, fuzzy_operator: prod, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 17, sol_per_pop: 5
R-NMR	combination: mean, n_estimators: 50
R-NTSK	combination: mean, n_estimators: 50

Table 76 – Models' hyperparameters for Yulara 1

Model	hyperparameters
KNN [218]	n_neighbors: 11
Regression Tree [219]	max_depth: 2, max_features: 6
Random Forest [112]	max_depth: 2, max_features: 2, n_estimators: 150
SVM [220]	C: 10, gamma: 0.01, kernel: linear
LS-SVM [221]	kernel: linear
GBM [116]	learning_rate: 0.1, max_depth: 2, max_features: 4, n_estimators: 50
XGBoost [117]	eta: 0.3, eval_metric: rmse, gamma: 0.3, max_depth: 2, min_child_weight: 2, n_estimators: 50
LGBM [118]	learning_rate: 0.01, max_depth: 4, max_features: 2, n_estimators: 200, verbosity: -1
MLP [222]	n_hidden: 1, n_neurons: 38, activation: tanh, learning_rate: 0.005, input_shape: 15
CNN [223]	n_hidden: 0, n_neurons: 0, learning_rate: 0.031
RNN [224]	n_hidden: 3, n_neurons: 66, learning_rate: 0.44
LSTM [225]	n_neurons: 22, n_lstm_hidden: 1, neurons_dense: 1, dropout_rate: 0, n_dense_hidden: 1, learning_rate: 0.009
GRU [226]	filters: 4, kernel_size: 5, strides: 3, n_neurons: 83, n_gru_hidden: 2, neurons_dense: 0, dropout_rate: 0, n_dense_hidden: 0, learning_rate: 0.383
WaveNet [227]	dilation_rate: (1, 2), repeat: 1, learning_rate: 0.007
eTS [68]	omega: 1000, r: 5
Simpl_eTS [69]	omega: 50, r: 0.3
exTS [70]	mu: 0.7, omega: 250
ePL [72]	alpha: 0.001, beta: 0.1, lambda1: 0.001, r: 0.1, s: 100
eMG [74]	alpha: 0.01, lambda1: 0.1, omega: 10000, sigma: 0.003, w: 10
ePL+ [73]	alpha: 0.001, beta: 0.1, e_utility: 0.03, lambda1: 0.25, omega: 10000, pi: 0.3, sigma: 0.1
ePL-KRLS-DISCO [75]	alpha: 0.1, beta: 0.1, e_utility: 0.05, lambda1: 0.001, sigma: 50
NMR	fuzzy_operator: minmax, rules: 8
NTSK (RLS)	adaptive_filter: RLS, fuzzy_operator: prod, lambda1: 0.99, rules: 1
NTSK (wRLS)	adaptive_filter: wRLS, fuzzy_operator: prod, rules: 2
GEN-NMR	error_metric: RMSE, fuzzy_operator: prod, num_generations: 10, num_parents_mating: 5, parallel_processing: 10, rules: 19, sol_per_pop: 10
GEN-NTSK (RLS)	adaptive_filter: RLS, error_metric: MAE, fuzzy_operator: max, lambda1: 0.98, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 1, sol_per_pop: 5
GEN-NTSK (wRLS)	adaptive_filter: wRLS, error_metric: MAE, fuzzy_operator: max, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 19, sol_per_pop: 5
R-NMR	combination: median, n_estimators: 50
R-NTSK	combination: weighted_average, n_estimators: 50

Table 77 – Models' hyperparameters for Yulara 5

Model	hyperparameters
KNN [218]	n_neighbors: 11
Regression Tree [219]	max_depth: 4, max_features: 2
Random Forest [112]	max_depth: 2, max_features: 4, n_estimators: 150
SVM [220]	C: 10, gamma: 0.01, kernel: linear
LS-SVM [221]	kernel: linear
GBM [116]	learning_rate: 0.01, max_depth: 2, max_features: 10, n_estimators: 150
XGBoost [117]	eta: 0.3, eval_metric: rmse, gamma: 0.3, max_depth: 2, min_child_weight: 2, n_estimators: 50
LGBM [118]	learning_rate: 0.01, max_depth: 2, max_features: 2, n_estimators: 100, verbosity: -1
MLP [222]	n_hidden: 0, n_neurons: 0, activation: relu, learning_rate: 0.001, input_shape: 15
CNN [223]	n_hidden: 3, n_neurons: 91, learning_rate: 0.031
RNN [224]	n_hidden: 4, n_neurons: 64, learning_rate: 0.001
LSTM [225]	n_neurons: 35, n_lstm_hidden: 1, neurons_dense: 1, dropout_rate: 0, n_dense_hidden: 2, learning_rate: 0.006
GRU [226]	filters: 16, kernel_size: 5, strides: 3, n_neurons: 46, n_gru_hidden: 1, neurons_dense: 0, dropout_rate: 0, n_dense_hidden: 0, learning_rate: 0.022
WaveNet [227]	dilation_rate: (1, 2), repeat: 1, learning_rate: 0.044
eTS [68]	omega: 50, r: 0.7
Simpl_eTS [69]	omega: 50, r: 0.1
exTS [70]	mu: 0.1, omega: 50
ePL [72]	alpha: 0.1, beta: 0.1, lambda1: 0.1, r: 0.1, s: 100
eMG [74]	alpha: 0.001, lambda1: 0.5, omega: 10000, sigma: 0.001, w: 10
ePL+ [73]	alpha: 0.001, beta: 0.01, e_utility: 0.03, lambda1: 0.25, omega: 100, pi: 0.3, sigma: 0.1
ePL-KRLS-DISCO [75]	alpha: 0.05, beta: 0.25, e_utility: 0.05, lambda1: 0.001, sigma: 50
NMR	fuzzy_operator: max, rules: 2
NTSK (RLS)	adaptive_filter: RLS, fuzzy_operator: prod, lambda1: 0.99, rules: 1
NTSK (wRLS)	adaptive_filter: wRLS, fuzzy_operator: min, rules: 5
GEN-NMR	error_metric: RMSE, fuzzy_operator: min, num_generations: 10, num_parents_mating: 5, parallel_processing: 10, rules: 15, sol_per_pop: 10
GEN-NTSK (RLS)	adaptive_filter: RLS, error_metric: MAE, fuzzy_operator: minmax, lambda1: 0.99, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 1, sol_per_pop: 5
GEN-NTSK (wRLS)	adaptive_filter: wRLS, error_metric: RMSE, fuzzy_operator: max, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 13, sol_per_pop: 5
R-NMR	combination: median, n_estimators: 50
R-NTSK	combination: mean, n_estimators: 50

