TECHNICAL UNIVERSITY OF CRETE

School of Electrical And Computer Engineering

Diploma Thesis

Application and Deep Learning Algorithms for the Short-Term and Medium-Term Wind Speed Prediction

Author:
Dimitrios Kourtidis

ThesisComitee:
George Stavrakakis(Supervisor)
Michael Zervakis
Fotios Kanellos

A thesis submitted in partial fulllment of the requirements for the Diploma of Electrical and Computer Engineering

Chania, Greece
January 2024

Acknowledgements

After a series of unexpected events and situations that emerged during my undergraduate studies, I would like to thank the people who, with their role, have helped me to accomplish this important goal in my life. Heartfelt thanks to my family and friends for their unwavering encouragement, understanding, and patience during the challenging phases of this academic journey. Their support has been my source of strength and motivation.

Also I would like to thank my supervisor professor George Stavrakakis who gave me valuable knowledge in the field of renewable resources and electric machines and for his trust who accepted me. Without his contribution this diploma would not have been realized.

Last but not least, I would like to thank my thesis advisor Marios Antonakakis, for giving me the inspiration and the will to learn and to work for new scientic fields. His support was invaluable, whenever I asked for it.

ABSTRACT

Wind speed forecasting plays a pivotal role in various industries, particularly in the realms of environmental management and investments. Accurate predictions of wind speed are crucial for optimizing the performance of wind energy systems, aiding in efficient power generation and grid integration. Machine learning methods for wind speed forecasting leverage historical meteorological data and advanced algorithms to enhance prediction accuracy. Supervised and unsupervised learning techniques, including regression and neural networks, analyze past weather patterns to identify relationships and patterns for predicting future wind speeds.

A big challenge in machine learning lies in the size and quality of the available datasets. When dealing with small datasets, models may struggle to generalize effectively, potentially leading to overfitting and limited predictive capabilities. Due to the small of the used dataset in this study, one of the primary goals is to increase its size using signal augmentation techniques. This was necessary for increasing the efficiency of wind speed forecasting which counts as the second goal of this thesis. The generation of the new data was done in the frequency domain according to the Adjusted Amplitude Fourier Transform (AAFT) algorithm. Then, the goal was to develop and apply advanced deep learning for wind speed forecasting. The two models were a stacked autoencoder (SAE) and stacked independently recurrent autoencoder (SIRAE). The innovation in the present work lies in the design, implementation and application of SAE and SIRAE on the basis of signal based augmentation by means of AAFT.

We first apply the AAFT algorithm to increase the size of the used dataset and we then divide it by month due to the seasonality of the wind speed,then we create subsequences, of different sizes, of the data that will be taken into account for the current forecast and then we train the models. The aim of this work is to create models with very high success rates and robustness and to be compared with the results from literature as well as the most suitable data subsequence size. The comparison between these two models made taking into account the MSE and RMSE of the forecasts. Both models showed high performance, so to draw clearer conclusions the RMSE of the two models was averaged.

In coclusion the SIRAE model has bigger accuracy than SAE model, specifically, we will show that the SIRAE alone has a 12.5% higher performance than the SAE for the medium-term forecast and 15.7% for the short-term forecast respectively. Also, the most appropriate size of the subsequences was shown to be equal to 5, in this case we had the best results for both models. Finally the percentages mentioned were obtained by calculating the average of the RMSE for both models with subsequences size equals to 5.

ABSTRACT

Η πρόβλεψη ταχύτητας ανέμου διαδραματίζει κεντρικό ρόλο σε διάφορους κλάδους, ιδιαίτερα στους τομείς της περιβαλλοντικής διαχείρισης και των επενδύσεων. Οι ακριβείς προβλέψεις της ταχύτητας του ανέμου είναι ζωτικής σημασίας για τη βελτιστοποίηση της απόδοσης των συστημάτων αιολικής ενέργειας, βοηθώντας στην αποδοτική παραγωγή ενέργειας και στην ενοποίηση του δικτύου. Οι μέθοδοι μηχανικής εκμάθησης για την πρόβλεψη της ταχύτητας του ανέμου αξιοποιούν ιστορικά μετεωρολογικά δεδομένα και προηγμένους αλγόριθμους για τη βελτίωση της ακρίβειας πρόβλεψης. Οι εποπτευόμενες και μη εποπτευόμενες τεχνικές εκμάθησης, συμπεριλαμβανομένων της παλινδρόμησης και των νευρωνικών δικτύων, αναλύουν προηγούμενα καιρικά μοτίβα για να εντοπίσουν σχέσεις και μοτίβα για την πρόβλεψη μελλοντικών ταχυτήτων ανέμου.

Μια μεγάλη πρόκληση στη μηχανική μάθηση έγκειται στο μέγεθος και την ποιότητα των διαθέσιμων συνόλων δεδομένων. Όταν ασχολούμαστε με μικρά σύνολα δεδομένων, τα μοντέλα μπορεί να δυσκολεύονται να γενικεύσουν αποτελεσματικά, οδηγώντας δυνητικά σε υπερβολική προσαρμογή και περιορισμένες δυνατότητες πρόβλεψης. Λόγω του μικρού συνόλου δεδομένων που χρησιμοποιείται σε αυτή τη μελέτη, ένας από τους πρωταρχικούς στόχους είναι να αυξηθεί το μέγεθός του χρησιμοποιώντας τεχνικές αύξησης σήματος. Αυτό ήταν απαραίτητο για την αύξηση της αποτελεσματικότητας της πρόβλεψης ταχύτητας ανέμου που μετράει ως ο δεύτερος στόχος αυτής της διατριβής. Η παραγωγή των νέων δεδομένων έγινε στο πεδίο συχνοτήτων σύμφωνα με τον αλγόριθμο προσαρμοσμένου πλάτους μετασχηματισμού Fourier (AAFT). Στη συνέχεια, ο στόχος ήταν να αναπτυχθεί και να εφαρμοστεί προηγμένη βαθιά εκμάθηση για την πρόβλεψη ταχύτητας ανέμου. Τα δύο μοντέλα ήταν ένας αυτοκωδικοποιητής στοίβαξης (SAE) και αυτοκωδικοποιητές επαναλαμβανόμενων ανεξάρτητων νευρωνικών δικτύων (SIRAE). Η καινοτομία στην παρούσα εργασία έγκειται στον σχεδιασμό, την υλοποίηση και την εφαρμογή του SAE και του SIRAE με βάση την ενίσχυση σήματος μέσω του AAFT.

Αρχικά εφαρμόζουμε τον αλγόριθμο AAFT για να αυξήσουμε το μέγεθος του χρησιμοποιούμενου συνόλου δεδομένων και στη συνέχεια το διαιρούμε ανά μήνα λόγω της εποχικότητας της ταχύτητας του ανέμου και μετά δημιουργούμε υποακολουθίες, διαφορετικών μεγεθών, των δεδομένων που θα ληφθούν υπόψη για τη τρέχουσα πρόβλεψη και μετά εκπαιδεύουμε τα μοντέλα. Στόχος αυτής της εργασίας είναι η δημιουργία μοντέλων με πολύ υψηλά ποσοστά επιτυχίας και ευρωστίας και η σύγκριση με τα αποτελέσματα από τη βιβλιογραφία καθώς και το καταλληλότερο μέγεθος υποακολουθίας δεδομένων. Η σύγκριση μεταξύ αυτών των δύο μοντέλων έγινε λαμβάνοντας υπόψη το MSE και το RMSE των προβλέψεων. Και τα δύο μοντέλα εμφάνισαν υψηλές επιδόσεις, επομένως για να βγουν πιο ξεκάθαρα συμπεράσματα το RMSE των δύο μοντέλων υπολογίστηκε κατά μέσο όρο.

Συμπερασματικά το μοντέλο SIRAE έχει μεγαλύτερη ακρίβεια από το μοντέλο SAE, συγκεκριμένα, θα δείξουμε ότι μόνο το SIRAE έχει 12,5% υψηλότερη απόδοση από το SAE για τη μεσοπρόθεσμη πρόβλεψη και 15,7% για τη βραχυπρόθεσμη πρόβλεψη αντίστοιχα. Επίσης, το καταλληλότερο μέγεθος των υποακολουθιών φάνηκε να είναι ίσο με 5, σε αυτή την περίπτωση είχαμε τα καλύτερα αποτελέσματα και για τα δύο μοντέλα. Τέλος, τα ποσοστά που αναφέρονται προέκυψαν με τον υπολογισμό του μέσου όρου του RMSE και για τα δύο μοντέλα με μέγεθος υποακολουθιών ίσο με 5.

Contents

Acknowledgements......3
Abstract......5
Abstract......7
Contents......9
List of Figures......10
List of Tables 11
List of Abbreviations......12
CHAPTER 1: Introduction......13
CHAPTER 2: Data Description 16
CHAPTER 3: Artificial Intelligence 17
3.1: Machine Learning 18
3.2: Deep Learning 21
3.3: State of the Art 22
3.4: Methods for Time-series forecast 24
3.5: LSTM Model 28
3.6: SAE, SIRAE Models......30
CHAPTER 4: Data Augmentation......36
4.1: Data Surrogation Time and Frequency Domain......37
4.2: AAFT Algorithm......39
CHAPTER 5:Imputation......41
5.1: Other Methods......41
CHAPTER 6: Model Design......43
6.1: Medium-Term Forecasting......43
6.2: Short-Term Forecasting......46
CHAPTER 7: Results Medium-Term & Short- Term Forecastings......49
7.1: Traditional methods of forecasting......49
7.2: Results of Medium-term forecasting......52
7.3: Results of Short-term forecasting......55
CHAPTER 8: Comparison of Results......58
CHAPTER 9: Conclusions Observations......66
Bibliography......68
Figure Attachments......72

List of Figures

Figure 1: Wind Energy Park....13
Figure 2:Total installed MW in Greece per year....14
Figure 3: The Dia Island....16
Figure 4: Neuron and Machine Learning....17
Figure 5: Supervised Learning....18
Figure 6: Unsupervised Learning....18
Figure 7: Rainforcement Learning....19
Figure 8: Deep Learning....21
Figure 9: The Schematic Of Lstm....29
Figure 10: The Structure Of Stacked Autoencoders With Lstm Layers....31
Figure 11: The SIRAE Structure With IndRNN Layers....35
Figure 12: The Histogramm Of All Surrogates For a Month....40
Figure 13: The Bounds Of Data Surrogates For January....43
Figure 14: Block Diagramm For Wind Speed Forecasting....46
Figure 15: Block Diagramm Imputation And Data Surrogation....48
Figure 16: The fixed or sliding rolling window....49
Figure 17: The expanding rolling window....50
Figure 18: The bar plot of MSE performance for SAE (Medium-Term)....59
Figure 19: The bar plot of MSE performance for SIRAE (Medium-Term)....60
Figure 20: The bar plot of MSE performance for SAE (Short-Term)....61
Figure 21: The bar plot of MSE performance for SIRAE (Short-Term)....62
Figure 22: RMSE average performance (Medium-Term)....63
Figure 23: RMSE average performance (Short-Term)....65

List of Tables

Table 1: The SAE performance results for the medium term forecast (MSE)....................................58

Table 2: The SIRAE performance results for the medium term forecast (MSE)................................59

Table 3: The SAE performance results for the short term forecast (MSE)..60

Table 4: The SIRAE performance results for the short term forecast (MSE)....................................61

Table 5: The SAE performance results for the medium term forecast (RMSE).................................62

Table 6: The SIRAE performance results for the medium term forecast (RMSE).............................63

Table 7: Mean values of SAE model (RMSE) medium-term forecasting...63

Table 8: Mean values of SIRAE model (RMSE) medium-term forecasting.......................................63

Table 9: The SAE performance results for the short term forecast (RMSE).....................................64

Table 10: The SIRAE performance results for the short term forecast (RMSE)...............................64

Table 11: Mean values of SAE model (RMSE) short-term forecasting...64

Table 12: Mean values of SIRAE model (RMSE) short-term forecasting...64

List of Abbreviations

AI:	Artificial Intelligence
ML:	Machine Learning
DL:	Deep Learning
ARMA:	Autoregressive moving average
AR:	Autoregressive
ARIMA:	Autoregressive integrated moving average
AAFT:	Amplitude Adjusted Fourier Transform
SAE:	Stacked Autoencoders
SIRAE:	Stacked Independently Reccurent Neural Networks Autoencoders
LSTM:	Long- Short Term Memory
IoT:	Internet of Things
ReLU:	Rectified Linear Unit
RNN:	Reccurent Neural Networks
GRU:	Gated Recurrent Units
TCN:	Temporal Convolutional Network
AutoML:	Automated Machine Learning
MA:	Moving Average
GDP:	Gross Domestic Product
SARIMA:	Seasonal Autoregressive integrated moving average
NLP:	Natural Language Processing
MSE:	Mean Squared Error
MAPE:	Mean absolute percentage error
MAE :	Mean absolute error
SMAPE :	Symmetric mean absolute percentage error
NMSE :	Normalized mean squared error
NRMSE :	Normalized root mean squared error
NMAPE :	Normalized mean absolute percentage error
IndRNN :	Indepentently Recurrent Neural Network

CHAPTER 1
Introduction

Wind speed prediction holds a pivotal role across various industries, especially in environmental management and investments. Accurate forecasts of wind speed are essential for optimizing the performance of wind energy systems, contributing to efficient power generation and seamless grid integration. In the environmental sector, precise predictions facilitate better planning for renewable energy projects, ensuring sustainability and minimizing environmental impact. Investors heavily rely on wind speed forecasts to make informed decisions about the feasibility and profitability of wind energy ventures.

Figure 1: Wind-Energy Park

Machine learning techniques for time series prediction, particularly in the context of wind speed forecasting, have proven to be invaluable in enhancing the accuracy and efficiency of predictions. Traditional methods often struggle to capture the complex and dynamic nature of wind patterns. In contrast, machine learning algorithms, such as recurrent neural networks (RNNs), long short-term memory networks (LSTMs), and convolutional neural networks (CNNs), excel in recognizing temporal dependencies and non-linear patterns within time series data. These models can effectively extract meaningful features from historical wind speed data, allowing them to adapt to changing weather conditions and seasonal variations. Additionally, ensemble techniques, like random forests and gradient boosting, offer robustness by combining multiple models to mitigate individual weaknesses. Incorporating meteorological features, such as temperature, pressure, and humidity, further enhances the predictive capabilities of machine learning models. Overall, the application of machine learning in wind speed prediction facilitates more accurate forecasts, aiding in the optimization of renewable energy resources and supporting efficient energy management systems.

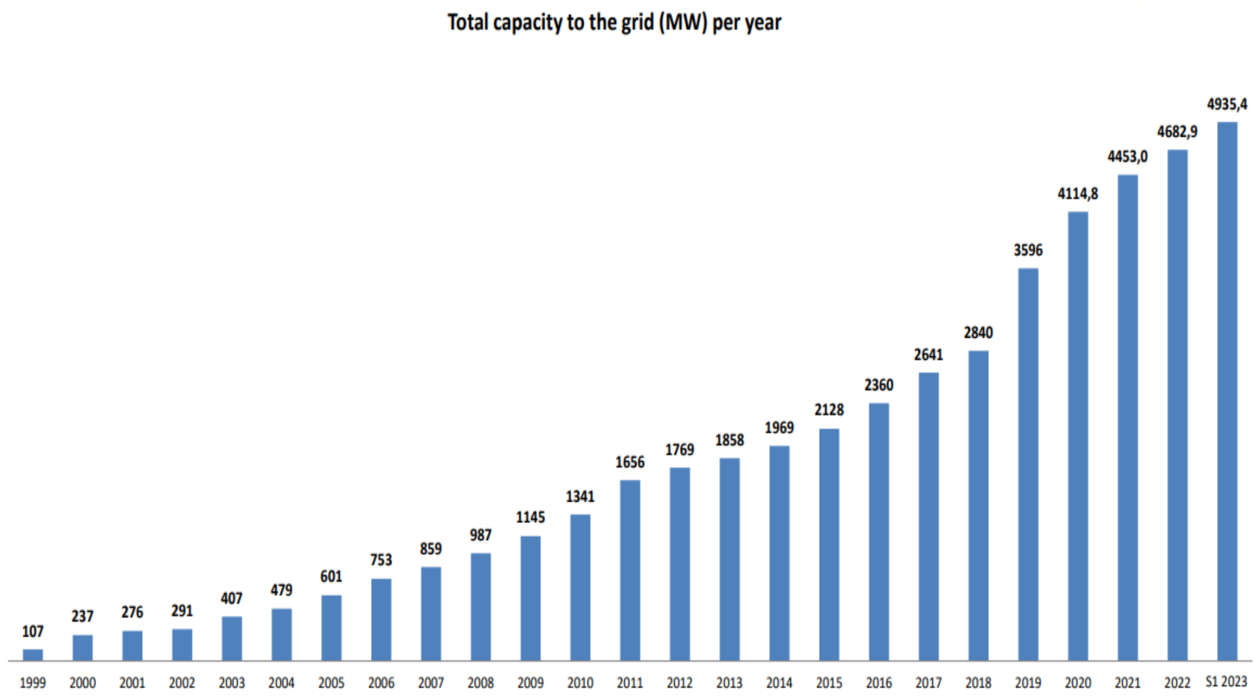


Figure 2: Total installed MW in Greece per year

Despite the advancements in machine learning techniques for time series prediction in wind speed, there are still some challenges and restrictions that need to be addressed. One notable limitation is the sensitivity of these models to the quality and availability of data. Incomplete or inaccurate historical wind speed data can lead to suboptimal predictions, emphasizing the need for high-quality datasets. Additionally, the interpretability of complex machine learning models remains a concern, as stakeholders often require transparent insights into the decision-making process. Ensuring the trustworthiness and accountability of these models is crucial, especially in applications where the consequences of inaccurate predictions can be significant. Another challenge lies in the dynamic nature of weather patterns, as sudden changes or extreme events can pose difficulties for models trained on historical data. Continual model adaptation and robustness to outliers are essential for addressing these issues. Despite these current restrictions, ongoing research and advancements in machine learning methodologies are working towards overcoming these challenges and further improving the reliability of wind speed predictions.

The need to carry out this diplomacy arises from two issues. the first is to solve the problem of lack of data both quantitatively and qualitatively. Without a competent and complete data set, no functional prediction from any model can work. Next, the goal of data generation was achieved in the field of frequency through the AAFT. The second equally important requirement was to create highly accurate models because as we mentioned a good forecast has multiple profits. Thus we implemented two models (SAE-LSTM),(SIRAE-LSTM) that until now are mainly found in applications where images are processed as data. So the purpose was to see the behavior of these models with time series as data and finally to conclude how efficient they are.

The structure of the diplomatic thesis is as follows: In chapter 2 the data and the area where the measurements were made are presented. In chapter 3 the theoretical background, related to the techniques and the models applied for the medium- and short-term forecast. In chapter 4 there is a discussion about data augmentation for the fields in which it can be implemented and with which algorithms, an analysis is also made for the AAFT algorithm. In chapter 5 an analysis is made for the imputation method but also for other methods. In chapter 6 we present the design of the models implemented for both medium and short term forecasting. Then the results of the forecasts for the medium and short term are presented in chapter 7. In chapter 8 compares the results for medium and short term forecasting. In chapter 9 we draw conclusions and observations and at the end ideas for future research are listed.

CHAPTER 2

Data Description

Dia is an islet of the Cretan Sea which is located north of Heraklion, from which it is 7 nautical miles away. It has an area of 11.91 km^2, with a maximum length and width of 5 and 3 km respectively. It belongs to the community of Elia of the Municipality of Hersonissos in the Regional Unit of Heraklion of Crete.

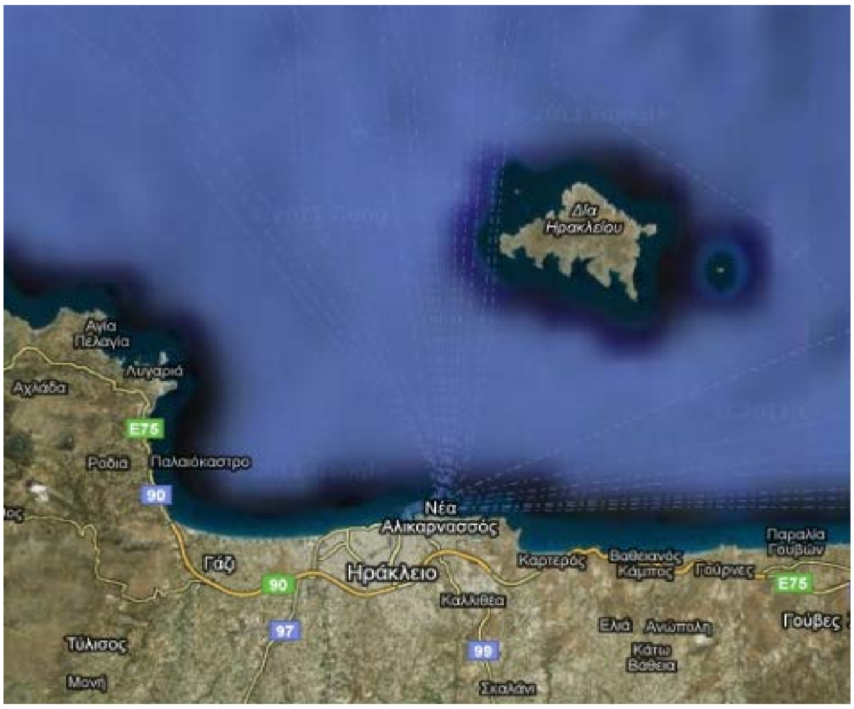


Figure 3:The Dia island

The measurements we used for this work, which are part of an ongoing survey carried out in the same laboratory [1], were made on this island at a height of 10m. Specifically, the following quantities have been recorded:

- Temperature (°C)
- Relative Humidity (%)
- Wind Speed (m/s)
- Wind Direction (°)
- Vapor Pressure (hPa)
- Solar Radiation (W/m2)
- Average Rainfall (mm)

The provided figures represent hourly data records, specifically showcasing updated rates for the years 2007, 2008, and 2009. Before these data points undergo modeling, they undergo a pre-processing phase involving two key steps. The first step involves addressing the issue of missing data, ensuring a comprehensive dataset for analysis. The second step entails normalization, a process designed to standardize the data, making it more uniform and facilitating meaningful comparisons across different variables. It's important to highlight that these steps are crucial in preparing the data for subsequent analyses, enhancing its reliability and usability in various models. In this thesis , of all the data given to us, we will only use the wind speed measurements because we will set up univariate models.

CHAPTER 3
Artificial Intelligence

Artificial intelligence (AI) draws inspiration from the intricate architecture of the human brain, where the concepts of dendrites, axons, and synapses play a crucial role in information processing. In the realm of AI, dendrites can be likened to the input layer of neural networks, receiving and transmitting signals to the artificial neurons. The axons, analogous to the output layer, carry the processed information to produce the final output. Meanwhile, synapses represent the connections between artificial neurons, where weights are assigned to regulate the strength of these connections. The process of learning in AI involves adjusting these synaptic weights based on the input data and desired output, akin to the strengthening or weakening of connections in the human brain through experience. This bio-inspired model allows artificial neural networks to adapt and learn from data, enabling AI systems to perform complex tasks, recognize patterns, and make decisions, mirroring the remarkable cognitive capabilities observed in the human brain[2].

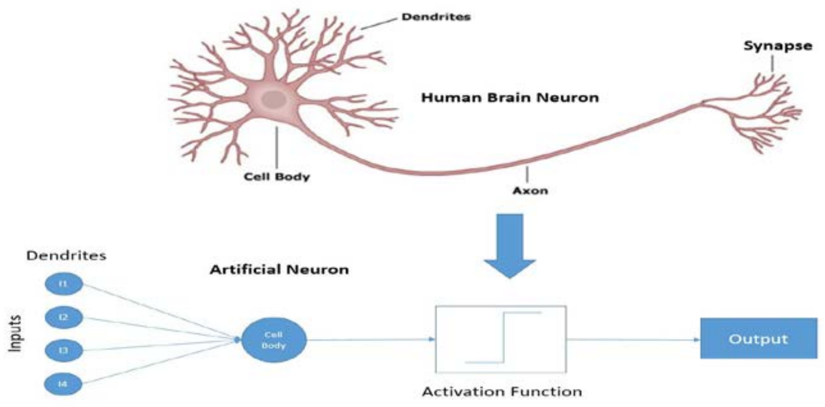


Figure 4: Neuron and Machine Learning

The applications of artificial intelligence (AI) span across various industries, revolutionizing the way tasks are performed and problems are addressed. In healthcare, AI is employed for diagnostic purposes, drug discovery, and personalized medicine, enhancing the accuracy and efficiency of medical processes. In finance, AI algorithms analyze vast datasets to predict market trends, manage risks, and automate trading strategies. In transportation, AI is integral to the development of autonomous vehicles, optimizing traffic flow, and improving safety. In manufacturing, AI-driven robotics and automation streamline production processes, increasing efficiency and reducing costs. Natural Language Processing (NLP) and computer vision technologies, both subsets of AI, find applications in customer service, content moderation, and image recognition. Moreover, AI is transforming education, agriculture, cybersecurity, and many other fields, showcasing its versatility and potential to drive innovation and positive change across diverse sectors of society.

3.1: Machine Learning

Machine Learning (ML) stands at the forefront of technological innovation, revolutionizing industries and reshaping the way we interact with the world. It is a subset of artificial intelligence (AI) that empowers systems to learn and improve from experience without explicit programming. At its core, machine learning involves the development of algorithms that allow computers to analyze data, identify patterns, and make intelligent decisions or predictions. There are three main types of machine learning: supervised learning, unsupervised learning, and reinforcement learning.

1. Supervised Learning: This involves training a model on a labeled dataset, where the algorithm learns to map input data to the corresponding output. It is commonly used in tasks such as classification and regression.

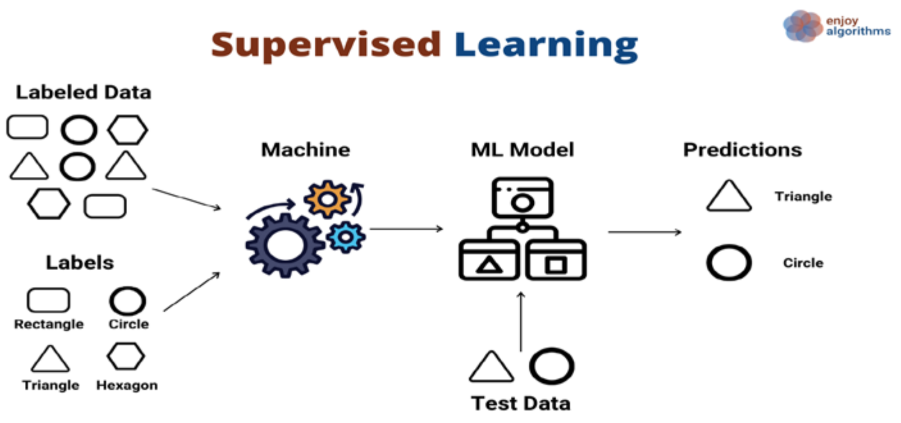


Figure 5: Supervised Learning

2. Unsupervised Learning: In this approach, the algorithm is given unlabeled data and must identify patterns or relationships within the dataset. Clustering and dimensionality reduction are common applications of unsupervised learning.

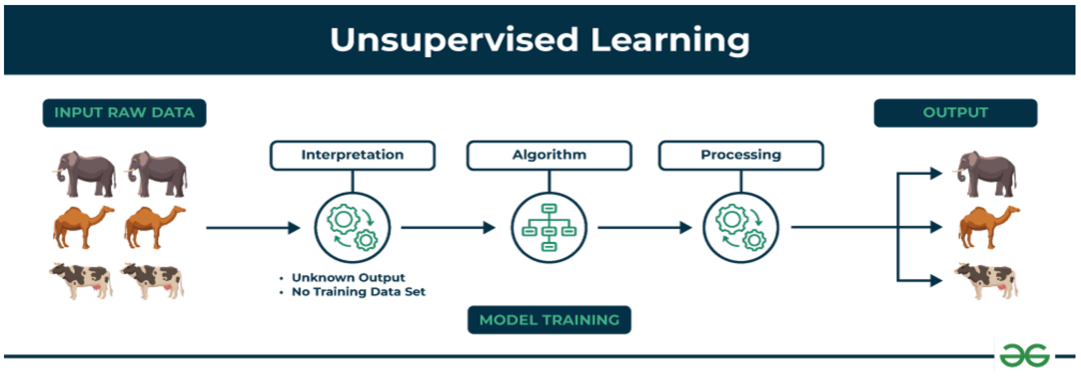


Figure 6: Unsupervised Learning

3. Reinforcement Learning: This type of learning is inspired by behavioral psychology, where an agent learns to make decisions by interacting with an environment and receiving feedback in the form of rewards or penalties.

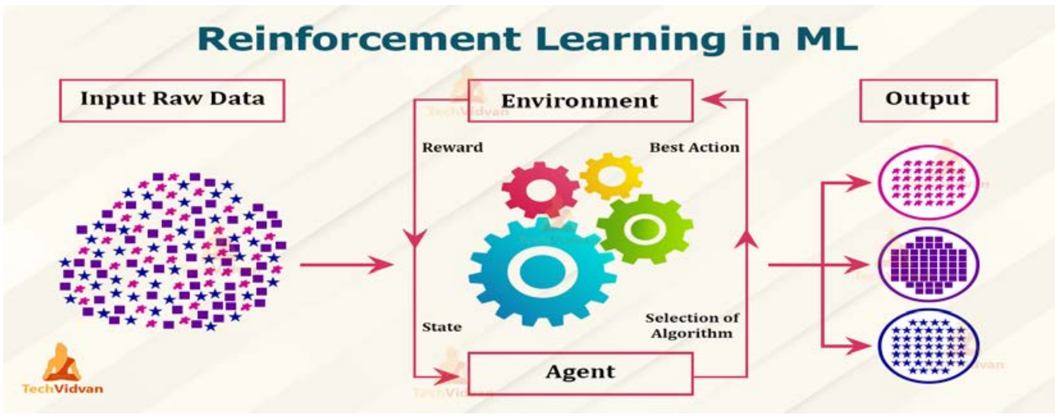


Figure 7: Rainforcement Learning

Classification:

In this category there are discrete data, instead of continuous. The prices output, correspond to categories in which the vectors are classified entrance. Consequently, in presenting new data, we do not attempt to approximate the value of a function, but rank our input some of the categories that are known. These charges usually they are called classes and can have numerical values, be symbols, etc. Generally, depending on the nature of the problem being studied, the classes into which the data is grouped. The methodologies that applied to categorization problems differ quite a bit, in terms of how they try to group the elements provided to them. In their general form, however, what they are trying to what they do is to calculate the boundaries between the categories (decision boundaries) [3].

Regression:

Regression, in the context of machine learning, is a statistical technique focused on modeling the relationship between a dependent variable and one or more independent variables. Unlike classification, where the goal is to assign instances to predefined categories, regression aims to predict a continuous numeric output. The process involves training a model on a dataset with known outcomes, allowing it to learn the underlying patterns and associations between input features and the target variable. Common regression algorithms include linear regression, polynomial regression, and support vector regression. The trained model can then be used to make predictions on new data, providing insights into trends, correlations, and forecasting future value. The evaluation of regression models often relies on metrics like mean squared error or R-squared, reflecting the accuracy and reliability of the predictions. Overall, regression is a powerful tool in machine learning for making informed numerical predictions based on observed data patterns [4].

The versatility of machine learning is reflected in its wide range of applications across various industries:

1. Healthcare: ML is being utilized for disease diagnosis, personalized treatment plans, and drug discovery. Predictive models can analyze patient data to identify potential health risks and improve overall healthcare outcomes.
2. Finance: In the financial sector, machine learning is employed for fraud detection, risk assessment, and algorithmic trading. These applications enhance security and streamline decision-making processes.
3. Marketing: ML algorithms analyze consumer behavior, preferences, and trends to optimize marketing strategies. Personalized recommendations, targeted advertisements, and customer segmentation are some of the ways ML is transforming the marketing landscape.
4. Autonomous Vehicles: Machine learning plays a crucial role in the development of self-driving cars. Algorithms process real-time data from sensors to make decisions such as navigation, obstacle avoidance, and traffic management.

While machine learning holds immense potential, it also poses certain challenges:

1. Data Quality: ML models heavily rely on high-quality and diverse datasets. Poor data quality can lead to biased models and inaccurate predictions.
2. Interpretability: The "black-box" nature of some machine learning algorithms can make it challenging to understand how decisions are reached. This lack of interpretability can be a barrier in critical applications such as healthcare and finance.
3. Ethical Concerns: Machine learning models may inadvertently perpetuate existing biases present in the training data. Addressing ethical concerns and ensuring fairness in algorithmic decision-making is an ongoing challenge.

3.2: Deep Learning

Deep learning is a subset of machine learning that involves the use of artificial neural networks to model and solve complex problems. The term "deep" refers to the depth of the neural networks, which consist of multiple layers (deep architectures) allowing them to automatically learn hierarchical representations of data. Deep learning algorithms, often implemented using deep neural networks, excel at capturing intricate patterns and features from large volumes of unstructured data, such as images, text, and audio. One of the key strengths of deep learning lies in its ability to automatically extract relevant features and representations from raw data, reducing the need for manual feature engineering. Popular deep learning architectures include convolutional neural networks (CNNs) for image analysis, recurrent neural networks (RNNs) for sequence data, and transformer models like BERT for natural language processing tasks. Deep learning has achieved remarkable success in various domains, including computer vision, speech recognition, and natural language understanding, contributing to advancements in technologies such as autonomous vehicles, virtual assistants, and medical diagnostics[5].

Key Components of Deep Learning:

1. Neural Networks: The fundamental building blocks of deep learning, neural networks comprise layers of interconnected nodes. Input layers receive data, hidden layers process it, and output layers produce the model's predictions or classifications.
2. Activation Functions: These mathematical operations introduce non-linearities to the neural network, enabling it to learn complex patterns. Common activation functions include the sigmoid, tanh, and rectified linear unit (ReLU).
3. Loss Functions: Loss functions measure the difference between the model's predictions and the actual outcomes. During training, the goal is to minimize this loss, improving the model's accuracy.

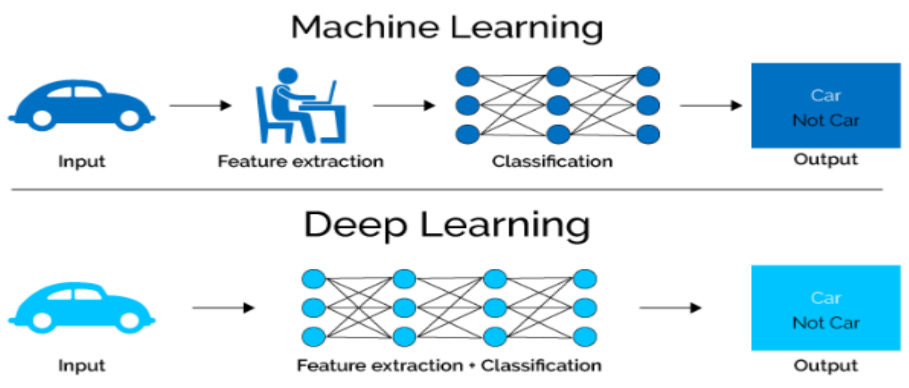


Figure 8: Deep Learning

3.3:State of the Art

In the ever-evolving landscape of time series forecasting, the integration of neural networks and advanced modeling techniques has emerged as a revolutionary force. The current state of the art in time series forecasting, shedding light on the transformative role played by neural networks and sophisticated models in unraveling the intricacies of temporal data.

Neural Networks in Time Series Forecasting:

1. Recurrent Neural Networks (RNNs): RNNs have garnered attention for their ability to capture temporal dependencies within sequential data. However, they face challenges such as the vanishing gradient problem, limiting their effectiveness in capturing long-term dependencies.
2. Long Short-Term Memory (LSTM) Networks: LSTMs, an evolution of RNNs, address the vanishing gradient problem, making them highly effective in modeling long-range dependencies. Their architecture, featuring memory cells, allows them to retain and utilize information over extended periods, making them well-suited for time series forecasting.
3. Gated Recurrent Units (GRUs): Similar to LSTMs, GRUs are designed to capture long-term dependencies. They are computationally more efficient than LSTMs due to a simplified architecture, making them a popular choice for certain time series forecasting applications.
4. Transformer Models: Originally designed for natural language processing, transformer models, such as the Transformer architecture and its variants like BERT and GPT, have demonstrated effectiveness in capturing temporal relationships in sequential data, expanding their applicability to time series forecasting.

Advanced Models in Time Series Forecasting:

1. Temporal Convolutional Networks (TCNs): TCNs leverage convolutional layers to capture hierarchical features in temporal data. Their parallelized structure allows for efficient training and makes them suitable for capturing both short and long-term dependencies.
2. Ensemble Models: Combining the predictions of multiple models often results in improved accuracy. Ensembling techniques, such as stacking and bagging, offer a robust approach by leveraging diverse neural network architectures or combining neural networks with traditional time series forecasting methods.
3. Hybrid Architectures: Combining the strengths of different neural network architectures or integrating neural networks with traditional statistical models creates hybrid models with enhanced forecasting capabilities.

Despite their successes, neural networks and advanced models in time series forecasting face challenges such as interpretability, overfitting, and the need for large datasets. Researchers are actively exploring techniques to enhance model interpretability, mitigate overfitting, and develop models that perform well in scenarios with limited data.

The integration of neural networks in automated machine learning (AutoML) platforms has democratized access to advanced forecasting models. As forecasting models become more complex, the need for explainability has grown, leading to the development of techniques to interpret and communicate the decisions made by neural networks.

Neural networks and advanced models have reshaped the landscape of time series forecasting, offering unprecedented capabilities to capture complex temporal relationships. As technology advances, the synergy between innovative neural network architectures and sophisticated modeling techniques promises further breakthroughs in accurately predicting future trends. Navigating the horizon of time series forecasting with neural networks and advanced models opens up exciting possibilities for businesses and industries seeking to make data-driven decisions in an increasingly dynamic and uncertain world.

3.4: Methods for Time-series forecast

AUTOREGRESSIVE MODEL(AR)

In the realm of time series analysis, the Autoregressive (AR) model stands as a foundational and versatile tool for understanding and predicting sequential data. Introduced as part of the broader autoregressive integrated moving average (ARIMA) family, the AR model specifically focuses on capturing the temporal dependencies within a time series, offering valuable insights into trends and patterns over time. This article provides an in-depth exploration of the AR model, elucidating its principles, applications, and significance in the context of time series analysis.

Principles of the Autoregressive Model:

The Autoregressive model, denoted as AR(p), revolves around the concept of autocorrelation, where a variable's current value is modeled as a linear combination of its past values. The 'p' in AR(p) denotes the order of the autoregressive process, indicating the number of lagged observations considered in predicting the future values.

The AR(p) model is mathematically represented as follows:

$$X_t = c + \phi_1 \cdot X_{t-1} + \phi_2 \cdot X_{t-2} + \ldots + \phi_p \cdot X_{t-p} + \varepsilon_t \qquad (1)$$

Where:

- X_t is the current observation.
- ϕ_i represents the autoregressive coefficients.
- 'c' is a constant term.
- ε_t is a white noise error term.

Each term $\phi_i \cdot X_{t-i}$ reflects the influence of the previous observations on the current one, capturing the inherent serial correlation in the time series [6].

