Lecture-14-15
Course: Applied Data Science

Time Series Forecasting

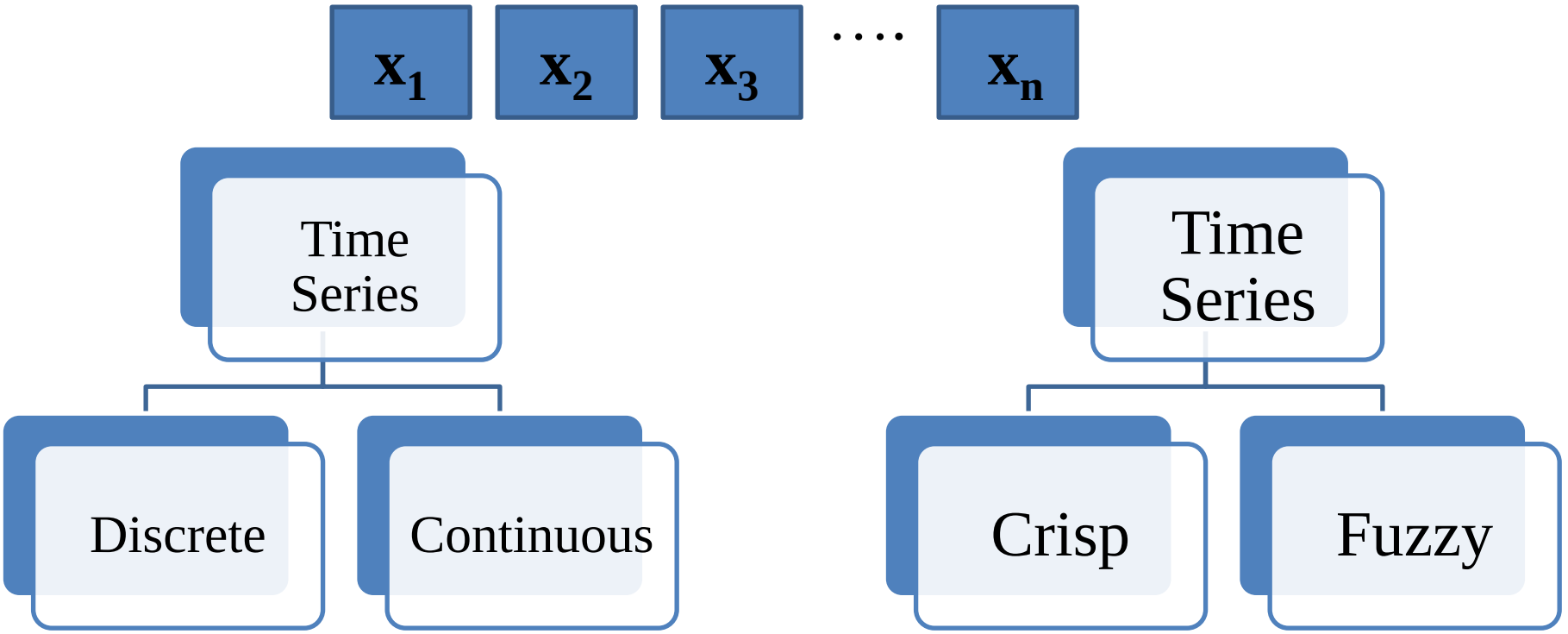
By
Dr. Sibarama Panigrahi
Senior Member, IEEE
Assistant Professor, Department of Computer Sc. & Engineering
National Institute of Technology, Rourkela, Odisha, 769008, India
Mobile No.: +91-7377302566
Email: panigrahis[at]nitrkl[dot]ac[dot]in
panigrahi[dot]sibarama[at]gmail[dot]com

Outlines

- Introduction and Motivation
- **Time Series Forecasting using Optimized Deep Learning Models**
- Hybrid Time Series Forecasting using Deep Learning Models
  - Additive Hybrid Models
  - Multiplicative Hybrid Models
  - Parallel Hybrid Models
  - Decomposition Based Hybrid Models
- References

INTRODUCTION

- Time Series
  - A time series is a set of observations of the same variable measured sequentially through time.

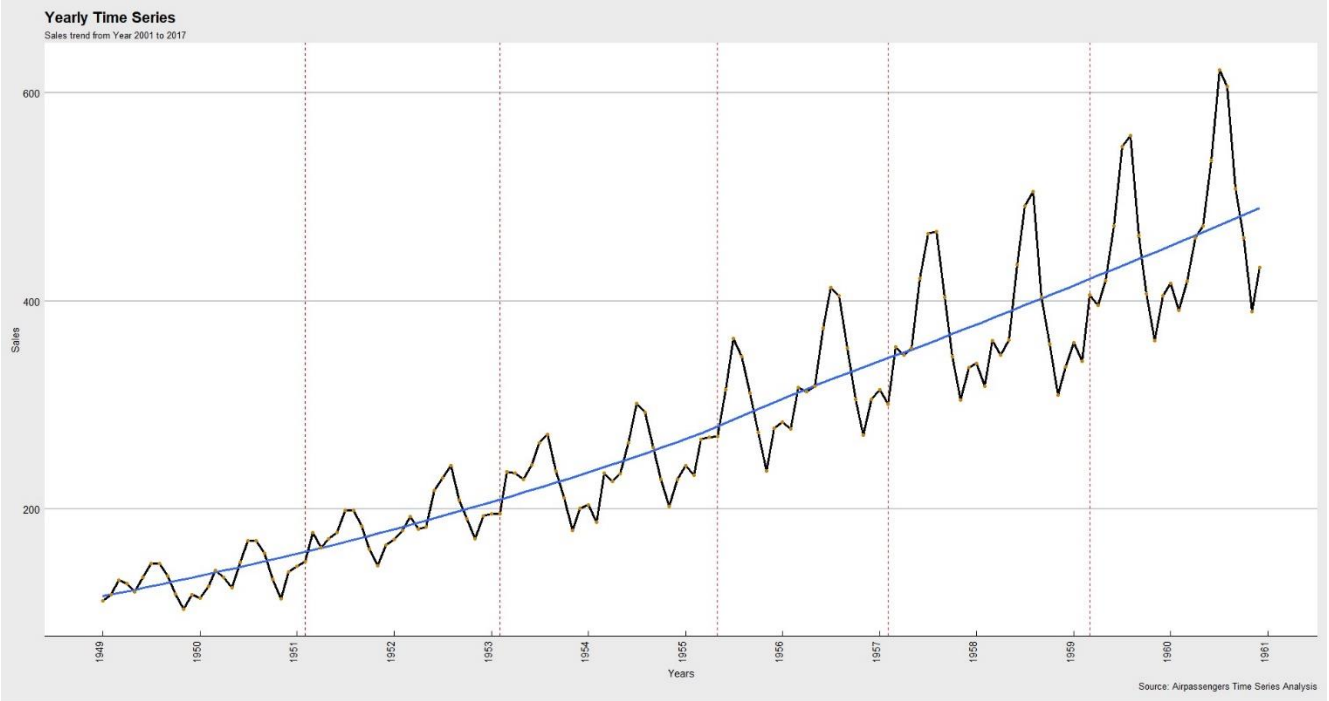


INTRODUCTION

- **Components of Time Series**
  - Trend
  - Seasonal
  - Cyclical
  - Irregular