Table 78 – Models' hyperparameters for S&P 500 (horizon = 1)

Model	hyperparameters
KNN [218]	n_neighbors: 2
Regression Tree [219]	max_depth: 16, max_features: 6
Random Forest [112]	max_depth: 2, max_features: 2, n_estimators: 50
SVM [220]	C: 0.01, gamma: 0.01, kernel: linear
LS-SVM [221]	C: 0.01, gamma: 0.01, kernel: linear
GBM [116]	learning_rate: 0.01, max_depth: 2, max_features: 2, n_estimators: 50
XGBoost [117]	eta: 0.3, eval_metric: rmse, gamma: 0.3, max_depth: 2, min_child_weight: 5, n_estimators: 100
LGBM [118]	learning_rate: 0.01, max_depth: 2, max_features: 2, n_estimators: 50, verbosity: -1
MLP [222]	n_hidden: 1, n_neurons: 39, activation: relu, learning_rate: 3.153e-05, input_shape: 4
CNN [223]	n_hidden: 2, n_neurons: 96, learning_rate: 0.124
RNN [224]	n_hidden: 5, n_neurons: 96, learning_rate: 0.335
LSTM [225]	n_neurons: 58, n_lstm_hidden: 4, neurons_dense: 1, dropout_rate: 0, n_dense_hidden: 3, learning_rate: 0.005
GRU [226]	filters: 8, kernel_size: 2, strides: 5, n_neurons: 42, n_gru_hidden: 1, neurons_dense: 0, dropout_rate: 0, n_dense_hidden: 0, learning_rate: 0.01
WaveNet [227]	dilation_rate: (1, 2, 4, 8), repeat: 1, learning_rate: 0.025
eTS [68]	omega: 500, r: 0.7
Simpl_eTS [69]	omega: 50, r: 0.1
exTS [70]	mu: 0.1, omega: 50
ePL [72]	alpha: 0.001, beta: 0.2, lambda1: 0.1, r: 0.25, s: 10000
eMG [74]	alpha: 0.01, lambda1: 0.1, omega: 10000, sigma: 0.001, w: 10
ePL+ [73]	alpha: 0.001, beta: 0.01, e_utility: 0.03, lambda1: 0.75, omega: 100, pi: 0.3, sigma: 0.1
ePL-KRLS-DISCO [75]	alpha: 0.05, beta: 0.25, e_utility: 0.03, lambda1: 1e-07, sigma: 10
NMR	fuzzy_operator: minmax, rules: 6
NTSK (RLS)	adaptive_filter: RLS, fuzzy_operator: prod, lambda1: 0.99, rules: 1
NTSK (wRLS)	adaptive_filter: wRLS, fuzzy_operator: prod, rules: 1
GEN-NMR	error_metric: RMSE, fuzzy_operator: prod, num_generations: 10, num_parents_mating: 5, parallel_processing: 10, rules: 9, sol_per_pop: 10
GEN-NTSK (RLS)	adaptive_filter: RLS, error_metric: MAE, fuzzy_operator: min, lambda1: 0.98, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 1, sol_per_pop: 5
GEN-NTSK (wRLS)	adaptive_filter: wRLS, error_metric: RMSE, fuzzy_operator: prod, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 1, sol_per_pop: 5
R-NMR	combination: weighted_average, n_estimators: 50
R-NTSK	combination: weighted_average, n_estimators: 50

Table 79 – Models' hyperparameters for S&P 500 (horizon = 5)