Applications of the AR Model:

1. Economics and Finance: AR models find extensive use in modeling economic indicators, such as stock prices and interest rates. They help analysts understand and predict the future values of financial time series based on historical patterns.
2. Environmental Sciences: In fields like climatology, AR models are applied to study and forecast climate variables, such as temperature and precipitation. The ability to capture temporal dependencies aids in predicting future climate trends.
3. Signal Processing: AR models are employed in signal processing for tasks like speech recognition and audio signal analysis. By modeling the sequential nature of signals, AR models contribute to accurate prediction and analysis.
4. Stock Market Analysis: Understanding the autocorrelation in stock prices is crucial for traders and investors. AR models help in modeling and predicting stock price movements based on historical data.

While powerful, AR models have limitations. They assume a linear relationship between past and current observations and may not capture complex non-linear patterns. Additionally, the order 'p' must be carefully selected to balance model complexity and predictive accuracy.

The Autoregressive model serves as a fundamental building block in time series analysis, offering a clear and interpretable framework for understanding the temporal dependencies within data. Its applications span various domains, providing valuable insights into trends and patterns that drive decision-making processes. As part of the broader family of ARIMA models, the AR model continues to be a cornerstone in the field, complementing other techniques and contributing to the comprehensive analysis of time-dependent datasets.

Autoregressive Integrated Moving Average (ARIMA)

The Autoregressive Integrated Moving Average (ARIMA) model stands as a powerful and versatile tool for forecasting and understanding temporal data patterns. ARIMA represents a synthesis of autoregressive (AR) and moving average (MA) models, coupled with differencing to achieve stationarity.

Principles of the ARIMA Model:

ARIMA models are denoted as ARIMA(p, d, q), where 'p' is the autoregressive order, 'd' is the differencing order, and 'q' is the moving average order. The three components signify key aspects of the ARIMA modeling process:

1. Autoregressive (AR) Component: Represents the dependency of the current observation on its own past values. The 'p' order captures the number of lagged observations considered in predicting the future values.
2. Integrated (I) Component: Encompasses the differencing process required to achieve stationarity. The 'd' order signifies the number of differencing steps applied to the time series data.
3. Moving Average (MA) Component: Captures the dependency of the current observation on past white noise error terms. The 'q' order reflects the number of lagged error terms considered in the model.

Mathematically, the ARIMA(p, d, q) model is represented as:

$$Y_t = c + \phi_1 \cdot Y_{t-1} + \phi_2 \cdot Y_{t-2} + \ldots + \phi_p \cdot Y_{t-p} + \theta_1 \cdot \varepsilon_{t-1} + \theta_2 \cdot \varepsilon_{t-2} + \ldots + \theta_q \cdot \varepsilon_{t-q} + \varepsilon_t \qquad (2)$$

Where:

- Y_t is the current observation.
- ϕ_i and θ_i are the autoregressive and moving average coefficients, respectively.
- 'c' is a constant term.
- ε_t is a white noise error term. [7]

Applications of the ARIMA Model:

1. Economics and Finance: ARIMA models are widely employed in forecasting economic indicators such as GDP, inflation rates, and stock prices.
2. Supply Chain Management: Businesses use ARIMA models for demand forecasting, inventory management, and optimizing supply chain processes.
3. Public Health: In epidemiology, ARIMA models are applied for forecasting disease outbreaks, aiding in resource allocation and public health planning.

4. Energy Consumption: ARIMA models help predict energy consumption patterns, facilitating efficient resource allocation and sustainability efforts.

Considerations and Limitations:

- Model Selection: The appropriate selection of 'p', 'd', and 'q' parameters is crucial for effective model performance. This requires careful analysis and understanding of the underlying data.
- Stationarity: ARIMA models assume stationarity, and differencing is employed to achieve this. However, excessive differencing can introduce model complexity and impact interpretability.
- Seasonality: While powerful, ARIMA models may struggle with capturing seasonal patterns. Seasonal ARIMA models (SARIMA) are often preferred for such scenarios.

The Autoregressive Integrated Moving Average (ARIMA) model is a cornerstone in time series analysis, providing a robust framework for forecasting and understanding complex temporal patterns. Its adaptability, versatility, and applicability across diverse domains make it a go-to tool for analysts and researchers seeking to unravel the intricacies of time-dependent datasets. As the field of time series analysis continues to evolve, ARIMA models persist as a fundamental building block, contributing to the broader landscape of predictive analytics and informed decision-making.

Autoregressive Moving Average (ARMA)

Autoregressive Moving Average (ARMA) models provide a powerful framework for understanding and forecasting sequential data. Representing a balance between autoregressive (AR) and moving average (MA) components, ARMA models are adept at capturing the temporal dependencies and random fluctuations inherent in time series datasets. This article delves into the principles, applications, and significance of ARMA models, offering insights into their role in time series analysis.

Principles of the ARMA Model:

The ARMA model is denoted as ARMA(p, q), where 'p' is the autoregressive order and 'q' is the moving average order. The model integrates autoregressive terms, which capture the dependency of the current observation on its own past values, and moving average terms, which account for the dependency on past white noise error terms.

Mathematically, the ARMA(p, q) model is represented as:

$$Y_t = c + \phi_1 \cdot Y_{t-1} + \phi_2 \cdot Y_{t-2} + \ldots + \phi_p \cdot Y_{t-p} + \varepsilon_t - \theta_1 \cdot \varepsilon_{t-1} - \theta_2 \cdot \varepsilon_{t-2} - \ldots - \theta_q \cdot \varepsilon_{t-q} \quad (3)$$

Where:

- Y_t is the current observation.
- ϕ_i and θ_i are the autoregressive and moving average coefficients, respectively.
- 'c' is a constant term.
- ε_t is a white noise error term. [8]

Applications of the ARMA Model:

1. Financial Time Series: ARMA models are widely used in finance for modeling and forecasting stock prices, exchange rates, and other financial indicators.
2. Economic Indicators: ARMA models are applied to analyze and forecast economic indicators such as inflation rates, unemployment rates, and GDP growth.
3. Climate Data Analysis: In meteorology, ARMA models are utilized to analyze and predict temperature, precipitation, and other climatic variables.
4. Operations Research: ARMA models find applications in operations research for tasks like demand forecasting, helping optimize inventory management and resource allocation.

Considerations and Limitations:

- Stationarity Assumption: ARMA models assume stationarity in the time series data. It is essential to assess and potentially transform the data to meet this assumption.
- Model Order Selection: Choosing the appropriate values for 'p' and 'q' is crucial for effective model performance. This often involves statistical techniques and diagnostic checks.
- Limited Ability for Long-Term Forecasting: ARMA models may struggle with capturing long-term trends or seasonality. In such cases, extended models like ARIMA or SARIMA may be more suitable.

The Autoregressive Moving Average (ARMA) model stands as a valuable tool in time series analysis, providing a balanced approach to capturing temporal dependencies and random fluctuations in sequential data. Its versatility and applicability across various domains make it a preferred choice for analysts and researchers aiming to gain insights and make informed predictions. As the field of time series analysis advances, ARMA models continue to contribute to the foundational understanding of time-dependent datasets, providing a solid framework for forecasting and decision-making.

3.5:LSTM Model

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture designed to overcome the limitations of traditional RNNs in capturing long-term dependencies in sequential data. It was introduced by Sepp Hochreiter and Jürgen Schmidhuber in 1997.

The primary challenge that LSTMs address is the vanishing gradient problem, which occurs when training RNNs on long sequences. As information passes through the network, gradients can become extremely small, making it difficult for the network to learn and retain information from distant time steps.

Here are key components of an LSTM:

1. Cell State (c_t): The cell state is the long-term memory of the network. LSTMs use a mechanism to selectively add or remove information from the cell state through a set of gates.
2. Hidden State (h_t): The hidden state is the short-term memory or the output of the LSTM at a particular time step. It is a function of the cell state and the input at that time step.
3. Gates:
    - Forget Gate (f_t): Determines what information from the cell state should be discarded or kept. It takes the previous hidden state (h_{t-1}) and the current input (x_t) as inputs and outputs a value between 0 and 1 for each element in the cell state. A value of 0 means "forget," and a value of 1 means "remember."
    - Input Gate (i_t): Decides what new information should be added to the cell state. Similar to the forget gate, it takes the previous hidden state and current input, processes them, and outputs values between 0 and 1.
    - Cell Gate (g_t): Creates a vector of new candidate values that could be added to the cell state. It is a candidate update that is weighted by the input and previous hidden state.
    - Output Gate (o_t): Determines the next hidden state based on the updated cell state. It decides what part of the cell state should be exposed as the output.

The computations within the LSTM are governed by these gates, which are implemented using a combination of sigmoid and hyperbolic tangent (tanh) activation functions. The architecture allows LSTMs to capture and propagate relevant information over long sequences, making them effective for tasks involving sequential data, such as natural language processing, speech recognition, and time-series analysis.

The fundamental structure of an LSTM (Long Short-Term Memory) cell involves three key gates within a memory cell. The input gates play a crucial role in tracking and storing the latest information in the memory cell. On the other hand, the output gate is responsible for controlling the flow of the most current information throughout the rest of the network. The third type of gates, known as forget gates, serves the purpose of deciding whether to discard information based on the state of the preceding cell. Equations (4)-(11) provide a detailed explanation of how to implement and update the LSTM cell state, as well as compute the final LSTM outputs [9].

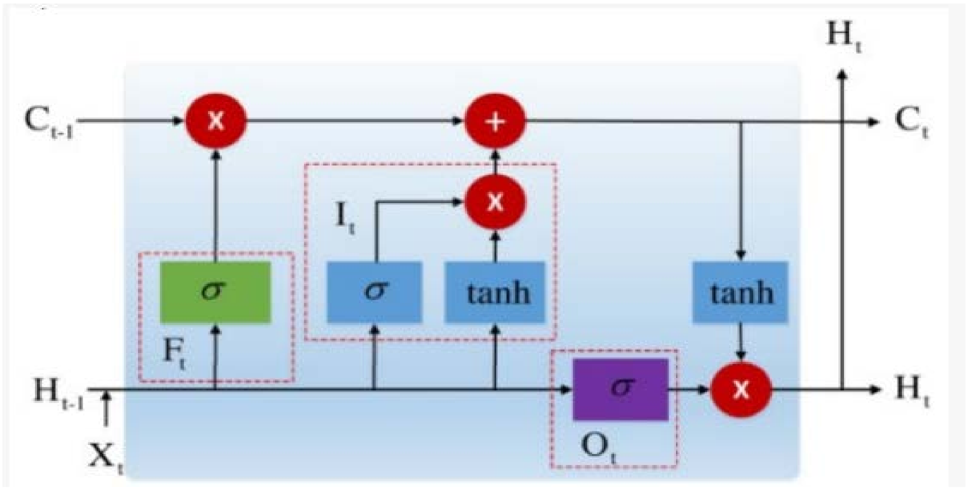


Figure 9: The schematic of lstm

$$F_t = \sigma\left(W_{xf}X_t + W_{hf}H_{(t-1)} + B_f\right) \quad (4)$$

$$I_t = \sigma\left(W_{xi}X_t + W_{hi}H_{(t-1)} + B_i\right) \quad (5)$$

$$\widehat{C_t} = \sigma\left(W_{xf}X_t + W_{hf}H_{(t-1)} + B_f\right) \quad (6)$$

$$C_t = F_t * C_{(t-1)} + I_t * \widehat{C_t} \quad (7)$$

$$O_t = \sigma\left(W_{xo}X_t + W_{ho}H_{(t-1)} + B_o\right) \quad (8)$$

$$H_t = O_t \tanh(C_t) \quad (9)$$

$$Y_t = \sigma\left(W_{hy}H_t + B_y\right) \quad (10)$$

$$\sigma(x) = \frac{1}{\exp(-x)} \quad (11)$$

Where X_t = input vector; Y_t = output vector; I_t = input gate outcome; F_t = forget gate outcome;O_t= output gate outcome;C_t= finishing state in memory block;$\widehat{C_t}$= temporary; σ = sigmoid function;$W_{xf}, W_{xi}, W_{xc}$and$W_{xo}$are input weight matrices;W_{hf}, W_{hi}, W_{xc} and$W_{ho}$are recurrent weight matrices;$W_{hy}$is output weight matrix; and$B_f, B_i, B_c, B_o$and$B_y$are the related bias vectors.

3.6: SAE and SIRAE Models

STACKED AUTOENCODERS LSTM

In the ever-evolving landscape of artificial intelligence and deep learning, researchers and practitioners continuously explore innovative architectures to tackle complex tasks. One such powerful combination is the marriage of Long Short-Term Memory (LSTM) networks and stacked autoencoders.

Autoencoders, a fundamental concept in unsupervised learning, consist of an encoder and a decoder. The encoder compresses input data into a lower-dimensional representation, while the decoder reconstructs the original input from this compressed representation. Stacked autoencoders take this idea a step further by employing multiple layers of encoders and decoders, creating a deep neural network architecture.

Long Short-Term Memory (LSTM): LSTM, a variant of recurrent neural networks (RNNs), is specifically designed to handle sequential data by capturing long-term dependencies. Featuring memory cells and gates, LSTMs excel in modeling sequences with significant time lags between relevant events. They are particularly useful for tasks such as time-series prediction and natural language processing.

LSTM-Stacked Autoencoder Model: The synergy of LSTM and stacked autoencoders involves leveraging LSTM layers as both the encoder and decoder in an autoencoder architecture. This integration combines the ability of LSTMs to capture sequential dependencies with the feature learning prowess of stacked autoencoders.

Architecture in Action:

1. Encoding Sequential Information: The input sequence is fed into the LSTM encoder, which excels in capturing sequential patterns. This initial encoding step compresses the input information while preserving the temporal relationships between data points.
2. Enhancing Complexity with Stacked LSTMs: The compressed representation undergoes further processing through a stack of additional LSTM layers. This stacking enhances the model's capacity to capture intricate and complex dependencies within the data, making it adept at handling challenging sequential tasks.
3. Decoding for Reconstruction: The decoder, composed of LSTM layers in reverse order, reconstructs the input sequence from the enriched and compressed representation. The objective is to minimize the reconstruction error, guiding the model to learn a meaningful and hierarchical representation of the sequential input.

Applications:

1. Time-Series Prediction: The LSTM component helps in capturing temporal dependencies, making the model well-suited for time-series prediction tasks. Stacked autoencoders contribute by learning hierarchical features, enabling the model to make accurate predictions based on learned representations.
2. Feature Learning: The stacked autoencoder structure excels in learning hierarchical representations of the input data, while the LSTM layers contribute by capturing sequential patterns. This makes the model valuable in scenarios where understanding complex features in sequential data is crucial.

The LSTM-stacked autoencoder model represents a potent fusion of sequential modeling and feature learning. Its applications span diverse domains, from predicting stock prices to analyzing natural language. As the field of deep learning continues to evolve, understanding and harnessing the strengths of such innovative architectures become essential. The LSTM-stacked autoencoder stands as a testament to the ongoing exploration and synergy of neural network components, paving the way for advancements in handling complex sequential data [10].

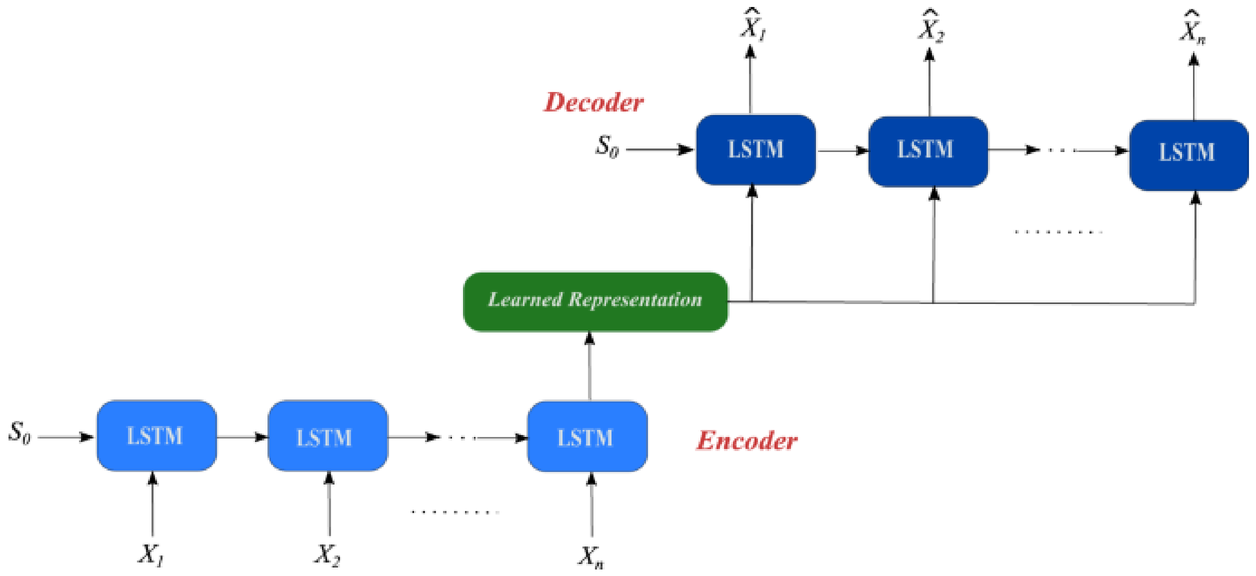


Figure 10: The Structure of Stacked Autoencoders With LSTM Layers

1. LSTM Encoder:

   - Let *X* represent the input sequence, h_i denote the hidden state, and c_i represent the cell state of the LSTM unit at time step *i*.

     $h_i, c_i = LSTM_{encoder}(X_i, h_{i-1}, c_{i-1})$

   - The LSTM equations for the encoder can be written as:
   - The output of the last time step of the encoder is the compressed representation: $Z = h_{last}$

2. Stacked LSTM Layers:

   - If there are *N* stacked LSTM layers in the model, the output of one layer becomes the input to the next:

     $h_i^{(j)}, c_i^{(j)} = LSTM_{layer j}(h_i^{(j-1)}, c_i^{(j-1)})$

for each layer *j*, where $h_i^{(0)} = Z$ is the compressed representation from the encoder.

3. LSTM Decoder:
    - Similar to the encoder, the LSTM decoder operates on the output sequence *Y*, which is the reconstructed version of the input sequence *X*:

      $\widehat{X_\iota}, \widehat{h_\iota}, \widehat{c_\iota} = LSTM_{decoder}(Y_i, \widehat{h_{\iota-1}}, \widehat{c_{\iota-1}})$
    - The output of the last time step of the decoder represents the reconstructed sequence: $\hat{X} = \widehat{h_{last}}$
4. Objective Function:
    - The objective is typically to minimize the difference between the input and the reconstructed output. This can be represented using a loss function like Mean Squared Error (MSE): $L = \frac{1}{T}\sum_{i=1}^{T} \left||X_i - \widehat{X_\iota}|^2\right|$

In practice, the parameters (weights and biases) of the LSTM cells in the encoder and decoder, as well as any additional layers, are learned through backpropagation and optimization algorithms like stochastic gradient descent.

STACKED INDEPENDENTLY RECURRENT AUTOENCODER

In the ever-evolving field of deep learning, the quest for efficient representation learning has given rise to various architectures and techniques. One such powerful approach is the utilization of stacked independently trained autoencoders. This method, rooted in the fundamentals of neural networks, empowers models to capture hierarchical and abstract features from complex datasets. We will explore the intricacies of stacked autoencoders, their layer-wise training process, and the applications that make them a cornerstone in the realm of unsupervised learning.

Before delving into stacked autoencoders, let's revisit the foundational concept of autoencoders. An autoencoder is a neural network architecture comprised of two primary components – an encoder and a decoder. The encoder compresses the input data into a lower-dimensional representation, and the decoder reconstructs the original data from this representation. The primary objective is to minimize the reconstruction error, encouraging the model to learn a meaningful representation of the input.

The SIRAE model, or Stacked Independently Recurrent Auto Encoder, is a sophisticated neural network architecture designed for sequential data processing tasks, particularly in the domain of natural language processing and time-series analysis. This model consists of multiple layers of recurrent neural networks (RNNs), where each layer operates independently, encoding input sequences into latent representations. Unlike traditional autoencoders, SIRAE employs recurrent connections within each layer to capture temporal dependencies effectively. By stacking these recurrent layers, the model can learn hierarchical representations of input sequences at different levels of abstraction. SIRAE demonstrates remarkable performance in tasks such as sequence generation, anomaly detection, and feature extraction, making it a powerful tool for various applications requiring sequential data analysis.

The SIRAE model augmented with IndRNN layers integrates the innovative IndRNN (Independently Recurrent Neural Network) architecture within its stacked structure to enhance its sequential data processing capabilities. Unlike traditional RNNs, IndRNNs leverage element-wise multiplication, enabling each neuron to maintain its internal state independently across time steps. This unique property alleviates the vanishing gradient problem, allowing for more effective long-term temporal modeling. By incorporating IndRNN layers into the SIRAE architecture, the model gains increased robustness in capturing intricate temporal dependencies within sequential data. Each layer of the SIRAE-IndRNN model operates as an independent encoder, progressively extracting hierarchical representations of the input sequences. This fusion of SIRAE with IndRNN layers empowers the model with superior learning capacity and generalization ability, making it well-suited for a wide range of sequential data tasks, including natural language processing, time-series analysis, and beyond.

IndRNN- layer (Encoder):

For the encoders the IndRNN layer is used the mathematical equation is:
$h_t = \sigma(W \cdot x_t + u * h_{t-1} + b)$

Where W is the encoding weight matrix, u is the recurrent weight matrix, b is the encoding bias vector and σ is ReLU function.

Dense-layer(Decoder):

For the decoders the Dense layer is used the mathematical equation is:

$y_t = g(W \cdot h_t + c)$

Where W is the decoding matrix, c is the decoding bias vector and g is ReLU function [11] .

SAE vs SIRAE

At encoding phase there is the first important difference between SAE and SIRAE model. SAE model uses LSTM layers for encoding so the neurons do not operate independently this expands and at training phase. In IndRNN layer each neuron operate independently this leading to improved gradient flow and potentially better performance on tasks involving long-term dependencies. At the end the second difference of these two models is the layers that they use at decoding phase, SAE model uses again LSTM-layer and the SIRAE model Dense-layer. Dense layers are standard feedforward layers that process each input independently, while LSTM layers are specialized recurrent layers designed to capture temporal dependencies and maintain memory across time steps, making them suitable for sequential data processing tasks

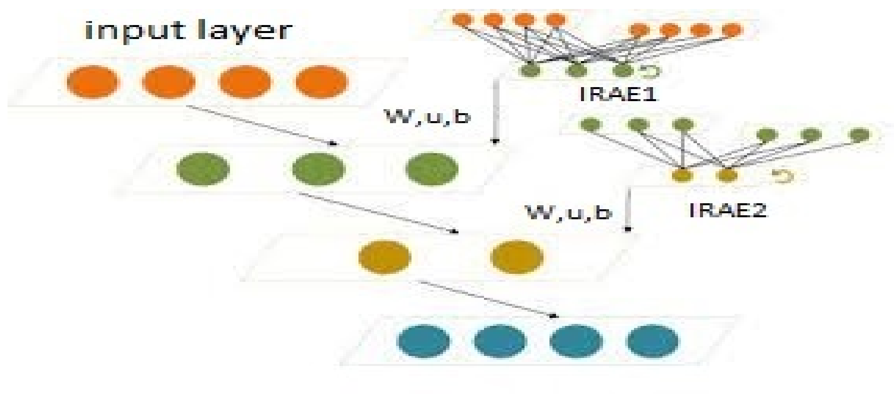


Figure 11: The SIRAE structure with IndRNN layers

CHAPTER 4: *Data Augmentation*

In the field of machine learning and deep learning, the quality and quantity of training data are critical factors influencing model performance. However, obtaining large, diverse datasets can be challenging, particularly in domains such as medical imaging or natural language processing. To address this limitation, data augmentation has emerged as a powerful technique, artificially expanding the diversity of training data to improve model generalization and robustness [12].

Data augmentation involves applying a set of predefined transformations to the existing dataset, generating variations in input data. These transformations simulate real-world scenarios and introduce diversity without requiring additional labeled samples. While applicable across various domains, data augmentation finds prominence in computer vision tasks like image classification, object detection, and segmentation.

For image data, transformations include rotation, flipping (horizontal or vertical), scaling, translation, zooming, and adjustments to brightness, contrast, and color. In text data, strategies involve synonym replacement, random insertion or deletion of words, and word swapping [13].

Major deep learning frameworks, such as TensorFlow and PyTorch, embed support for data augmentation. In TensorFlow's Keras, the (ImageDataGenerator) facilitates image data augmentation, while PyTorch's(transforms) module caters to augmenting image data. Standalone libraries like Albumentations and imgaug offer comprehensive sets of augmentation techniques.

Data augmentation proves advantageous when dealing with a limited dataset, emphasizing the need for improved model generalization. It finds common application in training deep learning models for image classification, object detection, and segmentation.

While applicable across diverse domains, data augmentation's prevalence is notable in computer vision, especially image-related tasks. In medical imaging, where labeled data is often scarce, data augmentation becomes a crucial strategy. In Natural Language Processing (NLP), though less common, it contributes to enhanced model performance in text-related tasks [14].

4.1: Data Surrogation Time and Frequency Domain

Data augmentation, a pivotal technique in machine learning, extends its applicability into both the time and frequency domains. The choice of domain depends on the characteristics of the data and the specific requirements of the task at hand.

Time Domain:

In the time domain, data augmentation involves direct transformations applied to input signals or sequences. This approach is frequently employed in tasks such as speech recognition, time-series analysis, and signal processing. Augmentation techniques include:

Time Warping: Slightly stretching or compressing the time axis of a signal.

Jittering: Introducing small random perturbations to the data points.

Time Shifting: Shifting the entire signal in time.

Amplitude Scaling: Adjusting the amplitude of the signal [15].

Frequency Domain:

Alternatively, data augmentation in the frequency domain is beneficial for signals with a well-defined frequency spectrum. This approach is valuable in tasks such as audio processing, image processing, and wireless communication. Augmentation techniques include:

Spectral Warping: Distorting the frequency content of the signal.

Random Phase Modifications: Introducing random changes to the phase of the frequency components.

Band-pass Filtering: Selectively emphasizing or attenuating specific frequency bands.

Time Domain Applications:

Speech Recognition: Augmenting audio signals with time warping or speed variations.

Time-Series Analysis: Enhancing models dealing with temporal patterns using time-based augmentations.

Signal Processing: Improving the robustness of signal processing models with various time domain transformations.

Frequency Domain Applications:

Audio Processing: Augmenting audio data in the frequency domain for tasks like music classification or audio synthesis.

Image Processing: Utilizing Fourier domain augmentations for image data with a frequency representation.

Wireless Communication: Enhancing the training of models for signal demodulation by introducing variations in the frequency domain.

<u>Considerations:</u>

The choice between time and frequency domain augmentation hinges on the characteristics of the data and the specific task requirements. Combining both time and frequency domain augmentations may be advantageous in capturing a broader range of variations.

4.2: AAFT Algorithm

The AAFT (Amplitude Adjusted Fourier Transform) algorithm is a method for surrogate data generation in the context of time series analysis. Surrogate data is artificial data created to preserve certain statistical properties of the original time series while breaking any underlying temporal dependencies [16]. The AAFT algorithm is particularly useful for testing hypotheses and assessing the significance of observed patterns in time series data. Here's a description of the AAFT algorithm:

Step (i): Rank-Ordered Remapping onto a Gaussian Distribution

$x_n=G(y_n)$ (12)

Here, x_n is the remapped data obtained by applying the function G to the original time series y_n. This remapping is done to transform the time series onto a Gaussian distribution, conserving dynamic non-linearities.

Step (ii): Fourier Transform of X_n

$X_n(f)=F[x_n]$ (13)

Compute the Fourier transform $X_n(f)$ of the remapped data x_{n}.. This involves transforming the time-domain data into the frequency domain.

Additional Step (iii): Generate a Random Shuffling Realization of x_n

$X_{shuffled}(f)=F[x_n]$ (14)

Generate a random shuffling realization of x_n. and calculate its Fourier transform. This is crucial for obtaining uncorrelated uniformly distributed Fourier phases in the interval $[-\pi,\pi]$, given that the amplitude distribution in real space is Gaussian.

Step (iv): Combine Fourier Amplitudes and Random Phases

$X_0(f)=|X_n(f)|\cdot e^{i\Theta(f)}$(15)

Combine the Fourier amplitudes $|X_n(f)|$ of x_n with the random Fourier phases $\Theta(f)$ obtained from the shuffling realization. This results in a new set of Fourier coefficients $X_0(f)$.

Step (v): Perform an Inverse Fourier Transformation

$x_{0n}(t)= F^{-1} [X_0(f)]$ (16)

Perform the inverse Fourier transformation on $X_0(f)$ to obtain the surrogate time series $x_{0n}(t)$. This is the resulting time series after the first iteration

Step (vi): Sort x_{0n} and Evaluate the Fourier Transform of z_n

$z_n = S(x_{0n})$

Sort the surrogate time series x_{0n} to obtain z_n. Evaluate the Fourier transform of z_n. Steps (i), (ii), (iii), (iv), (v) and (vi) together describe the AAFT algorithm.

Use the set of surrogate time series to establish a baseline distribution for statistical testing. Compare the properties of the original time series with the properties of the surrogate set to assess the significance of observed patterns or features.The primary purpose of the AAFT algorithm is to create surrogate data that preserves certain statistical characteristics of the original time series.It is commonly used in hypothesis testing to determine whether observed patterns in the original time series are statistically significant or if they could have occurred by chance. The success of the AAFT algorithm relies on the assumption that the temporal dependencies in the original time series are predominantly due to the phase information, which is destroyed during the random phase shuffling.

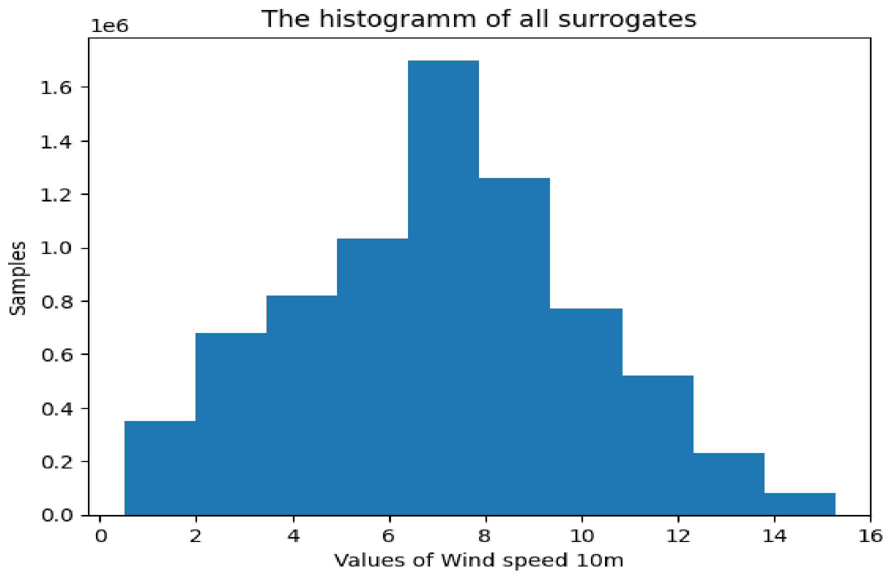


Figure 12: The histogramm of all surrogates for a month

CHAPTER 5
Imputation

Transforming a time series from an hourly to a more granular base holds significant importance in enhancing the precision and responsiveness of forecasting models. While hourly data provides a general overview of trends and patterns, more frequent intervals enable a finer understanding of temporal dynamics. This increased granularity is particularly crucial in industries where rapid changes or fluctuations occur within short time frames, such as energy consumption, financial markets, or manufacturing processes. By transitioning to smaller minute intervals, analysts and forecasting algorithms gain the ability to capture subtle variations, respond promptly to fluctuations, and generate more accurate predictions. This finer temporal resolution not only improves the accuracy of forecasting models but also facilitates better decision-making in real-time scenarios, contributing to more efficient resource allocation, cost management, and overall operational optimization [17]. In this approach the imputation was done as follows, between two hours we used the function randn with definitions as lower bound the value for the first hour of the interval and as upper bound the value for the second hour of the interval. The numpy.random.randn function in Python is a convenient tool for generating an array of random numbers sampled from a standard normal distribution. The standard normal distribution has a mean of 0 and a standard deviation of 1.

5.1:OTHER METHODS

INTERPOLATION

Interpolation is a common imputation method for filling in missing values in a time series. This technique involves estimating the values of missing data points by making educated guesses based on the observed values at neighboring time points. In the context of time series imputation, linear interpolation is frequently employed.

Linear interpolation assumes a linear relationship between two adjacent observed points and uses this assumption to estimate the values at intermediate time points. The process involves connecting the observed values with a straight line and determining the values at the missing points along that line.

Here's a basic outline of how linear interpolation works for time series imputation:

1. Identify Missing Values:
   - Identify the time points where data is missing within the time series.
2. Select Neighboring Points:
   - For each missing value, identify the nearest observed values before and after the missing point.
3. Calculate Slope:
   - Calculate the slope of the line connecting the two observed values. The slope represents the rate of change between the two points.
4. Interpolate Missing Values:
   - Use the calculated slope to estimate the values at the missing time points along the line connecting the observed values.

The linear interpolation method assumes a steady and linear trend between observed points. While this assumption might be suitable for certain types of data, it might not capture more complex patterns present in the time series. Additionally, linear interpolation is sensitive to outliers and abrupt changes in the data.

Other types of interpolation, such as cubic spline interpolation, can also be employed to address these limitations. Cubic spline interpolation fits a piecewise cubic polynomial to the data, providing a smoother and more flexible interpolation between observed points.

When applying interpolation for imputation in a time series, it's important to carefully consider the characteristics of the data and the potential impact on subsequent analyses or modeling. It's advisable to explore various imputation methods and assess their suitability based on the specific features and patterns present in the time series dataset [18].

RANDOM FORREST

While linear interpolation is a deterministic method, Random Forest imputation is a machine learning-based approach for handling missing values in a time series. Unlike linear interpolation, which assumes a linear relationship between observed points, Random Forest imputation leverages the predictive power of a Random Forest model to estimate missing values.

Here's an overview of how Random Forest imputation can be used for time series imputation:

1. Train Random Forest Model:
    - Split the time series data into two parts: one with complete observations and another with missing values. The complete dataset serves as the training set for the Random Forest model.
2. Feature Selection:
    - Identify relevant features (variables) that can help predict the missing values. These features could include the time of day, day of the week, and other relevant contextual information.
3. Train the Model:
    - Train a Random Forest model on the complete dataset, using the identified features to predict the target variable (the variable with missing values).
4. Predict Missing Values:
    - Use the trained Random Forest model to predict the missing values in the dataset with incomplete observations.
5. Impute Predicted Values:
    - Substitute the predicted values for the missing values in the time series dataset.

Random Forest imputation has the advantage of capturing non-linear relationships and interactions between variables, making it suitable for time series with complex patterns. It's a robust method that can handle both numerical and categorical variables. However, it may introduce some level of uncertainty, as predictions are based on the learned patterns in the training data.

When applying Random Forest imputation, it's crucial to validate the model's performance and assess its suitability for the specific characteristics of the time series data. Additionally, consider the potential computational costs associated with training and using a machine learning model, especially for large datasets [19].

CHAPTER 6
Model Design

6.1 : Medium Term Forecasting

Stacked Autoencoders with LSTM

To commence, we acquire the data pertaining to the year 2009. Firstly we get data of a month from whole data set, we employ the AAFT algorithm to produce novel signals. Initially, we generate 10,000 signals with a Gaussian distribution, resulting in 10,000 values for each hour throughout a month. Following this, we eliminate any duplicated values that may arise from this process. We then proceed to arrange the surrogate data, imposing a constraint to ensure that the values closely resemble real data points. Specifically, we avoid values that are bigger than the real value '+' standard deviation of real signal and smaller than the real value '-' standard deviation of real signal for every time point corresponding. Through this approach, we identify a set of points that align with all the underlying assumptions. Subsequently, we select the dataset derived from these points for predicting wind speed.

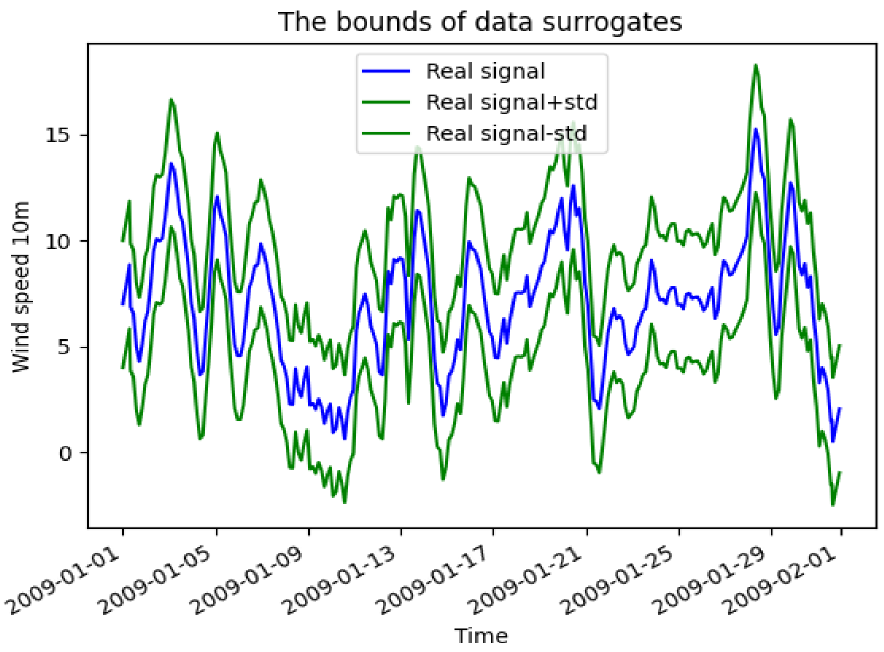


Figure 13: The bounds of data surrogates for January

We used the example for the month of January, as can be seen in the image, the data that we reproduced as described previously have values that are within the area enclosed by the green waveforms.

1. Input Data:
    - The input data is a 2D matrix the columns represent the value of wind speed at a specific hour and the rows represent the months. It is a univariate time series, with each data point capturing the wind speed at a specific hour.
2. Data Preprocessing:
    - The MinMaxScaler is used to normalize the data, ensuring that all values are within the [0, 1] range. Normalization is crucial for neural network training, especially when using activation functions like ReLU.
3. Data Splitting:
    - The dataset is split into three parts: a training set with the 80% of total data set, a validation set with the next 10%, and a test set with the remaining 10% data. This division allows for training the model, tuning hyperparameters with the validation set, and evaluating final performance on the test set.
4. Sequence Generation:
    - Sequences are generated from the training data for using as input in predicting phase. Each sequence represents a chunk of time steps, and the LSTM model is trained to capture temporal dependencies within these sequences.
5. LSTM Autoencoder Architecture:
    - Encoder:
        - The encoder is designed with three LSTM layers, each building a hierarchy of features.
        - The first LSTM layer with 50 units captures complex patterns at a higher level.
        - The second layer with 25 units refines the learned features, and the third layer with 12 units produces a compressed representation of the input sequence.
        - The final state of the third layer is repeated using RepeatVector(1) to match the time steps in the output sequence.
    - Decoder:
        - The decoder, mirroring the encoder, consists of three LSTM layers to reconstruct the input sequence.
        - Each LSTM layer in the decoder progressively expands the information.
        - The TimeDistributed dense layer with a single unit at the output predicts the wind speed for each time step.
6. Model Compilation:
    - The choice of the Adam optimizer with a learning rate of 0.01 plays a pivotal role in the optimization process. Adam, an adaptive optimization algorithm, combines the benefits of both RMSprop and Momentum methods. The adaptive learning rates for

individual parameters facilitate quicker adaptation to varying parameter importance, while the inclusion of a momentum term aids in overcoming optimization challenges

such as local minima. With the learning rate set at 0.01, this choice represents a balance between a step size that promotes swift convergence and one that mitigates the risk of overshooting optimal parameter values. The inclusion of bias correction in Adam further refines the accuracy of moment estimates, particularly in the early stages of training. Fine-tuning the learning rate is a critical aspect of model optimization, and the experimentation with the chosen value involves monitoring the model's convergence dynamics. While Adam is a widely-used optimizer, other alternatives such as SGD, RMSprop, and AdaGrad can also be considered based on specific data characteristics and model architecture. Ultimately, the choice of the Adam optimizer and its associated learning rate reflects a thoughtful decision in the pursuit of an effective and stable training process for the autoencoder LSTM model.