INTRODUCTION

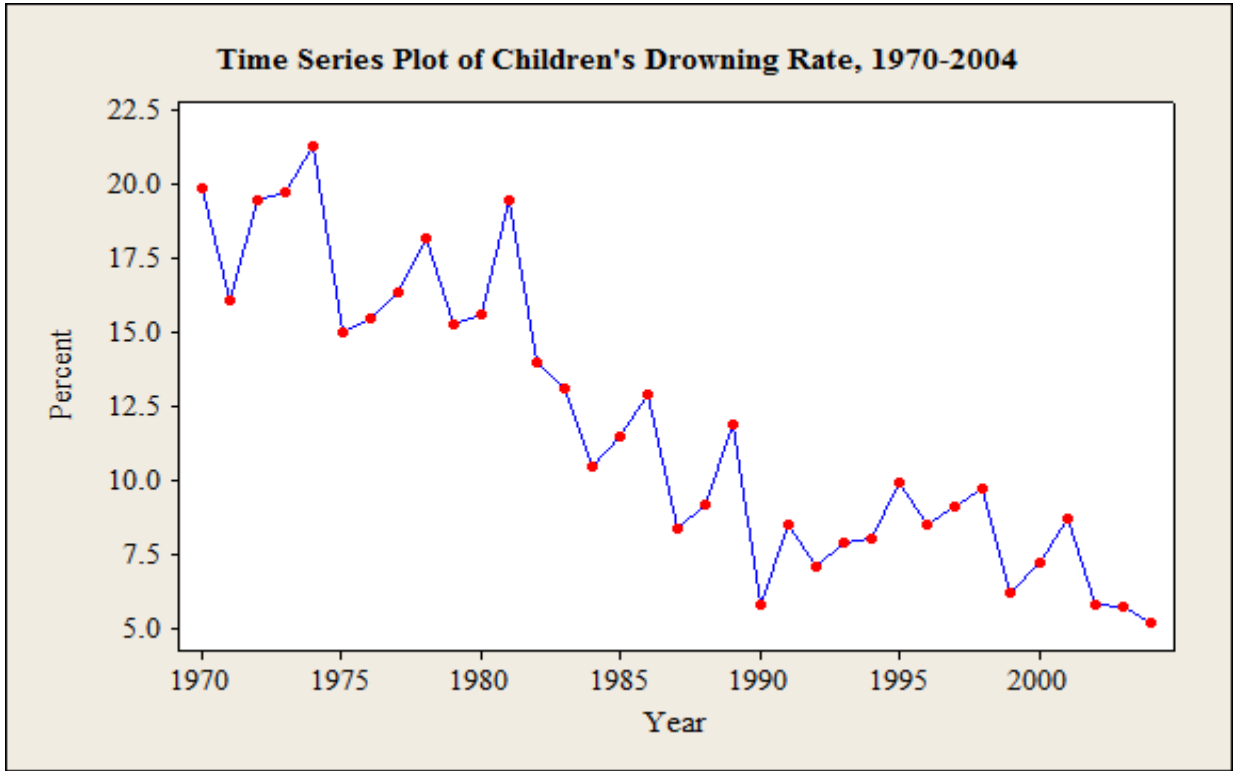
- **Components of Time Series**
  - **Trend:** Trend is a pattern in data that shows the movement of a series to relatively **higher or lower values over a long period of time**.
  - Seasonal
  - Cyclical
  - Irregular



Decreasing Trend

INTRODUCTION

- **Components of Time Series**
  - **Trend:** Trend is a pattern in data that shows the movement of a series to relatively higher or lower values over a long period of time.
  - Seasonal
  - Cyclical
  - Irregular



INTRODUCTION

- **Components of Time Series**
  - **Trend:** Trend is a pattern in data that shows the movement of a series to relatively higher or lower values over a long period of time.
    - **Trend component can be treated using differencing (desired order).**
    - e.g. Time Series: 8, 13, 18, 14, 13, 18,..
    - 1st Differenced Time Series: 5, 5, -4, -1, 5
    - Reverse De-Trending : 8

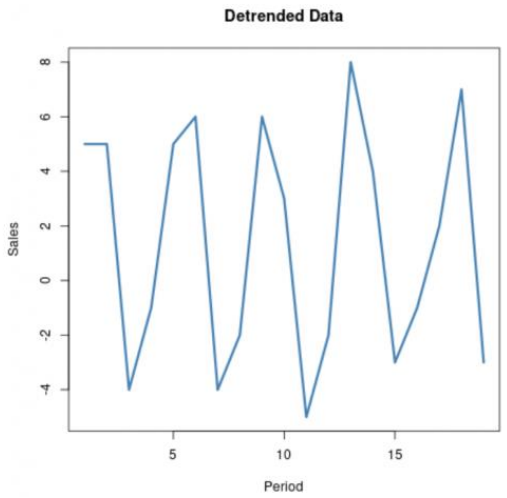
      8+5= 13

      13+5=18

      18+-4 =14

      14+-1 =13

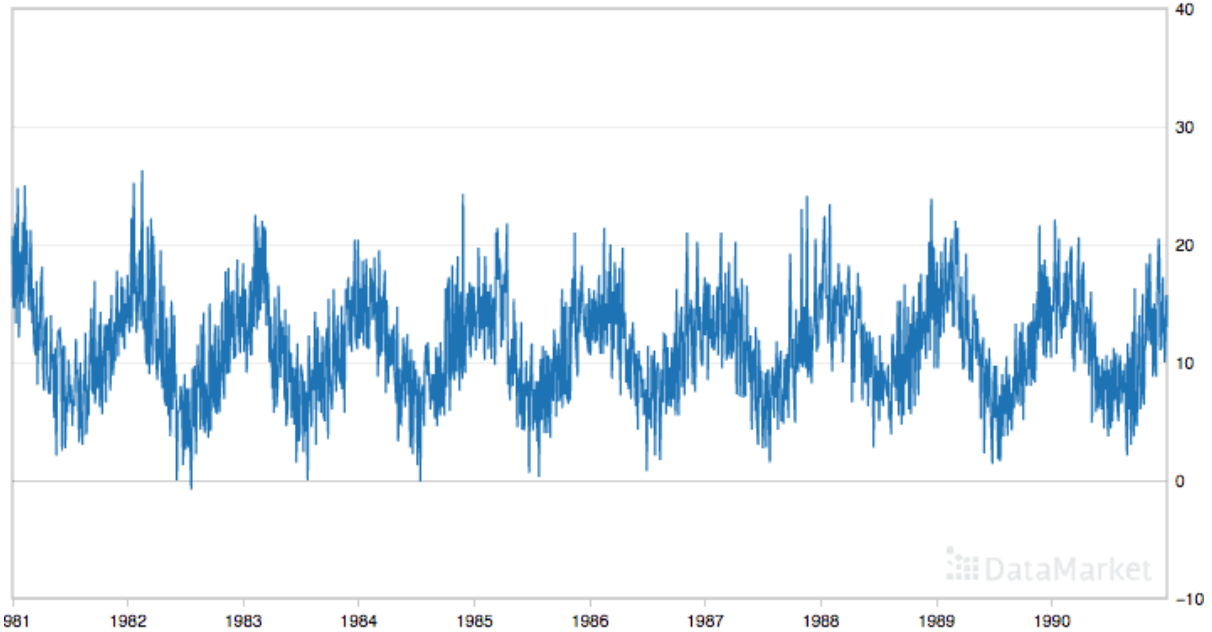
      13+5 =18



  - Python Code Link: https://machinelearningmastery.com/remove-trends-seasonality-difference-transform-python/

INTRODUCTION

- **Components of Time Series**
  - Trend
  - **Seasonal:** Seasonality, as its name suggested, refers to the seasonal characteristics of the time series data. It is the predictable pattern that repeats at a certain frequency within one year, such as weekly, monthly, quarterly, etc.
  - Cyclical
  - Irregular



INTRODUCTION

- **Components of Time Series**
  - Trend
  - **Seasonal:** Seasonality, as its name suggested, refers to the seasonal characteristics of the time series data. It is the predictable pattern that repeats at a certain frequency within one year, such as weekly, monthly, quarterly, etc.
    - **Seasonal component can be treated using seasonal differencing.**
      - We can remove seasonality in the data using differencing, which calculates the difference between the current value and its value in the previous season.