Model	hyperparameters
KNN [218]	n_neighbors: 2
Regression Tree [219]	max_depth: 12, max_features: 2
Random Forest [112]	max_depth: 4, max_features: 8, n_estimators: 150
SVM [220]	C: 0.01, gamma: 0.01, kernel: linear
LS-SVM [221]	C: 0.01, gamma: 0.01, kernel: linear
GBM [116]	learning_rate: 0.05, max_depth: 4, max_features: 2, n_estimators: 150
XGBoost [117]	eta: 0.3, eval_metric: rmse, gamma: 0.3, max_depth: 2, min_child_weight: 2, n_estimators: 100
LGBM [118]	learning_rate: 0.01, max_depth: 2, max_features: 2, n_estimators: 50, verbosity: -1
MLP [222]	n_hidden: 0, n_neurons: 0, activation: relu, learning_rate: 0.067, input_shape: 4
CNN [223]	n_hidden: 3, n_neurons: 99, learning_rate: 0.007
RNN [224]	n_hidden: 1, n_neurons: 80, learning_rate: 0.19
LSTM [225]	n_neurons: 72, n_lstm_hidden: 5, neurons_dense: 1, dropout_rate: 0, n_dense_hidden: 3, learning_rate: 0.12
GRU [226]	filters: 8, kernel_size: 1, strides: 5, n_neurons: 58, n_gru_hidden: 2, neurons_dense: 1, dropout_rate: 0, n_dense_hidden: 4, learning_rate: 0.009
WaveNet [227]	dilation_rate: (1, 2, 4, 8, 16), repeat: 2, learning_rate: 0.003
eTS [68]	omega: 50, r: 0.5
Simpl_eTS [69]	omega: 250, r: 0.1
exTS [70]	mu: 0.7, omega: 50
ePL [72]	alpha: 0.01, beta: 0.001, lambda1: 0.001, r: 0.25, s: 10000
eMG [74]	alpha: 0.01, lambda1: 0.5, omega: 10000, sigma: 0.001, w: 10
ePL+ [73]	alpha: 0.01, beta: 0.01, e_utility: 0.05, lambda1: 0.5, omega: 100, pi: 0.3, sigma: 0.1
ePL-KRLS-DISCO [75]	alpha: 0.1, beta: 0.25, e_utility: 0.03, lambda1: 1e-07, sigma: 50
NMR	fuzzy_operator: min, rules: 3
NTSK (RLS)	adaptive_filter: RLS, fuzzy_operator: prod, lambda1: 0.98, rules: 1
NTSK (wRLS)	adaptive_filter: wRLS, fuzzy_operator: prod, rules: 6
GEN-NMR	error_metric: CPPM, fuzzy_operator: prod, num_generations: 10, num_parents_mating: 5, parallel_processing: 10, rules: 15, sol_per_pop: 10
GEN-NTSK (RLS)	adaptive_filter: RLS, error_metric: RMSE, fuzzy_operator: prod, lambda1: 0.95, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 1, sol_per_pop: 5
GEN-NTSK (wRLS)	adaptive_filter: wRLS, error_metric: RMSE, fuzzy_operator: prod, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 13, sol_per_pop: 5
R-NTSK	combination: median, n_estimators: 50

Table 80 – Models' hyperparameters for NASDAQ (horizon = 1)

Model	hyperparameters
KNN [218]	n_neighbors: 2
Regression Tree [219]	max_depth: 8, max_features: 2
Random Forest [112]	max_depth: 8, max_features: 2, n_estimators: 100
SVM [220]	C: 0.01, gamma: 0.5, kernel: sigmoid
LS-SVM [221]	C: 0.01, gamma: 10, kernel: rbf
GBM [116]	learning_rate: 0.01, max_depth: 8, max_features: 4, n_estimators: 50
XGBoost [117]	eta: 0.4, eval_metric: rmse, gamma: 0.3, max_depth: 8, min_child_weight: 5, n_estimators: 150
LGBM [118]	learning_rate: 0.05, max_depth: 8, max_features: 2, n_estimators: 150, verbosity: -1
MLP [222]	n_hidden: 0, n_neurons: 0, activation: relu, learning_rate: 0.258, input_shape: 4
CNN [223]	n_hidden: 0, n_neurons: 0, learning_rate: 0.344
RNN [224]	n_hidden: 2, n_neurons: 63, learning_rate: 0.417
LSTM [225]	n_neurons: 77, n_lstm_hidden: 4, neurons_dense: 1, dropout_rate: 0, n_dense_hidden: 2, learning_rate: 0.006
GRU [226]	filters: 16, kernel_size: 4, strides: 2, n_neurons: 90, n_gru_hidden: 4, neurons_dense: 1, dropout_rate: 0, n_dense_hidden: 4, learning_rate: 0.104
WaveNet [227]	dilation_rate: (1, 2), repeat: 1, learning_rate: 0.017
eTS [68]	omega: 100, r: 0.1
Simpl_eTS [69]	omega: 250, r: 0.3
exTS [70]	mu: 0.1, omega: 50
ePL [72]	alpha: 0.1, beta: 0.005, lambda1: 0.001, r: 0.1, s: 10000
eMG [74]	alpha: 0.01, lambda1: 0.1, omega: 10000, sigma: 0.003, w: 10
ePL+ [73]	alpha: 0.001, beta: 0.01, e_utility: 0.05, lambda1: 0.75, omega: 10000, pi: 0.5, sigma: 0.1
ePL-KRLS-DISCO [75]	alpha: 0.05, beta: 0.25, e_utility: 0.03, lambda1: 1e-07, sigma: 1
NMR	fuzzy_operator: prod, rules: 2
NTSK (RLS)	adaptive_filter: RLS, fuzzy_operator: prod, lambda1: 0.95, rules: 1
NTSK (wRLS)	adaptive_filter: wRLS, fuzzy_operator: min, rules: 15
GEN-NMR	error_metric: MAE, fuzzy_operator: prod, num_generations: 10, num_parents_mating: 5, parallel_processing: 10, rules: 3, sol_per_pop: 10
GEN-NTSK (RLS)	adaptive_filter: RLS, error_metric: MAE, fuzzy_operator: max, lambda1: 0.96, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 1, sol_per_pop: 5
GEN-NTSK (wRLS)	adaptive_filter: wRLS, error_metric: MAE, fuzzy_operator: max, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 19, sol_per_pop: 5
R-NMR	combination: median, n_estimators: 50
R-NTSK	combination: mean, n_estimators: 50

Table 81 – Models' hyperparameters for NASDAQ (horizon = 5)