- Mean Squared Error (MSE) is selected as the loss function, quantifying the difference between predicted and actual wind speed values. The chosen loss function for the autoencoder LSTM model holds significant importance in the context of predicting wind speed. MSE is well-suited for regression tasks, providing a differentiable and interpretable metric that quantifies the average squared difference between predicted and actual wind direction values. By squaring deviations, MSE accentuates larger errors, making it sensitive to significant discrepancies in predictions. Its ability to balance positive and negative deviations ensures a comprehensive assessment of the model's overall performance. Minimizing MSE during training aligns with the objective of the autoencoder LSTM, encouraging the model to accurately reconstruct the input sequence.

7. Model Training:

- The model is trained using the training set for 30 epochs with a batch size of 32. Training is monitored on the validation set to prevent overfitting. During the training phase, the autoencoder LSTM model undergoes an iterative learning process, refining its parameters to accurately capture the underlying patterns in the wind direction time series. The model is trained for 30 epochs, signifying 30 complete passes through the entire training dataset, striking a balance between achieving convergence and avoiding excessive computational cost. A batch size of 32 is employed, meaning that the model updates its parameters based on 32 data points at a time. This batching approach enhances computational efficiency and allows the model to generalize well to different subsets of the data. Importantly, training progress is continuously monitored on the validation set, which serves as an independent dataset not seen by the model during training. This validation monitoring is pivotal for preventing overfitting, as it enables the early detection of signs that the model may be memorizing the training data rather than learning generalizable patterns.

8. Prediction:

- In the prediction phase, the autoencoder LSTM model follows a one-step-ahead forecasting strategy, utilizing information from the last (5,3,1) hours of every month to predict the wind speed at the next time step. Notably, each new prediction is made independently of the model's own forecasts and is not incorporated into subsequent predictions. This approach, commonly employed in time series forecasting,
- emphasizes simplicity and interpretability. By relying solely on observed historical data without considering the model's own predictions in subsequent forecasts, the methodology avoids potential compounding of errors. This conservative strategy ensures that each prediction is based on the most recent and directly observed information, aligning with the goal of making cautious and reliable forecasts for the wind direction time series

9. Evaluation:

- The Mean Squared Error (MSE) on the test set is computed to assess how well the model generalizes to unseen data.

10. Inverse Scaling:

- Predicted values are inverse scaled to bring them back to the original data range. This is essential for meaningful comparison and interpretation of the model's performance.

11. Visualization:

- The real vs. predicted values are visualized using matplotlib. This allows for a qualitative assessment of the model's ability to capture the temporal patterns in the wind direction.

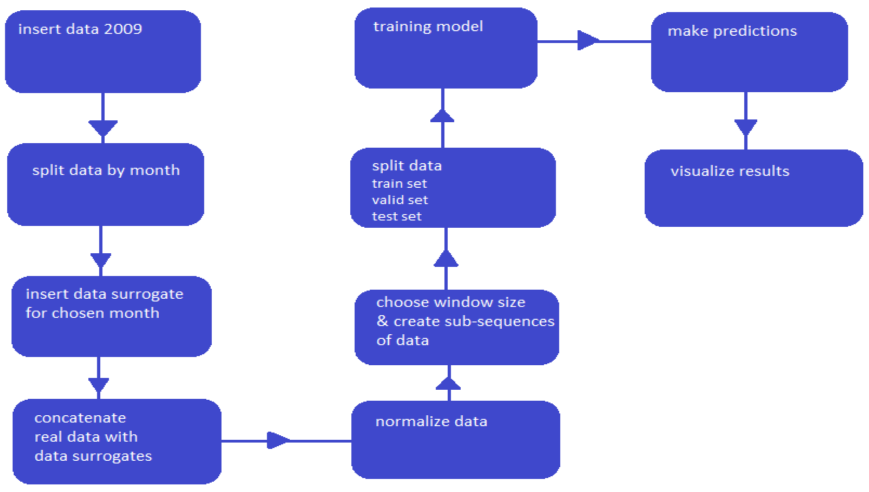

insert data 2009
split data by month
insert data surrogate for chosen month
concatenate real data with data surrogates
normalize data
choose window size & create sub-sequences of data
split data train set valid set test set
training model
make predictions
visualize results

Figure 14: Block diagramm for wind speed forecasting

Stacked Independently Reccurent Neural Networks Autoencoders with LSTM

For this model in the first encoder layer, the input data is processed by an IndRNN (Independently Recurrent Neural Network) layer with 50 units, followed by another IndRNN layer with the same configuration, which serves as the hidden representation of the encoder. The second encoder layer further compresses the representation obtained from the first encoder. It consists of an IndRNN layer with 25 units, followed by another IndRNN layer with the same configuration for hidden representation. The third encoder layer further compresses the representation obtained from the second encoder into a fixed-length vector. It consists of an IndRNN layer with 12 units, followed by another IndRNN layer with the same configuration. This layer does not return sequences, resulting in a single output vector. This vector is then replicated to match the shape of the expected decoder input using RepeatVector. Each decoder layer attempts to reconstruct the compressed representation from its corresponding encoder layer. The decoder layers mirror the configurations of their corresponding encoder layers, with Dense layers of appropriate sizes and ReLU activation functions. The output layer is configured to produce a reconstruction for each time step in the sequence using TimeDistributed Dense layer with a single output unit. Finally, the model is compiled using the Adam optimizer with a learning rate of 0.01 and Mean Squared Error (MSE) loss function. The other parameters were the same as previous in SAE.

6.2 : Short-Term Forecasting

For short-term forecasts it was necessary from a data set originally recorded at an hourly frequency, to perform a thorough transformation to enrich the data set with 15-minute intervals. The challenge lay in bridging the temporal gaps to ensure a seamless transition from hourly to quarter-hourly data. Employing a sophisticated imputation method, the missing values were judiciously estimated and filled, meticulously aligning the temporal resolution. This imputation strategy which been followed was, firstly we get the data of one month from all data set for 2009 and we made the transformation from hourly to 15-minutes intervals putting in random values between the hours. Between a pair of hours, the random values that enter are obtained having as limits the values we receive at the specific hours of this pair. After we convert the hourly time series into a 15-minute interval time serie, we use the AAFT algorithm to generate new signals to grow the dataset. At the begin we generate 10000 signals with gaussian distribution,so we have 10000 values for every hour of the month, then we exclude the resulting duplicate values. We continue sorting the surrogates data, we want to have values similar real values so we put a restriction to avoid values which are bigger than the real value '+' standard deviation of real signal and smaller than the real value '-' standard deviation of real signal for every time point corresponding. We arrive at a set of points that fulfill all the assumptions, and based on them we choose the data set that we will use to predict the wind speed. We repeat this process for all months of data set seperately.

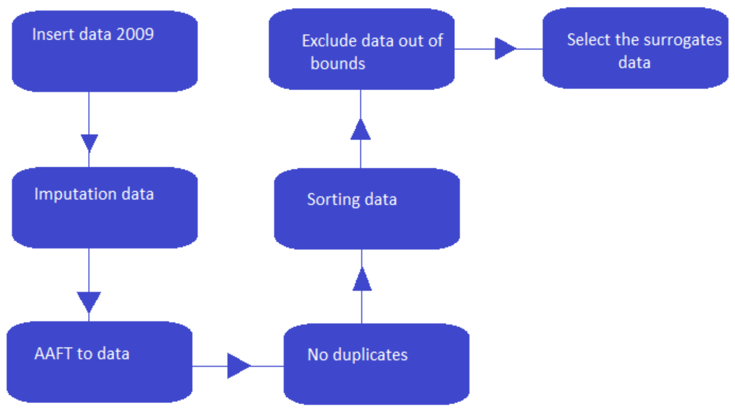


Figure 15: Block diagramm imputation and data surrogation

After we processed our data and we have the appropriate data set, we started the forecasting process. For both models in the short-term forecast the same applies exactly as in the medium-term as we described previously.

CHAPTER 7

Results Medium-Term & Short-term Forecastings

7.1 Traditional Methods of Forecasting

The importance of traditional forecasting methods lies in their ability to provide with a structured and systematic approach to predicting future trends and outcomes. These methods leverage historical data and expert judgment to offer insights that aid decision-making processes. By analyzing past patterns and trends, can make informed predictions. Traditional forecasting methods also provide a baseline for comparison with more advanced techniques, to validate and refine their predictions. Let's describe the most common methods.

1. *Rolling window forecast:*This method involves gradually adding one value from the test set to the training set at a time, training a new model on the updated training set, and using the model to predict the next value in the test set. This process is repeated until the entire test set has been predicted. There are 2 different ways to using the rolling window forecast. The first is the (Fixed or sliding rolling window), the same size of window is used throughout the forecast process and the second is (Expanding rolling window), the window size increases as the forecast process progresses [20].

prediction(t+1)=model(obs(t-1), obs(t-2), ... , obs(t-n))

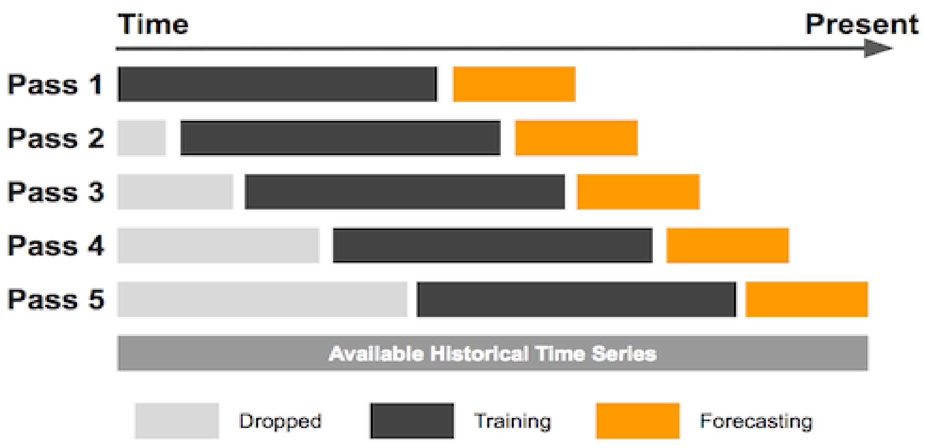


Figure 16: The fixed or sliding rolling window

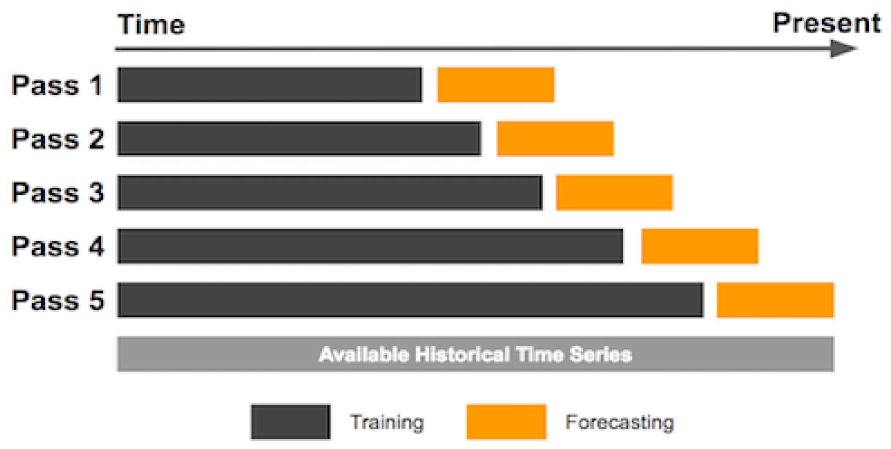


Figure 17: The expanding rolling window

2. *Recursive multistep prediction*: The recursive strategy involves the use of a model of step many times, where the prediction for the previous step is used as an input for the forecast for the next time period. In case of predicting the price of a product for the next two hours, a race prediction model is developed. This model is then used to predict time 1, and then this

forecast to be used as an input for the time forecast 2.

prediction(t+1)=model(obs(t-1), obs(t-2), ... , obs(t-n))
prediction(t+2)=model(prediction(t+1), obs(t-1), obs(t-2), ... , obs(t-n))

3. *Direct-recursive multi-step hybrid prediction*: This strategy, as it is obvious, results from the combination of two previous ones. a separate model can be built for each time step to be predicted, but each model can use the predictions made by models in previous times steps, as entry price. We can see how this can work for prediction of the price of a product for the next two hours, where two models, but the output from the first model is used as input to the second model.

prediction(t+1)=model(obs(t-1), obs(t-2), ... , obs(t-n))
prediction(t+2)=model 2(prediction(t+1), obs(t-1), obs(t-2), ... , obs(t-n))

4. *Multiple prediction strategy*:This strategy involves the development of a model that can to predict the entire prediction sequence in one way. In case predicting the price of a product for the next two hours, we would grow one model and we would use it to predict the next two hours as one operation.
prediction(t+1),prediction(t+2)=model(obs(t-1), obs(t-2), ... , obs(t-n))

Such models are more complex as they can be learned dependency structure between inputs and outputs as well as between outputs. To be more complex can mean they are slower and require more data during their training.

In our case we choose the method (Rolling window forecast) with fixed or sliding rolling window to make predictions. The wind speed forecast results are presented like these emerged after applying the models. are listed comparatively results of the models in relation to the real prices and in parallel the deviations (errors) of predicted and actual price are presented. On the x-axis are the hours of each day and by extension each hour and on the y axis the wind speed but also the deviation of predicted with actual. The forecasts were made using a window, taking into account 1,3,5 previous hours from the forecast time. for each value of the window we have the corresponding subsequences of the data for the prediction.

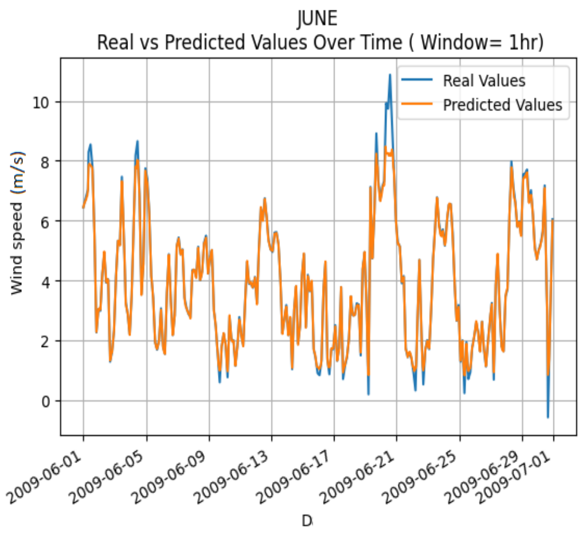
The results of forecasting wind speed are presented like these emerged after applying the models. Are listed comparatively results of the models in relation to the real prices and in parallel the deviations (errors) of predicted and actual price are presented. On the x-axis are the hours of each day for a specific month and on the y-axis the wind speed for every hour correspondingly. Below we list the results for each month for each window for both models, with orange color are the predicted values and with blue color are the real values.

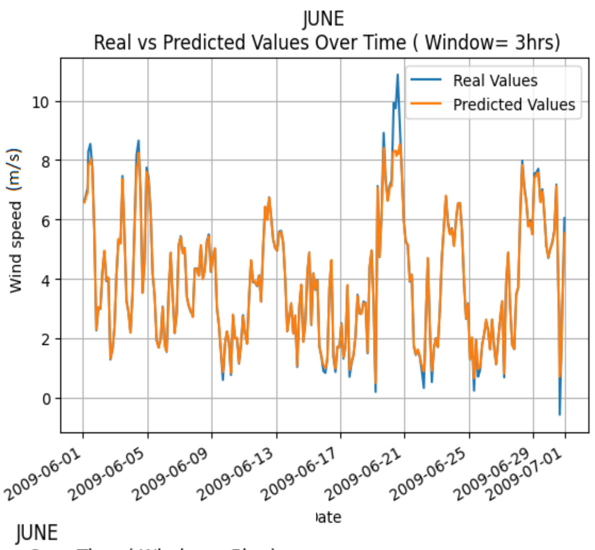
Due to the fact that the number of waveforms of the results is very large because only for the medium-term forecast, having three forecast windows results in the fact that we will have 36 waveforms only for one model. Therefore, it follows that for all the predictions of all windows and for both models, 144 waveforms are obtained, for this reason we list only those waveforms that we consider important enough to draw conclusions. Of course, you can find all the results in the appendix at the end of the text.

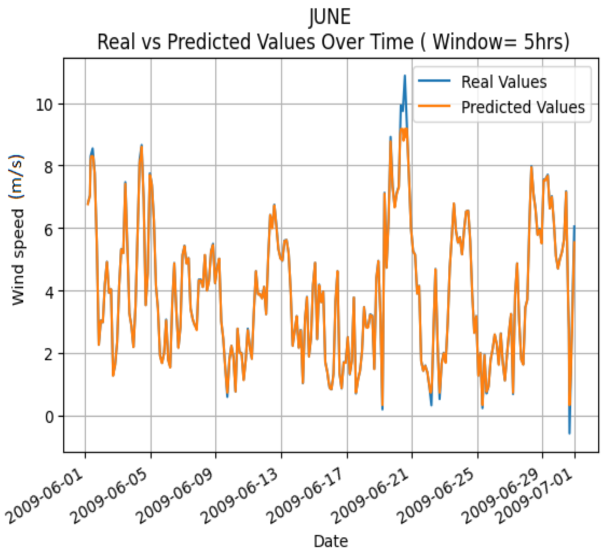
7.2 Results of Medium-term forecasting:

June:

SAE

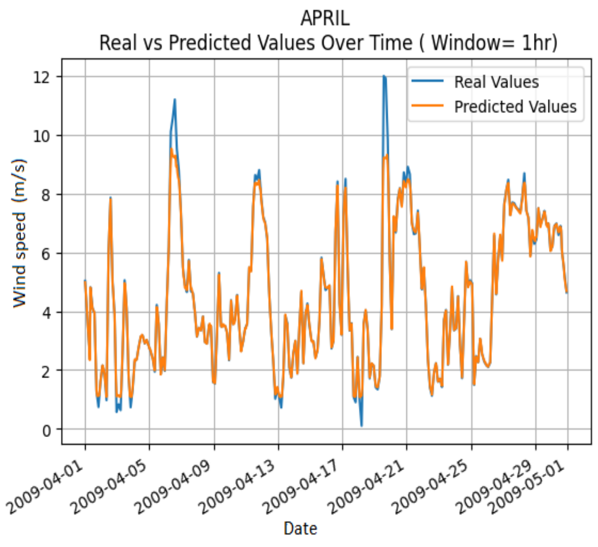


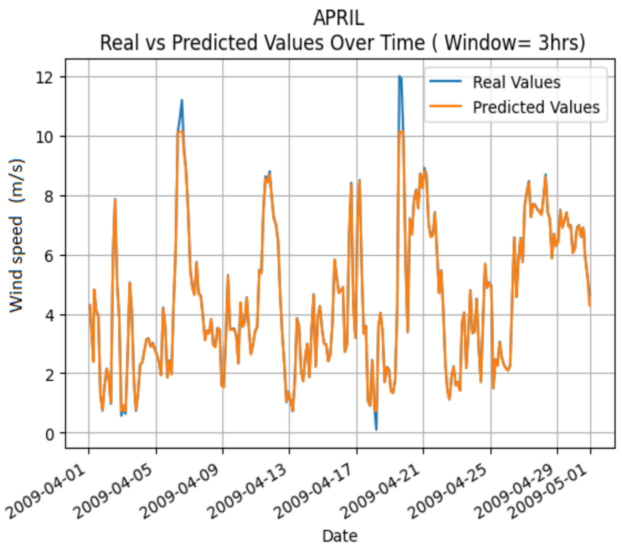


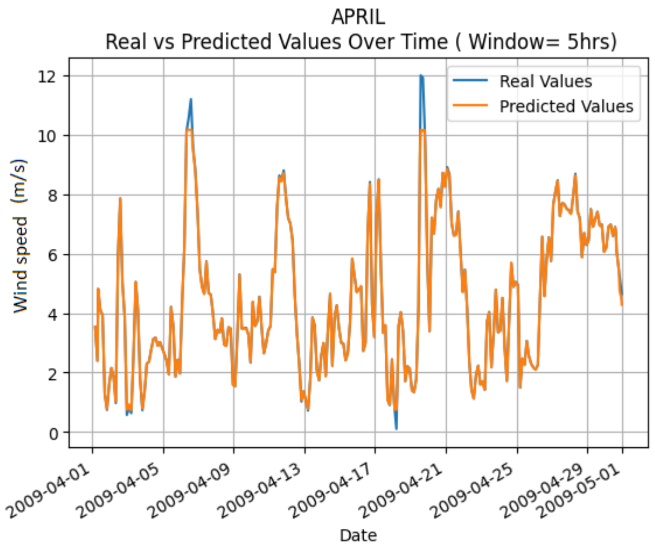


As we can see from the wavefroms about the medium-term forecasting for the month June for all windows. Mainly at these 3 results it is noticable that we have difference between predicted and real values at the edges of wavefrom We see that for window size 5 the line with orange color which represent the predicted values, come closer than the other two predictions, to the real values at the edges of wavefrom.

SIRAE







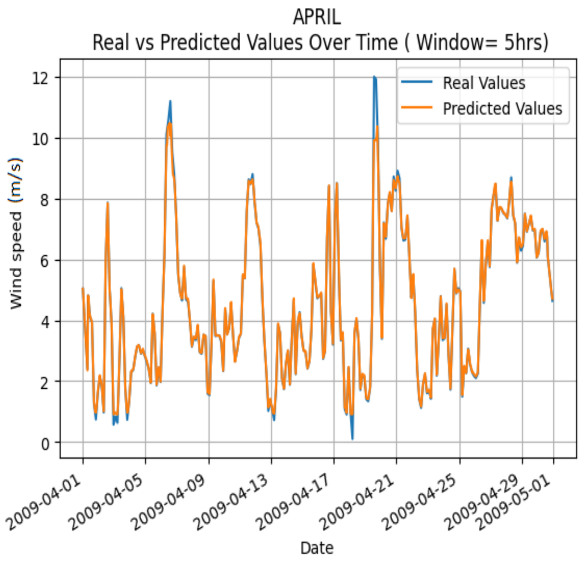
As previous we see the same results for SIRAE model for window size 5 we have better perfomance than the 2 others window sizes.

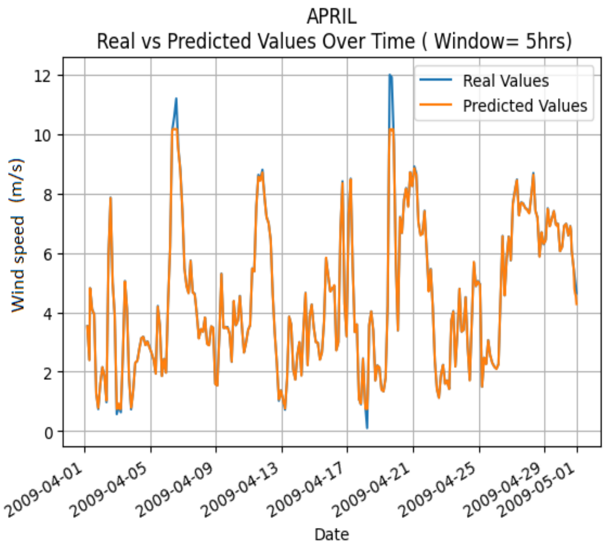
Now let's make a comparison between the two models and not for window sizes. So we will represent the results for the month April for both models with the same window size equals 5.

<u>April:</u>

Window size=5

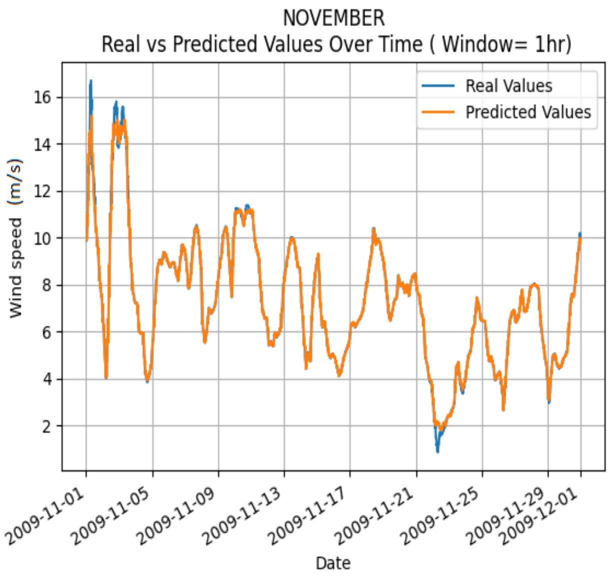
<u>SAE</u> ***<u>SIRAE</u>***

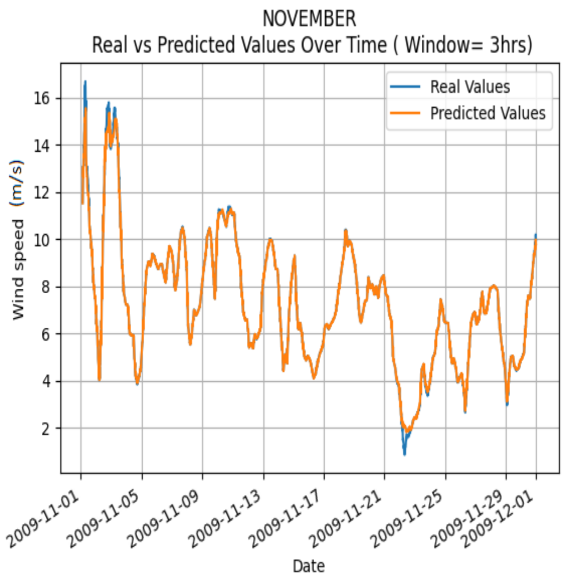


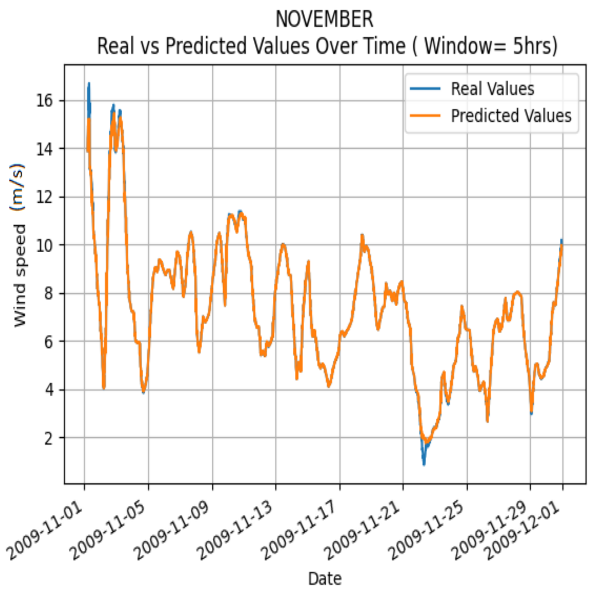


It's obvious that we have a better performance with SIRAE model than the SAE model, we see again the line with color orange is closer to the blue line for SIRAE model.

7.3 Results of Short-term forecasting: *SAE*

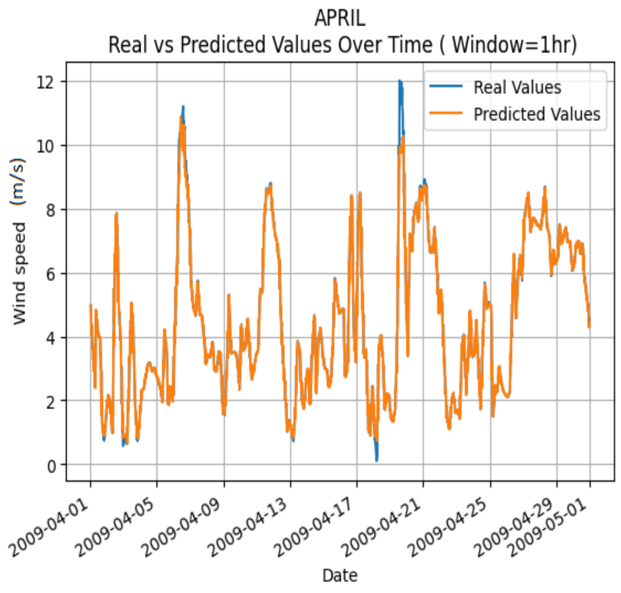


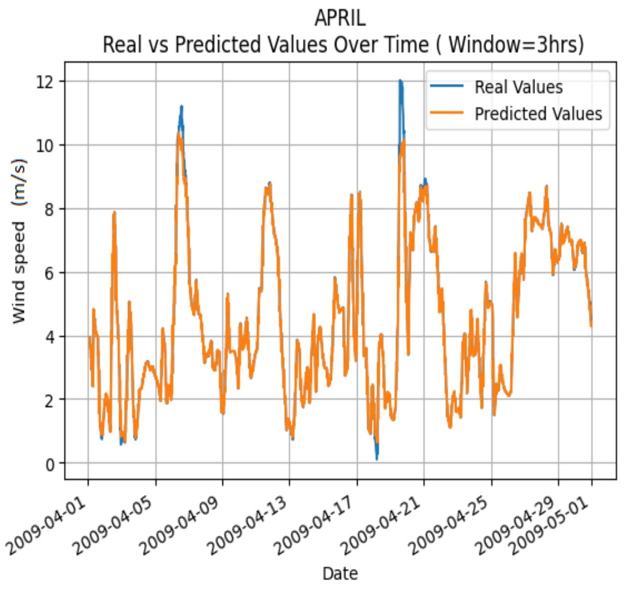


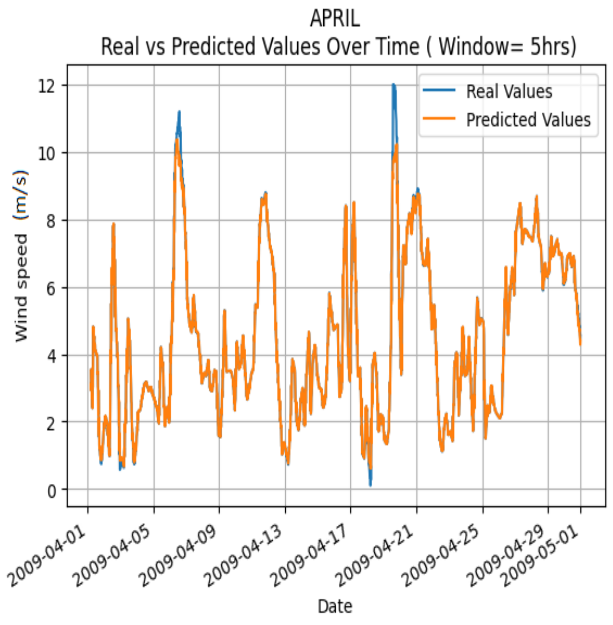


For short-term forecasting with SAE model we observe again as previously at medium-term that when we use window size 5 we have better results than the other two windows. The wavefrom with predicted values has smaller distance than the wavefrom with real values.

SIRAE



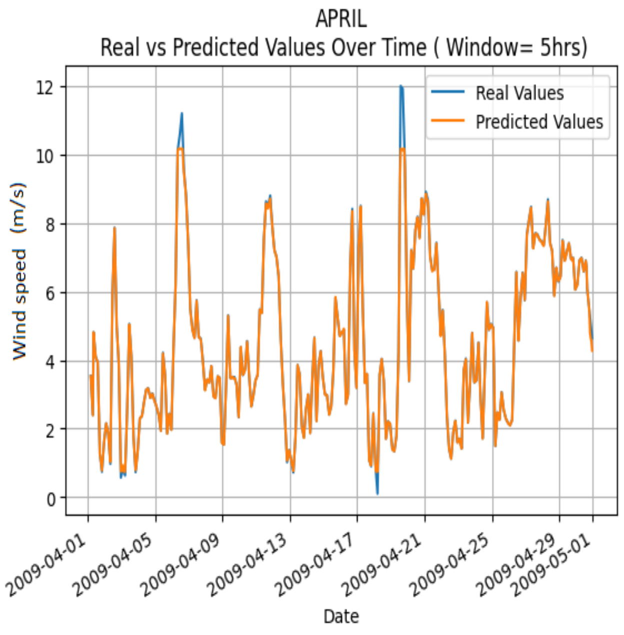


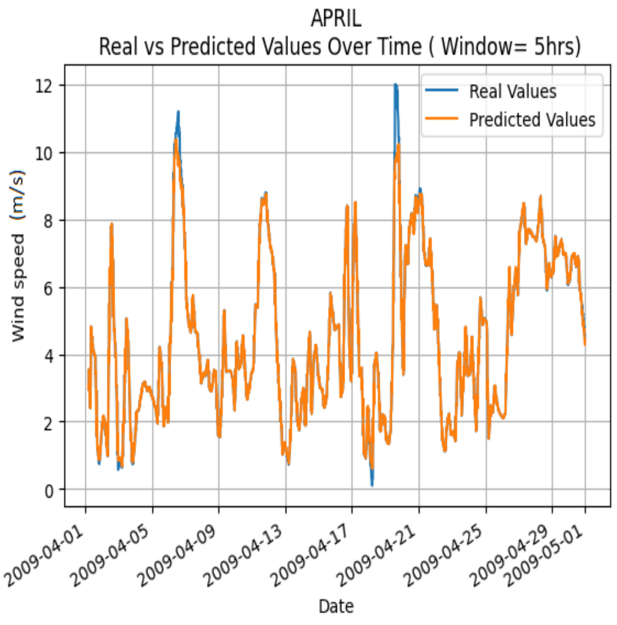


For short-term forecasting with SIRAE model due to that we have very high perfomance the differences between the predictions with different window size are very small, so it's more difficult to understand which selection of window size is better than others from wavefroms. For this reason at the next chapter we quote the measurements MSE and RMSE for all predictions for every window size for both models.

The last comparison we will try from wavefroms for short-term forecasting is between two models which one has better perfomance.

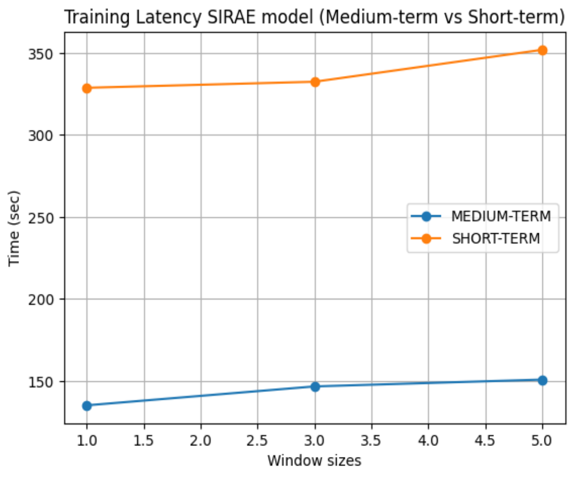
<u>SAE</u> **<u>SIRAE</u>**

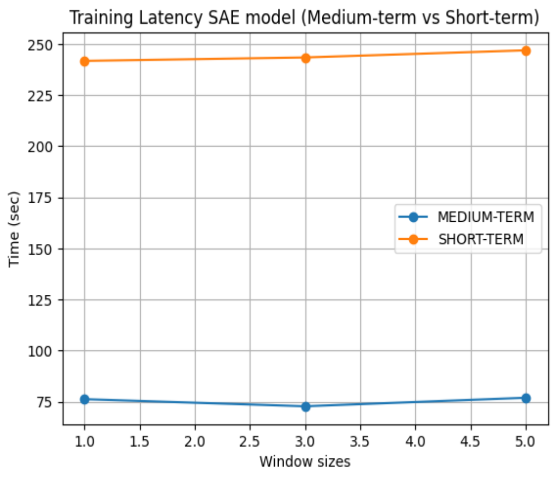




In this case it's clear that the SIRAE model has bigger accuracy than the SAE model as we observe the differences at the edges of wavefroms.

<u>Training Latency of models</u>:





It is important to mention the time delay of training the models, that is why we have illustrated the time delays resulting from training the models for each window. It is clear that as the size of the window increases, we have a greater delay, although the differences are not very big. Where we encounter a significant delay difference is between medium-term and short-term forecasting, where training the models in the short-term forecast requires more time, this is due to the larger amount of data that we enter into the models. At the end we see that the SIRAE model is slower tha SAE model in every case.

CHAPTER 8
Comparison of Results

Medium-term forecasting

The evaluation of the results of the wind speed forecast will be based on some relationships, which measure the deviation (error) between predicted and actual values. The most famous relations which are used mainly in such species forecast problems are:

- Mean squared error (MSE)
- Root mean squared error (RMSE)
- Mean absolute percentage error (MAPE)
- Mean absolute error (MAE)
- Coefficient of determination (R^2)
- Symmetric mean absolute percentage error (SMAPE)
- Normalized mean squared error (NMSE)
- Normalized root mean squared error (NRMSE)
- Normalized mean absolute percentage error (NMAPE)

To compare the results of the medium-term forecast we used the MSE and RMSE as comparison measure. MSE has been described in more detail previously, the RMSE as the name implies, it is the square root of the MSE (RMSE= sqrt(MSE)). Because MSE is squared, its units are not match those of the original output. Given that the MSE and RMSE effects of the differences (residuals) squared, are affected likewise the extreme values. RMSE is proportional to standard deviation (MSE to variance) and shows the distribution of difference results. . Then we list the model performance results table for all windows and for both models.

Then we list the model performance results table for all windows and for both models.