    - e.g. Time Series: 1,3,-5,0,4,-4
    - Seasonal Differenced Time Series: -1,1,1, … (Seasonal Length-3)
    - Reverse process : 1+-1= 0
      3+1= 4
      -5+1= -4

  - Python Code Link: https://machinelearningmastery.com/remove-trends-seasonality-difference-transform-python/

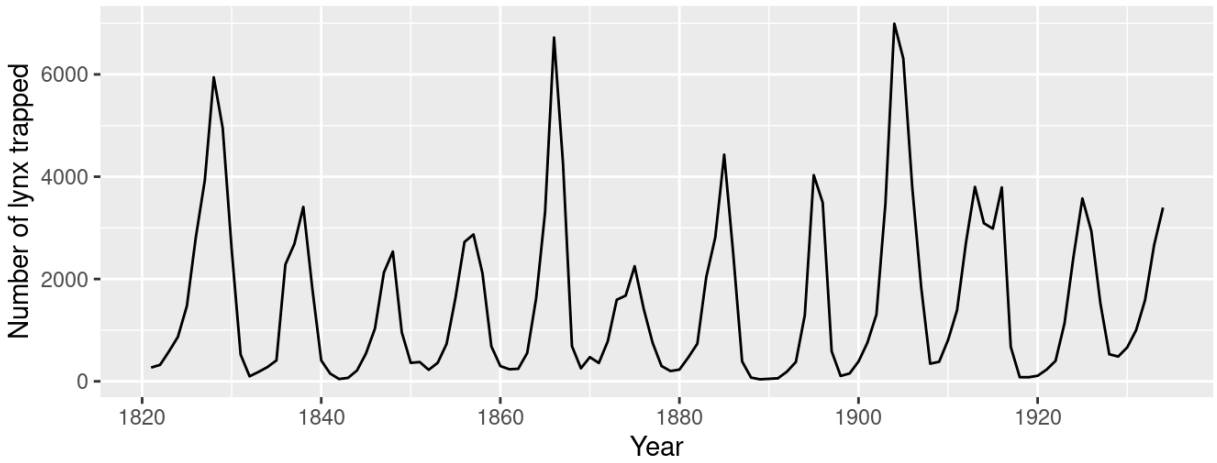
INTRODUCTION

- **Components of Time Series**
  - Trend
  - **Seasonal:** Seasonality, as its name suggested, refers to the seasonal characteristics of the time series data. It is the predictable pattern that repeats at a certain frequency within one year, such as weekly, monthly, quarterly, etc.
    - **Seasonal component can be treated using seasonal differencing using seasonal average.**
      - We can remove seasonality in the data using differencing by seasonal average, which calculates the difference between the current value and its seasonal average.

    - e.g. Time Series: 1,3,-5,1, 3, -5…
    - Seasonal Average (Season Length:3): 1, 3, -5
    - Seasonal Differenced Time Series: 0, 0, 0, 0, 0, 0 … (Seasonal Length-3)
    - Reverse process : 0+-1= 1
      0+3= 3
      0+-5= -5

INTRODUCTION

- **Components of Time Series**
  - Trend
  - Seasonal
  - **Cyclical:** The cyclical component of a time series refers to (regular or periodic) fluctuations excluding the irregular component.
  - Irregular



INTRODUCTION

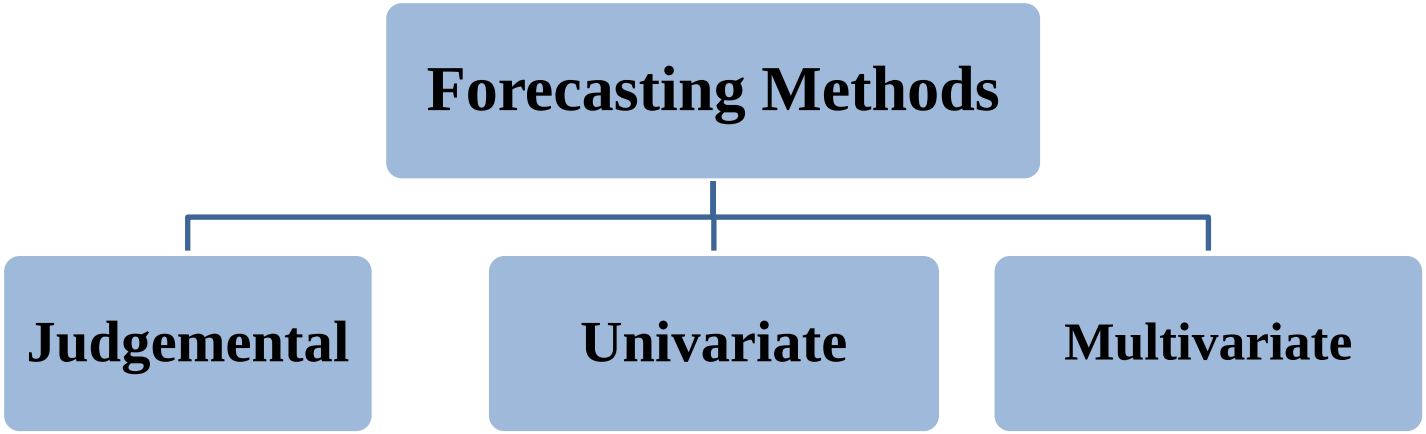
- **Components of Time Series**
  - Trend
  - Seasonal
  - Cyclical
  - **Irregular:** This component is unpredictable. Every time series has some unpredictable component that makes it a random variable.

INTRODUCTION

- **Time Series Forecasting**
  - **Time series forecasting (TSF)** is the *process of predicting* the *future outcomes* of a phenomenon by ***systematically analyzing its past observations***.