Model	hyperparameters
KNN [218]	n_neighbors: 11
Regression Tree [219]	max_depth: 16, max_features: 2
Random Forest [112]	max_depth: 4, max_features: 2, n_estimators: 150
SVM [220]	C: 0.01, gamma: 0.01, kernel: linear
LS-SVM [221]	C: 0.1, gamma: 1, kernel: rbf
GBM [116]	learning_rate: 0.1, max_depth: 2, max_features: 2, n_estimators: 200
XGBoost [117]	eta: 0.4, eval_metric: rmse, gamma: 0.3, max_depth: 16, min_child_weight: 5, n_estimators: 150
LGBM [118]	learning_rate: 0.05, max_depth: 8, max_features: 2, n_estimators: 200, verbosity: -1
MLP [222]	n_hidden: 0, n_neurons: 0, activation: relu, learning_rate: 0.032, input_shape: 4
CNN [223]	n_hidden: 2, n_neurons: 73, learning_rate: 0.008
RNN [224]	n_hidden: 1, n_neurons: 70, learning_rate: 0.351
LSTM [225]	n_neurons: 58, n_lstm_hidden: 1, neurons_dense: 1, dropout_rate: 0, n_dense_hidden: 1, learning_rate: 0.151
GRU [226]	filters: 64, kernel_size: 1, strides: 3, n_neurons: 22, n_gru_hidden: 5, neurons_dense: 1, dropout_rate: 0.1, n_dense_hidden: 3, learning_rate: 0.015
WaveNet [227]	dilation_rate: (1, 2, 4, 8, 16), repeat: 1, learning_rate: 0.01
eTS [68]	omega: 50, r: 0.1
Simpl_eTS [69]	omega: 50, r: 0.1
exTS [70]	mu: 0.3, omega: 50
ePL [72]	alpha: 0.01, beta: 0.001, lambda1: 0.001, r: 0.1, s: 100
eMG [74]	alpha: 0.01, lambda1: 0.5, omega: 10000, sigma: 0.003, w: 10
ePL+ [73]	alpha: 0.001, beta: 0.1, e_utility: 0.03, lambda1: 0.25, omega: 100, pi: 0.3, sigma: 0.1
ePL-KRLS-DISCO [75]	alpha: 0.05, beta: 0.25, e_utility: 0.05, lambda1: 1e-07, sigma: 0.5
NMR	fuzzy_operator: min, rules: 4
NTSK (RLS)	adaptive_filter: RLS, fuzzy_operator: prod, lambda1: 0.97, rules: 1
NTSK (wRLS)	adaptive_filter: wRLS, fuzzy_operator: max, rules: 8
GEN-NMR	error_metric: RMSE, fuzzy_operator: prod, num_generations: 10, num_parents_mating: 5, parallel_processing: 10, rules: 15, sol_per_pop: 10
GEN-NTSK (RLS)	adaptive_filter: RLS, error_metric: RMSE, fuzzy_operator: prod, lambda1: 0.97, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 1, sol_per_pop: 5
GEN-NTSK (wRLS)	adaptive_filter: wRLS, error_metric: CPPM, fuzzy_operator: prod, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 9, sol_per_pop: 5
R-NMR	combination: median, n_estimators: 50
R-NTSK	weighted_average, n_estimators: 50

Table 82 – Models' hyperparameters for TAIEX (horizon = 1)

Model	hyperparameters
KNN [218]	n_neighbors: 2
Regression Tree [219]	max_depth: 8, max_features: 8
Random Forest [112]	max_depth: 4, max_features: 2, n_estimators: 50
SVM [220]	C: 0.01, gamma: 0.01, kernel: linear
LS-SVM [221]	C: 1, gamma: 0.5, kernel: rbf
GBM [116]	learning_rate: 0.01, max_depth: 8, max_features: 4, n_estimators: 50
XGBoost [117]	eta: 0.4, eval_metric: rmse, gamma: 0.3, max_depth: 8, min_child_weight: 2, n_estimators: 50
LGBM [118]	learning_rate: 0.01, max_depth: 2, max_features: 2, n_estimators: 50, verbosity: -1
MLP [222]	n_hidden: 0, n_neurons: 0, activation: relu, learning_rate: 0.155, input_shape: 4
CNN [223]	n_hidden: 2, n_neurons: 36, learning_rate: 0.02
RNN [224]	n_hidden: 4, n_neurons: 96, learning_rate: 0.387
LSTM [225]	n_neurons: 6, n_lstm_hidden: 4, neurons_dense: 1, dropout_rate: 0.2, n_dense_hidden: 1, learning_rate: 0.316
GRU [226]	filters: 64, kernel_size: 3, strides: 5, n_neurons: 58, n_gru_hidden: 1, neurons_dense: 1, dropout_rate: 0, n_dense_hidden: 3, learning_rate: 0.029
WaveNet [227]	dilation_rate: (1, 2, 4, 8), repeat: 1, learning_rate: 0.005
eTS [68]	omega: 1000, r: 0.5
Simpl_eTS [69]	omega: 750, r: 0.1
exTS [70]	mu: 0.5, omega: 1000
ePL [72]	alpha: 0.01, beta: 0.1, lambda1: 0.001, r: 0.25, s: 10000
eMG [74]	alpha: 0.001, lambda1: 0.1, omega: 10000, sigma: 0.003, w: 10
ePL+ [73]	alpha: 0.001, beta: 0.01, e_utility: 0.05, lambda1: 0.5, omega: 10000, pi: 0.5, sigma: 0.1
ePL-KRLS-DISCO [75]	alpha: 0.05, beta: 0.01, e_utility: 0.05, lambda1: 1e-07, sigma: 1
NMR	fuzzy_operator: min, rules: 2
NTSK (RLS)	adaptive_filter: RLS, fuzzy_operator: prod, lambda1: 0.98, rules: 1
NTSK (wRLS)	adaptive_filter: wRLS, fuzzy_operator: prod, rules: 4
GEN-NMR	error_metric: CPPM, fuzzy_operator: max, num_generations: 10, num_parents_mating: 5, parallel_processing: 10, rules: 3, sol_per_pop: 10
GEN-NTSK (RLS)	adaptive_filter: RLS, error_metric: MAE, fuzzy_operator: minmax, lambda1: 0.99, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 1, sol_per_pop: 5
GEN-NTSK (wRLS)	adaptive_filter: wRLS, error_metric: RMSE, fuzzy_operator: prod, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 11, sol_per_pop: 5
R-NMR	combination: weighted_average, n_estimators: 50
R-NTSK	combination: mean, n_estimators: 50