Medium-term forecasting:

SAE

MSE	1hr	3hrs	5hrs
January	0,062	0,054	0,062
February	0,081	0,072	0,077
March	0,045	0,042	0,041
April	0,088	0,081	0,077
May	0,039	0,038	0,037
June	0,063	0,056	0,019
July	0,018	0,038	0,038
August	0,025	0,024	0,022
September	0,025	0,023	0,022
October	0,021	0,017	0,014
November	0,014	0,014	0,014
December	0,039	0,074	0,073

Table 1: The SAE performance results for the medium term forecast (MSE)

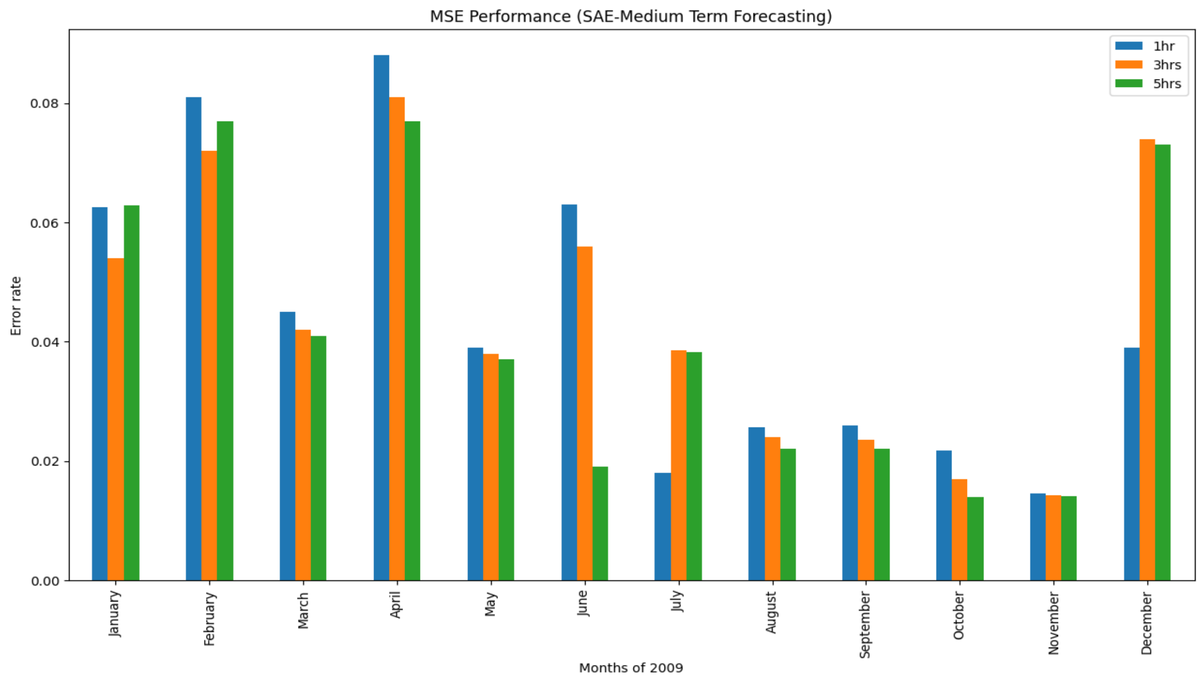


Figure 18: The bar plot of MSE performance for SAE (Medium-Term)

In order to have more distinct results for the performance of each model, we illustrated the MSE values from table 1 in a bar plot.

Firstly at the bar plots we have with blue color the window size equals 1hr,with orange color window size equals 3hrs and with green color the window size equals 5hrs. It's obvious that for the most months the smallest error rate has the window size equals 5hrs. We will do the same for SIRAE model.

SIRAE

MSE	1hr	3hrs	5hrs
January	0,048	0,051	0,036
February	0,035	0,031	0,029
March	0,041	0,04	0,02
April	0,084	0,027	0,026
May	0,033	0,017	0,017
June	0,04	0,017	0,049
July	0,017	0,036	0,019
August	0,017	0,022	0,012
September	0,016	0,01	0,011
October	0,021	0,007	0,007
November	0,058	0,069	0,064
December	0,058	0,038	0,07

Table 2: The SIRAE performance results for the medium term forecast (MSE)

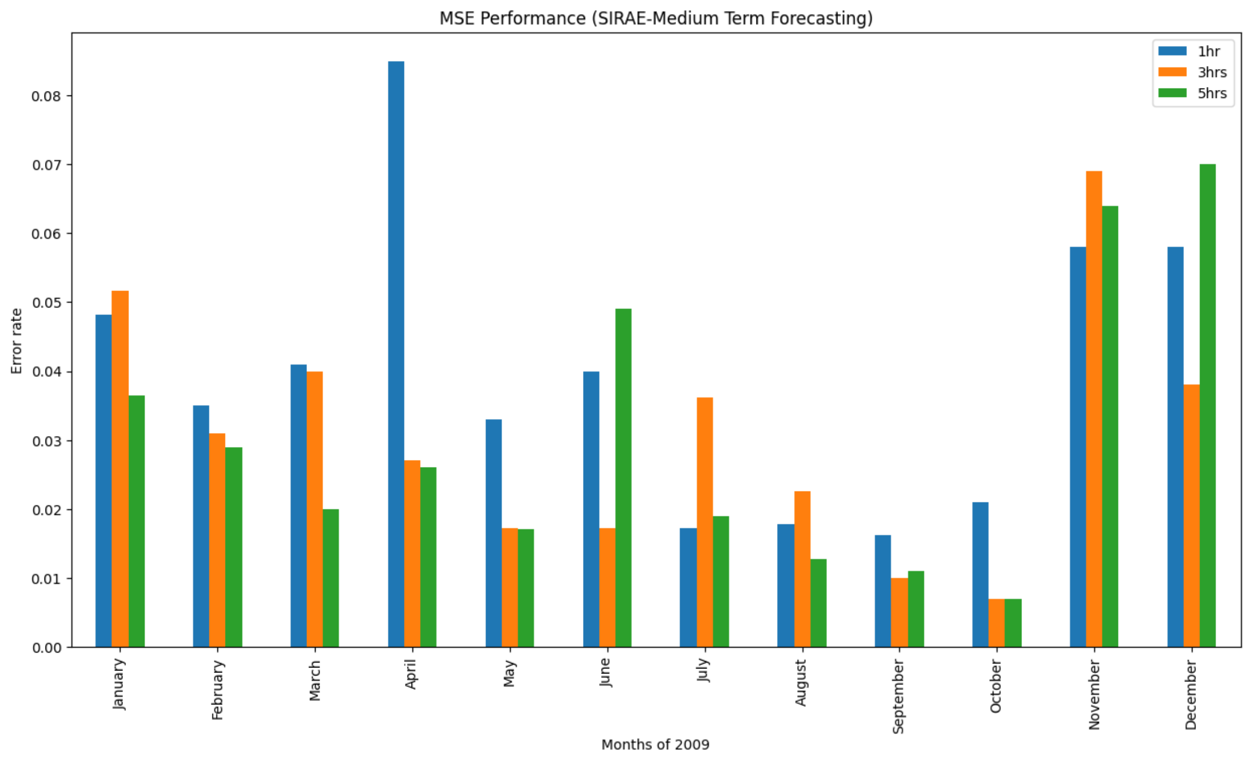


Figure 19: The bar plot of MSE performance for SIRAE (Medium-Term)

For SIRAE model with window size equals 5hrs we have the best performance than the others windows.

Short-term forecasting:

SAE

MSE	1hr	3hrs	5hrs
January	0,032	0,027	0,026
February	0,035	0,011	0,011
March	0,019	0,018	0,015
April	0,04	0,034	0,034
May	0,024	0,021	0,022
June	0,036	0,034	0,028
July	0,017	0,015	0,01
August	0,013	0,011	0,011
September	0,004	0,003	0,002
October	0,007	0,005	0,004
November	0,028	0,019	0,013
December	0,012	0,006	0,006

Table 3: The SAE performance results for the short term forecast (MSE)

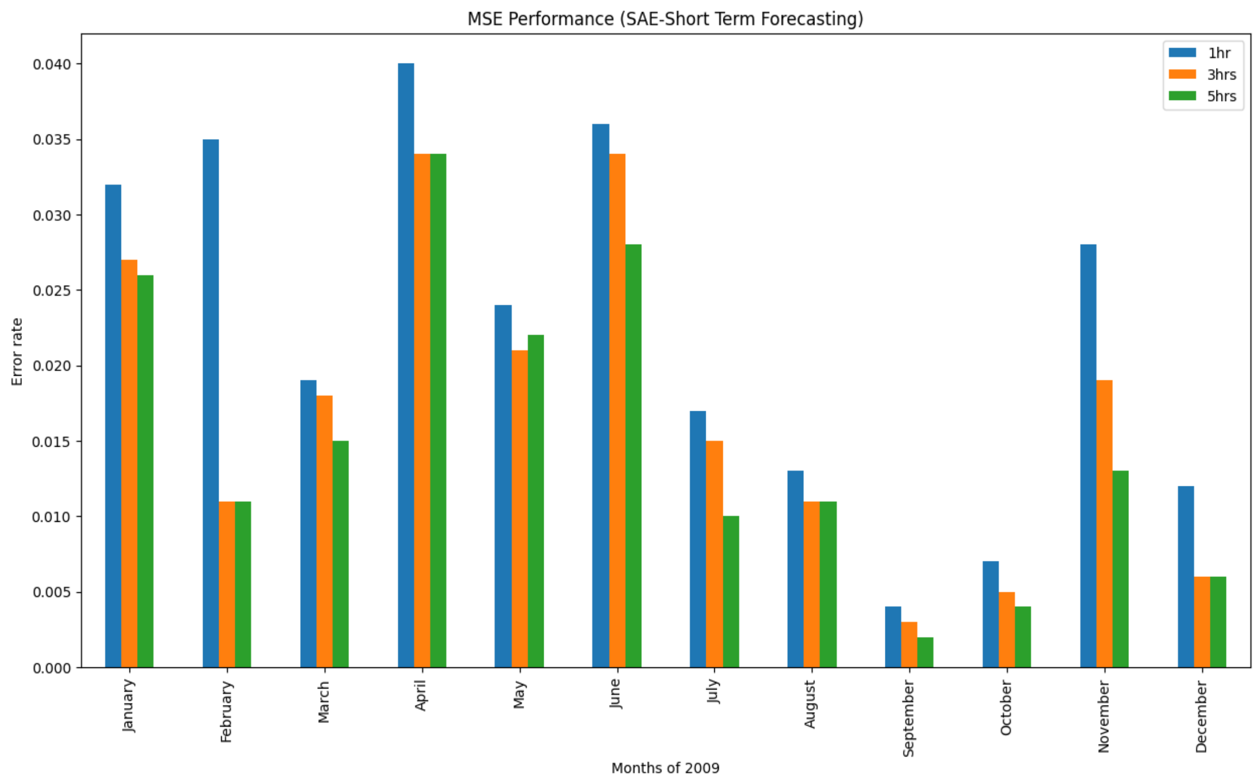


Figure 20: The bar plot of MSE performance for SAE (Short-Term)

For short-term forecasting using the bar plot of MSE perfomance for SAE model we conclude that the window size equals 5hrs has the biggest accuracy than others windows.

SIRAE

MSE	1hr	3hrs	5hrs
January	0,028	0,03	0,027
February	0,013	0,016	0,011
March	0,009	0,017	0,014
April	0,03	0,036	0,015
May	0,006	0,023	0,022
June	0,013	0,01	0,01
July	0,007	0,006	0,004
August	0,016	0,008	0,009
September	0,004	0,003	0,004
October	0,006	0,004	0,003
November	0,005	0,018	0,014
December	0,014	0,011	0,006

Table 4: The SIRAE performance results for the short term forecast (MSE)

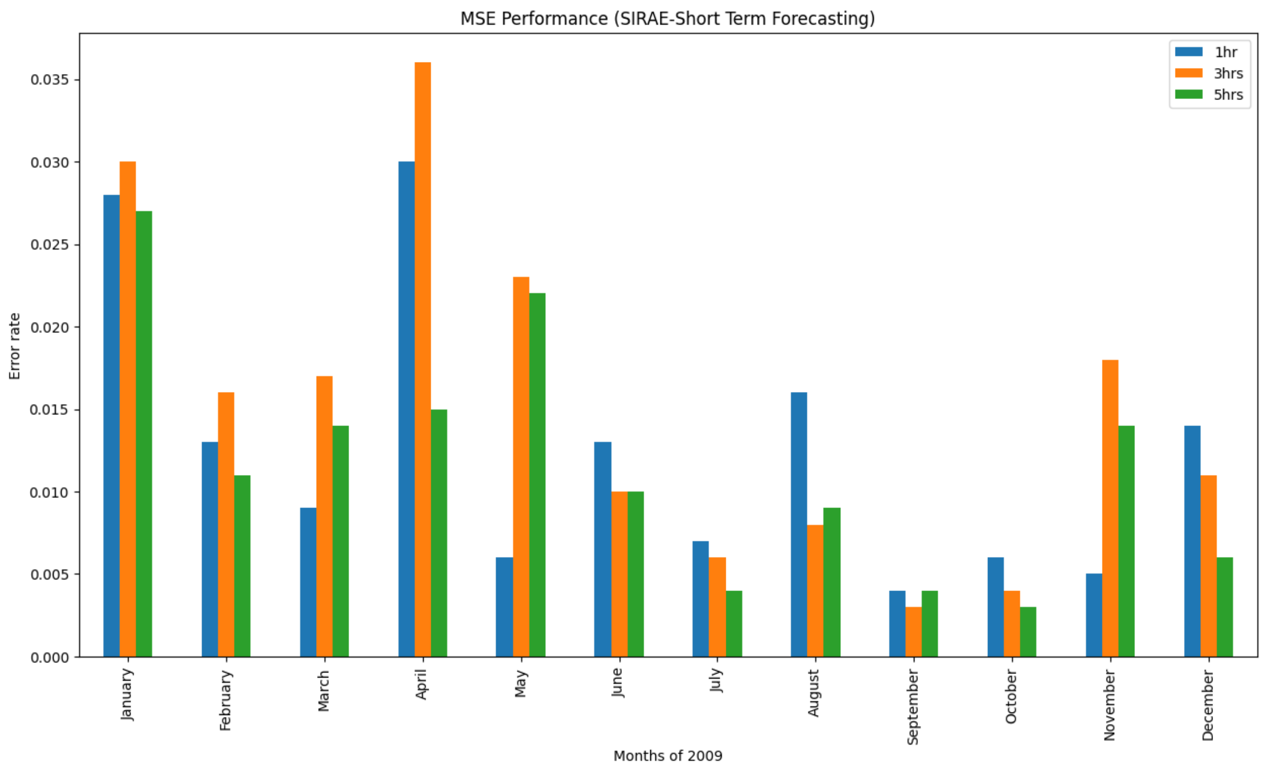


Figure 21: The bar plot of MSE performance for SIRAE (Short-Term)

We see again in case of short-term with SIRAE model that the window size equals 5hrs is more efficient than others windows.

For the comparison between SAE and SIRAE model about their performance we need to calculate the RMSE for both models. So thus, the tables with the RMSE measurements are listed, also the mean values of the RMSE were calculated for each window for both models.

Medium-term forecasting :

SAE

RMSE	1hr	3hrs	5hrs
January	0,25	0,232	0,25
February	0,284	0,268	0,277
March	0,212	0,204	0,202
April	0,296	0,284	0,277
May	0,197	0,194	0,192
June	0,25	0,236	0,137
July	0,134	0,196	0,195
August	0,16	0,154	0,148
September	0,16	0,153	0,148
October	0,147	0,13	0,118
November	0,12	0,119	0,118
December	0,197	0,272	0,27

Table 5: The SAE performance results for the medium term forecast (RMSE)

SIRAE

RMSE	1hr	3hrs	5hrs
January	0,219	0,227	0,191
February	0,187	0,176	0,17
March	0,202	0,2	0,141
April	0,291	0,164	0,161
May	0,181	0,131	0,13
June	0,2	0,131	0,221
July	0,131	0,19	0,137
August	0,133	0,15	0,113
September	0,127	0,1	0,104
October	0,144	0,083	0,083
November	0,24	0,262	0,252
December	0,24	0,194	0,264

Table 6: The SIRAE performance results for the medium term forecast (RMSE)

In order to have an easier and clearer comparison between the two models we calculated the average of the RMSE for each window.

MSE	1hr	3hrs	5hrs
AVERAGE	0,2	0,203	0,194

Table 7: Mean values of SAE model (RMSE)

RMSE	1hr	3hrs	5hrs
AVERAGE	0,191	0,167	0,163

Table 8: Mean values of SIRAE model (RMSE)

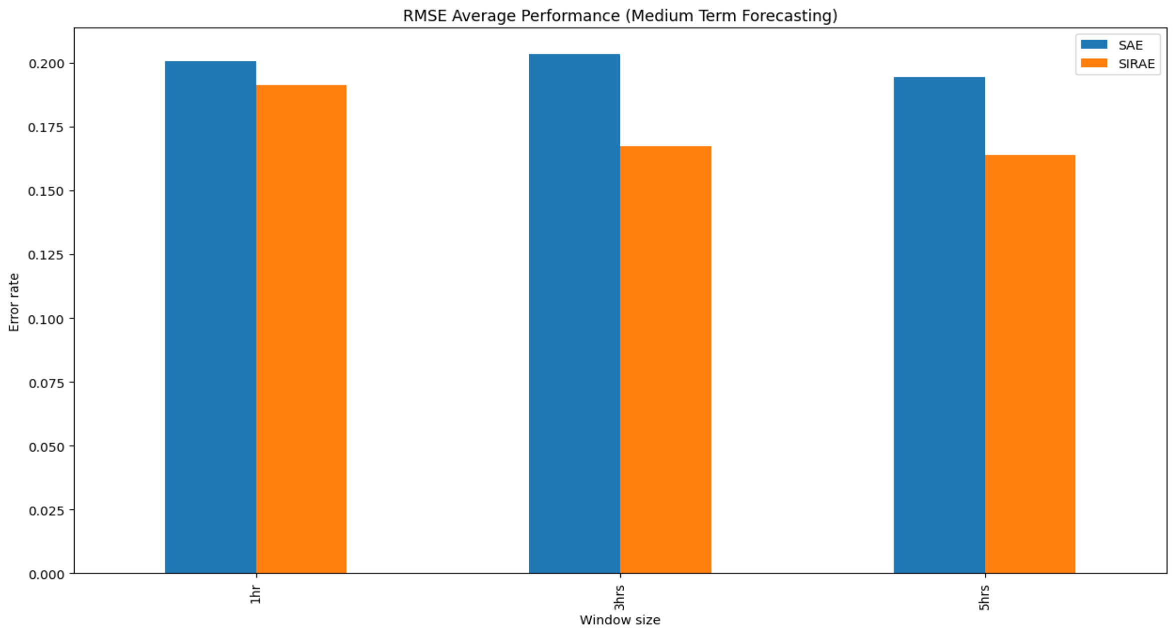


Figure 22: RMSE average performance (Medium-Term)

Short-term forecasting:

SAE

RMSE	1hr	3hrs	5hrs
January	0,178	0,164	0,161
February	0,187	0,104	0,104
March	0,137	0,134	0,122
April	0,2	0,184	0,184
May	0,154	0,144	0,148
June	0,189	0,184	0,167
July	0,13	0,122	0,1
August	0,114	0,104	0,104
September	0,063	0,054	0,044
October	0,083	0,07	0,063
November	0,167	0,137	0,114
December	0,109	0,077	0,077

Table 9: The SAE performance results for the short term forecast (RMSE)

SIRAE

RMSE	1hr	3hrs	5hrs
January	0,167	0,173	0,164
February	0,114	0,126	0,104
March	0,094	0,13	0,118
April	0,173	0,189	0,122
May	0,077	0,151	0,148
June	0,114	0,1	0,1
July	0,083	0,077	0,063
August	0,126	0,089	0,094
September	0,063	0,054	0,063
October	0,077	0,063	0,054
November	0,0707	0,134	0,118
December	0,118	0,104	0,077

Table 10: The SIRAE performance results for the short term forecast (RMSE)

MSE	1hr	3hrs	5hrs
AVERAGE	0,142	0,123	0,115

Table 11: Mean values of SAE model (RMSE) short-term forecasting

RMSE	1hr	3hrs	5hrs
AVERAGE	0,106	0,115	0,102

Table 12: Mean values of SIRAE model (RMSE) short-term forecasting

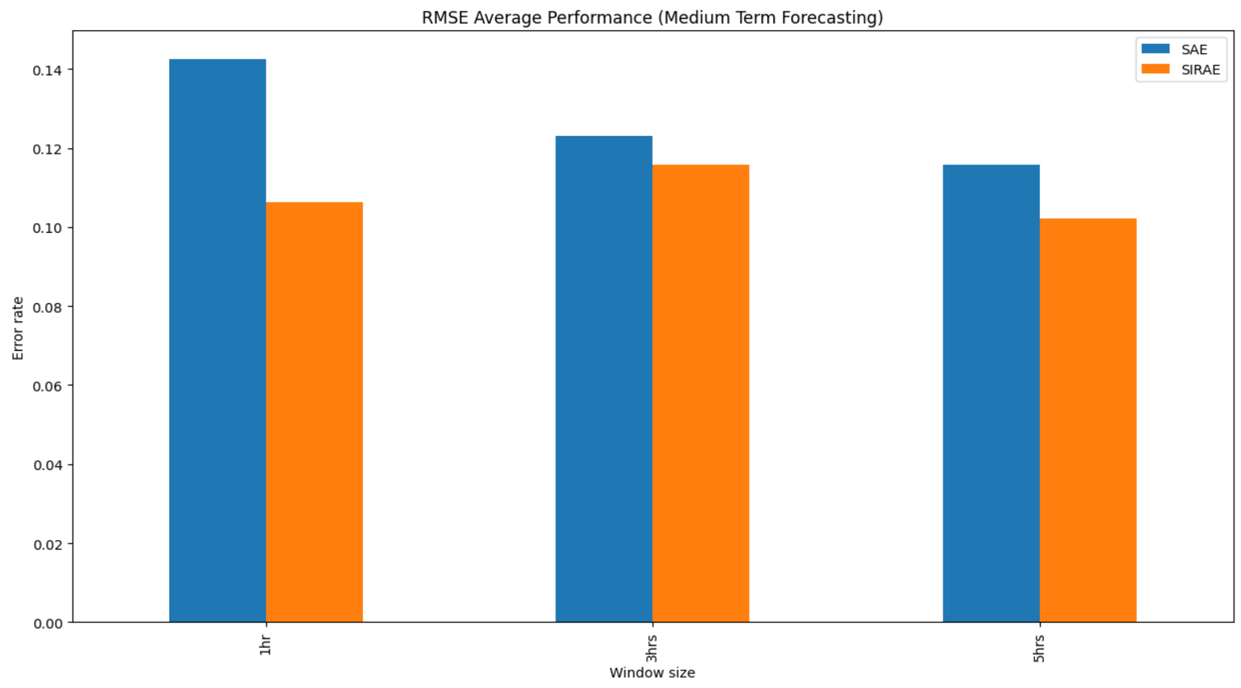


Figure 23: RMSE average performance (Short-Term)

Examining the MSE and RMSE performance tables and figures for short-term and medium-term forecasts reveals minimal distinctions between the measurements. Nevertheless, it is evident that the SIRAE model outperforms the SAE model. Notably, for both models, predictions generated with a window size of 5 demonstrated superior effectiveness. Also we can say that in short-term prediction we have bigger accuracy in relation with the medium term for both models and for all window sizes.

CHAPTER 9
Conclusions Observations

In this thesis, the medium-term and short-term forecast of the wind speed for one year for the island of DIA based on historical data received from the laboratory. Having data for the years 2007, 2008 and 2009, we found that it was not enough for the univariate prediction of the wind speed. Thus choosen the data of the year 2009 and with the (AAFT) method was created surrogate data for the predictions to be made. Then designed two models (SAE) and (SIRAE) in order to compare the performances of these two models. The forecasts were made with the rolling window method, where we had 3 cases of windows for (1,3,5) hours. Therefore, our study showed which model is more efficient and which window size is suitable. It is worth noting that for the short-term forecast we had to use the imputation method so that our data from where they had hourly values have values every 15 minutes. After this step was carried out we continued as for the mid-term forecast.

Medium-term forecasting

1)Prediction accuracy of the models: Regarding the accuracy of models using MSE and RMSE as performance metric we conclude that we have a very high accuracy prediction as long as these two metrics approaches zero values. Both models had a very high performance, but the SIRAE model excels in relation to the SAE. Also for both models we can say about the window size that we had better results when the window size was equal to 5, this is due to the fact that we enter more data for a specific forecast time point than the other windows.

2)Forecast accuracy by season:From the medium-term results prediction it is clear that the worst prediction accuracy is observed in winter months, in all forecast models. This is expected if we consider that in winter the weather is traditionally colder unstable until the summer months, where the weather conditions are better, the prediction accuracy improves and the errors become many small ones.

3)Training latency: For the training time the SIRAE model is slower compared to the SAE model, however we gain a better performance. Also the differences between the window sizes about the latency we conclude that as the window size getting bigger we have bigger latency.The parameters of the models used play an important role in the training and prediction time.

4)Comparison of results with existing literature: In bibliography beyond the classic models there are models which have arise through the combination of various techniques and models, the so-called hybrid models. Specifically, in the paper [11] the SIRAE model has an average improvement of the RMSE of approximately 35-36% is relation with SAE model, while in the specific diplomacy we have a percentage equal to 12.5%. This difference is due to the fact that in this work we had more accurate estimates for both models (SAE, SIRAE), despite this we still have a significant improvement in results between the two models.

Short-term forecasting:

1)In assessing the prediction accuracy of our models using Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) as performance metrics, it is evident that a good prediction is achieved when these metrics approach zero values. As previously mentioned, both models performed well, but the SIRAE model demonstrated superior performance compared to the SAE model. Notably, the choice of window size played a crucial role in model performance. We observed that a window size of 5 yielded the best results for both models. This can be attributed to the fact that using a window size of 5 allowed us to incorporate more relevant data for a specific forecast time point, leading to improved predictive accuracy.

2)Forecast accuracy by season: We have the same results with the medium-term forecasting about the accuracy by season, the months of summer we have better accuracy than the months of winter for both models.

3)Training latency: For the training time the SIRAE model is slower compared to the SAE model, however we gain a better performance. We have the same conclusion about the latency and the window sizes with the medium-term forecasting. Also it's important to mention that between the medium-term and short-term there is difference about the training time and predicting time. The short-term is significant slower than medium-term because in this case we have almost four times bigger data set for processing.

4)Comparison of results with existing literature: In this case the SIRAE model has an average improvement 15,7% than the SAE model which is again smaller than the improvement of paper [11]. The reasons are the same as we described before in the medium-term forecasting.

Future Research

To overcome the main limitations appeared of the present thesis, some extensions could potentially be implemented. For example, the same models could be used by using multiple variables on which the wind speed depends. In this manner, we would proceed with more information and possibly no need of a very large amount of historical data. Also, the AAFT-driven generated data played a vital role because an almost completely new dataset was essentially created and served the used dataset. An additional set of signal augmentation algorithms can be used in the near future in order to provide a more qualitative data which could potentially lead to better predictions. Moreover, the use of the method 'Rolling window forecast' with fixed window to make predictions could be replaced by using some other methods like 'Multiple prediction strategy' which could improve the overall forecasting time. Another aspect which was wasn't tackled in this thesis was the use of cross-validation strategies which could lead to more robust prediction performance. By investigating also methods to exclude secondary sequences reported in the test set from the training set could improve the adaptability of the model. In addition, the accurate prediction of the speed is necessary for the estimation of the electricity produced by wind systems. So new research related to wind systems or even related to wind energy can be based on this specific work. Finally using the tied weights method for autoencoders will reduce the risk of over-fitting and speed up the training process as its known from literature.

Bibliography

[1]Blazakis, K., Katsigiannis, Y.A. and Stavrakakis, G. (2022) 'One-Day-Ahead Solar Irradiation and Windspeed Forecasting with Advanced Deep Learning Techniques,' *Energies*, 15(12), p. 4361. https://doi.org/10.3390/en15124361.

[2]Wikipedia Contributors. (2018, December 6).*Artificial neural network*. Wikipedia; Wikimedia Foundation. https://en.wikipedia.org/wiki/Artificial_neural_network

[3]Chowdhury, S. (no date) *Research paper classification using Supervised Machine Learning Techniques*. Available at: https://ieeexplore.ieee.org/document/9249211

[4]Rong, S. and Bao-wen, Z. (2018). The research of regression model in machine learning field. *MATEC Web of Conferences* [online] 176, p.01033.doi:https://doi.org/10.1051/matecconf/201817601033.

[5]Sarker, I.H. (2021). Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions. *SN Computer Science*, [online] 2(6). doi:https://doi.org/10.1007/s42979-021-00815-1.

[6] Zhang, M. (2018).*Time Series: Autoregressive models AR, MA, ARMA, ARIMA Overview 1 Introduction of Time Series Categories and Terminologies White Noise and Random Walk Time Series Analysis*. [online] Available at: https://people.cs.pitt.edu/~milos/courses/cs3750/lectures/class16.pdf.

[7]www.sciencedirect.com. (n.d.).*Autoregressive Integrated Moving Average - an overview | ScienceDirect Topics*. [online] Available at: https://www.sciencedirect.com/topics/mathematics/autoregressive-integrated-moving-average.

[8]Rastogi, V. (2023).*The ARMA (Autoregressive Moving Average) model is a popular statistical model used in time series…*. [online] Medium. Available at: https://medium.com/@vaibhav1403/the-arma-autoregressive-moving-average-model-is-a-popular-statistical-model-used-in-time-series-24ce43049366 .

[9]Calzone, O. (2022).*An Intuitive Explanation of LSTM*. [online] Medium. Available at: https://medium.com/@ottaviocalzone/an-intuitive-explanation-of-lstm-a035eb6ab42c.

[10]Sujan Ghimire, Liu, B., Wang, H., Al-Musaylh, M.S., Casillas-Perez, D. and Sancho Salcedo-Sanz (2022). Stacked LSTM Sequence-to-Sequence Autoencoder with Feature Selection for Daily Solar Radiation Prediction: A Review and New Modeling Results. 15(3), pp.1061–1061. doi:https://doi.org/10.3390/en15031061.

[11]Wang, L., Tao, R., Hu, H. and Zeng, Y.-R. (2021). Effective wind power prediction using novel deep learning network: Stacked independently recurrent autoencoder. *Renewable Energy*, 164, pp.642–655. doi:https://doi.org/10.1016/j.renene.2020.09.108.

[12] Hallaj, P. (2023) 'Data augmentation in Python | By Pouya Hallaj | Medium,' *Medium*, 21 September. https://medium.com/@pouyahallaj/data-augmentation-benefits-and-disadvantages-38d8201aead.

[13] Shorten, C. and Khoshgoftaar, T.M. (2019b) 'A survey on Image Data Augmentation for Deep Learning,' *Journal of Big Data*, 6(1). https://doi.org/10.1186/s40537-019-0197-0.

[14]Takimoglu, A. (2021). *What is Data Augmentation? Techniques, Benefit and Examples*. [online] research.aimultiple.com. Available at: https://research.aimultiple.com/data-augmentation/.

[15] Iglesias, G. *et al.* (2023) 'Data Augmentation techniques in time series domain: a survey and taxonomy,' *Neural Computing and Applications*, 35(14), pp. 10123–10145. https://doi.org/10.1007/s00521-023-08459-3.

[16]Räth, C. and Monetti, R. (2018). *Surrogates with random Fourier Phases*. [online] Available at: https://arxiv.org/pdf/0812.2380.pdf

[17]Simplilearn.com. (2022). *Introduction to Data Imputation | Simplilearn*. [online] Available at: https://www.simplilearn.com/data-imputation-article.

[18]Agrawal, R. (2021). *Interpolation Techniques Guide & Benefits | Data Analysis (Updated 2023)*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2021/06/power-of-interpolation-in-python-to-fill-missing-values/.

[19]Dash, S.K. (2022). *Handling Missing Values with Random Forest*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2022/05/handling-missing-values-with-random-forest/.

[20]Andrés, D. and Andrés, D. (2023) 'Forecasting methods in Time Series - ML Pills,' *ML Pills - Machine Learning Pills*, 19 January. https://mlpills.dev/time-series/forecasting-in-time-series/.

[21]ieeexplore.ieee.org. (n.d.). *A brief comparison of different learning methods for wind speed forecasting | IEEE Conference Publication | IEEE Xplore*. [online] Available at: https://ieeexplore.ieee.org/document/9258733.

[22]*A Gentle Introduction to LSTM Autoencoders*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/lstm-autoencoders/.

[23]ieeexplore.ieee.org. (n.d.). *Deep Learning based Wind Speed Forecasting-A Review | IEEE Conference Publication | IEEE Xplore*. [online] Available at: https://ieeexplore.ieee.org/document/8776923.

[24]ieeexplore.ieee.org. (n.d.). *Robust deep neural network for wind speed prediction | IEEE Conference Publication | IEEE Xplore*. [online] Available at: https://ieeexplore.ieee.org/document/7391664.

[25]Meka, R., Alaeddini, A. and Bhaganagar, K. (2021). A robust deep learning framework for short-term wind power forecast of a full-scale wind farm using atmospheric variables. *Energy*, 221, p.119759. doi:https://doi.org/10.1016/j.energy.2021.119759.

[26]“Deep Learning to Predict the Generation of a Wind Farm”, J.M. Torres, R.M. Aguilar, and K.V. Zuniga-Meneses

[27] “Multi-Step Wind Speed Forecasting Based On Ensemble Empirical Mode Decomposition, Long Short Term Memory Network and Error Correction Strategy”, Yuansheng Huang , Lei Yang 1, Shijian Liu and Guangli Wan

[28]www.ijsgce.com. (n.d.). *Short-term wind power forecasting using long-short term memory based recurrent neural network model and variable selection - Vol. 8, No. 2, March 2019 - International Journal of Smart Grid and Clean Energy (SGCE).* [online] Available at: https://www.ijsgce.com/index.php?m=content&c=index&a=show&catid=78&id=418.

[29]Brahimi, T. (2019). Using Artificial Intelligence to Predict Wind Speed for Energy Application in Saudi Arabia. *Energies*, 12(24), p.4669. doi:https://doi.org/10.3390/en12244669.

[30]González-Sopeña, J.M., Pakrashi, V. and Ghosh, B. (2021). An overview of performance evaluation metrics for short-term statistical wind power forecasting. *Renewable and Sustainable Energy Reviews*, 138, p.110515. doi:https://doi.org/10.1016/j.rser.2020.110515.

[31]Bakshi, R. (2021). *Stacked Autoencoders.* [online] Medium. Available at: https://towardsdatascience.com/stacked-autoencoders-f0a4391ae282.

[32]PhD, E.G. (2023). *Unveiling the Power of Stacked Autoencoders*. [online] Medium. Available at: https://medium.com/@evertongomede/unveiling-the-power-of-stacked-autoencoders-675de2ce4273 [Accessed 25 Jan. 2024].

[33]Kugiumtzis, D. (n.d.). *COMPLICATIONS IN APPLYING THE METHOD OF SURROGATE DATA TO EEG*. [online] Available at: http://users.auth.gr/dkugiu/PDFs/ProcSurEEG.pdf

[34]Suzuki, T., Ikeguchi, T. and Suzuki, M. (2007). Algorithms for generating surrogate data for sparsely quantized time series. *Physica D: Nonlinear Phenomena*, [online] 231(2), pp.108–115. doi:https://doi.org/10.1016/j.physd.2007.04.006.

[35]www.altumintelligence.com. (n.d.). *Time Series Prediction Using LSTM Deep Neural Networks*. [online] Available at: https://www.altumintelligence.com/articles/a/Time-Series-Prediction-Using-LSTM-Deep-Neural-Networks.

[36]ieeexplore.ieee.org. (n.d.). *An LSTM-SAE-Based Behind-the-Meter Load Forecasting Method | IEEE Journals & Magazine | IEEE Xplore*. [online] Available at: https://ieeexplore.ieee.org/document/10124738.

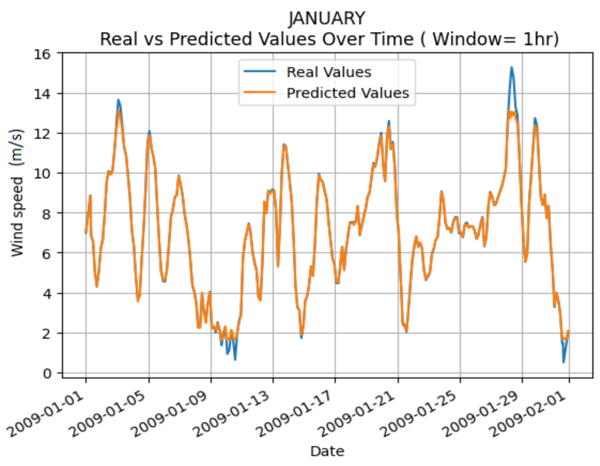
[37]Xu, X. and Ren, W. (2022). A hybrid model of stacked autoencoder and modified particle swarm optimization for multivariate chaotic time series forecasting. *Applied Soft Computing*, 116, p.108321. doi:https://doi.org/10.1016/j.asoc.2021.108321.