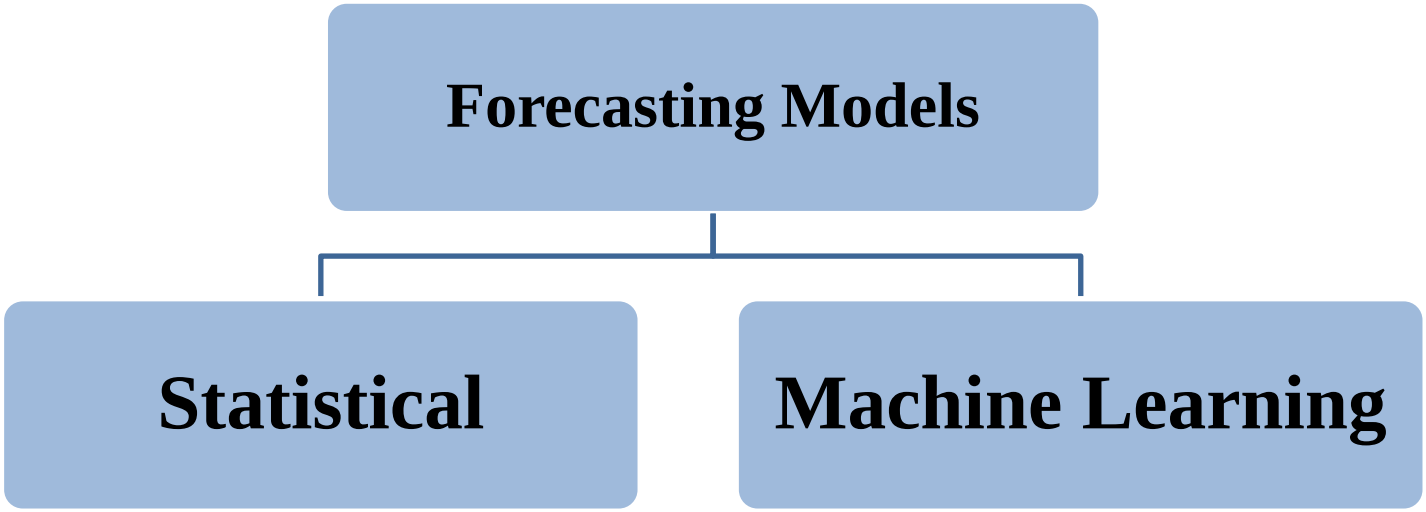
INTRODUCTION

- **Forecasting Method:**
  - A forecasting method is a procedure to predict the future values by systematically analyzing the past observations.



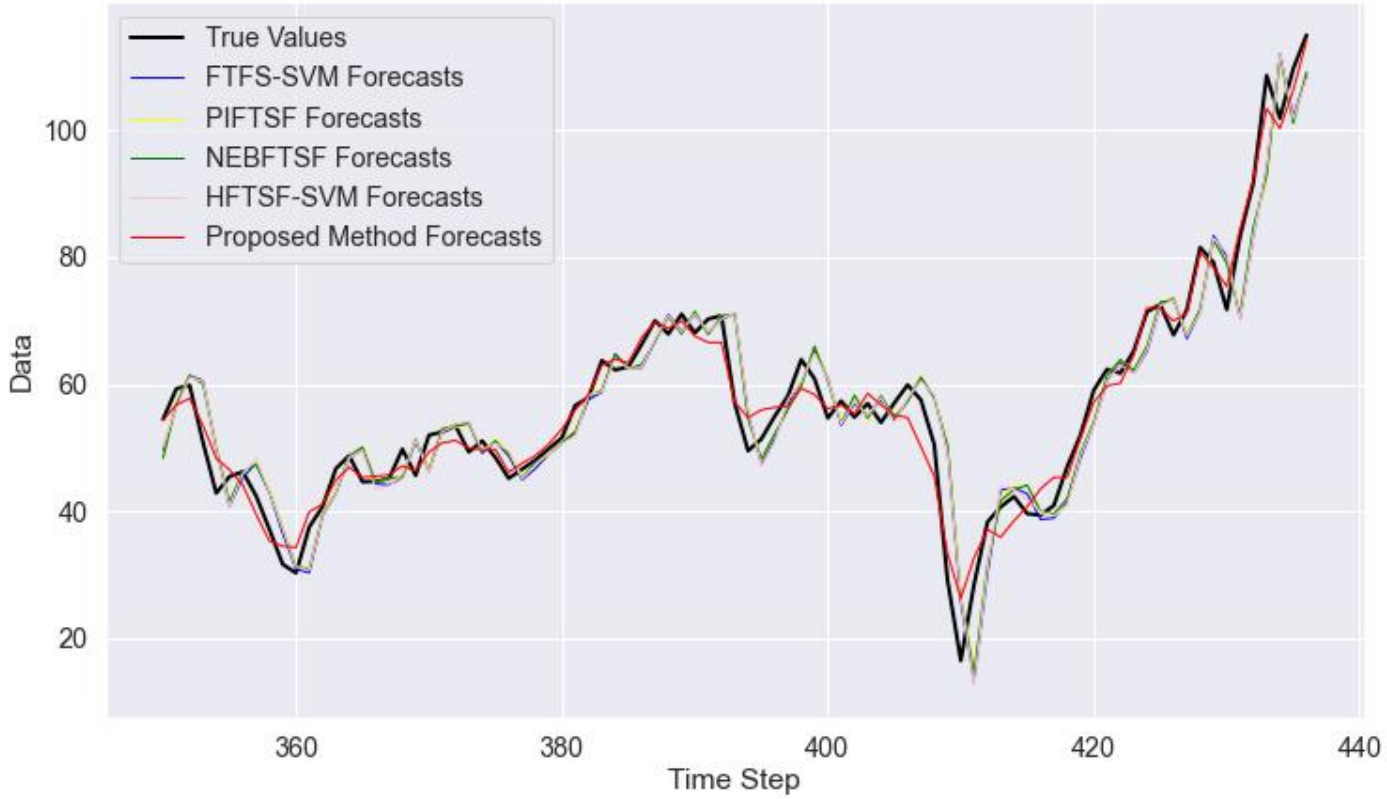
INTRODUCTION

- **Forecasting Model:**
  - A forecasting model is a tool to reveal the relationship existing in past data and using which the future values are predicted.



INTRODUCTION

- **Point vs Interval Forecasting**
  - **Point (Deterministic) Forecasting**



INTRODUCTION

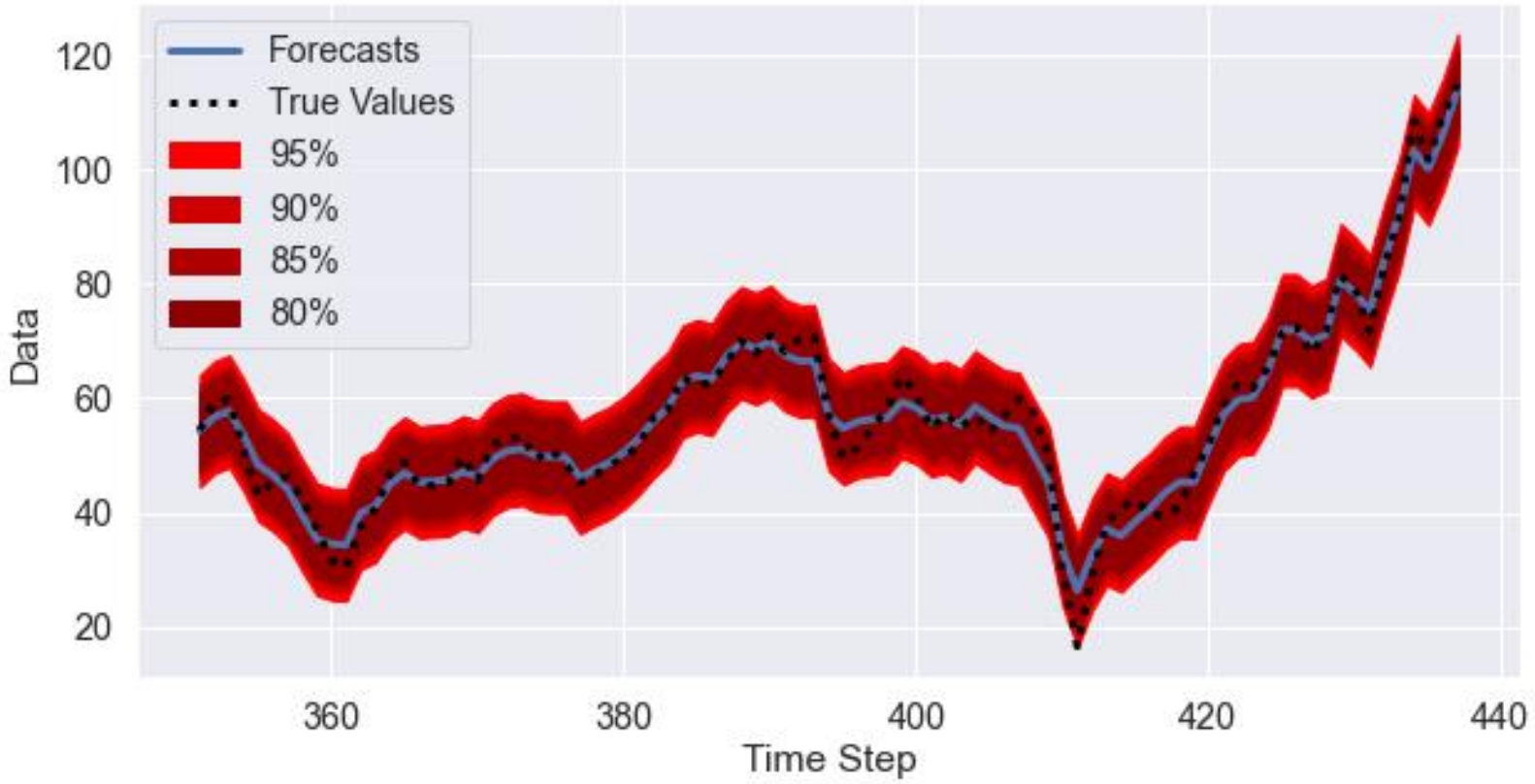
- **Point Forecasting Accuracy Measures**