Table 83 – Models' hyperparameters for TAIEX (horizon = 5)

Model	hyperparameters
KNN [218]	n_neighbors: 2
Regression Tree [219]	max_depth: 8, max_features: 2
Random Forest [112]	max_depth: 4, max_features: 2, n_estimators: 100
SVM [220]	C: 0.01, gamma: 0.5, kernel: rbf
LS-SVM [221]	C: 0.01, gamma: 1, kernel: rbf
GBM [116]	learning_rate: 0.01, max_depth: 4, max_features: 2, n_estimators: 100
XGBoost [117]	eta: 0.3, eval_metric: rmse, gamma: 0.3, max_depth: 16, min_child_weight: 5, n_estimators: 50
LGBM [118]	learning_rate: 0.01, max_depth: 4, max_features: 2, n_estimators: 50, verbosity: -1
MLP [222]	n_hidden: 0, n_neurons: 0, activation: relu, learning_rate: 0.041, input_shape: 4
CNN [223]	n_hidden: 1, n_neurons: 95, learning_rate: 0.022
RNN [224]	n_hidden: 1, n_neurons: 90, learning_rate: 0.433
LSTM [225]	n_neurons: 32, n_lstm_hidden: 2, neurons_dense: 1, dropout_rate: 0, n_dense_hidden: 3, learning_rate: 0.067
GRU [226]	filters: 4, kernel_size: 1, strides: 4, n_neurons: 39, n_gru_hidden: 2, neurons_dense: 1, dropout_rate: 0, n_dense_hidden: 3, learning_rate: 0.057
WaveNet [227]	dilation_rate: (1, 2, 4, 16, 32), repeat: 2, learning_rate: 0.002
eTS [68]	omega: 50, r: 0.1
Simpl_eTS [69]	omega: 250, r: 0.5
exTS [70]	mu: 0.1, omega: 50
ePL [72]	alpha: 0.001, beta: 0.001, lambda1: 0.001, r: 0.1, s: 100
eMG [74]	alpha: 0.001, lambda1: 0.1, omega: 10000, sigma: 0.001, w: 10
ePL+ [73]	alpha: 0.001, beta: 0.01, e_utility: 0.03, lambda1: 0.75, omega: 100, pi: 0.5, sigma: 0.1
ePL-KRLS-DISCO [75]	alpha: 0.05, beta: 0.25, e_utility: 0.03, lambda1: 0.001, sigma: 50
NMR	fuzzy_operator: min, rules: 4
NTSK (RLS)	adaptive_filter: RLS, fuzzy_operator: prod, lambda1: 0.96, rules: 1
NTSK (wRLS)	adaptive_filter: wRLS, fuzzy_operator: prod, rules: 13
GEN-NMR	error_metric: RMSE, fuzzy_operator: max, num_generations: 10, num_parents_mating: 5, parallel_processing: 10, rules: 5, sol_per_pop: 10
GEN-NTSK (RLS)	adaptive_filter: RLS, error_metric: RMSE, fuzzy_operator: prod, lambda1: 0.96, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 1, sol_per_pop: 5
GEN-NTSK (wRLS)	adaptive_filter: wRLS, error_metric: RMSE, fuzzy_operator: minmax, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 15, sol_per_pop: 5
R-NMR	combination: mean, n_estimators: 50
R-NTSK	combination: mean, n_estimators: 50

Table 84 – Models' hyperparameters for Bitcoin (horizon = 1)

Model	hyperparameters
KNN [218]	n_neighbors: 5
Regression Tree [219]	max_depth: 4, max_features: 2
Random Forest [112]	max_depth: 2, max_features: 4, n_estimators: 200
SVM [220]	C: 0.01, gamma: 50, kernel: sigmoid
LS-SVM [221]	C: 0.01, gamma: 0.01, kernel: linear
GBM [116]	learning_rate: 0.01, max_depth: 2, max_features: 4, n_estimators: 50
XGBoost [117]	eta: 0.3, eval_metric: rmse, gamma: 0.3, max_depth: 2, min_child_weight: 5, n_estimators: 150
LGBM [118]	learning_rate: 0.01, max_depth: 2, max_features: 2, n_estimators: 50, verbosity: -1
MLP [222]	n_hidden: 0, n_neurons: 0, activation: relu, learning_rate: 0.084, input_shape: 9
CNN [223]	n_hidden: 3, n_neurons: 76, learning_rate: 0.024
RNN [224]	n_hidden: 4, n_neurons: 72, learning_rate: 0.149
LSTM [225]	n_neurons: 44, n_lstm_hidden: 1, neurons_dense: 1, dropout_rate: 0, n_dense_hidden: 4, learning_rate: 0.003
GRU [226]	filters: 16, kernel_size: 2, strides: 4, n_neurons: 81, n_gru_hidden: 3, neurons_dense: 1, dropout_rate: 0.4, n_dense_hidden: 2, learning_rate: 0.192
WaveNet [227]	dilation_rate: (1, 2), repeat: 2, learning_rate: 0.044
eTS [68]	omega: 250, r: 0.1
Simpl_eTS [69]	omega: 500, r: 0.5
exTS [70]	mu: 0.5, omega: 50
ePL [72]	alpha: 0.001, beta: 0.1, lambda1: 0.001, r: 0.1, s: 10000
eMG [74]	alpha: 0.01, lambda1: 0.5, omega: 10000, sigma: 0.001, w: 10
ePL+ [73]	alpha: 0.001, beta: 0.01, e_utility: 0.05, lambda1: 0.75, omega: 10000, pi: 0.3, sigma: 0.1
ePL-KRLS-DISCO [75]	alpha: 0.1, beta: 0.01, e_utility: 0.05, lambda1: 1e-07, sigma: 10
NMR	fuzzy_operator: min, rules: 19
NTSK (RLS)	adaptive_filter: RLS, fuzzy_operator: prod, lambda1: 0.99, rules: 1
NTSK (wRLS)	adaptive_filter: wRLS, fuzzy_operator: prod, rules: 1
GEN-NMR	error_metric: CPPM, fuzzy_operator: max, num_generations: 10, num_parents_mating: 5, parallel_processing: 10, rules: 19, sol_per_pop: 10
GEN-NTSK (RLS)	adaptive_filter: RLS, error_metric: MAE, fuzzy_operator: minmax, lambda1: 0.99, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 1, sol_per_pop: 5
GEN-NTSK (wRLS)	adaptive_filter: wRLS, error_metric: MAE, fuzzy_operator: max, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 1, sol_per_pop: 5
R-NMR	combination: mean, n_estimators: 50
R-NTSK	combination: mean, n_estimators: 50