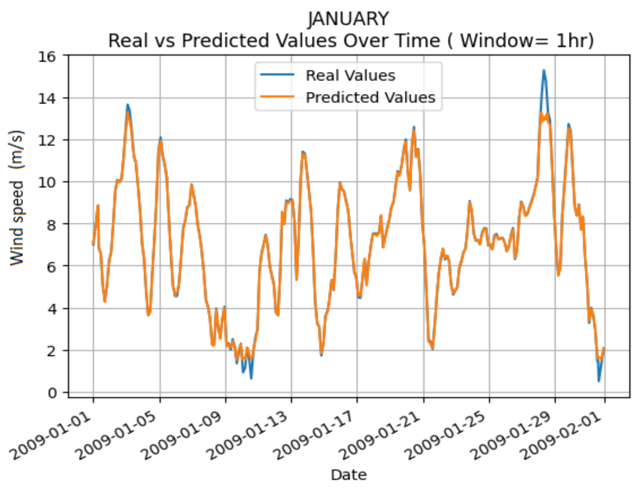
FIGURE ATTACHMENTS

January:

SAE ***SIRAE***

Window size=1

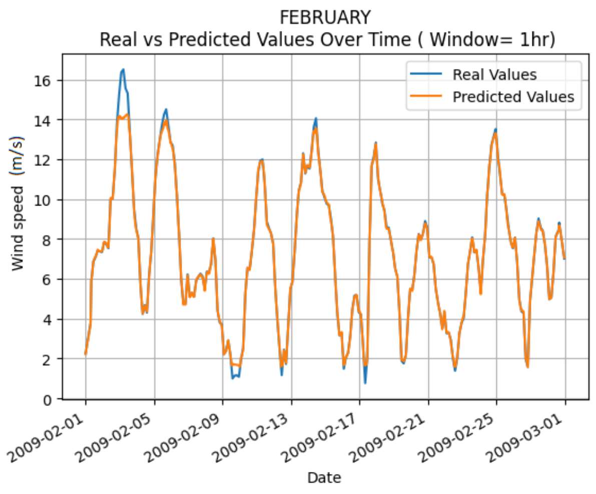


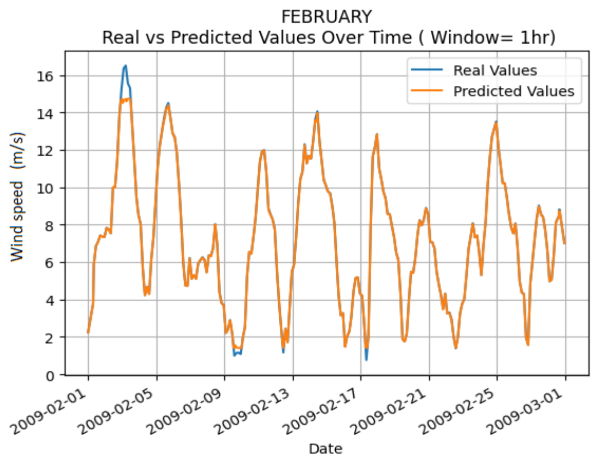


February:

Window size=1

SAE ***SIRAE***

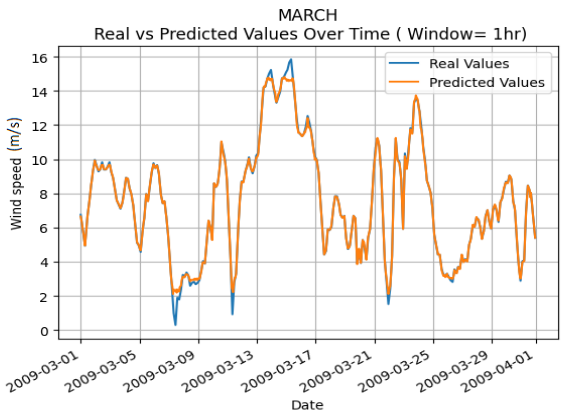


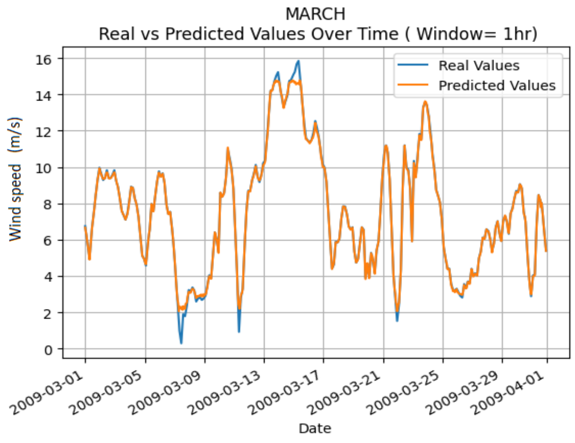


March:

Window size=1

SAE ***SIRAE***

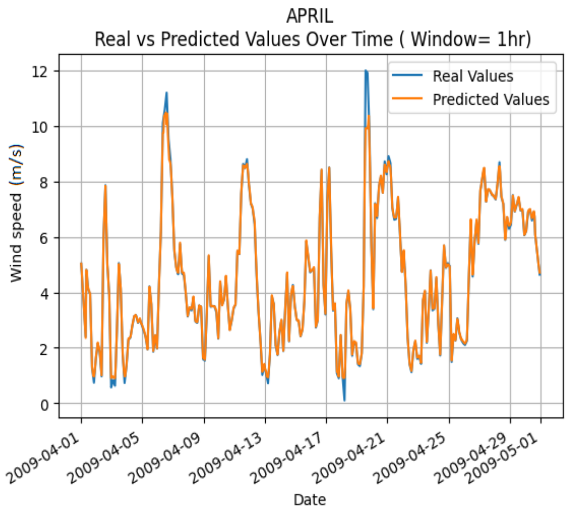


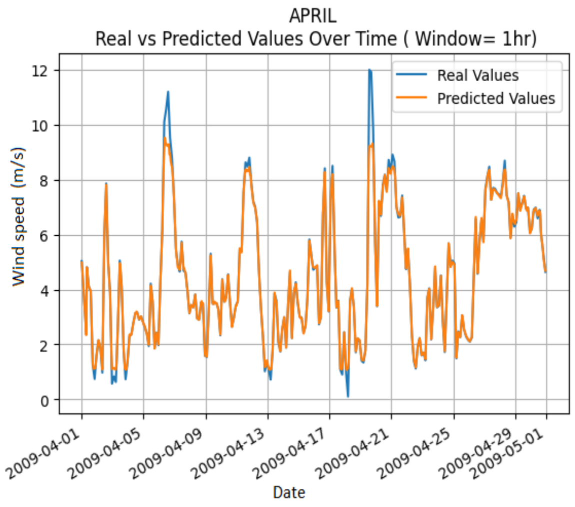


April:

Window size=1

SAE ***SIRAE***

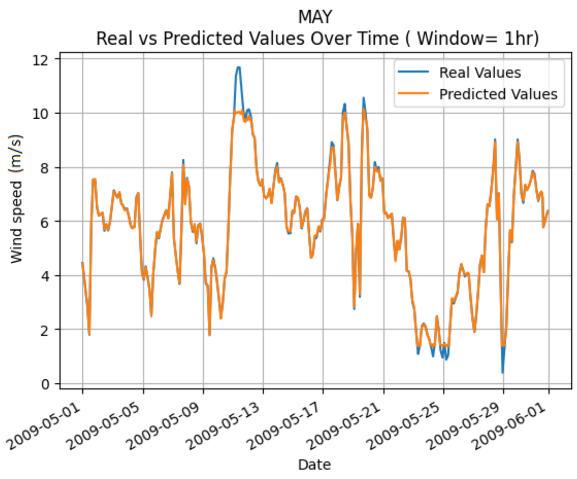


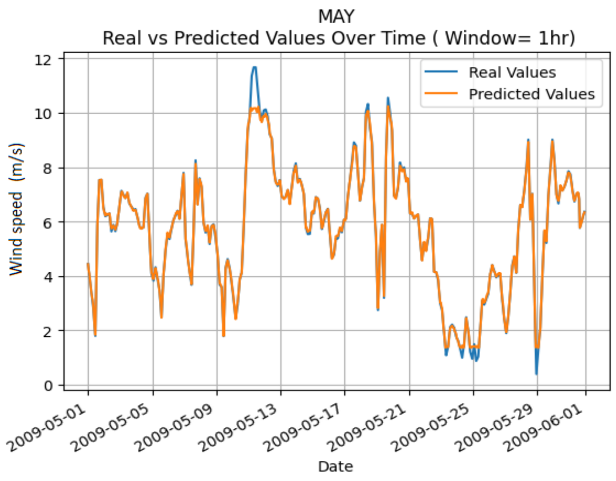


May:

SAE ***SIRAE***

Window size=1

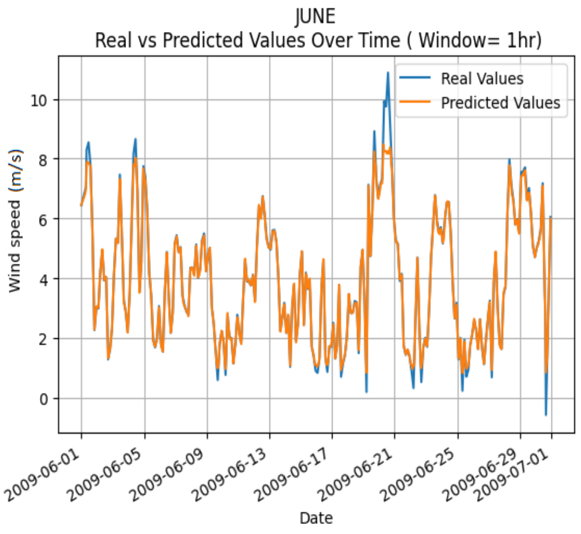


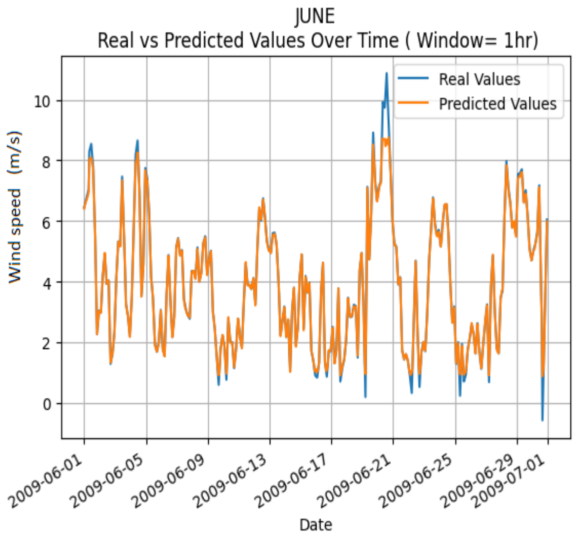


June:

Window size=1

SAE ***SIRAE***



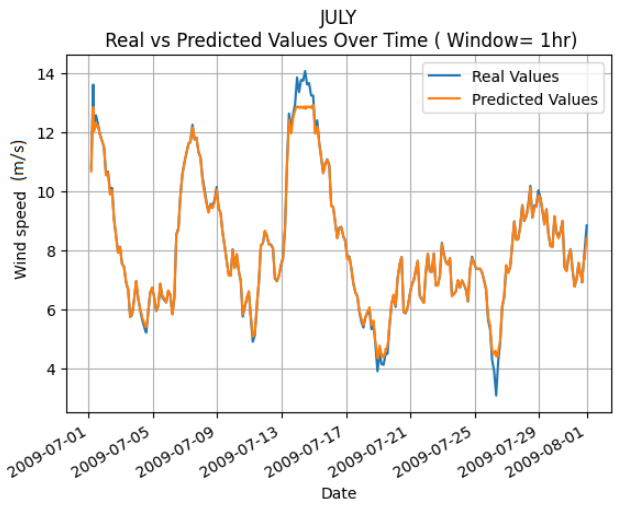


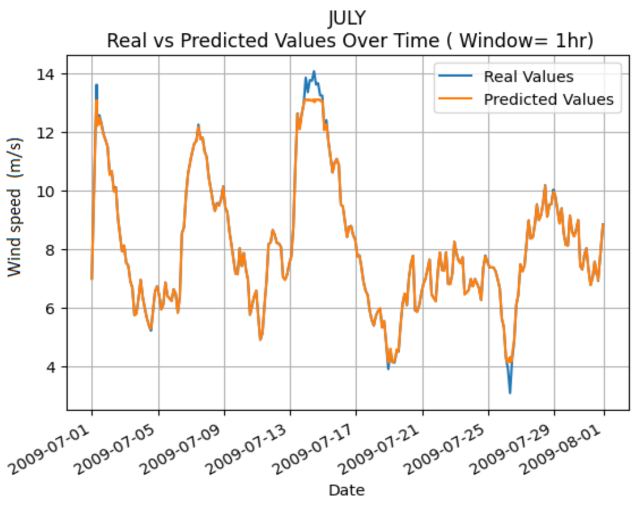
July:

Window size=1

SAE

SIRAE



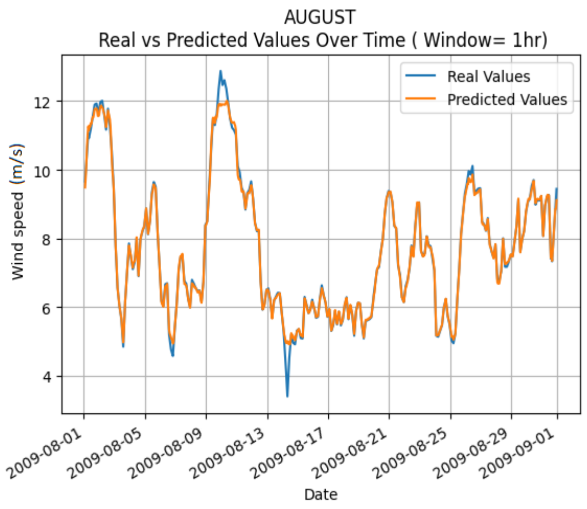


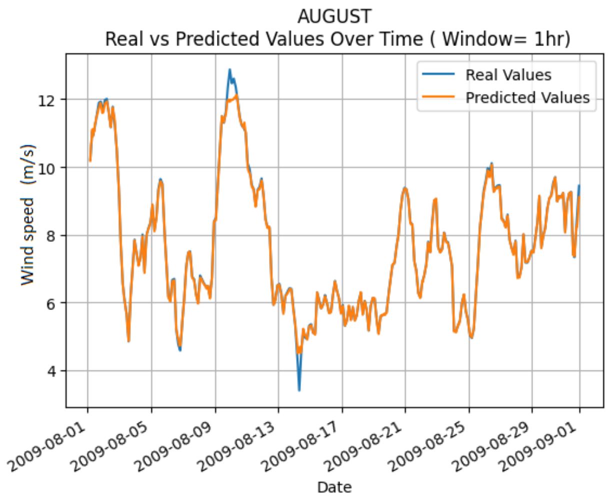
August:

Window size=1

SAE

SIRAE

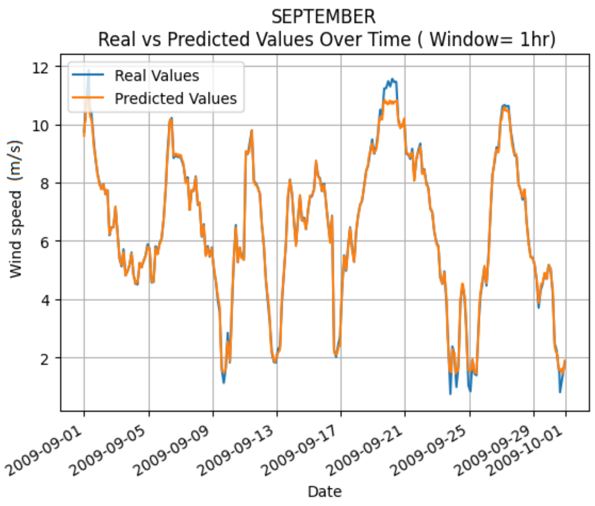


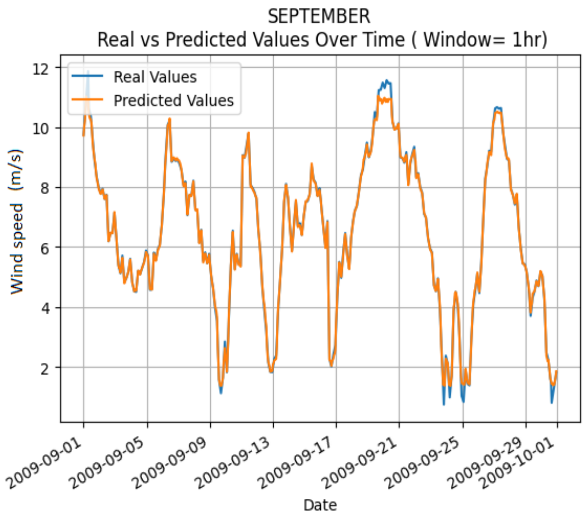


September:

Window size=1

SAE ***SIRAE***

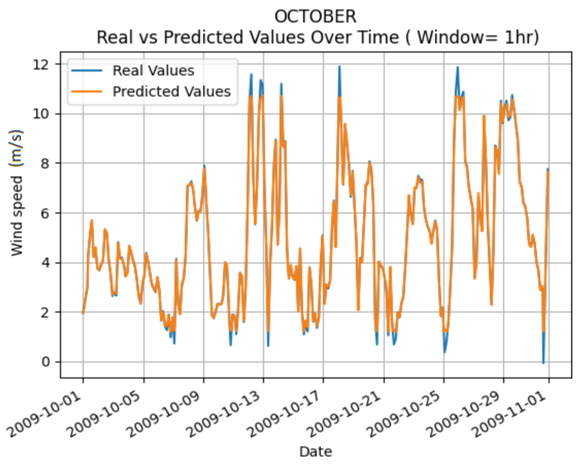


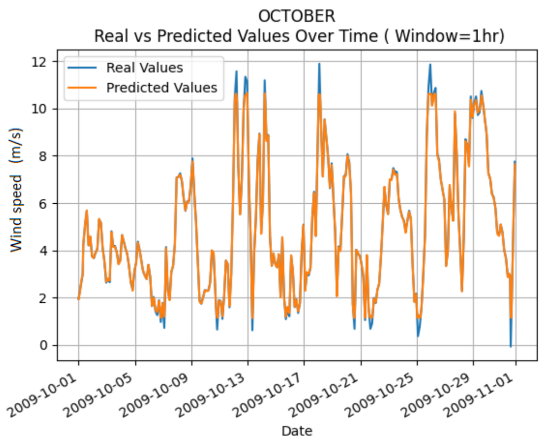


October:

Window size=1

SAE ***SIRAE***

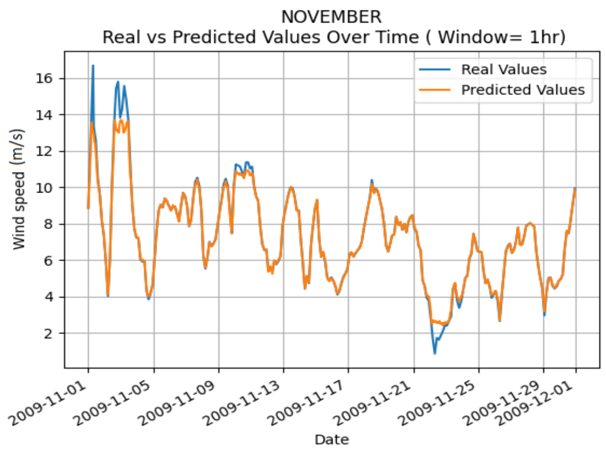


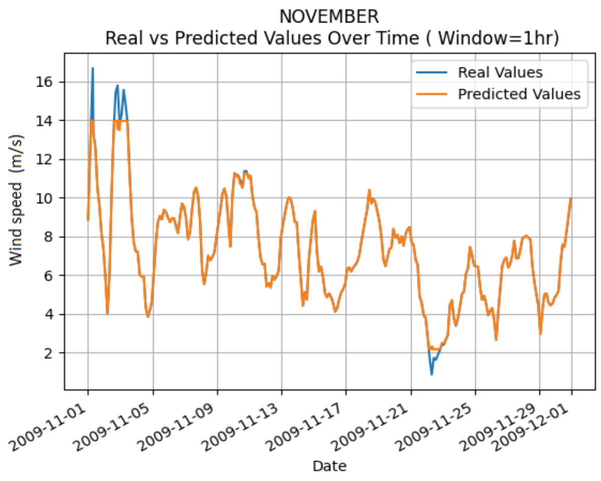


November:

Window size=1

SAE ***SIRAE***

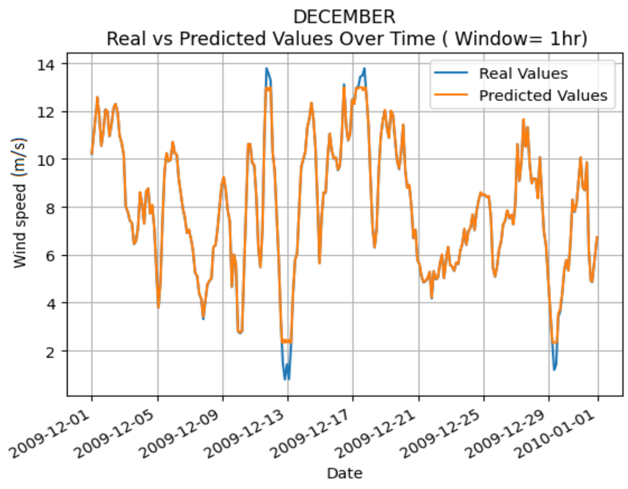


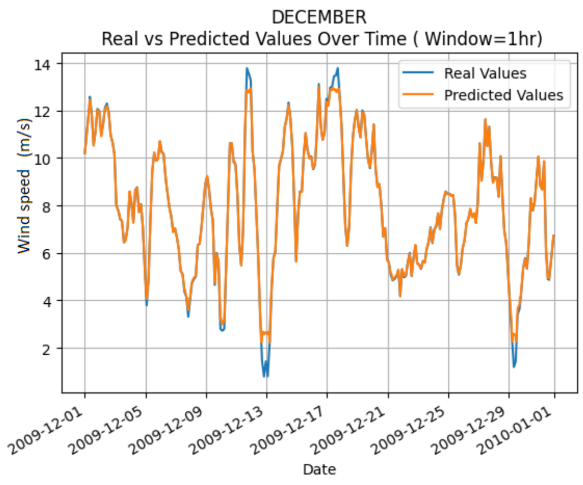


December:

Window size=1

SAE ***SIRAE***

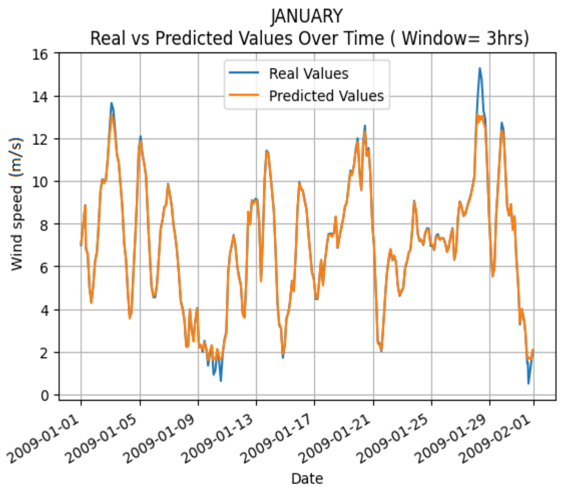


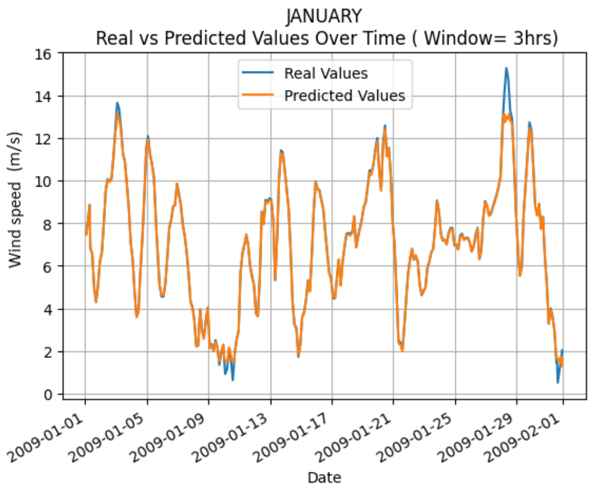


January:

Window size=3

SAE ***SIRAE***

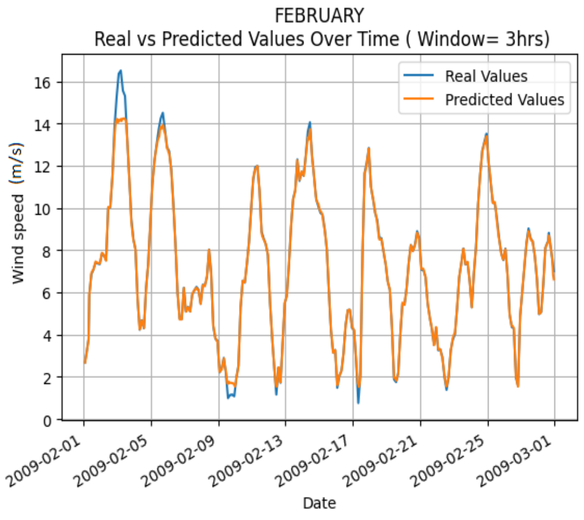


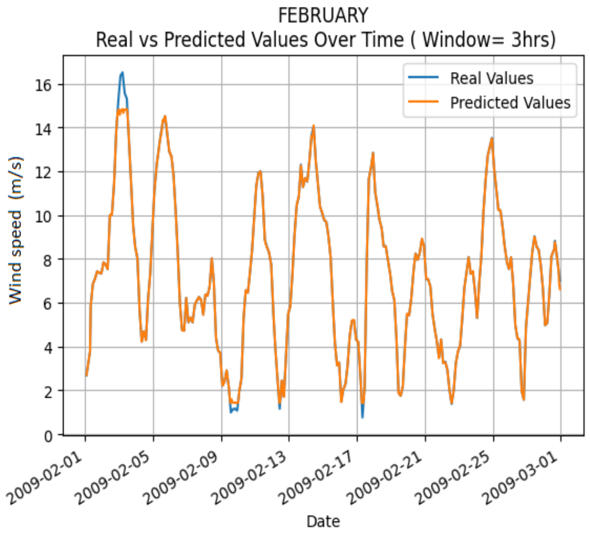


February:

Window size=3

SAE ***SIRAE***

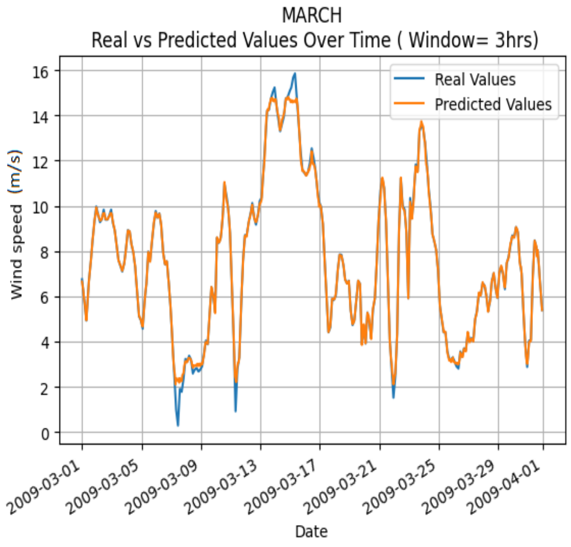


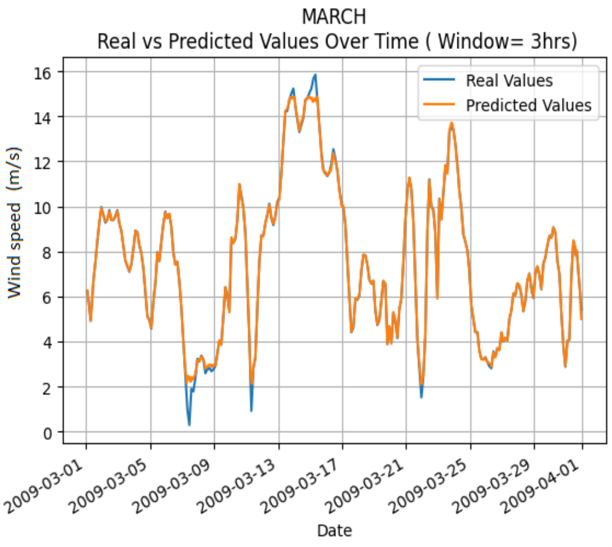


March:

Window size=3

SAE ***SIRAE***

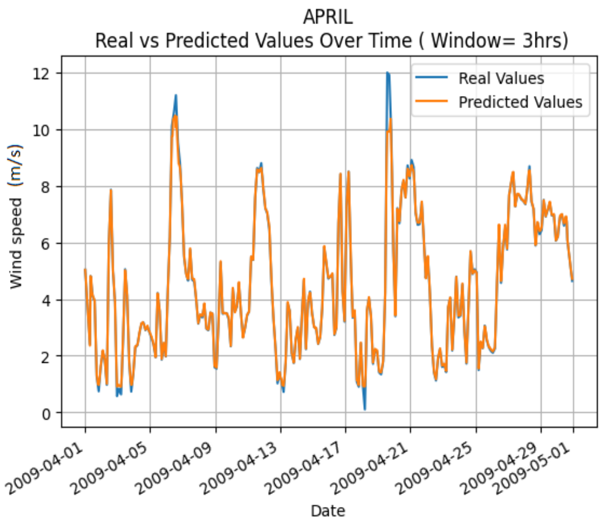


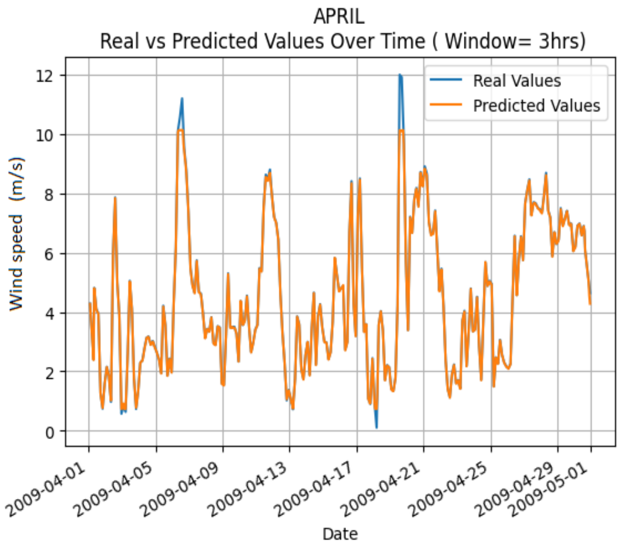


April:

Window size=3

SAE ***SIRAE***

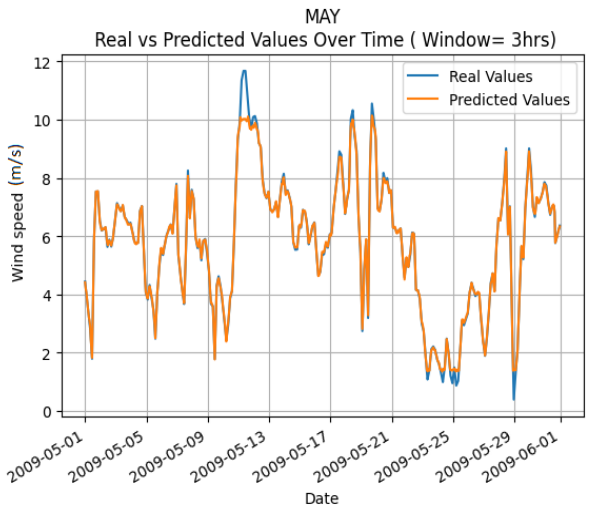


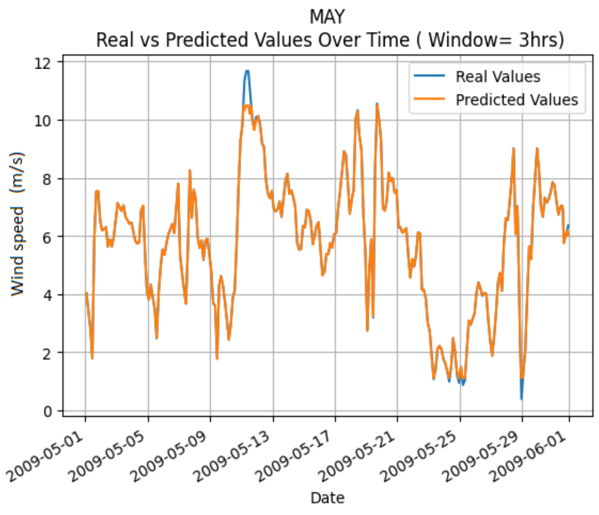


<u>May:</u>

Window size=3

<u>SAE</u> ***<u>SIRAE</u>***

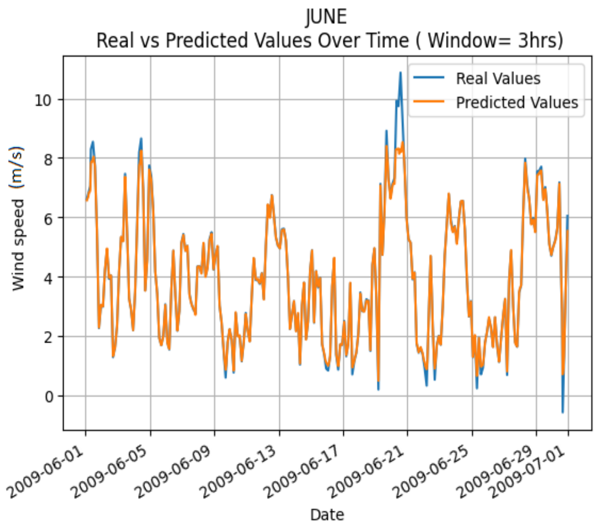


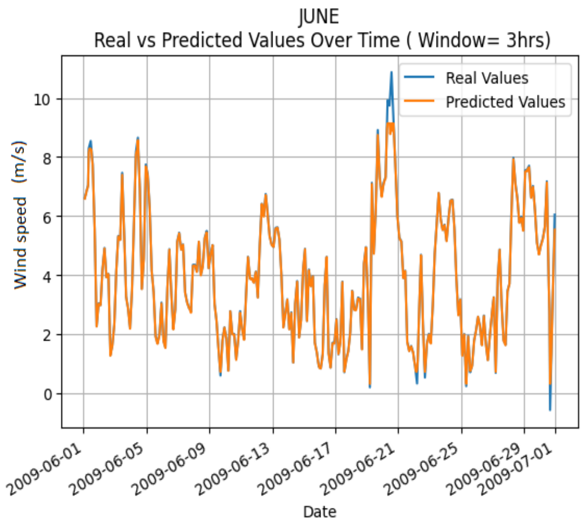


<u>June:</u>

Window size=3

<u>SAE</u> ***<u>SIRAE</u>***

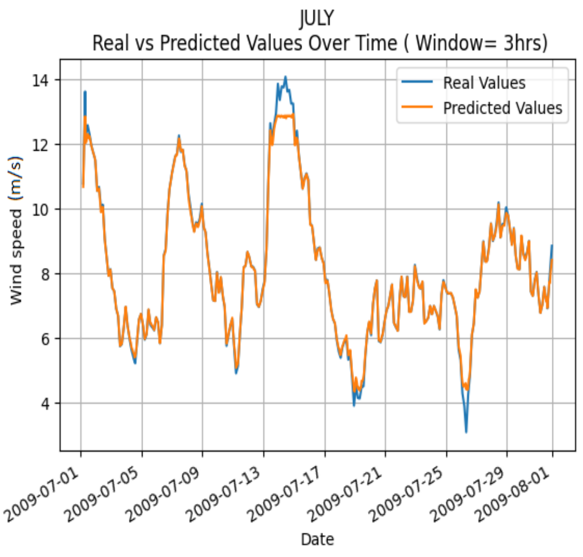




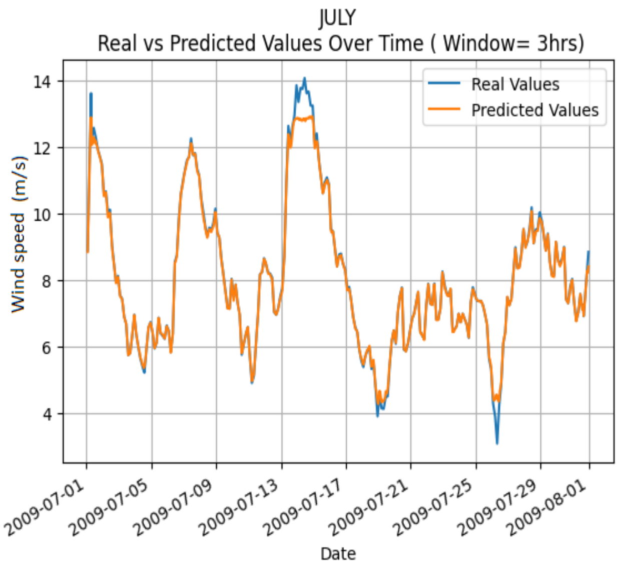
July:

Window size=3

SAE



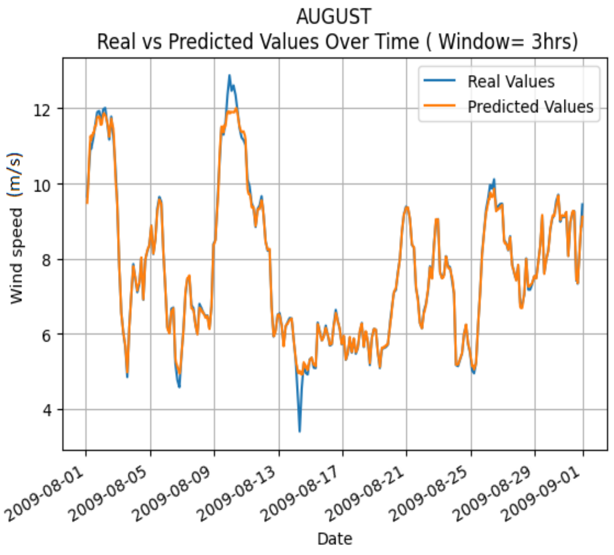
SIRAE



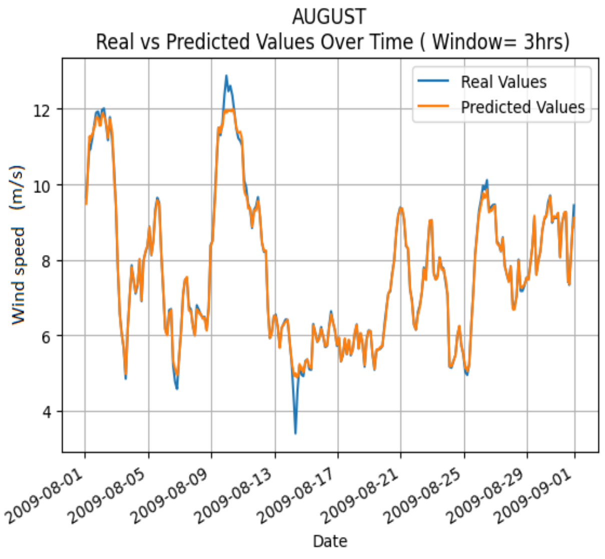
August:

Window size=3

SAE



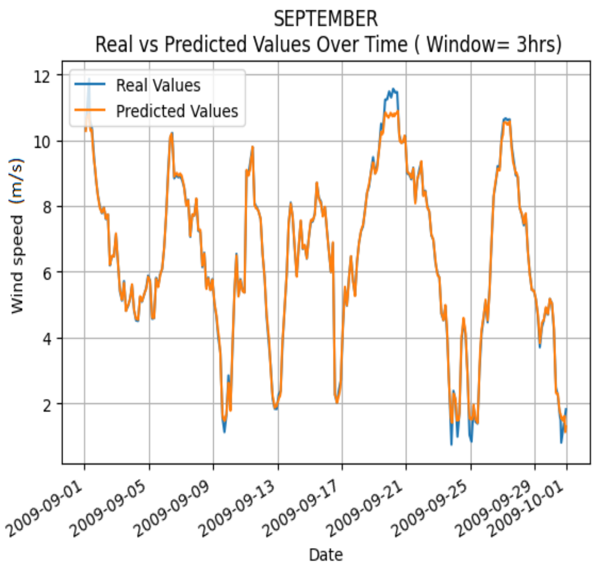
SIRAE



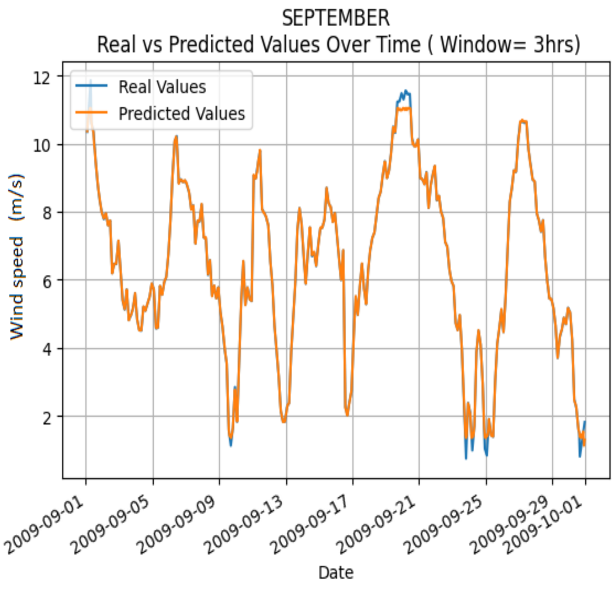
September:

Window size=3

SAE



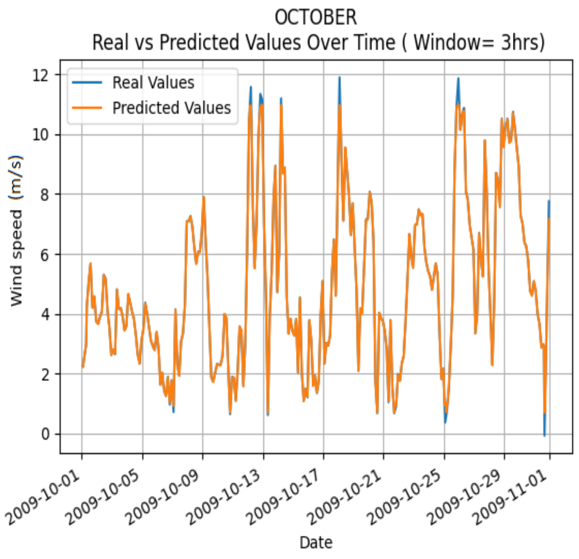
SIRAE



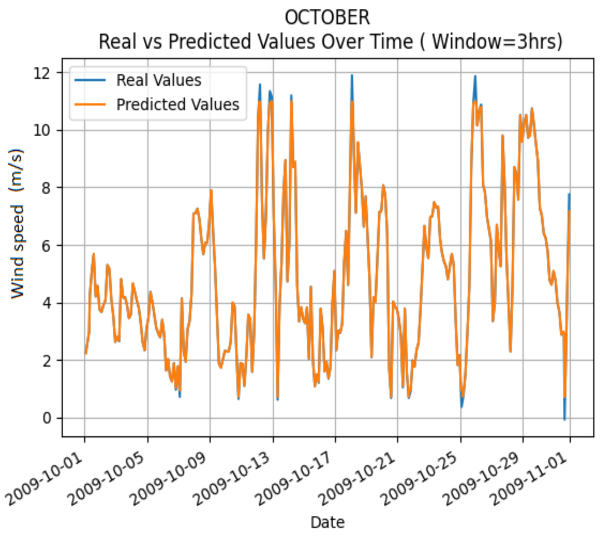
October:

Window size=3

SAE



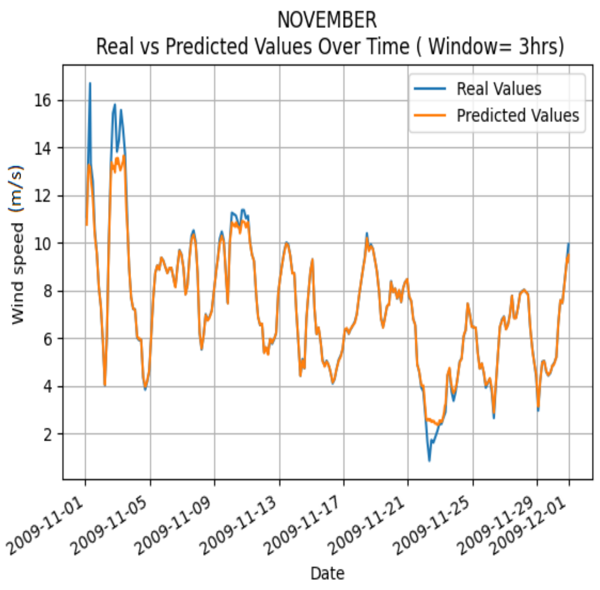
SIRAE

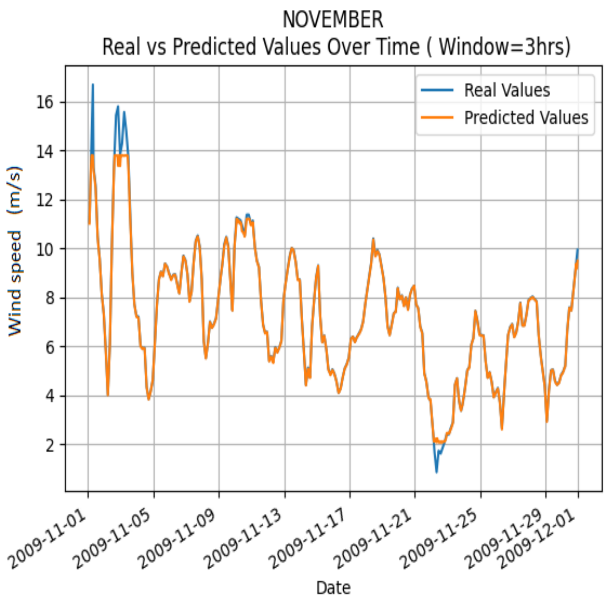


November:

Window size=3

SAE ***SIRAE***

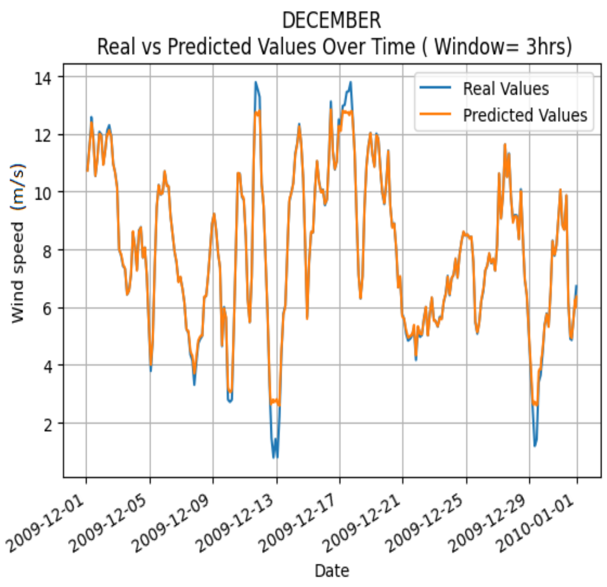


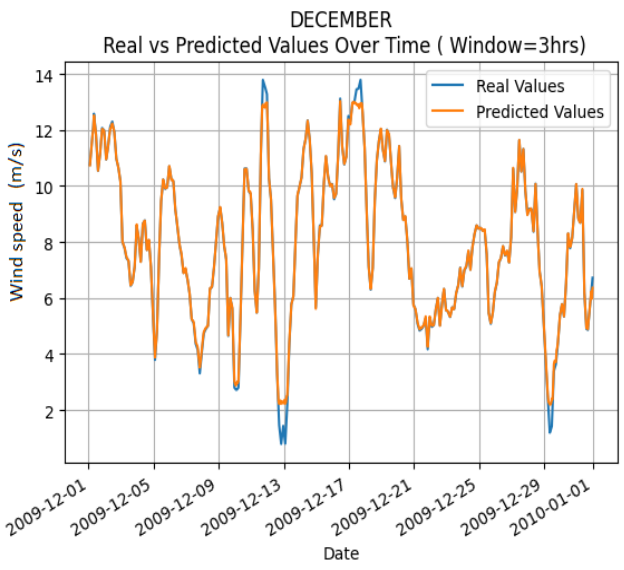


December:

Window size=3

SAE ***SIRAE***

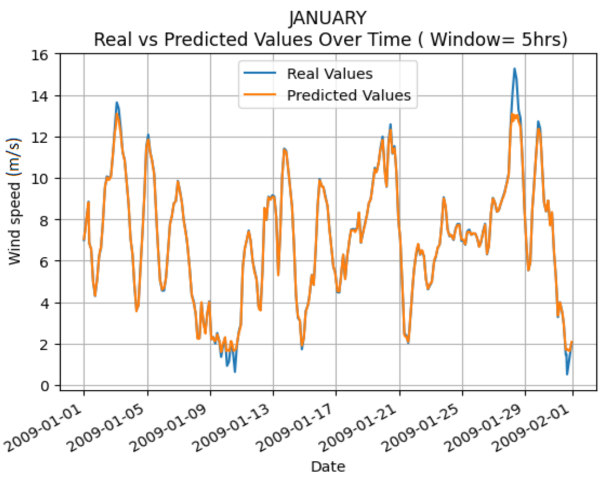


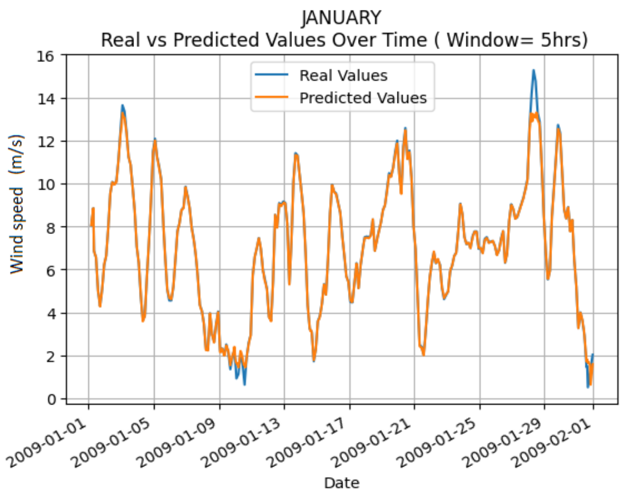


January:

Window size=5

SAE ***SIRAE***

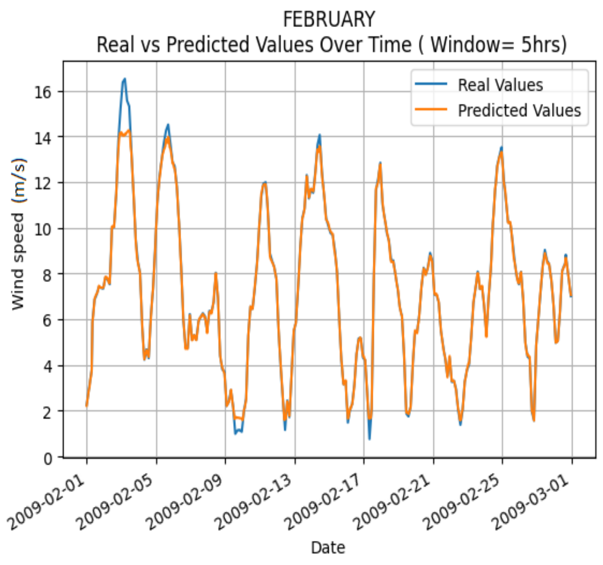


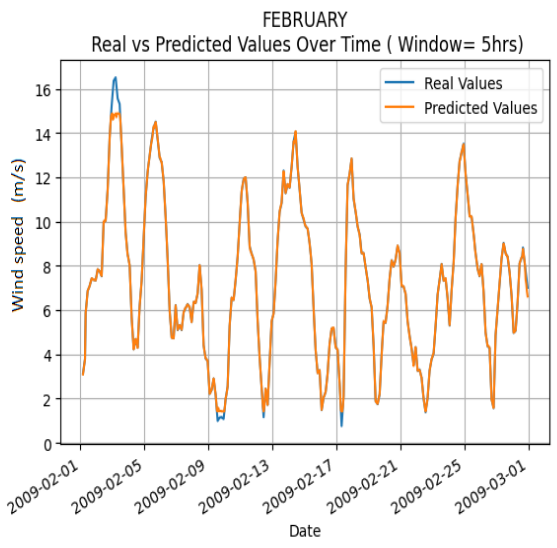


February:

Window size=5

SAE ***SIRAE***

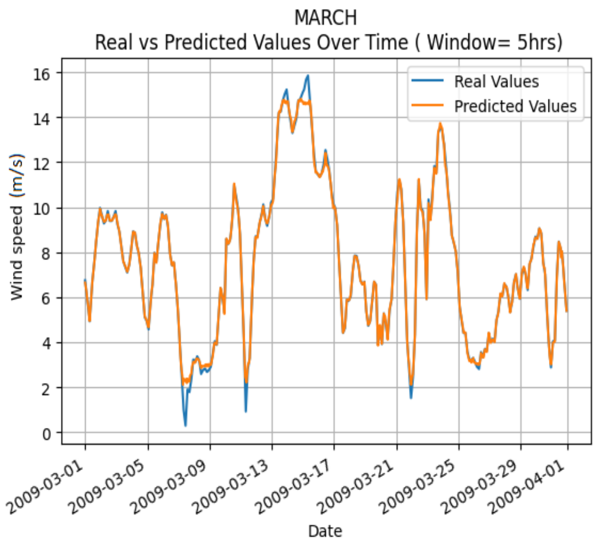




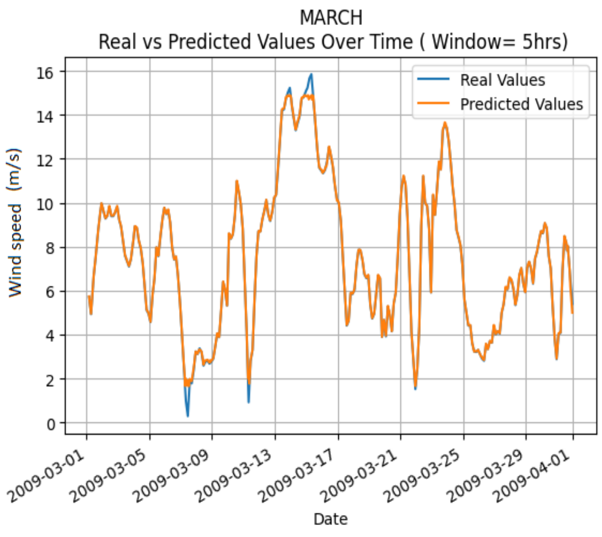
March:

Window size=5

SAE



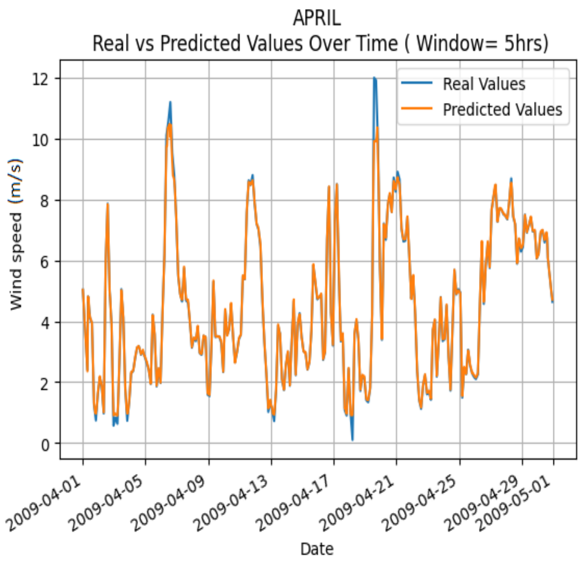
SIRAE



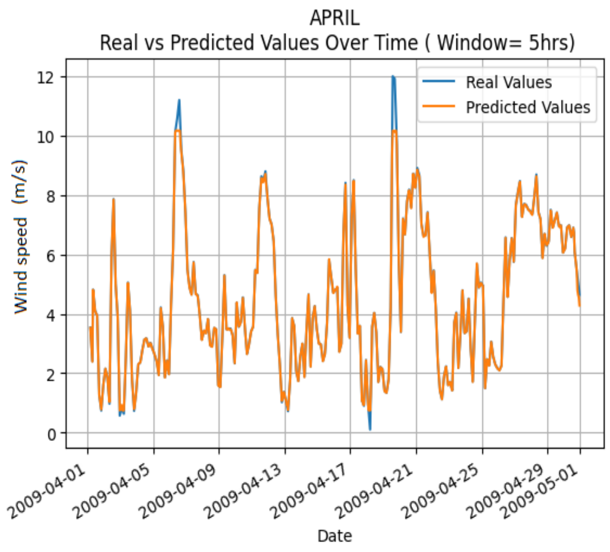
April:

Window size=5

SAE



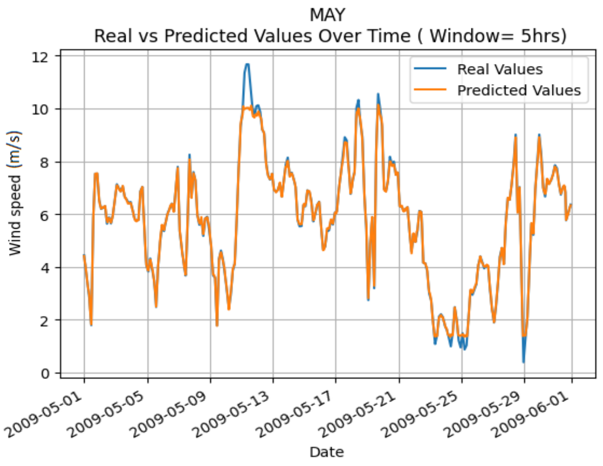
SIRAE

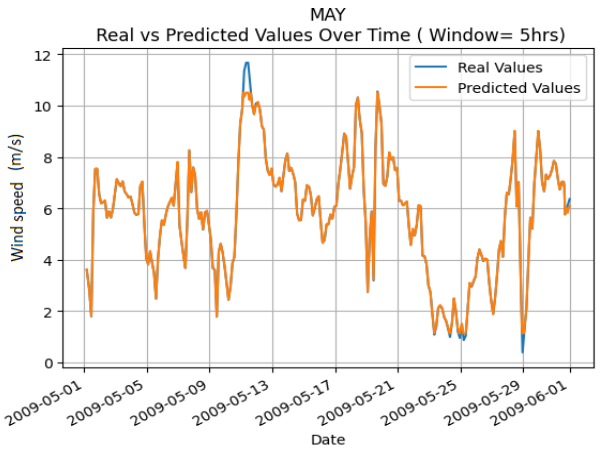


May:

Window size=5

SAE ***SIRAE***

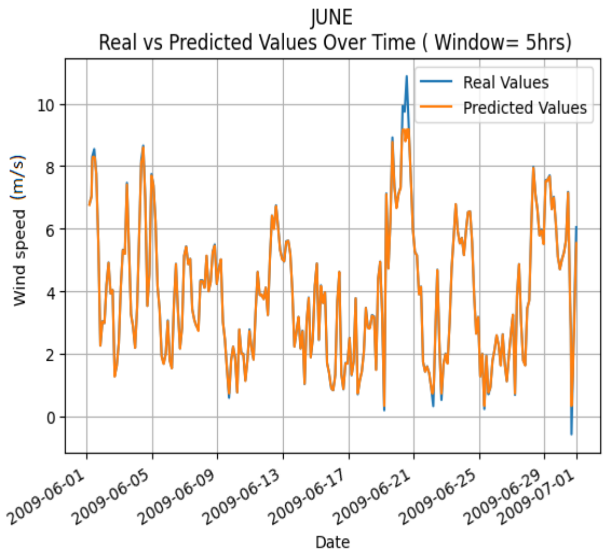


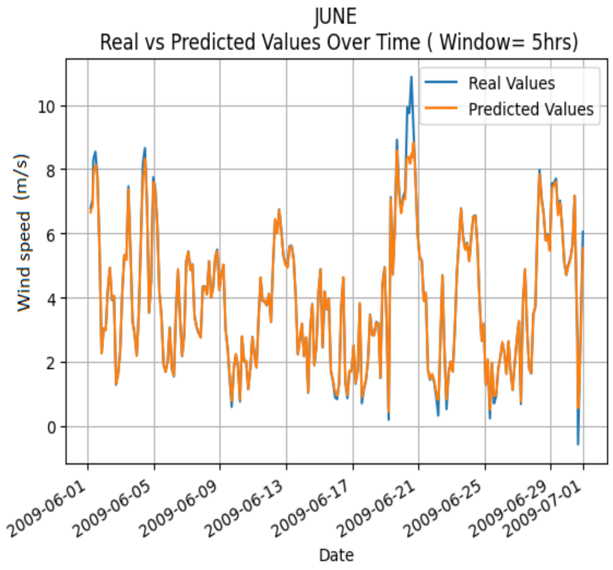


June:

Window size=5

SAE ***SIRAE***

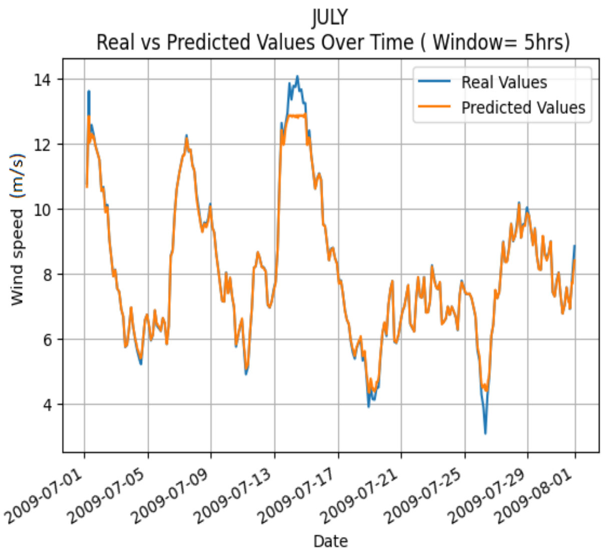




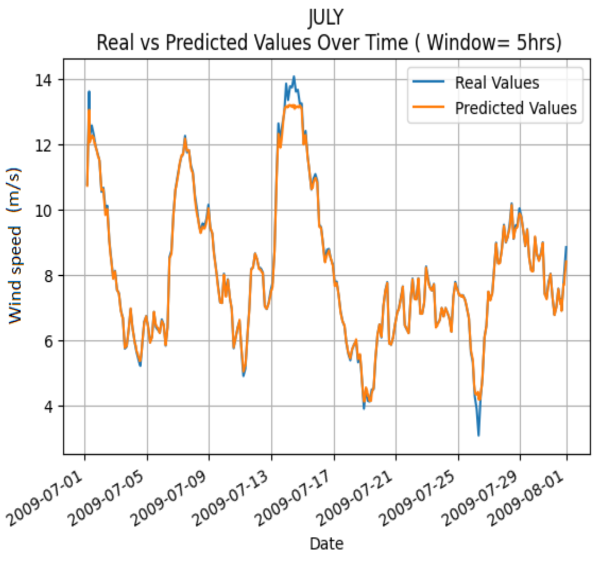
July:

Window size=5

SAE



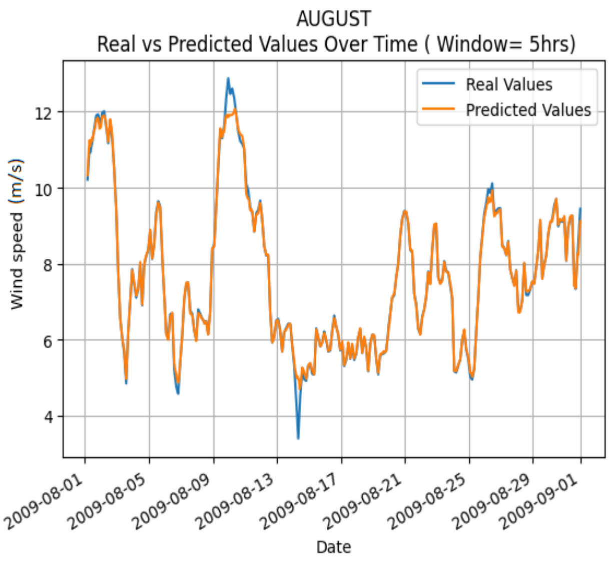
SIRAE



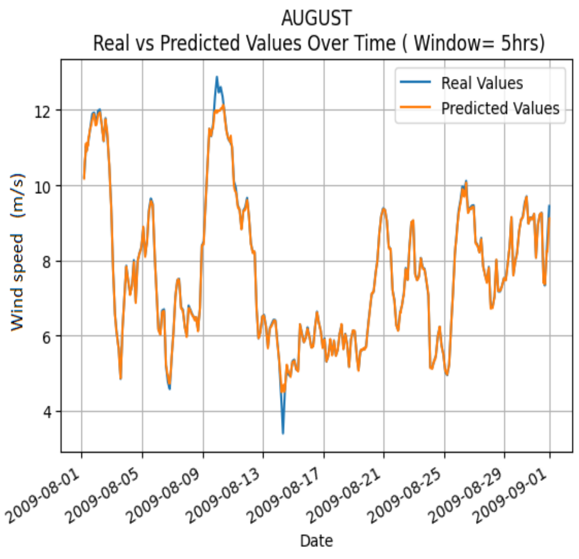
August:

Window size=5

SAE



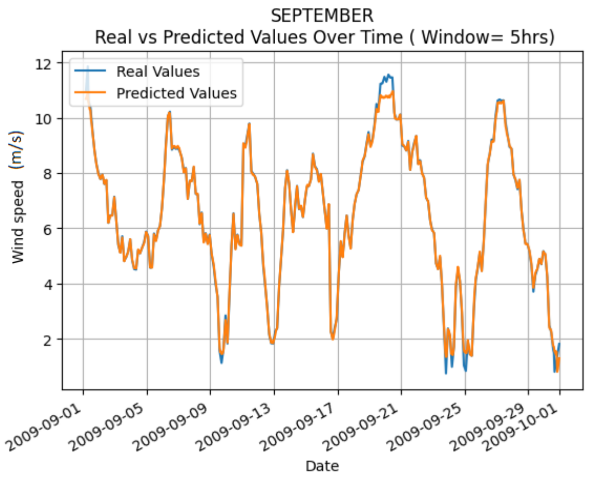
SIRAE



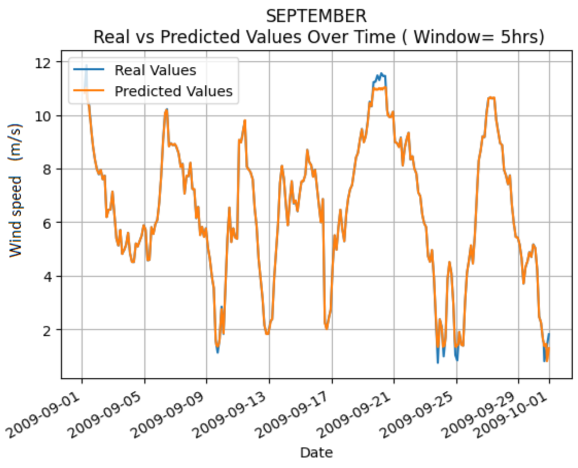
September:

Window size=5

SAE



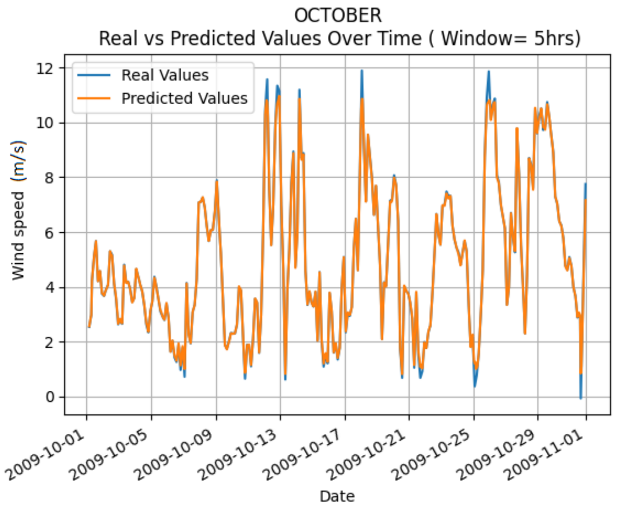
SIRAE



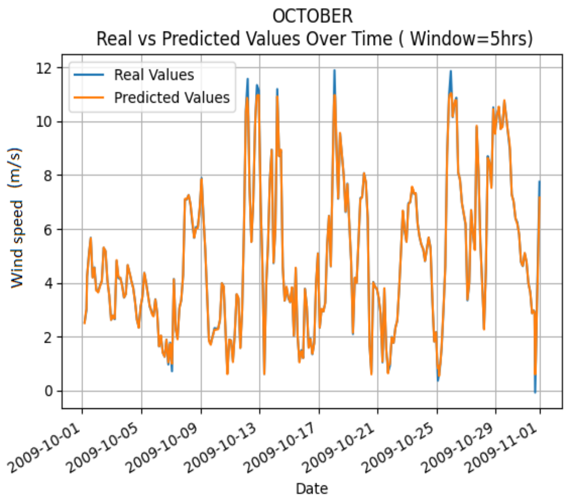
October:

Window size=5

SAE



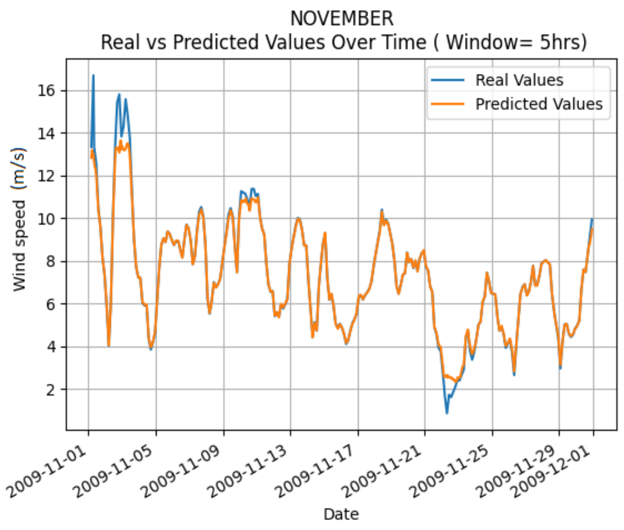
SIRAE

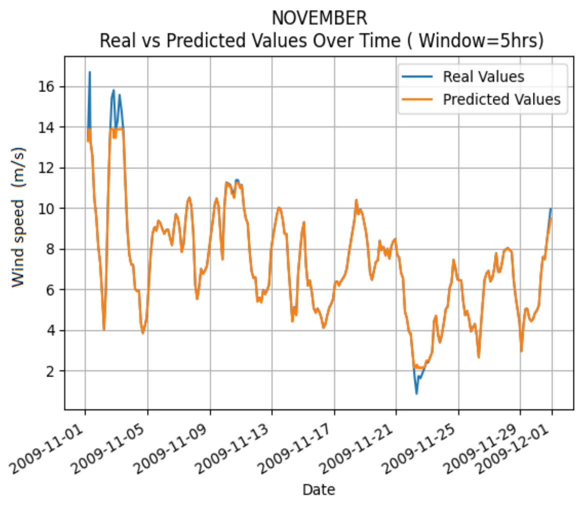


November:

Window size=5

SAE ***SIRAE***

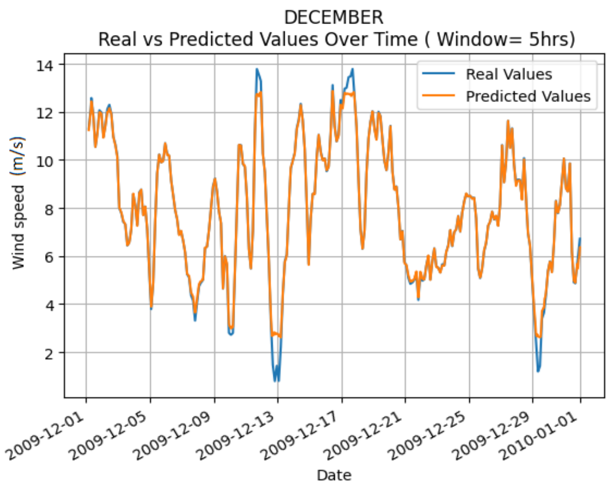


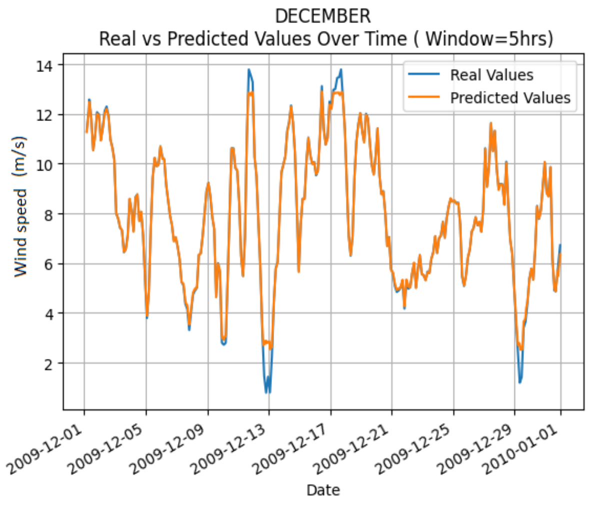


December:

Window size=5

SAE ***SIRAE***



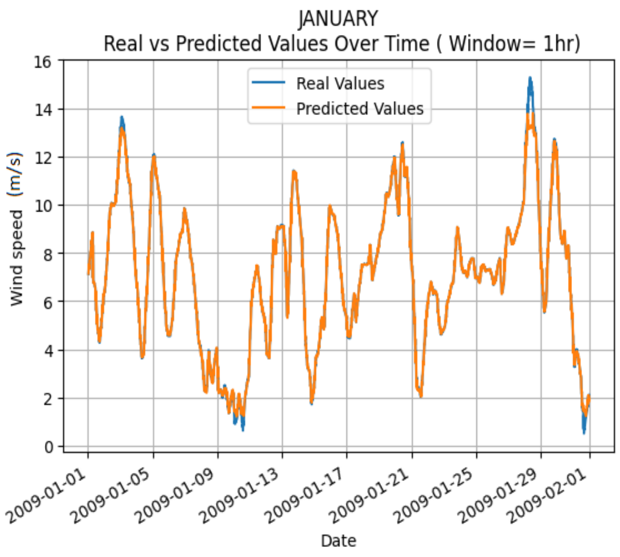


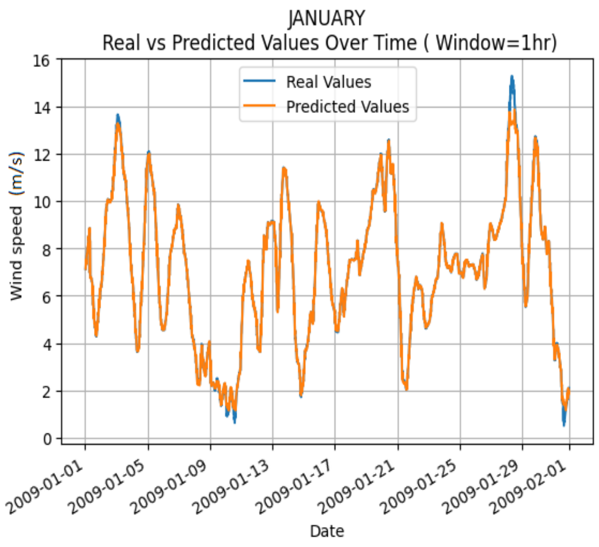
Results Short-Term Forecastings

January:

Window size=1

SAE ***SIRAE***

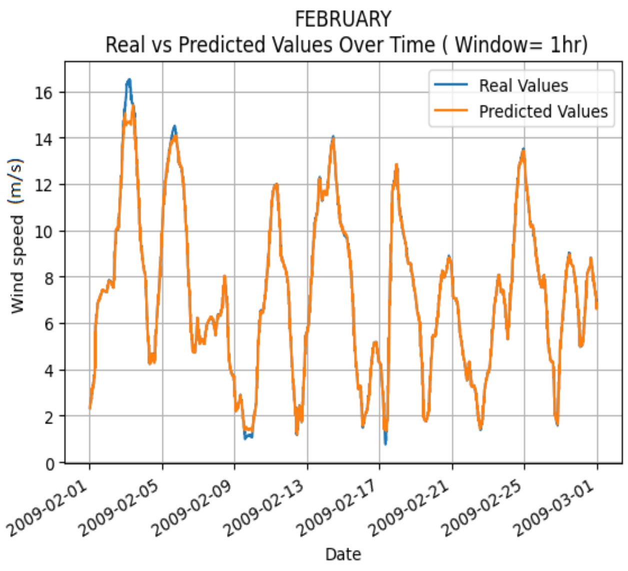


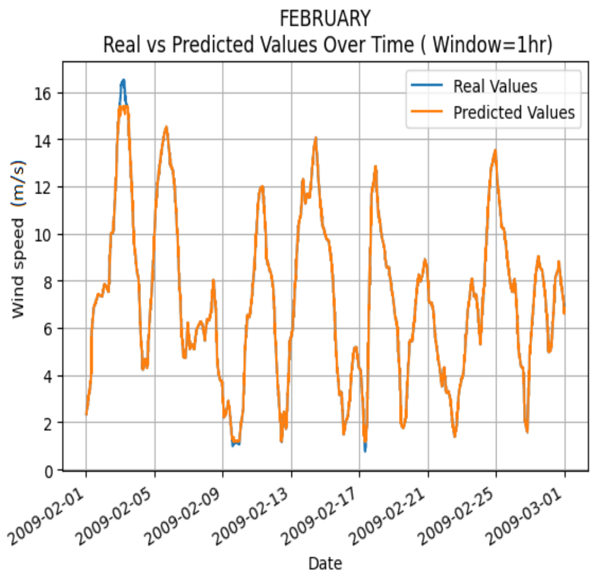


February:

Window size=1

SAE ***SIRAE***

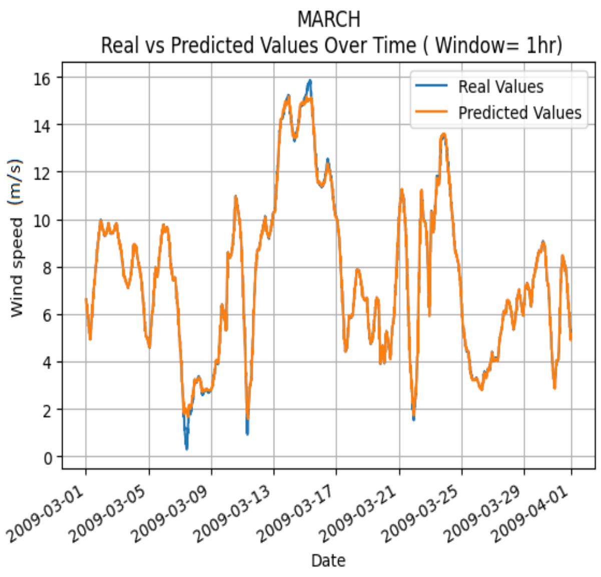


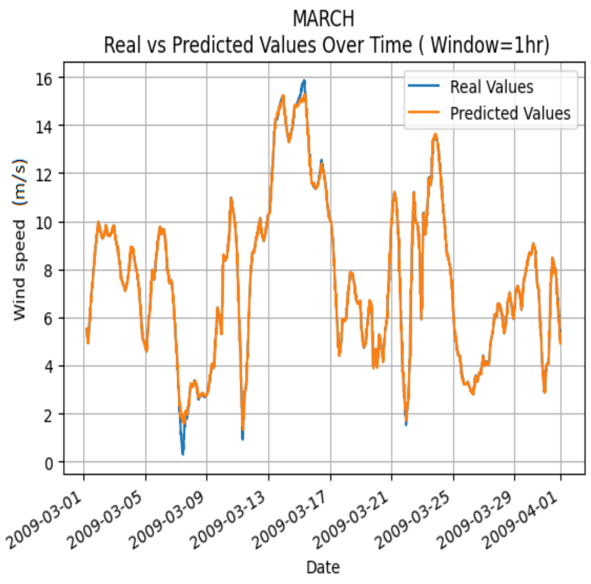


March:

Window size=1

SAE ***SIRAE***

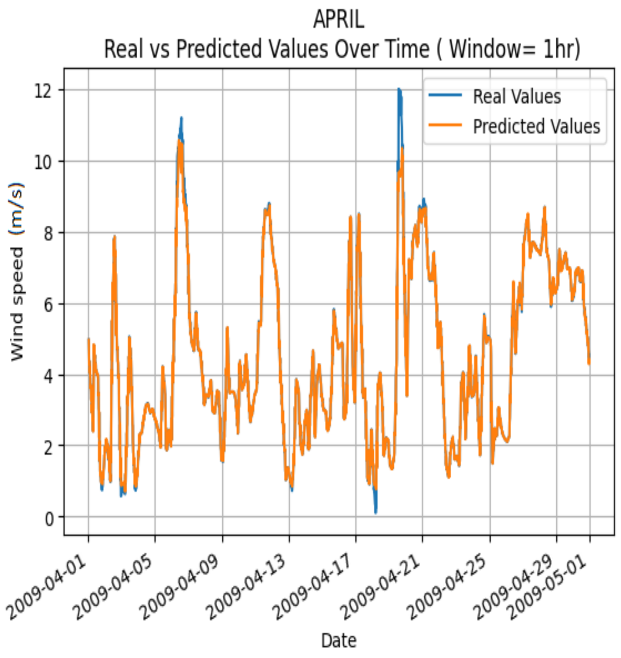


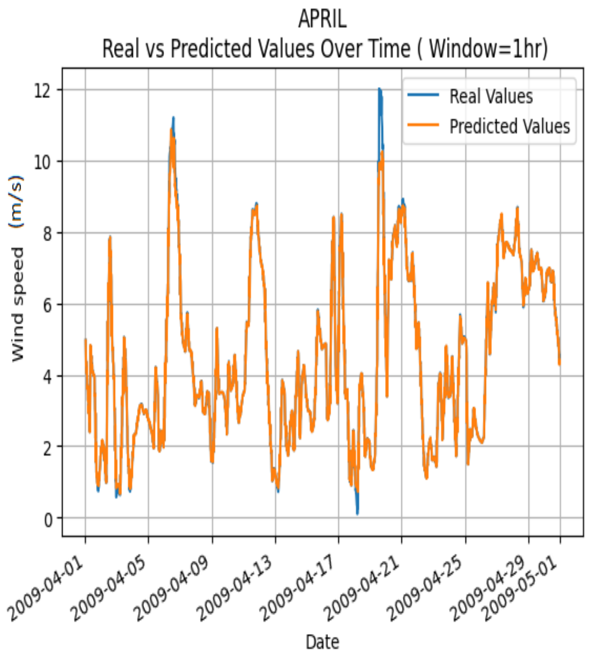


April:

Window size=1

SAE ***SIRAE***

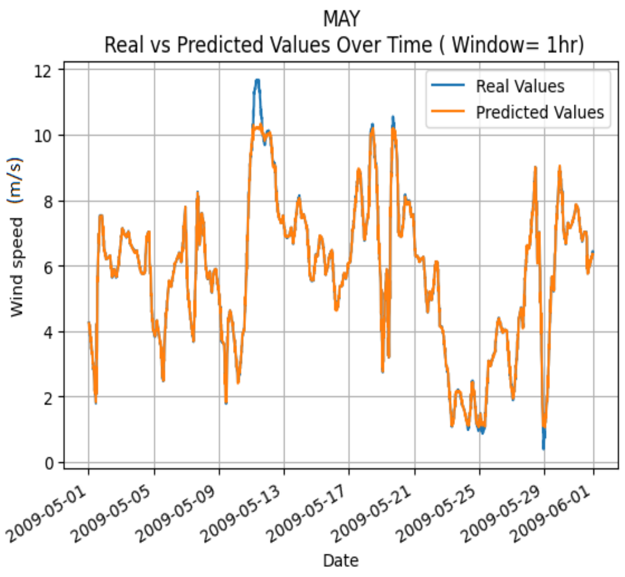


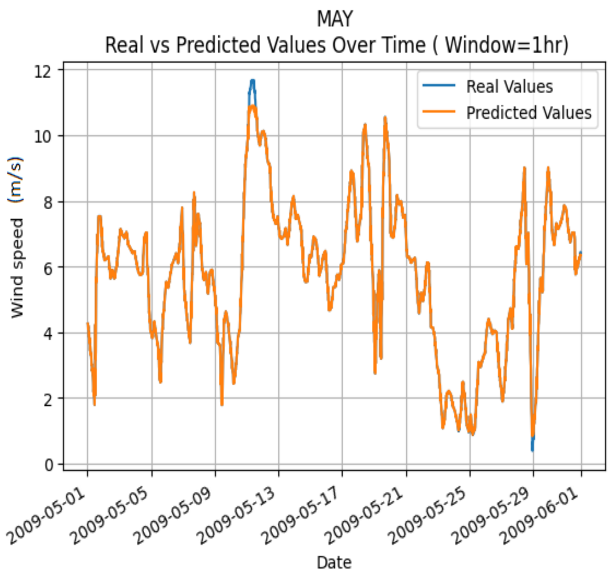


May:

Window size=1

SAE ***SIRAE***

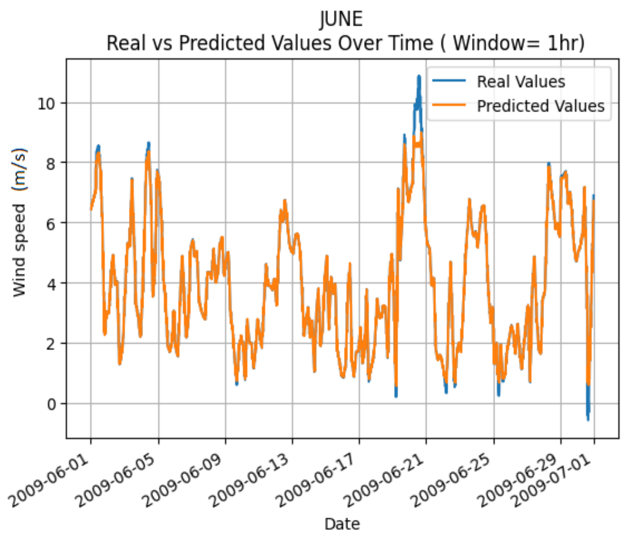


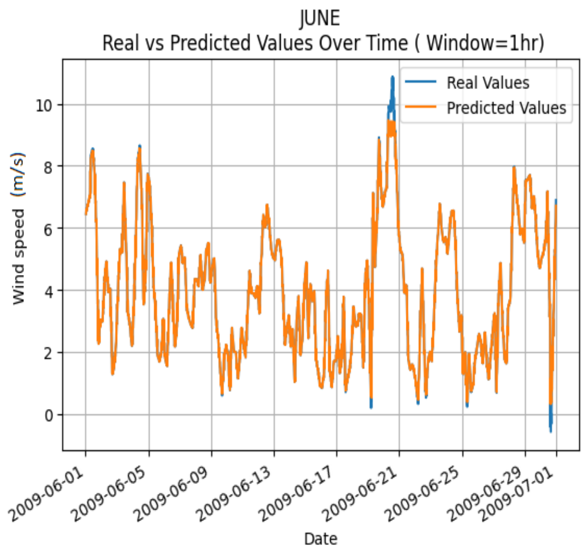


June:

Window size=1

SAE ***SIRAE***

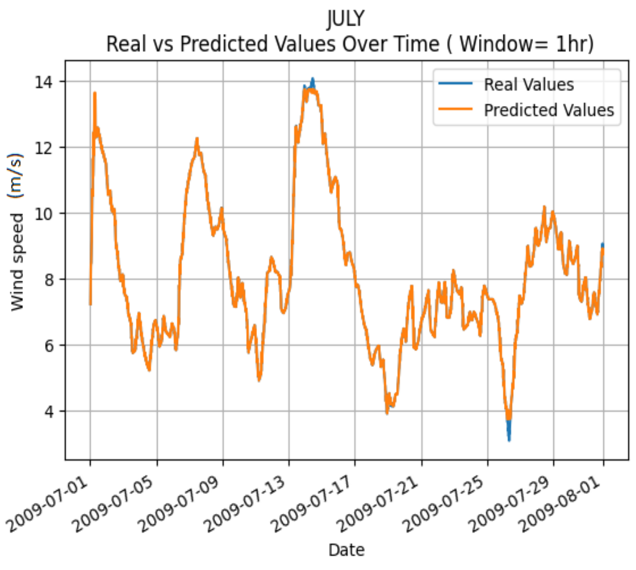




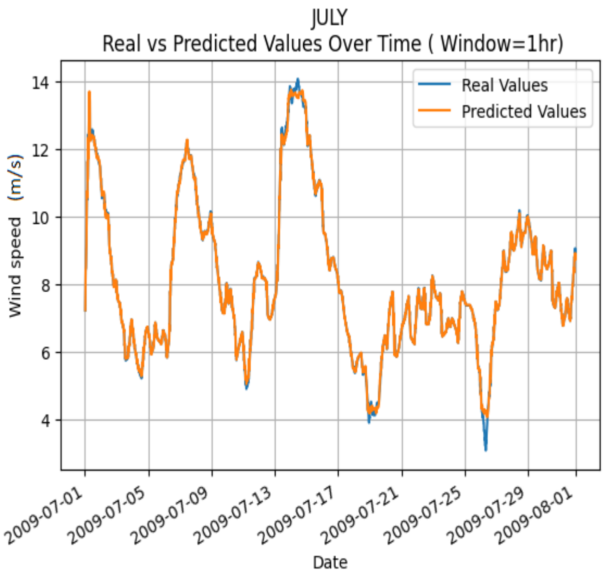
July:

Window size=1

SAE



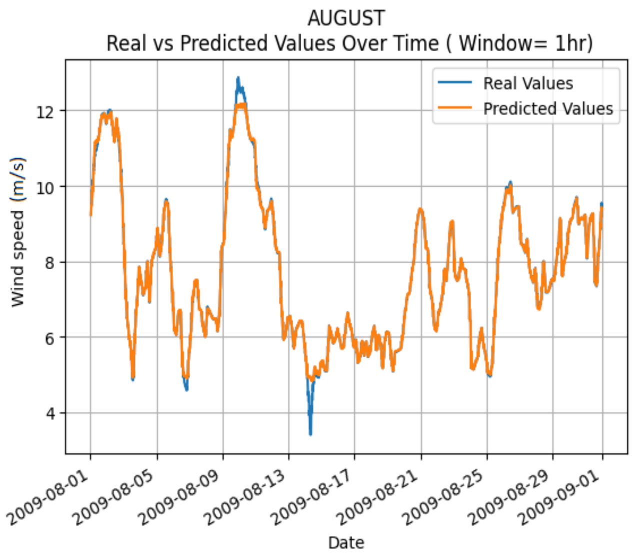
SIRAE



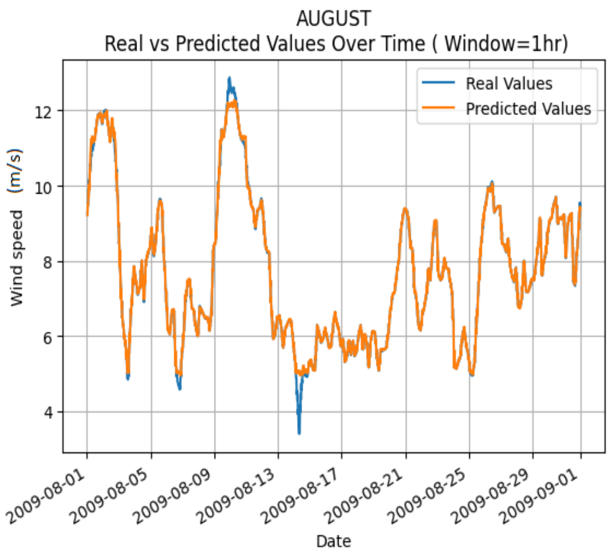
August:

Window size=1

SAE



SIRAE

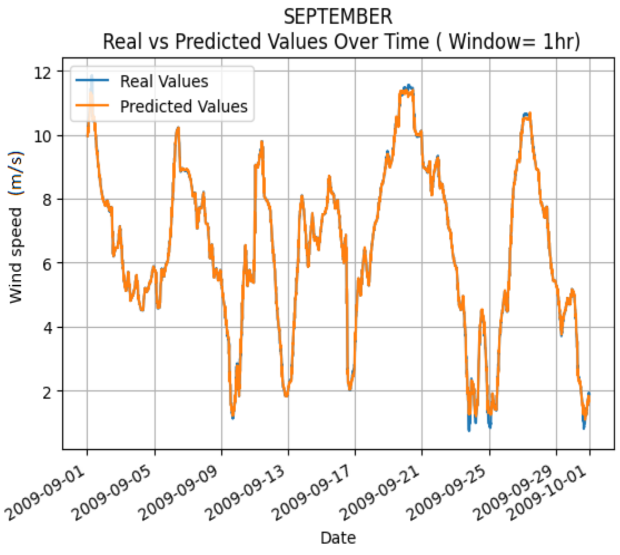


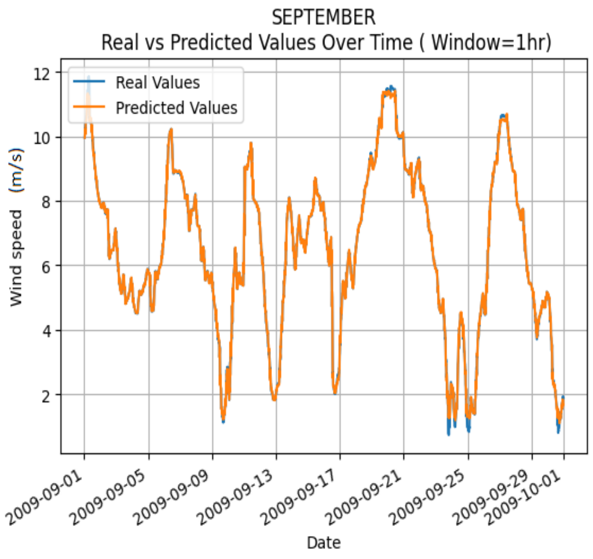
September:

Window size=1

SAE

SIRAE



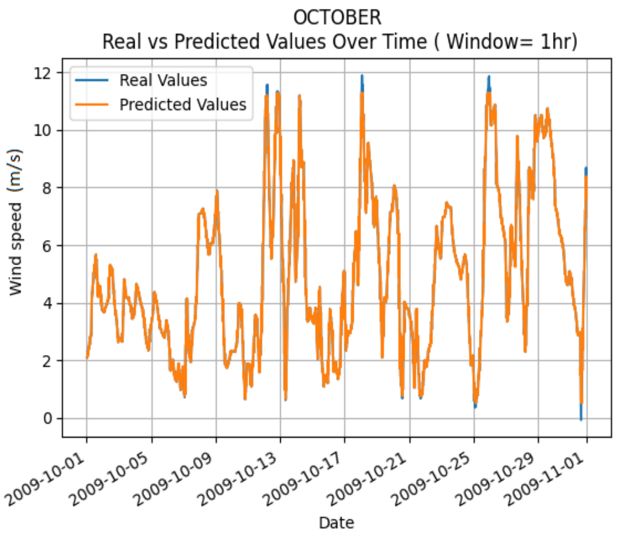


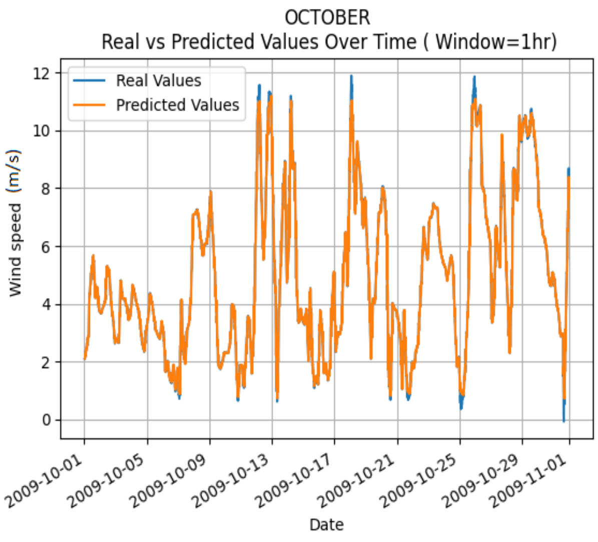
October:

Window size=1

SAE

SIRAE

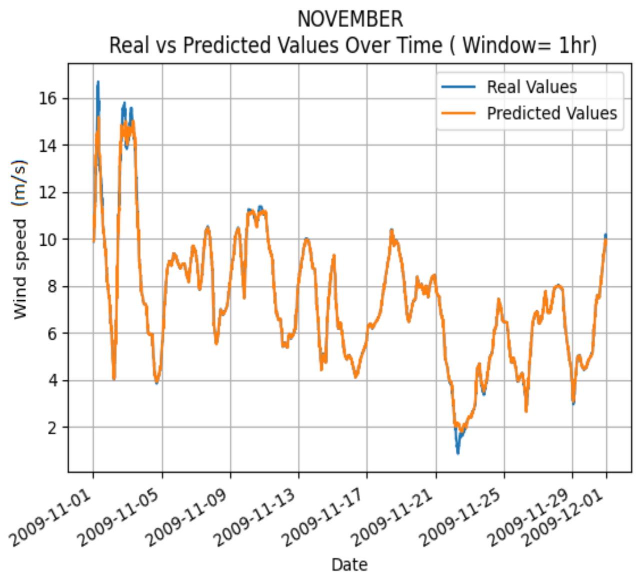


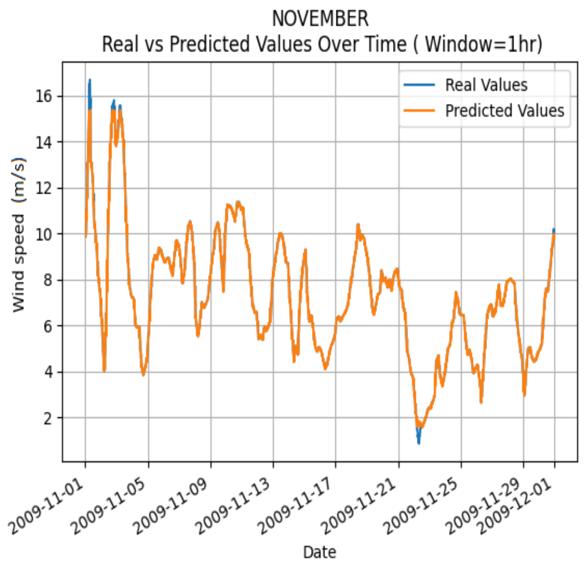


<u>November:</u>

Window size=1

<u>SAE</u> ***<u>SIRAE</u>***

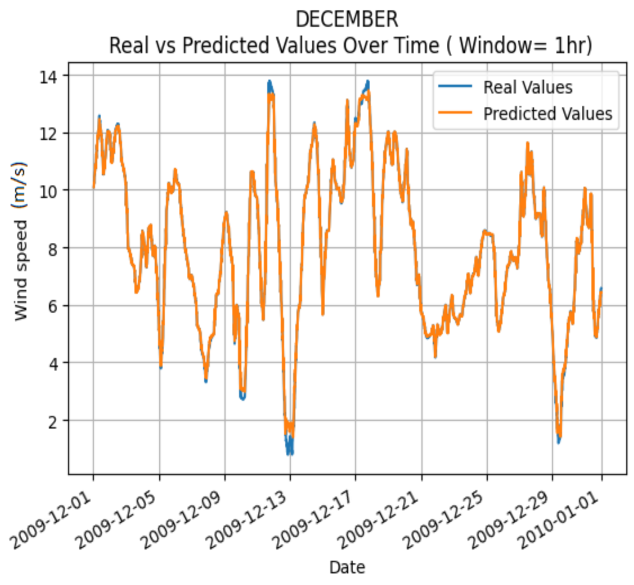


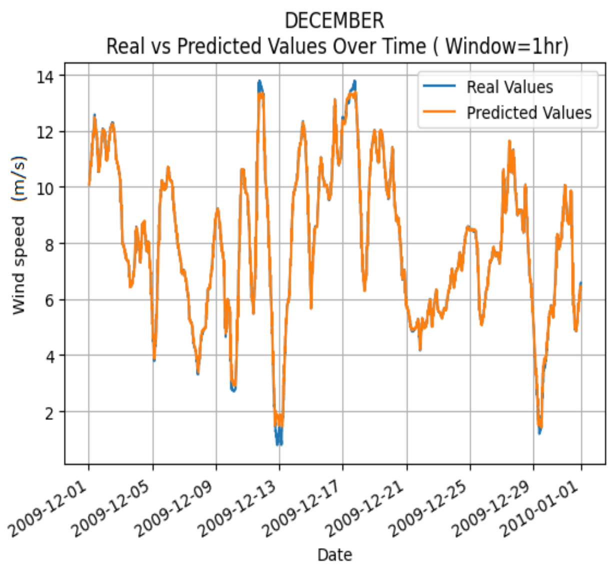


<u>December:</u>

Window size=1

<u>SAE</u> ***<u>SIRAE</u>***

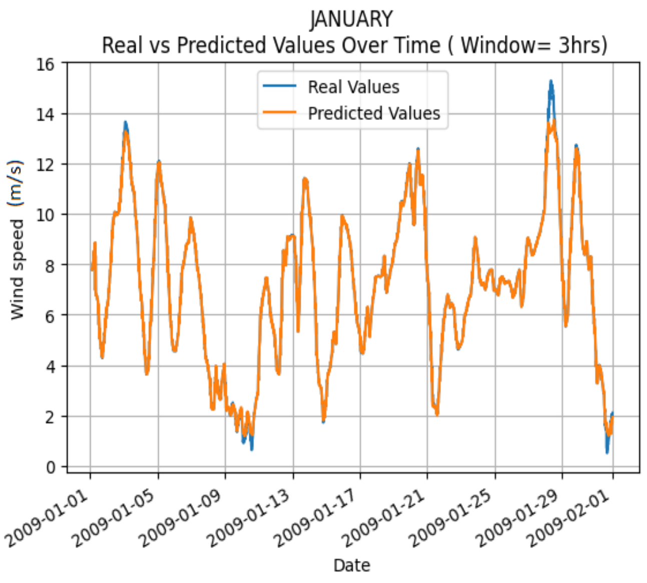


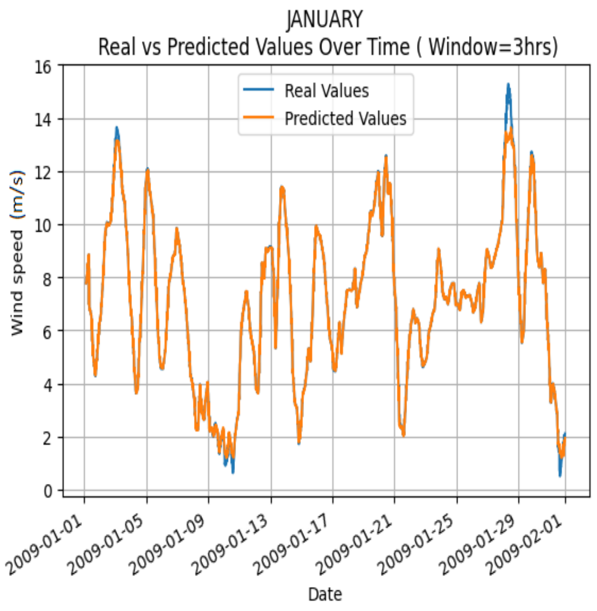


January:

Window size=3

SAE ***SIRAE***

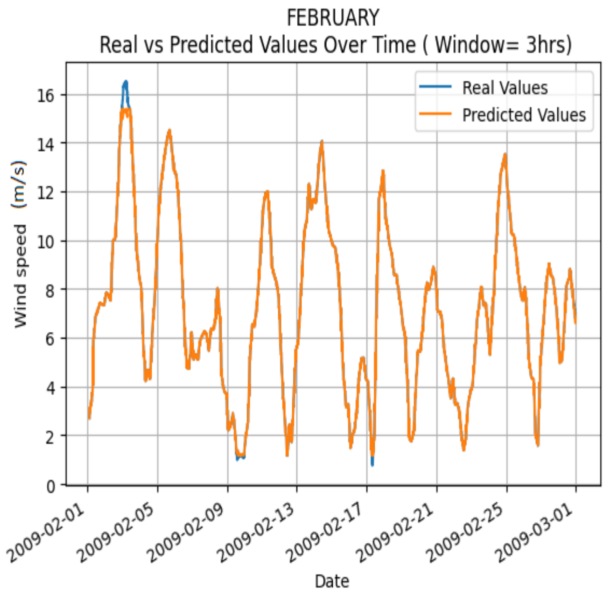


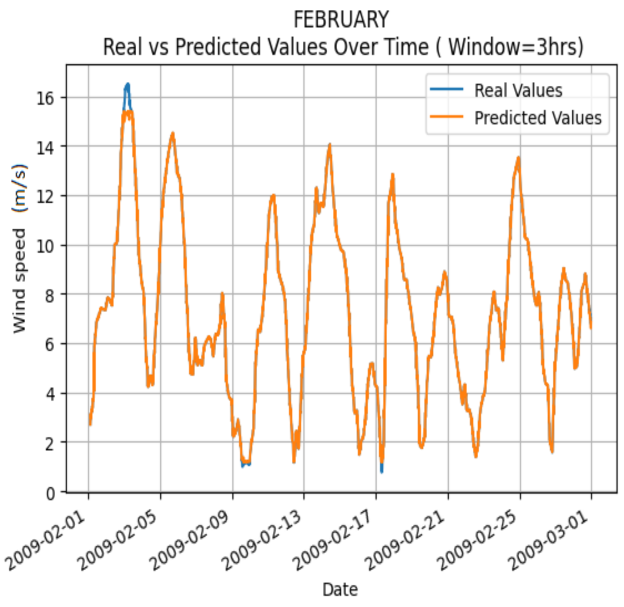


February:

Window size=3

SAE ***SIRAE***

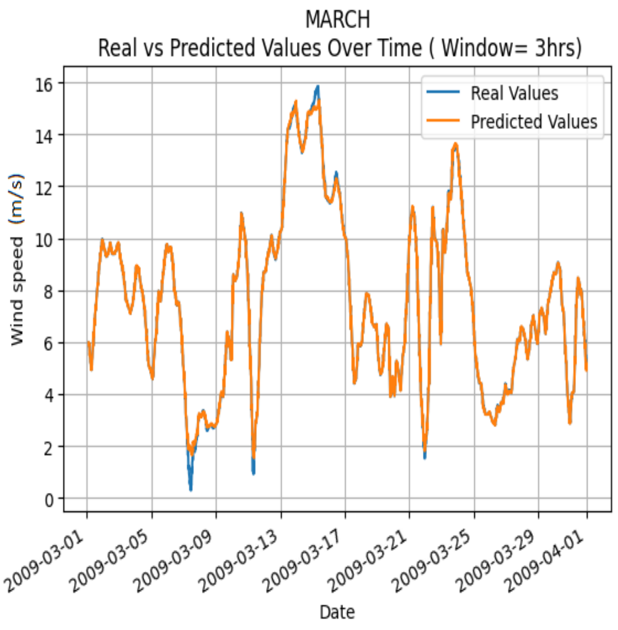


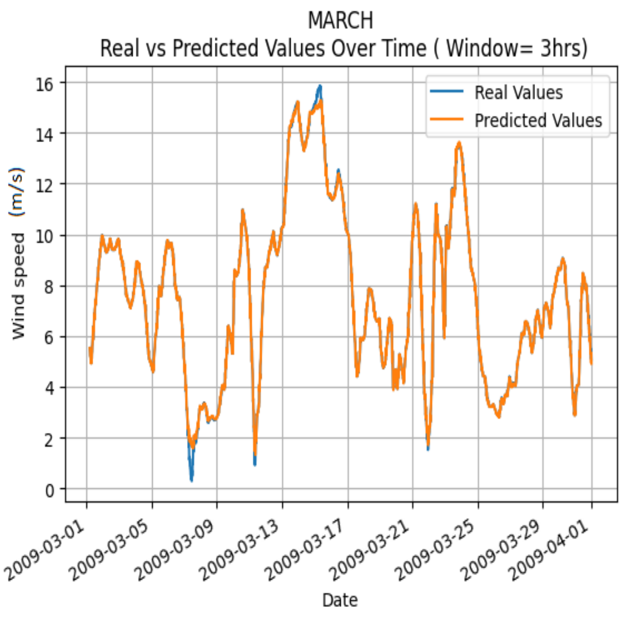


March:

Window size=3

SAE ***SIRAE***

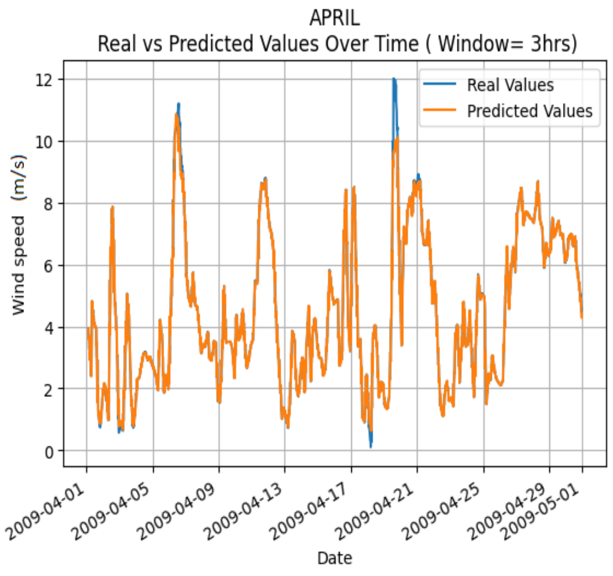


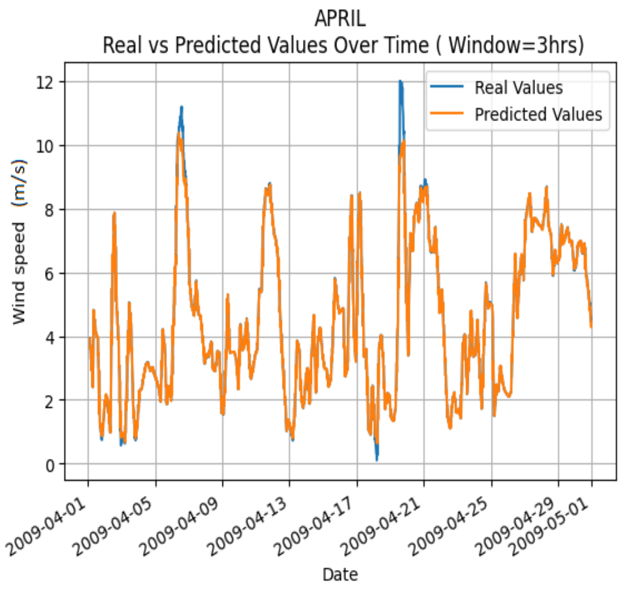


April:

Window size=3

SAE ***SIRAE***

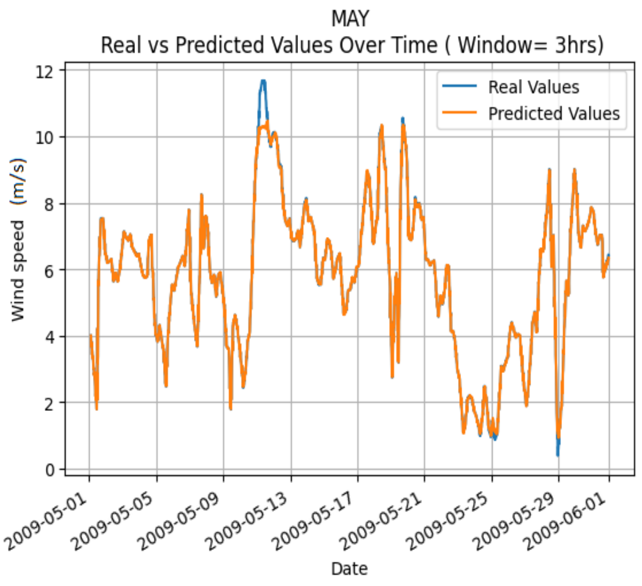


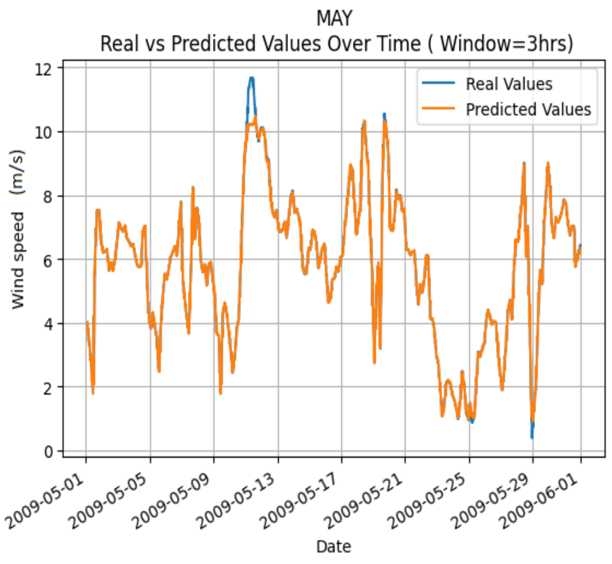


May:

SAE ***SIRAE***

Window size=3

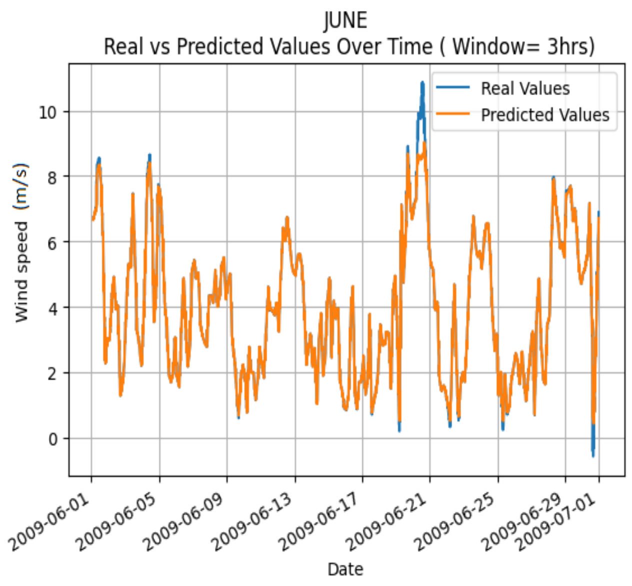


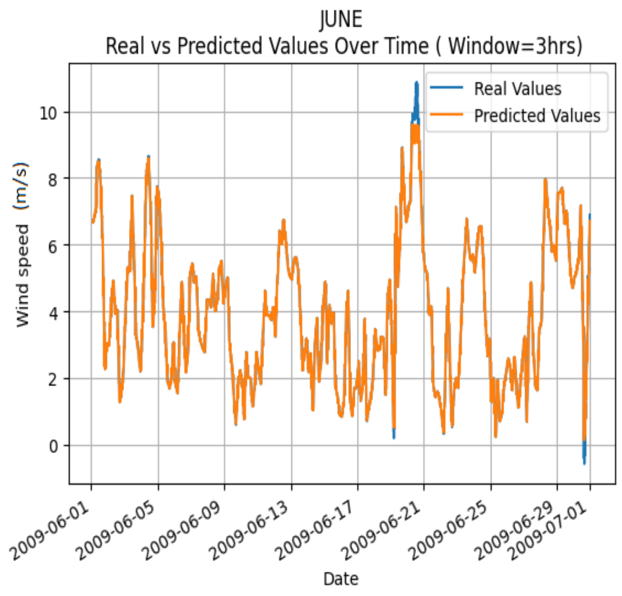


June:

Window size=3

SAE ***SIRAE***

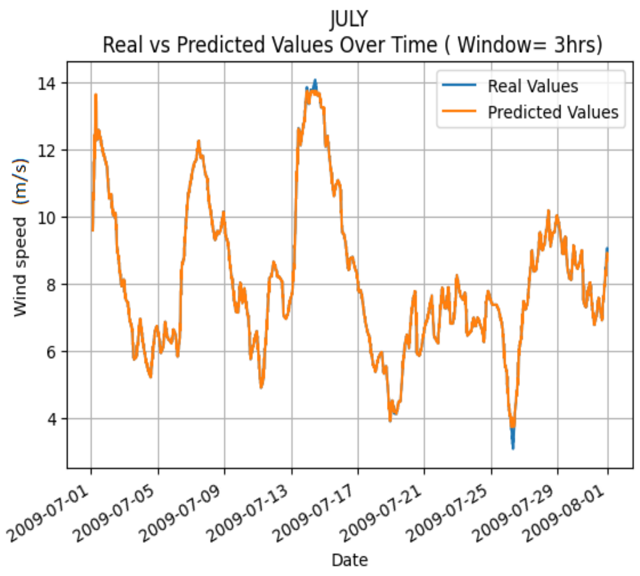




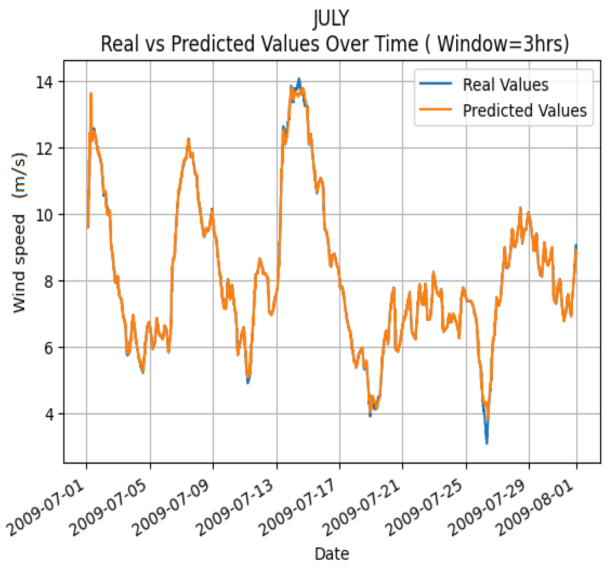
<u>July:</u>

Window size=3

<u>SAE</u>



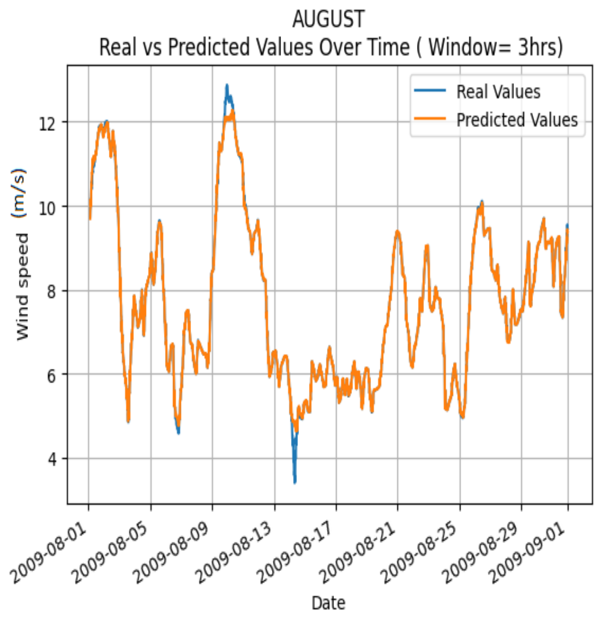
<u>SIRAE</u>



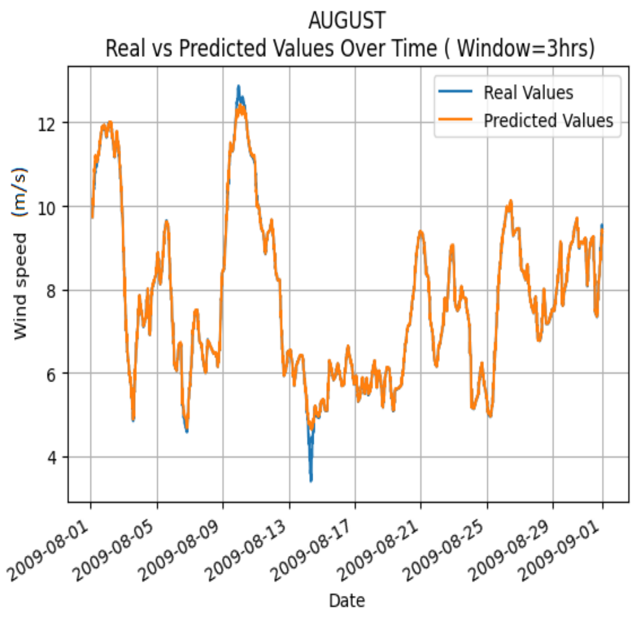
<u>August:</u>

Window size=3

<u>SAE</u>



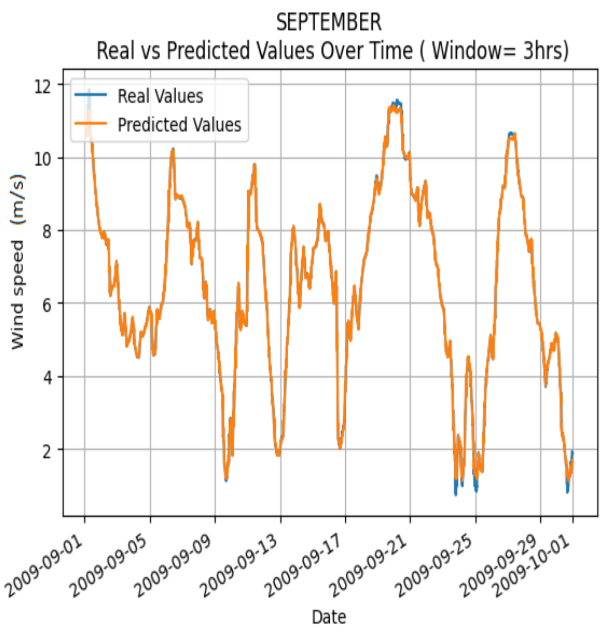
<u>SIRAE</u>

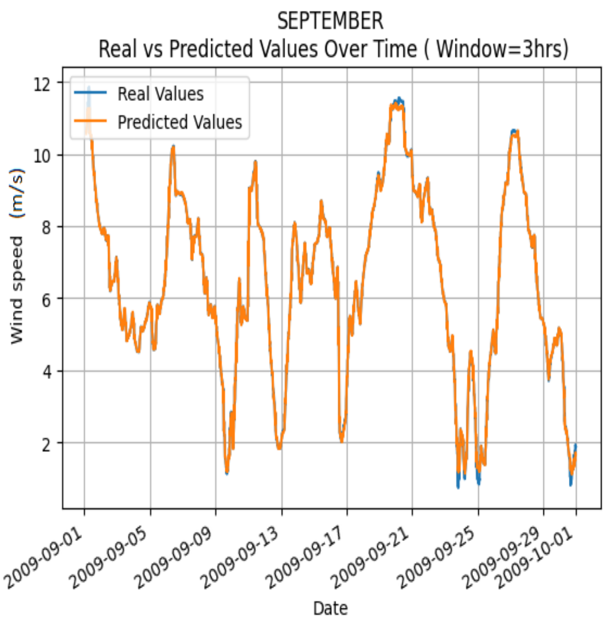


<u>September:</u>

Window size=3

<u>SAE</u> ***<u>SIRAE</u>***

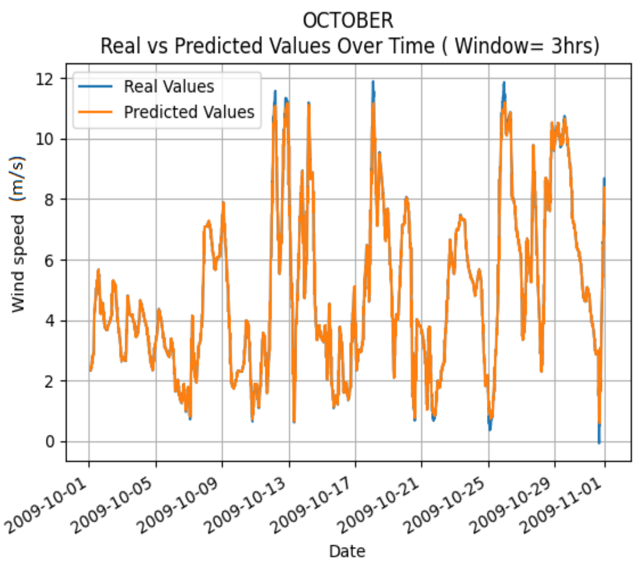


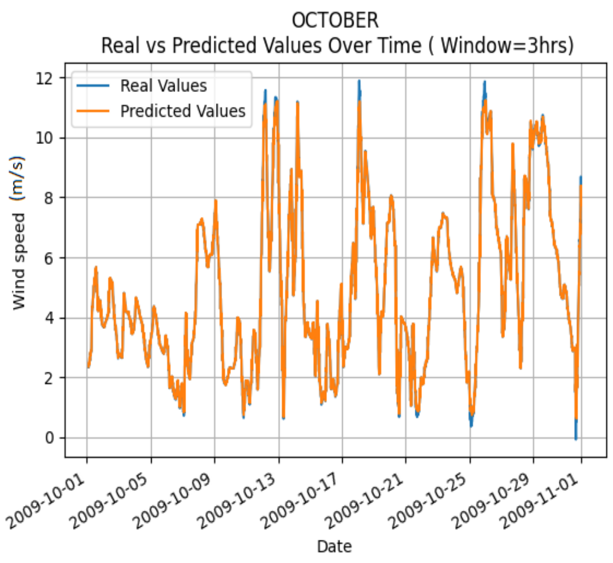


<u>October:</u>

Window size=3

<u>SAE</u> ***<u>SIRAE</u>***

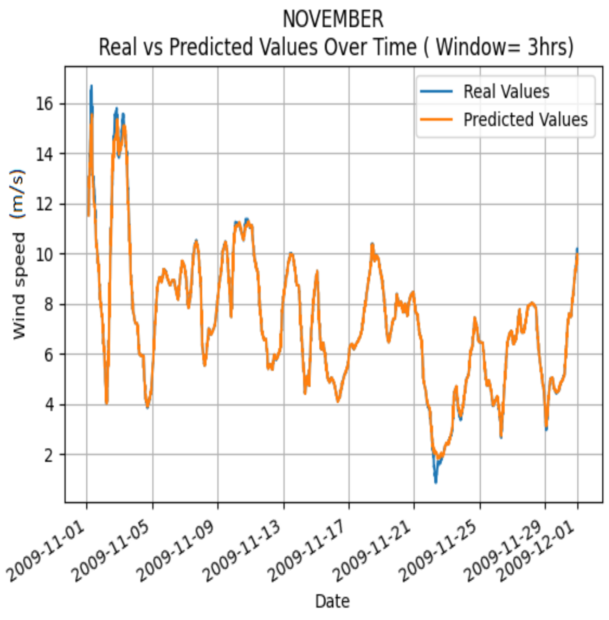




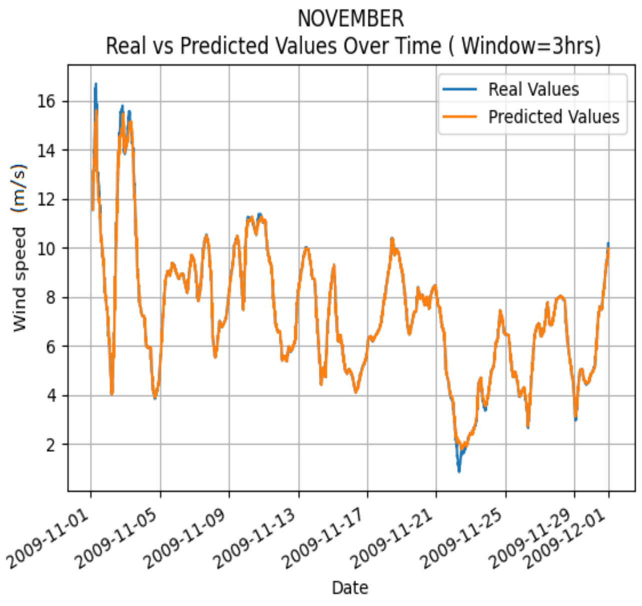
November:

Window size=3

SAE



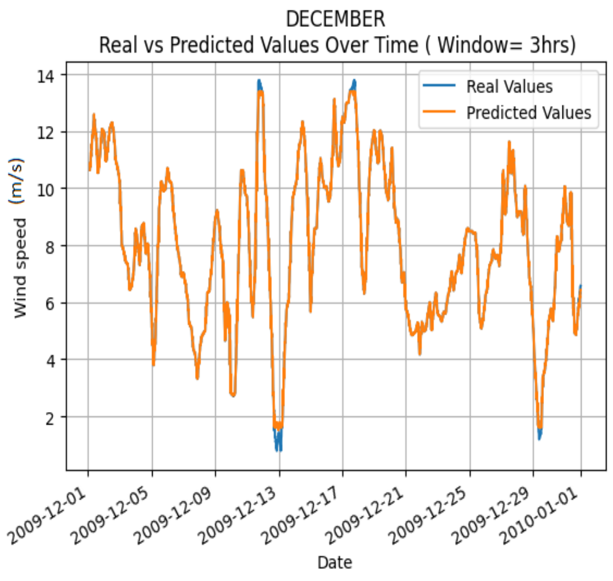
SIRAE



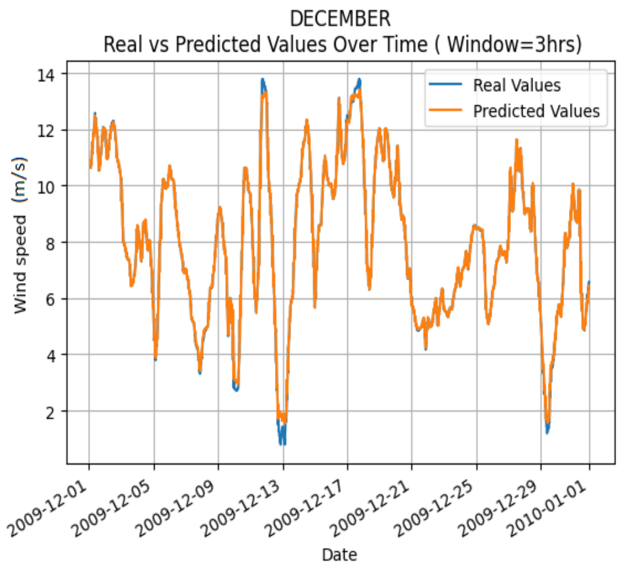
December:

Window size=3

SAE



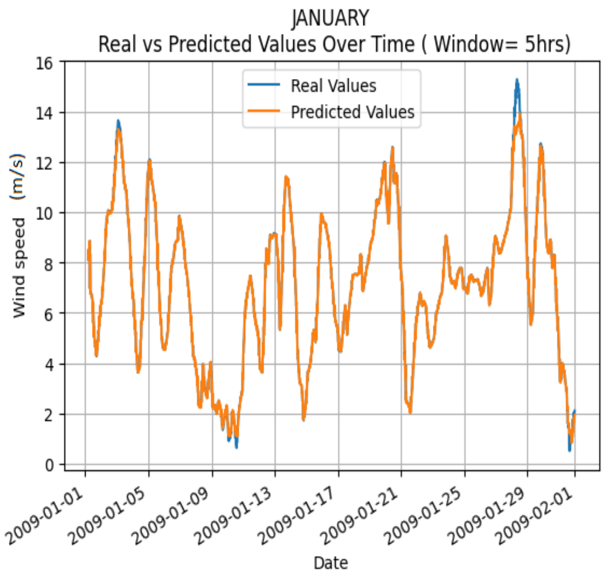
SIRAE

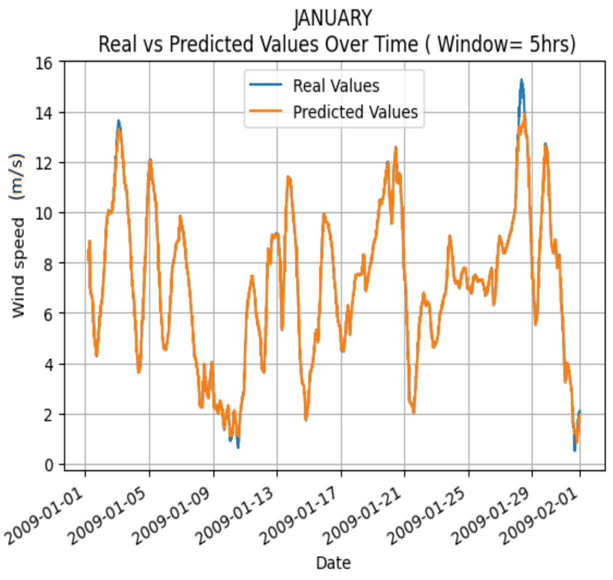


January:

Window size=5

SAE ***SIRAE***

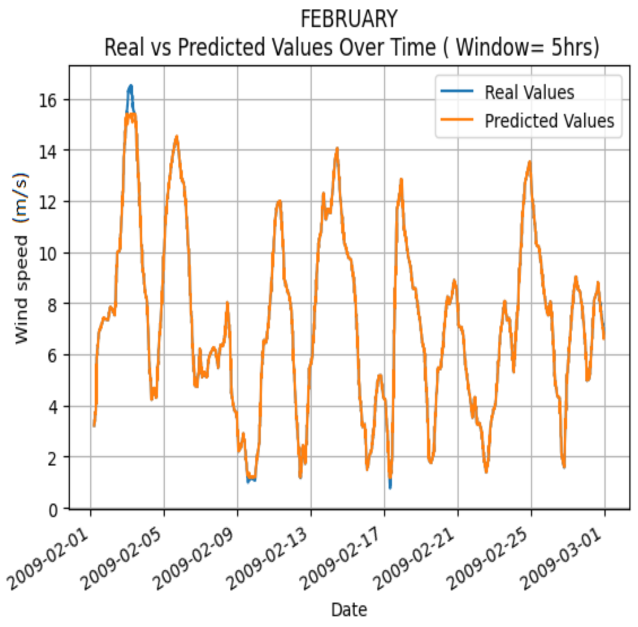


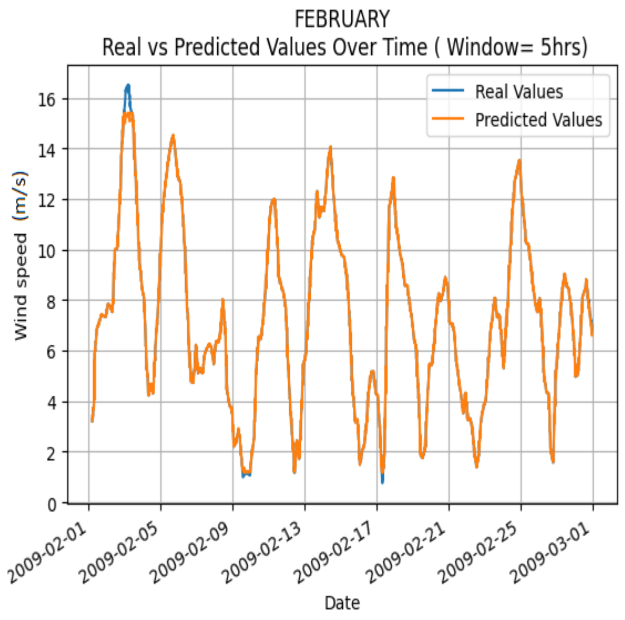


February:

Window size=5

SAE ***SIRAE***

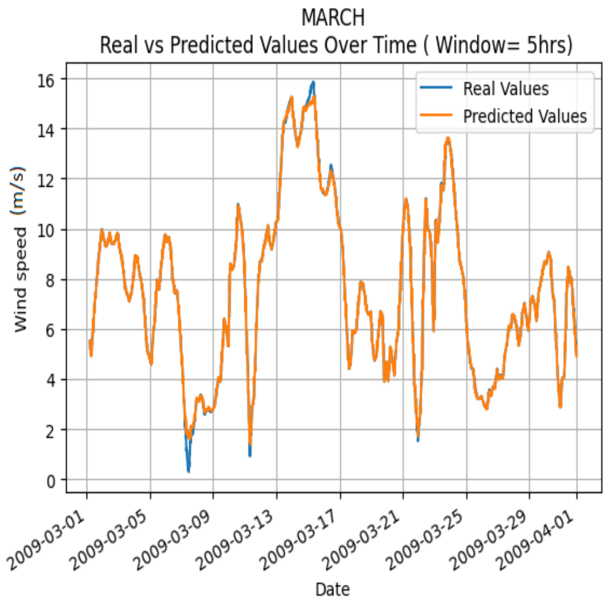


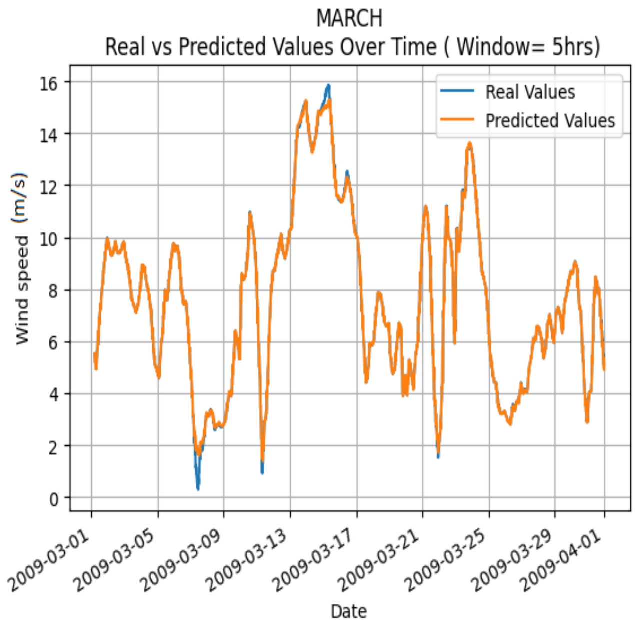


March:

Window size=5

SAE ***SIRAE***

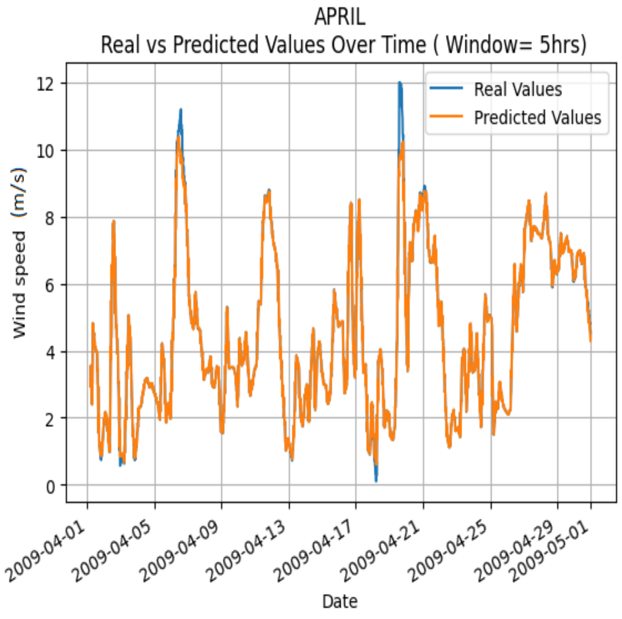


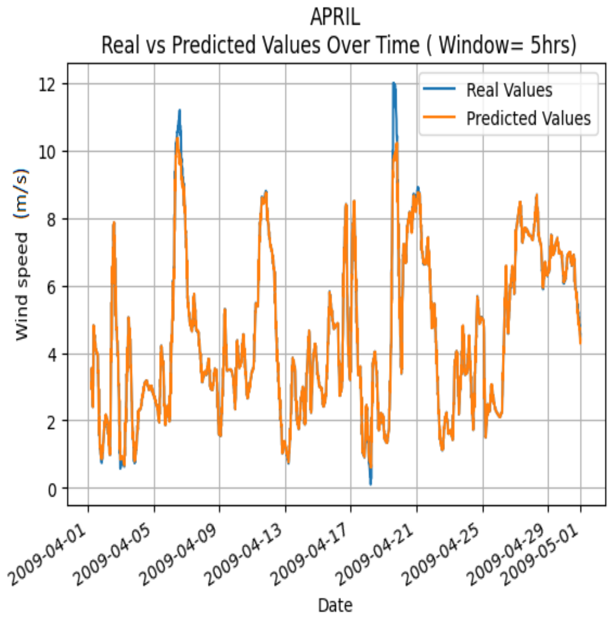


April:

Window size=5

SAE ***SIRAE***

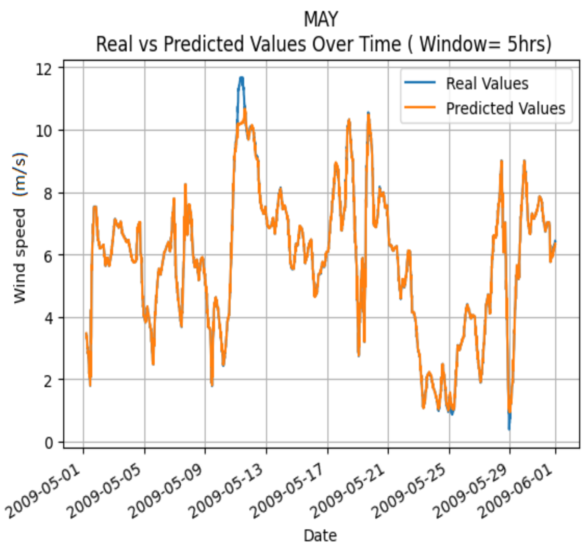


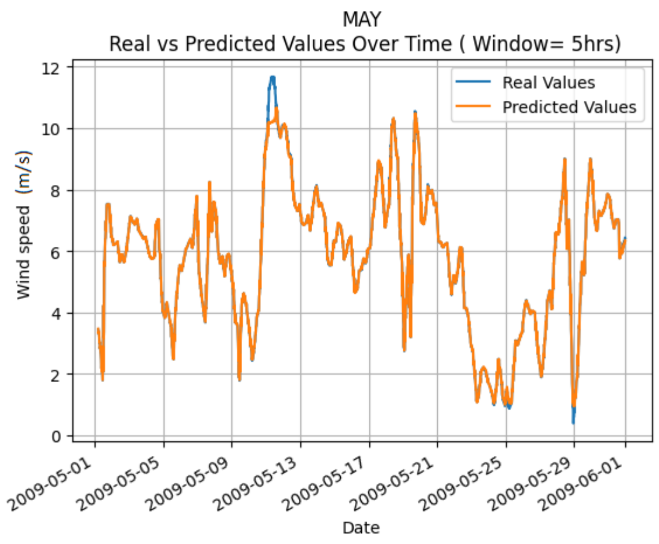


May:

Window size=5

SAE ***SIRAE***

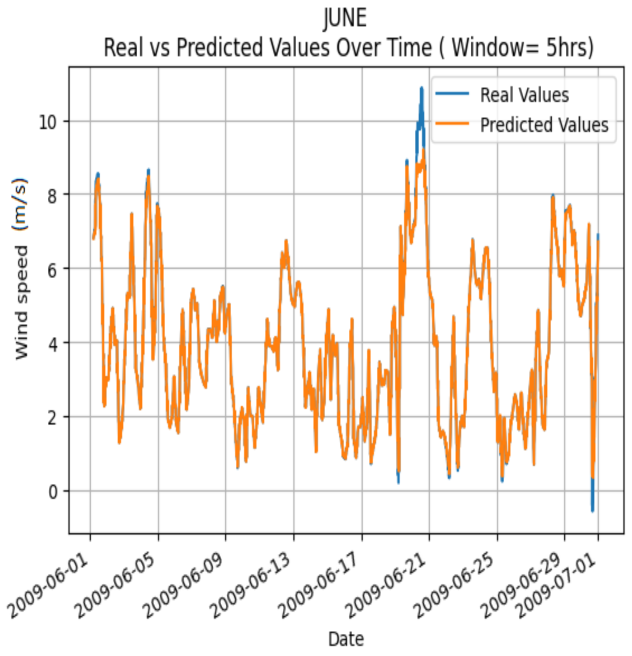


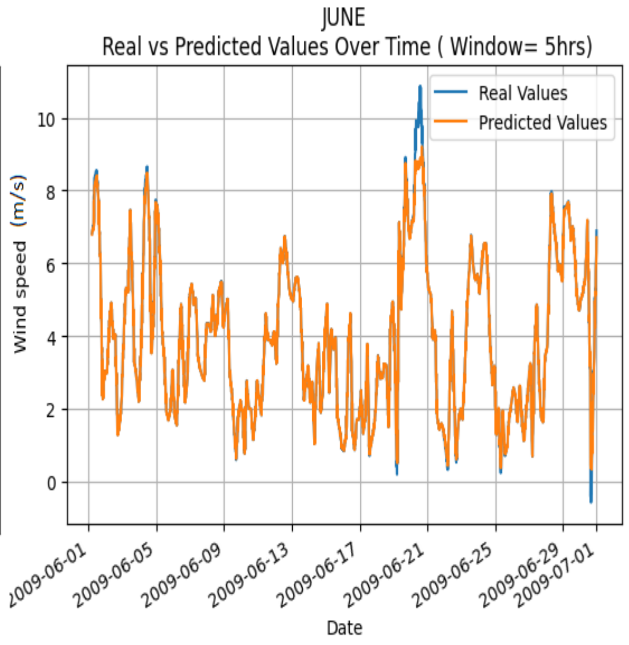


June:

Window size=5

SAE ***SIRAE***

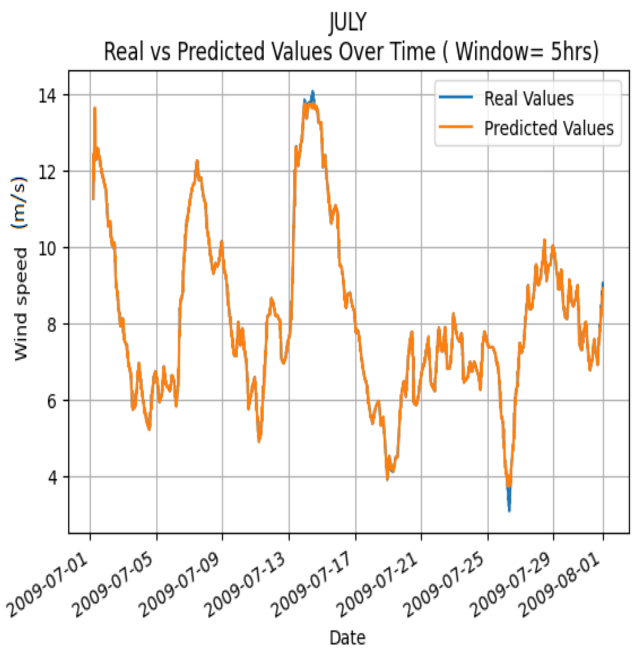


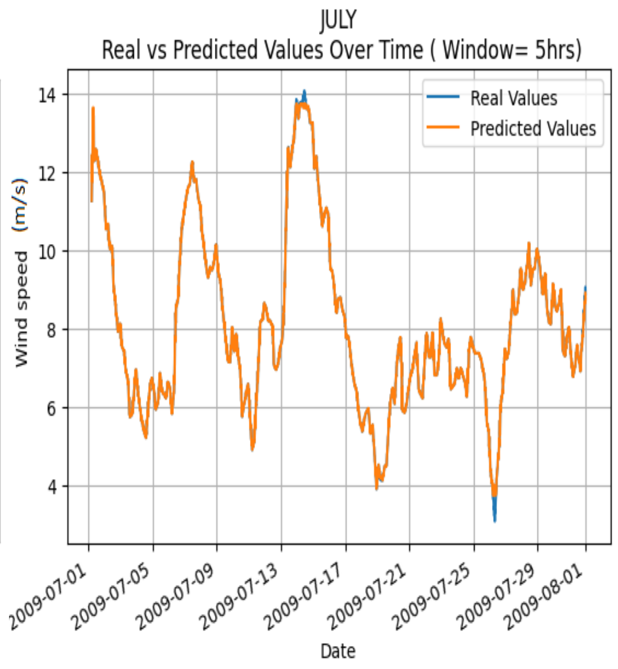


July:

Window size=5

SAE ***SIRAE***

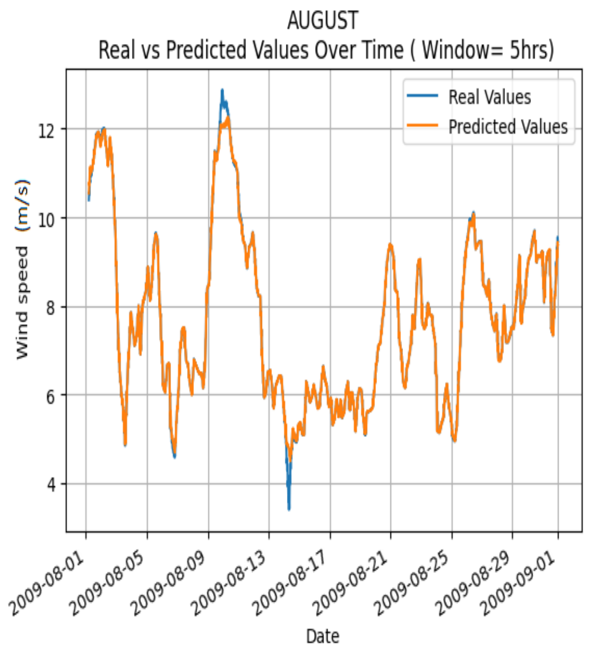


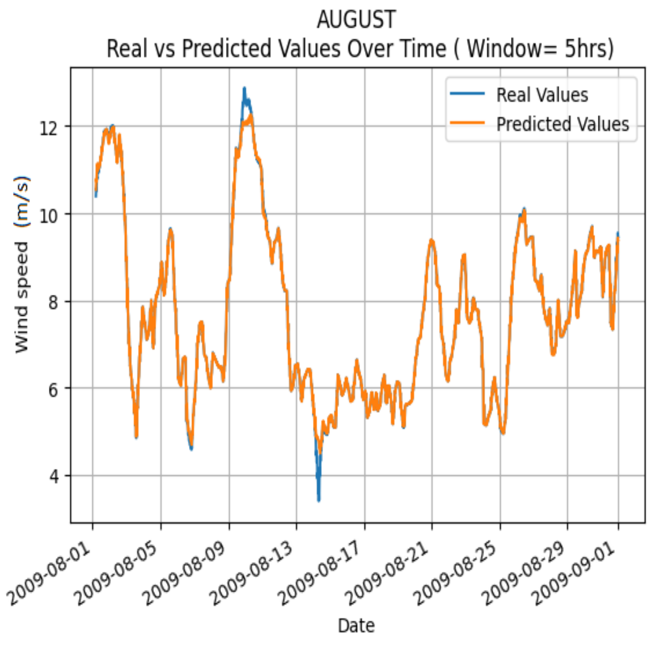


August:

Window size=5

SAE ***SIRAE***

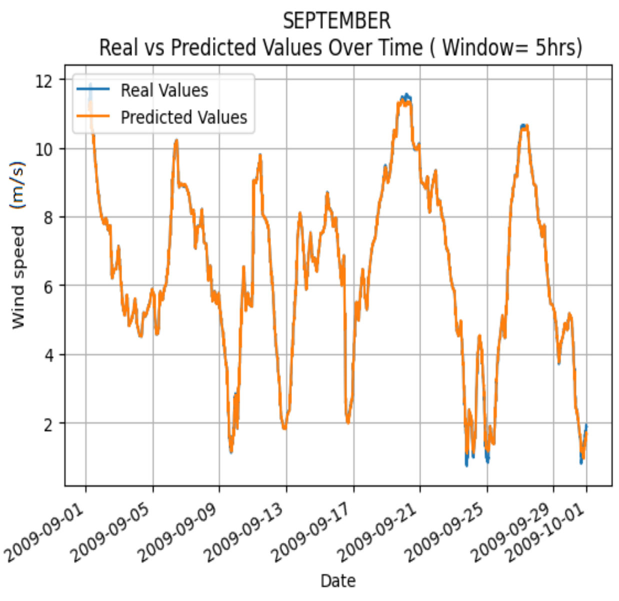




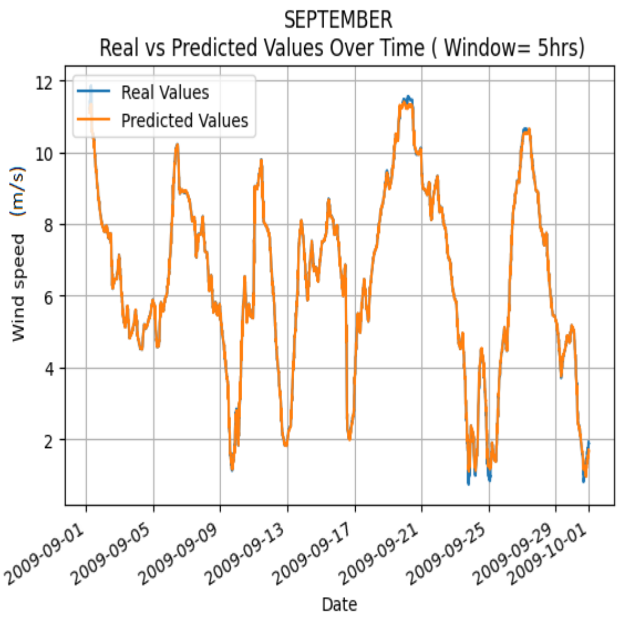
September:

Window size=5

SAE



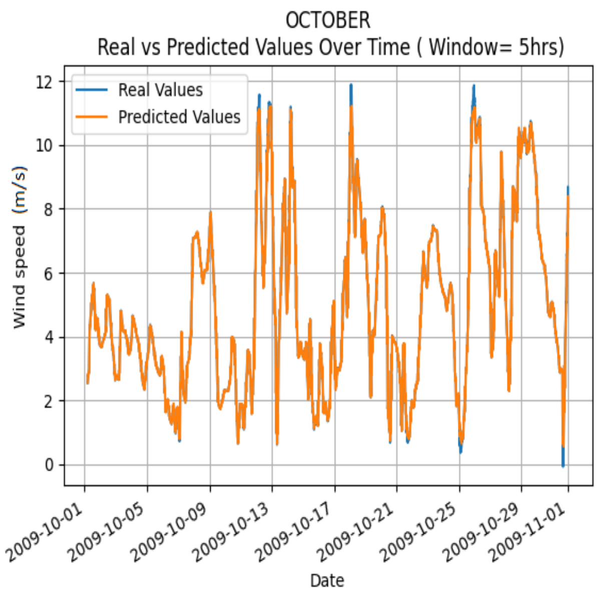
SIRAE



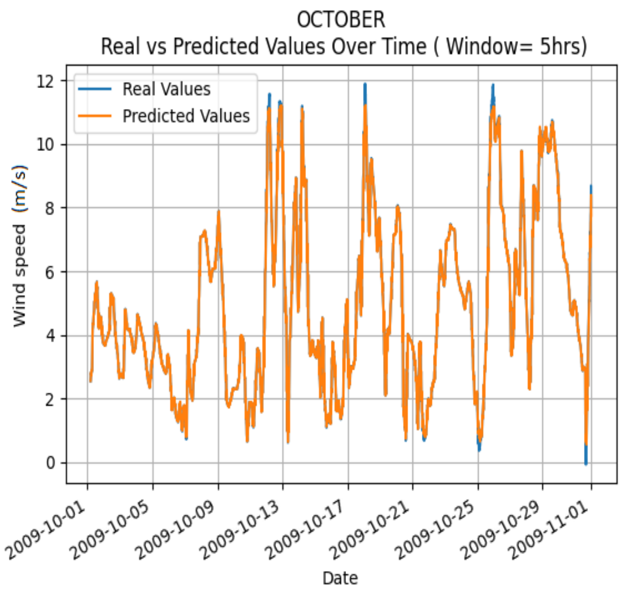
October:

Window size=5

SAE



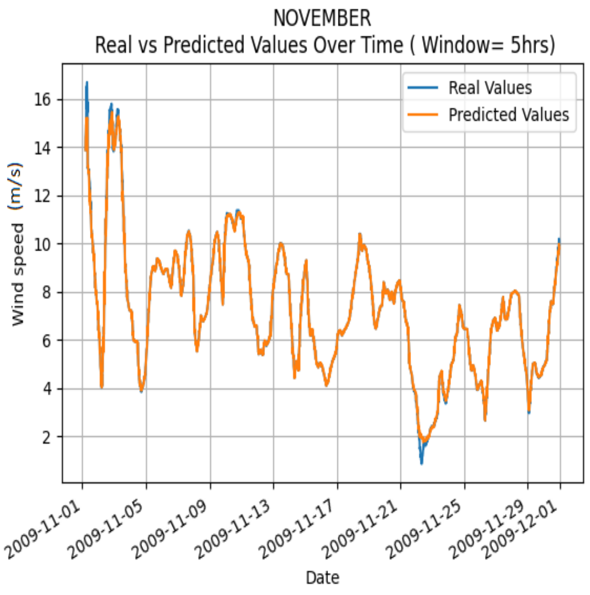
SIRAE

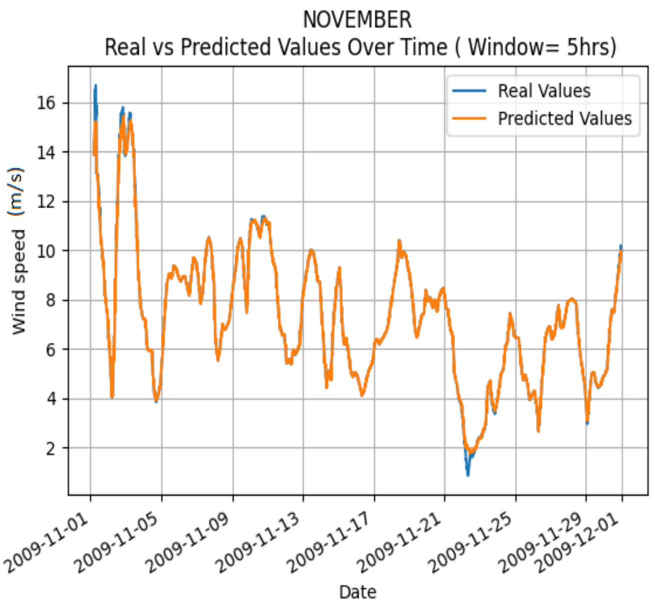


November:

Window size=5

SAE ***SIRAE***

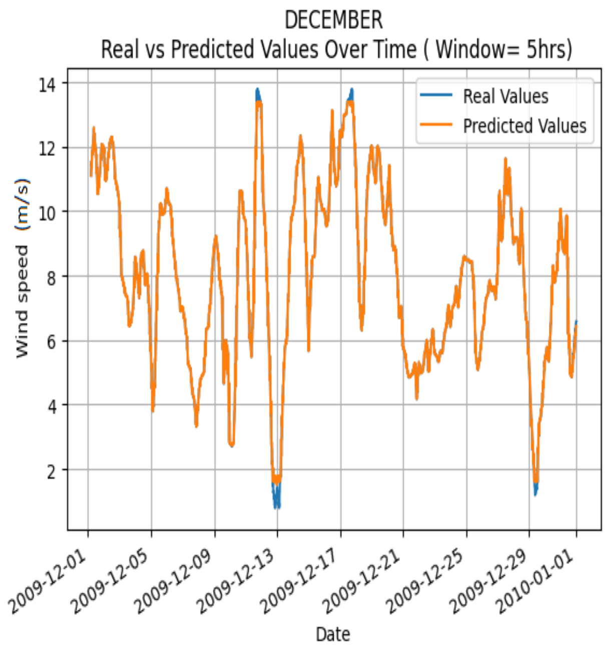


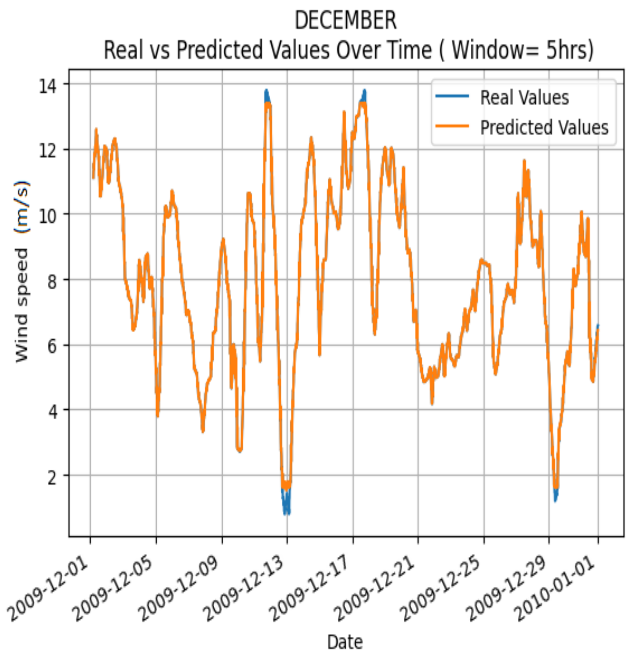


December:

Window size=5

SAE ***SIRAE***


DECEMBER
Real vs Predicted Values Over Time (Window= 5hrs)
Real Values
Predicted Values
Wind speed (m/s)
Date


DECEMBER
Real vs Predicted Values Over Time (Window= 5hrs)
Real Values
Predicted Values
Wind speed (m/s)
Date