Error-metric	Accuracy Measure	Formula
Scale-dependent	Mean Absolute Error (MAE) or Mean Absolute Deviation (MAD)	$\frac{1}{n}\sum_{i=1}^{n}\lvert y_i - \hat{y}_i\rvert$
	Geometric Mean Absolute Error (GMAE)	$\left(\prod_{i=1}^{n}\lvert y_i - \hat{y}_i\rvert\right)^{\frac{1}{n}}$
	Mean Square Error (MSE)	$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
	Root Mean Square Error (RMSE)	$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$
Percentage	Mean Absolute Percentage Error(MAPE)	$\frac{1}{n}\sum_{i=1}^{n}\frac{\lvert y_i - \hat{y}_i\rvert}{y_i} \times 100$
	Symmetric Mean Absolute Percentage Error(SMAPE)	$\frac{1}{n}\sum_{i=1}^{n}\frac{\lvert y_i - \hat{y}_i\rvert}{(\lvert y_i\rvert + \lvert\hat{y}_i\rvert)/2}$
Relative	Median Relative Absolute Error (MdRAE)	$median\left(\left\lvert\frac{y_i - \hat{y}_i}{y_i - \hat{y}_i^*}\right\rvert\right)$
	Geometric Mean Relative Absolute Error (GMRAE)	$\left(\prod_{i=1}^{n}\left\lvert\frac{y_i - \hat{y}_i}{y_i - \hat{y}_i^*}\right\rvert\right)^{\frac{1}{n}}$
Scale-free	Mean Absolute Scaled Error (MASE)	$\frac{1}{n}\sum_{i=1}^{n}\frac{\lvert y_i - \hat{y}_i\rvert}{\frac{1}{n-1}\sum_{j=2}^{n}\lvert y_i - y_{i-1}\rvert}$

where y_i and $\hat{y}_i$ denote the ith actual and forecasted value, n denotes the number of observations. **Lower the value better the forecasts.**

INTRODUCTION

- **Point vs Interval Forecasting**
  - **Interval (Probabilistic) Forecasting:**



INTRODUCTION

- **Interval Forecasting Accuracy Measures**

$$Prediction\ Interval\ Coverage\ Probabilty\ (PICP^{(\alpha)}) = \frac{1}{n}\sum_{i=1}^{n} C_i,\ C_i = \begin{cases} 1 & y_i \in [L_i^{(\alpha)}, U_i^{(\alpha)}] \\ 0 & y_i \notin [L_i^{(\alpha)}, U_i^{(\alpha)}] \end{cases}$$

$$Prediction\ Interval\ Normalized\ Average\ Width\ (PINAW^{(\alpha)}) = \frac{1}{nR}\sum_{i=1}^{n}(U_i^{\alpha} - L_i^{\alpha})$$

$$Accumulated\ Width\ Deviation\ (AWD^{(\alpha)}) = \frac{1}{n}\sum_{i=1}^{n} AWD_i^{\alpha}\ , AWD_i^{\alpha} = \begin{cases} \frac{L_i^{(\alpha)} - y_i}{U_i^{(\alpha)} - L_i^{(\alpha)}}, y_i < L_i^{(\alpha)} \\ 0\quad ,\ y_i \in [L_i^{(\alpha)}, U_i^{(\alpha)}] \\ \frac{y_i - U_i^{(\alpha)}}{U_i^{(\alpha)} - L_i^{(\alpha)}}, y_i > U_i^{(\alpha)} \end{cases}$$

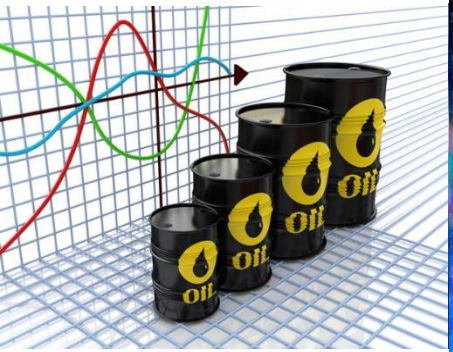
$$Average\ Coverage\ Error\ (ACE^{(\alpha)}) = PICP^{(\alpha)} - PINC^{(\alpha)}$$

α denotes the significance level, $L_i^{(\alpha)}$ and $U_i^{(\alpha)}$ denote the lower bound and upper bound for ith data point. **The lower values of PINAW, and AWD are desirable, while the higher values of PICP and ACE are desirable for a better model.**

INTRODUCTION

- **Basic Steps in Forecasting**
  - ***Step 1: Problem definition***
  - ***Step 2: Gathering information***
  - ***Step 3: Preliminary (exploratory) analysis***
  - ***Step 4: Choosing and fitting models***
  - ***Step 5: Using and evaluating a forecasting model***