Table 85 – Models' hyperparameters for Bitcoin (horizon = 5)

Model	hyperparameters
KNN [218]	n_neighbors: 11
Regression Tree [219]	max_depth: 2, max_features: 4
Random Forest [112]	max_depth: 2, max_features: 2, n_estimators: 50
SVM [220]	C: 0.01, gamma: 0.5, kernel: rbf
LS-SVM [221]	C: 0.01, gamma: 0.01, kernel: rbf
GBM [116]	learning_rate: 0.05, max_depth: 2, max_features: 2, n_estimators: 50
XGBoost [117]	eta: 0.5, eval_metric: rmse, gamma: 0.3, max_depth: 20, min_child_weight: 2, n_estimators: 50
LGBM [118]	learning_rate: 0.05, max_depth: 8, max_features: 2, n_estimators: 100, verbosity: -1
MLP [222]	n_hidden: 0, n_neurons: 0, activation: relu, learning_rate: 0.135, input_shape: 9
CNN [223]	n_hidden: 2, n_neurons: 39, learning_rate: 0.034
RNN [224]	n_hidden: 1, n_neurons: 82, learning_rate: 0.123
LSTM [225]	n_neurons: 59, n_lstm_hidden: 5, neurons_dense: 1, dropout_rate: 0, n_dense_hidden: 0, learning_rate: 0.342
GRU [226]	filters: 16, kernel_size: 2, strides: 2, n_neurons: 86, n_gru_hidden: 1, neurons_dense: 1, dropout_rate: 0.2, n_dense_hidden: 3, learning_rate: 0.022
WaveNet [227]	dilation_rate: (1, 2), repeat: 1, learning_rate: 0.15
eTS [68]	omega: 10000, r: 0.1
Simpl_eTS [69]	omega: 50, r: 0.1
exTS [70]	mu: 0.1, omega: 250
ePL [72]	alpha: 0.001, beta: 0.001, lambda1: 0.001, r: 0.1, s: 10000
eMG [74]	alpha: 0.01, lambda1: 0.5, omega: 10000, sigma: 0.003, w: 10
ePL+ [73]	alpha: 0.01, beta: 0.01, e_utility: 0.05, lambda1: 0.5, omega: 10000, pi: 0.5, sigma: 0.25
ePL-KRLS-DISCO [75]	alpha: 0.1, beta: 0.25, e_utility: 0.03, lambda1: 1e-07, sigma: 1
NMR	fuzzy_operator: max, rules: 2
NTSK (RLS)	adaptive_filter: RLS, fuzzy_operator: prod, lambda1: 0.96, rules: 1
NTSK (wRLS)	adaptive_filter: wRLS, fuzzy_operator: prod, rules: 13
GEN-NMR	error_metric: MAE, fuzzy_operator: minmax, num_generations: 10, num_parents_mating: 5, parallel_processing: 10, rules: 19, sol_per_pop: 10
GEN-NTSK (RLS)	adaptive_filter: RLS, error_metric: RMSE, fuzzy_operator: max, lambda1: 0.99, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 1, sol_per_pop: 5
GEN-NTSK (wRLS)	adaptive_filter: wRLS, error_metric: RMSE, fuzzy_operator: prod, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 9, sol_per_pop: 5
R-NMR	combination: median, n_estimators: 50
R-NTSK	combination: mean, n_estimators: 50

Table 86 – Models' hyperparameters for Ethereum (horizon = 1)