MOTIVATION



Crude Oil

Stock Price

Retail Industry

Internet Traffic

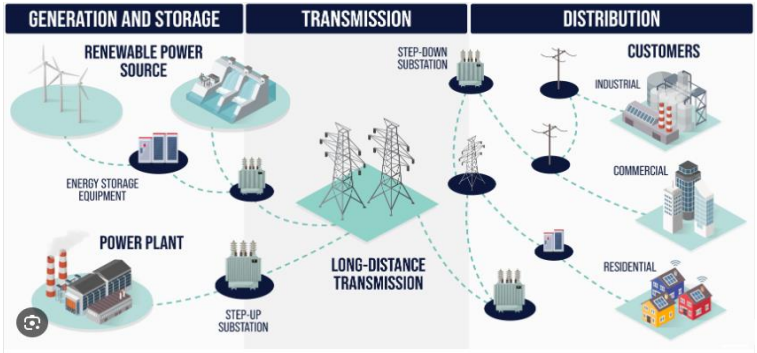
Electricity Price

Call Volume

Flood

Earthquake

MOTIVATION



Electricity Load Forecasting

Air Quality Index Forecasting

Rainfall Forecasting

Streamflow Forecasting

Agricultural Product Price Forecasting

Seed Demand Forecasting

Wind Speed Forecasting

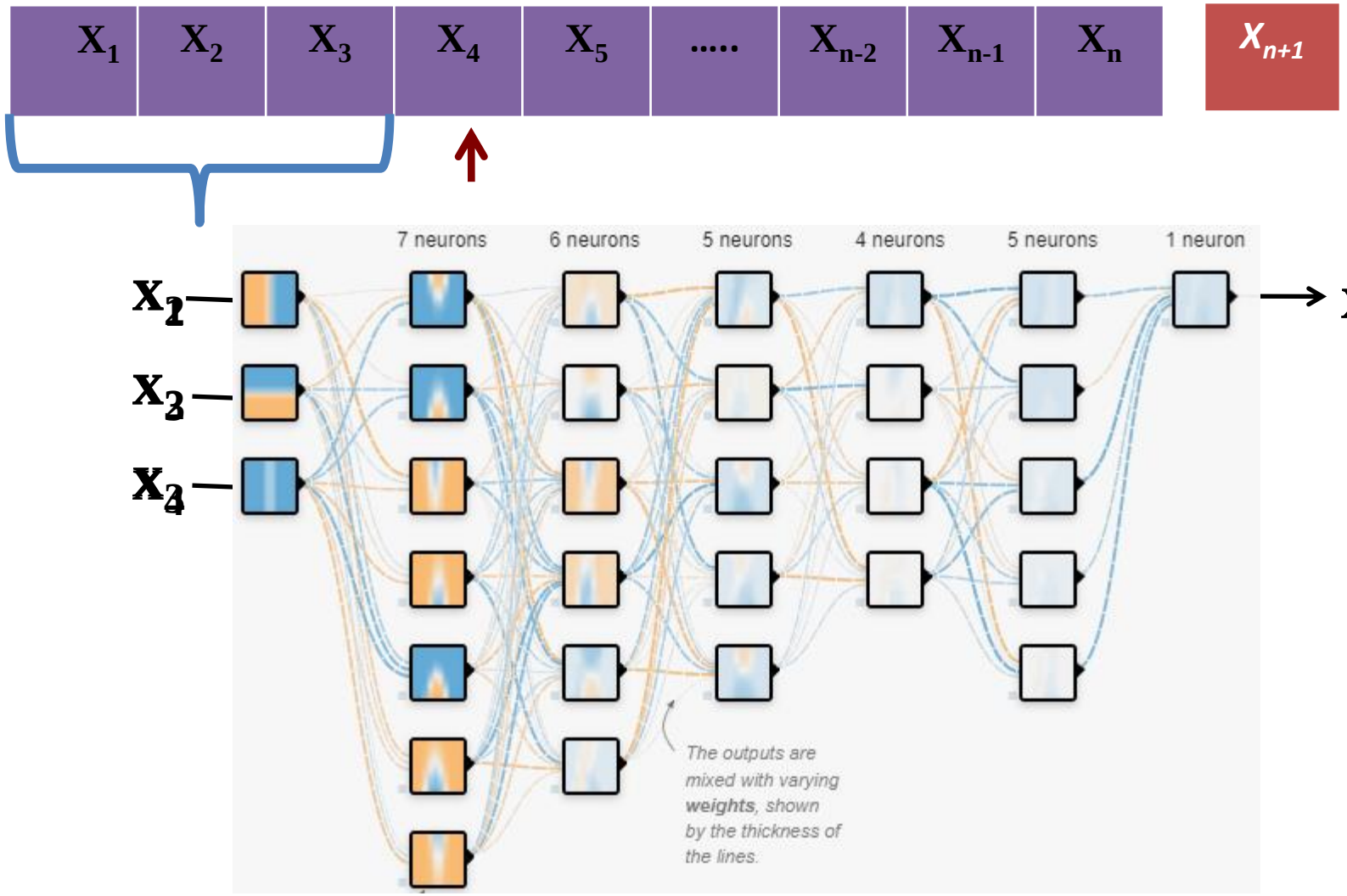
Temperature Forecasting

:

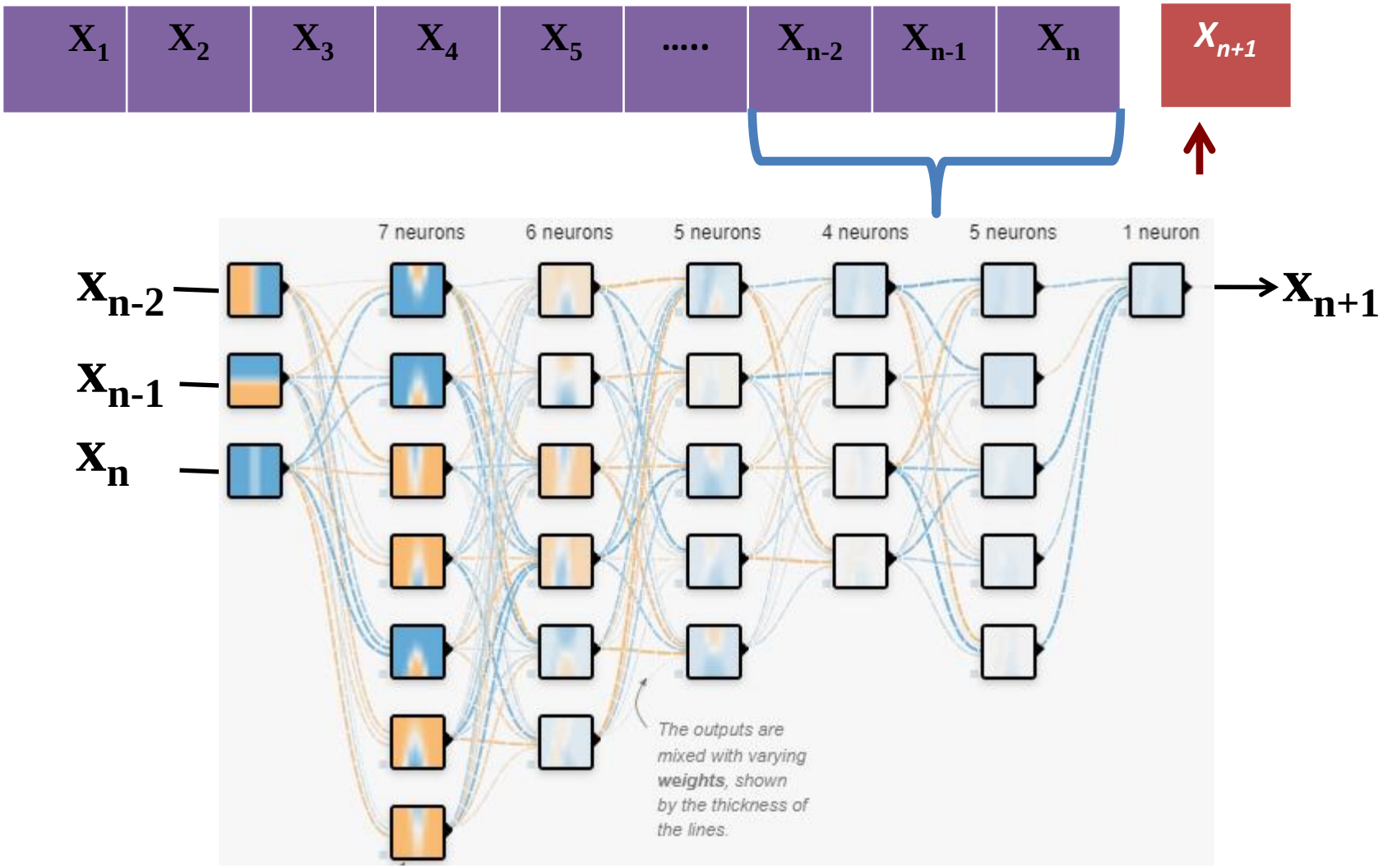
:

:

How to use Deep Learning Models in TSF



How Deep Neural Network in TSF



How Deep Neural Network for Multi-Step Ahead TSF

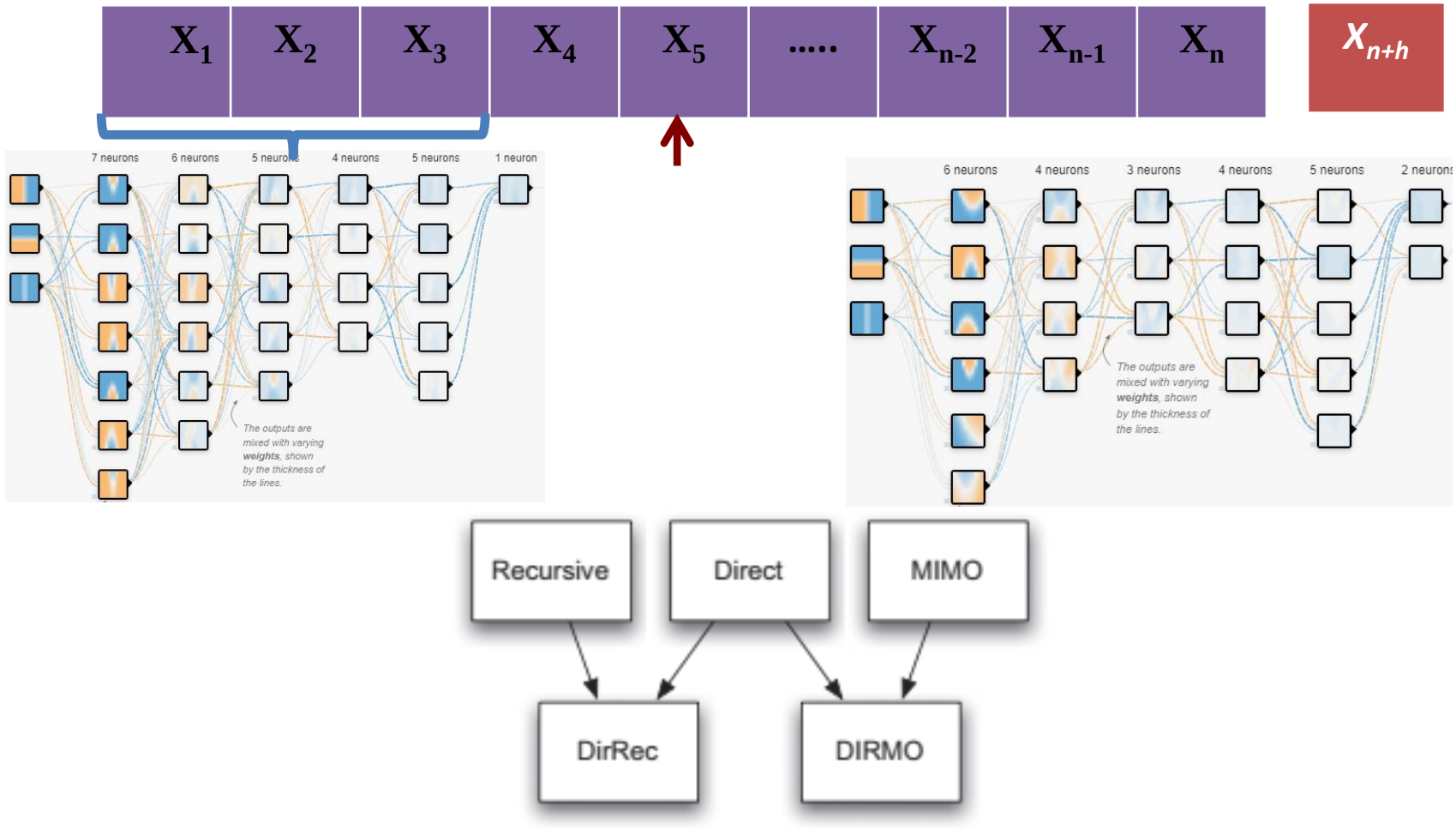


Fig. 1. The different forecasting strategies with the links showing their relationship.

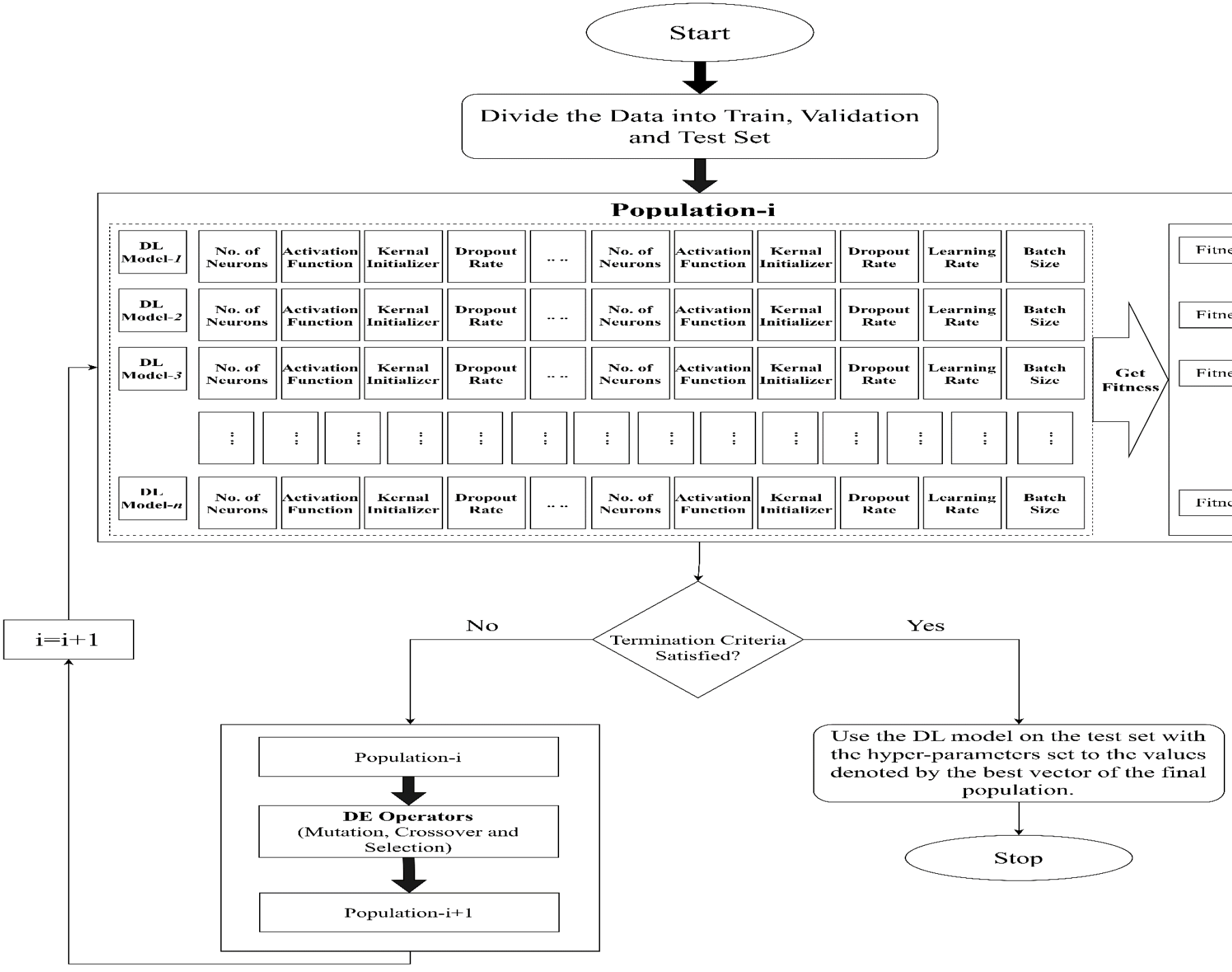
Taieb, S. B., Bontempi, G., Atiya, A. F., & Sorjamaa, A. (2012). A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. *Expert systems with applications*, *39*(8), 7067-7083.