Model	hyperparameters
KNN [218]	n_neighbors: 11
Regression Tree [219]	max_depth: 12, max_features: 2
Random Forest [112]	max_depth: 4, max_features: 6, n_estimators: 100
SVM [220]	C: 0.01, gamma: 0.01, kernel: linear
LS-SVM [221]	C: 0.01, gamma: 0.01, kernel: linear
GBM [116]	learning_rate: 0.1, max_depth: 2, max_features: 4, n_estimators: 100
XGBoost [117]	eta: 0.5, eval_metric: rmse, gamma: 0.3, max_depth: 2, min_child_weight: 2, n_estimators: 100
LGBM [118]	learning_rate: 0.05, max_depth: 2, max_features: 2, n_estimators: 100, verbosity: -1
MLP [222]	n_hidden: 0, n_neurons: 0, activation: relu, learning_rate: 0.137, input_shape: 9
CNN [223]	n_hidden: 0, n_neurons: 0, learning_rate: 0.366
RNN [224]	n_hidden: 1, n_neurons: 87, learning_rate: 0.002
LSTM [225]	n_neurons: 74, n_lstm_hidden: 5, neurons_dense: 1, dropout_rate: 0.2, n_dense_hidden: 4, learning_rate: 0.002
GRU [226]	filters: 4, kernel_size: 3, strides: 2, n_neurons: 90, n_gru_hidden: 3, neurons_dense: 1, dropout_rate: 0, n_dense_hidden: 3, learning_rate: 0.00084
WaveNet [227]	dilation_rate: (1, 2, 4), repeat: 1, learning_rate: 0.006
eTS [68]	omega: 100, r: 0.1
Simpl_eTS [69]	omega: 50, r: 0.7
exTS [70]	mu: 0.7, omega: 50
ePL [72]	alpha: 0.1, beta: 0.001, lambda1: 0.1, r: 0.5, s: 100
eMG [74]	alpha: 0.001, lambda1: 0.5, omega: 10000, sigma: 0.001, w: 10
ePL+ [73]	alpha: 0.01, beta: 0.01, e_utility: 0.05, lambda1: 0.25, omega: 100, pi: 0.3, sigma: 0.1
ePL-KRLS-DISCO [75]	alpha: 0.05, beta: 0.01, e_utility: 0.05, lambda1: 1e-07, sigma: 0.5
NMR	fuzzy_operator: prod, rules: 10
NTSK (RLS)	adaptive_filter: RLS, fuzzy_operator: prod, lambda1: 0.95, rules: 1
NTSK (wRLS)	adaptive_filter: wRLS, fuzzy_operator: prod, rules: 12
GEN-NMR	error_metric: CPPM, fuzzy_operator: min, num_generations: 10, num_parents_mating: 5, parallel_processing: 10, rules: 3, sol_per_pop: 10
GEN-NTSK (RLS)	adaptive_filter: RLS, error_metric: CPPM, fuzzy_operator: prod, lambda1: 0.96, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 1, sol_per_pop: 5
GEN-NTSK (wRLS)	adaptive_filter: wRLS, error_metric: CPPM, fuzzy_operator: prod, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 7, sol_per_pop: 5
R-NMR	combination: median, n_estimators: 50
R-NTSK	combination: mean, n_estimators: 50

Table 87 – Models' hyperparameters for Ethereum (horizon = 5)

Model	hyperparameters
KNN [218]	n_neighbors: 11
Regression Tree [219]	max_depth: 8, max_features: 2
Random Forest [112]	max_depth: 20, max_features: 4, n_estimators: 150
SVM [220]	C: 0.01, gamma: 0.5, kernel: sigmoid
LS-SVM [221]	C: 1, gamma: 0.01, kernel: linear
GBM [116]	learning_rate: 0.01, max_depth: 2, max_features: 2, n_estimators: 100
XGBoost [117]	eta: 0.3, eval_metric: rmse, gamma: 0.3, max_depth: 20, min_child_weight: 5, n_estimators: 150
LGBM [118]	learning_rate: 0.05, max_depth: 12, max_features: 2, n_estimators: 150, verbosity: -1
MLP [222]	n_hidden: 0, n_neurons: 0, activation: relu, learning_rate: 0.224, input_shape: 9
CNN [223]	n_hidden: 0, n_neurons: 0, learning_rate: 0.24
RNN [224]	n_hidden: 5, n_neurons: 21, learning_rate: 0.011
LSTM [225]	n_neurons: 80, n_lstm_hidden: 5, neurons_dense: 1, dropout_rate: 0.1, n_dense_hidden: 3, learning_rate: 0.001
GRU [226]	filters: 32, kernel_size: 5, strides: 4, n_neurons: 68, n_gru_hidden: 3, neurons_dense: 1, dropout_rate: 0.2, n_dense_hidden: 1, learning_rate: 0.001
WaveNet [227]	dilation_rate: (1, 2), repeat: 1, learning_rate: 0.04
eTS [68]	omega: 50, r: 0.1
Simpl_eTS [69]	omega: 50, r: 0.7
exTS [70]	mu: 0.1, omega: 250
ePL [72]	alpha: 0.001, beta: 0.01, lambda1: 0.001, r: 0.5, s: 100
eMG [74]	alpha: 0.01, lambda1: 0.1, omega: 10000, sigma: 0.003, w: 10
ePL+ [73]	alpha: 0.001, beta: 0.01, e_utility: 0.05, lambda1: 0.25, omega: 100, pi: 0.3, sigma: 0.25
ePL-KRLS-DISCO [75]	alpha: 0.05, beta: 0.01, e_utility: 0.05, lambda1: 1e-07, sigma: 10
NMR	fuzzy_operator: min, rules: 3
NTSK (RLS)	adaptive_filter: RLS, fuzzy_operator: prod, lambda1: 0.99, rules: 1
NTSK (wRLS)	adaptive_filter: wRLS, fuzzy_operator: prod, rules: 6
GEN-NMR	error_metric: MAE, fuzzy_operator: min, num_generations: 10, num_parents_mating: 5, parallel_processing: 10, rules: 3, sol_per_pop: 10
GEN-NTSK (RLS)	adaptive_filter: RLS, error_metric: CPPM, fuzzy_operator: minmax, lambda1: 0.98, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 1, sol_per_pop: 5
GEN-NTSK (wRLS)	adaptive_filter: wRLS, error_metric: RMSE, fuzzy_operator: min, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 19, sol_per_pop: 5
R-NMR	combination: weighted_average, n_estimators: 50
R-NTSK	combination: median, n_estimators: 50

Table 88 – Models' hyperparameters for Binance (horizon = 1)

Model	hyperparameters
KNN [218]	n_neighbors: 11
Regression Tree [219]	max_depth: 16, max_features: 4
Random Forest [112]	max_depth: 16, max_features: 10, n_estimators: 50
SVM [220]	C: 0.01, gamma: 0.01, kernel: rbf
LS-SVM [221]	C: 0.01, gamma: 0.01, kernel: rbf
GBM [116]	learning_rate: 0.1, max_depth: 8, max_features: 10, n_estimators: 150
XGBoost [117]	eta: 0.4, eval_metric: rmse, gamma: 0.3, max_depth: 8, min_child_weight: 5, n_estimators: 50
LGBM [118]	learning_rate: 0.05, max_depth: 4, max_features: 2, n_estimators: 150, verbosity: -1
MLP [222]	n_hidden: 0, n_neurons: 0, activation: relu, learning_rate: 0.318, input_shape: 9
CNN [223]	n_hidden: 3, n_neurons: 85, learning_rate: 0.002
RNN [224]	n_hidden: 3, n_neurons: 71, learning_rate: 0.0004
LSTM [225]	n_neurons: 75, n_lstm_hidden: 3, neurons_dense: 1, dropout_rate: 0, n_dense_hidden: 1, learning_rate: 0.00061
GRU [226]	filters: 2, kernel_size: 5, strides: 3, n_neurons: 44, n_gru_hidden: 2, neurons_dense: 0, dropout_rate: 0, n_dense_hidden: 0, learning_rate: 0.001
WaveNet [227]	dilation_rate: (1, 2), repeat: 2, learning_rate: 0.004
eTS [68]	omega: 50, r: 0.3
Simpl_eTS [69]	omega: 50, r: 0.1
exTS [70]	mu: 0.7, omega: 50
ePL [72]	alpha: 0.001, beta: 0.005, lambda1: 0.001, r: 0.75, s: 10000
eMG [74]	alpha: 0.01, lambda1: 0.5, omega: 10000, sigma: 0.003, w: 10
ePL+ [73]	alpha: 0.1, beta: 0.01, e_utility: 0.03, lambda1: 0.25, omega: 10000, pi: 0.3, sigma: 0.5
ePL-KRLS-DISCO [75]	alpha: 0.05, beta: 0.01, e_utility: 0.03, lambda1: 0.001, sigma: 1
NMR	fuzzy_operator: prod, rules: 14
NTSK (RLS)	adaptive_filter: RLS, fuzzy_operator: prod, lambda1: 0.99, rules: 1
NTSK (wRLS)	adaptive_filter: wRLS, fuzzy_operator: prod, rules: 5
GEN-NMR	error_metric: MAE, fuzzy_operator: max, num_generations: 10, num_parents_mating: 5, parallel_processing: 10, rules: 3, sol_per_pop: 10
GEN-NTSK (RLS)	adaptive_filter: RLS, error_metric: RMSE, fuzzy_operator: minmax, lambda1: 0.97, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 1, sol_per_pop: 5
GEN-NTSK (wRLS)	adaptive_filter: wRLS, error_metric: CPPM, fuzzy_operator: prod, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 1, sol_per_pop: 5
R-NMR	combination: mean, n_estimators: 50
R-NTSK	combination: mean, n_estimators: 50

Table 89 – Models' hyperparameters for Binance (horizon = 5)

Model	hyperparameters
KNN [218]	n_neighbors: 2
Regression Tree [219]	max_depth: 16, max_features: 2
Random Forest [112]	max_depth: 4, max_features: 2, n_estimators: 150
SVM [220]	C: 1, gamma: 0.01, kernel: linear
LS-SVM [221]	C: 0.01, gamma: 0.01, kernel: linear
GBM [116]	learning_rate: 0.1, max_depth: 4, max_features: 4, n_estimators: 200
XGBoost [117]	eta: 0.4, eval_metric: rmse, gamma: 0.3, max_depth: 8, min_child_weight: 2, n_estimators: 50
LGBM [118]	learning_rate: 0.5, max_depth: 20, max_features: 2, n_estimators: 200, verbosity: -1
MLP [222]	n_hidden: 0, n_neurons: 0, activation: relu, learning_rate: 0.107, input_shape: 9
CNN [223]	n_hidden: 3, n_neurons: 70, learning_rate: 0.001
RNN [224]	n_hidden: 5, n_neurons: 92, learning_rate: 0.0028
LSTM [225]	n_neurons: 53, n_lstm_hidden: 3, neurons_dense: 1, dropout_rate: 0, n_dense_hidden: 2, learning_rate: 0.00055
GRU [226]	filters: 32, kernel_size: 5, strides: 5, n_neurons: 45, n_gru_hidden: 3, neurons_dense: 0, dropout_rate: 0, n_dense_hidden: 0, learning_rate: 0.002
WaveNet [227]	dilation_rate: (1, 2), repeat: 1, learning_rate: 0.047
eTS [68]	omega: 50, r: 0.1
Simpl_eTS [69]	omega: 50, r: 0.3
exTS [70]	mu: 0.1, omega: 1000
ePL [72]	alpha: 0.001, beta: 0.01, lambda1: 0.001, r: 0.1, s: 10000
eMG [74]	alpha: 0.01, lambda1: 0.5, omega: 10000, sigma: 0.003, w: 10
ePL+ [73]	alpha: 0.001, beta: 0.01, e_utility: 0.05, lambda1: 0.25, omega: 100, pi: 0.5, sigma: 0.25
ePL-KRLS-DISCO [75]	alpha: 0.1, beta: 0.25, e_utility: 0.03, lambda1: 1e-07, sigma: 1
NMR	fuzzy_operator: prod, rules: 10
NTSK (RLS)	adaptive_filter: RLS, fuzzy_operator: prod, lambda1: 0.95, rules: 1
NTSK (wRLS)	adaptive_filter: wRLS, fuzzy_operator: prod, rules: 16
GEN-NMR	error_metric: CPPM, fuzzy_operator: min, num_generations: 10, num_parents_mating: 5, parallel_processing: 10, rules: 9, sol_per_pop: 10
GEN-NTSK (RLS)	adaptive_filter: RLS, error_metric: MAE, fuzzy_operator: max, lambda1: 0.99, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 1, sol_per_pop: 5
GEN-NTSK (wRLS)	adaptive_filter: wRLS, error_metric: CPPM, fuzzy_operator: prod, num_generations: 5, num_parents_mating: 5, parallel_processing: 10, rules: 1, sol_per_pop: 5
R-NMR	combination: median, n_estimators: 50
R-NTSK	combination: mean, n_estimators: 50