BIG QUESTIONS?

- How many Inputs?
- How many Layers?
- How many neurons in each layer?
- Which Activation Function?
- Which Kernel Initializer?
- Which Optimizer?
- What is the Batch Size?
- What is the Learning Rate?

POSSIBLE SOLUTIONS

- **Grid Search**
  - Inappropriate to handle continuous valued hyper-parameters like learning rate, dropout rate, etc.
  - Suffers from dimensionality issues when the number of hyper-parameters to tune becomes large.
- **Bayesian Optimization**
  - Inappropriate to handle discrete valued hyper-parameters like activation function, kernel initializer, batch size, number of neurons in each layer, etc.
  - Suffers from dimensionality issues when the number of hyper-parameters to tune becomes large.
- **Swarm and Evolutionary Algorithms**
  - Can handle both continuous valued and discrete valued hyper-parameters.
  - Scalable.


Start
Divide the Data into Train, Validation and Test Set
Population-i
DL Model-1
DL Model-2
DL Model-3
DL Model-n
No. of Neurons
Activation Function
Kernal Initializer
Dropout Rate
.. ..
Learning Rate
Batch Size
Get Fitness
Fitness-1
Fitness-2
Fitness-3
Fitness-n
Termination Criteria Satisfied?
No
Yes
i=i+1
Population-i
DE Operators (Mutation, Crossover and Selection)
Population-i+1
Use the DL model on the test set with the hyper-parameters set to the values denoted by the best vector of the final population.
Stop

SWARM AND EVOLUTIONARY ALGORITHM OPTIMIZED DL MODELS

Reference	Deep Learning Model	Hyper-Parameters	Optimization Technique	Encoding Scheme	Application
[1]	CNN	Dropout rate, Number of epochs, Number of filters, Number of outputs of each layer, Optimizer, Batch Size, Steps taken between pooling, , Size of Pooling, Layer type, Activation function, Initialization function, Rows and columns of the kernel, including bias, Behaviour of operator at the border.	Evolutionary Algorithm (EA)	Variable-length real encoding	Image Classification
[2]	CNN	Number of epochs, Contraction Expansion coefficient, measure times, Batch Size.	Particle Swarm Optimization (PSO)	Fixed-length binary encoding	Image Classification
[3]	CNN	Dropout rate, Kernel Size, No. of epochs, Batch Normalisation.	PSO	Variable-length real encoding	Image Classification
[4]	DNN, LSTM-RNN, DBN	Batch Size, Learning Rate, Type of Layer, Initialization function, Optimizer, Number of epochs, Dropout rate, Decay rate, Momentum.	PSO	Fixed-length binary encoding	Network Intrusion Detection
[5]	CNN	Dropout rate, Activation function, Kernel Initializer, Learning Rate, Batch Size.	PSO	Fixed-length hybrid binary encoding	Image Classification
[6]	CNN	Size of the mini-batch, dropout rate.	PSO	Fixed-length real encoding	Image Classification
[7]	CNN	Dropout rate, Dropout rate for lth block, Number of units in the dense layers, Momentum, Batch size, Number of epochs, Number of filters, Learning rate, Weight decay.	Fractal Decomposition Algorithm (FDA)	Fixed-length real encoding	Image Classification

SWARM AND EVOLUTIONARY ALGORITHM OPTIMIZED DL MODELS

Reference	Deep Learning Model	Hyper-Parameters	Optimization Technique	Encoding Scheme	Application
[8]	CNN	Learning Rate, Channels, Weight decay, Nesterov, Momentum, Optimizer, Batch Size, Number of epochs.	EA	Fixed-length real encoding	Image Classification
[9]	RNN	Activation function, Net input function, Learning Methods, Model of neuron, Number of neurons in output layer.	Genetic Algorithm (GA) + Gradient Descent Method	Fixed-length real encoding	Nuclear Reactor
[10]	CNN	Batch size, number of epochs, optimizer type, momentum rate, learning rate, dropout rate, activation function type, Number of convolutional layers, Number of filters, Convolution filter size, maxpooling size.	Cat Swarm Optimization (CSO)	Variable-length real encoding	Image Classification
[11]	CNN	Number of neurons in each convolutional layer, Number of convolutional layers.	PSO	Fixed-length binary encoding	Image Classification
[12]	CNN	Number of neurons in an Fully Connected layer, Pool type, Pool stride size, Pool kernel size, Convolution kernel size, Convolutional stride size, Length of CNN, number of feature maps.	Differential Evolution (DE)	Variable-length real encoding	Image Classification
[13]	CNN	Number of blocks, Number of nodes, Number of filters, Filter size, Pooling, Dropout, short connection, Batch normalization, Type of layer, Activation function, Dropout, Optimizer, Learning rate, Batch size, Initialization, Number of convolution layers, Activation function.	GA	Fixed-length Block-Based encoding	Image Classification
[14]	CNN	Dropout rate, Weight initialization method, Activation function, Kernel Initializer, Position of BN, BN Control parameter, No. of Hidden Nodes, No. of convolutional kernel	GA	Variable-length Block-Based encoding	Image Classification

SWARM AND EVOLUTIONARY ALGORITHM OPTIMIZED DL MODELS

Reference	Deep Learning Model	Hyper-Parameters	Optimization Technique	Encoding Scheme	Application
[15]	BiLSTM	No. of epochs of BP, learning rate, Number of iterations, maximum epochs of BiLSTM, Number of hidden neurons.	Improved Sine Cosine Algorithm (ISCA)	Fixed-length real encoding	Wind Speed Prediction
[16]	CNN/RNN/ GRU/LSTM	Network type, Weight value, Number of CNN layers, Kernel size of CNN, Whether bidirectional of the recurrent neural network, Number of layers of the recurrent neural network, Number of hidden nodes of the recurrent neural network.	GA	Variable-length real encoding	Time Series Forecasting
[17]	CNN	Learning Method, momentum rate, weight decay, drop path probability.	EA	Fixed-length real encoding	Image Classification
[18]	CNN	Batch size, epochs, SGD learning rate, momentum.	EA	Fixed-length binary encoding Variable-length real encoding	Image Classification
[19]	CNN	Learning rate for individual evaluation and fine-tuning, Number of epochs for individual evaluation and fine tuning.	EA	Variable-length binary encoding	Image Classification
[20]	CNN	Number of filters of Conv1D, Pool Size, Dropout rate, Optimizer.	EA	Variable-length real encoding	Image Classification

LIMITATION IN LITERATURE OF OPTIMIZING DL MODELS

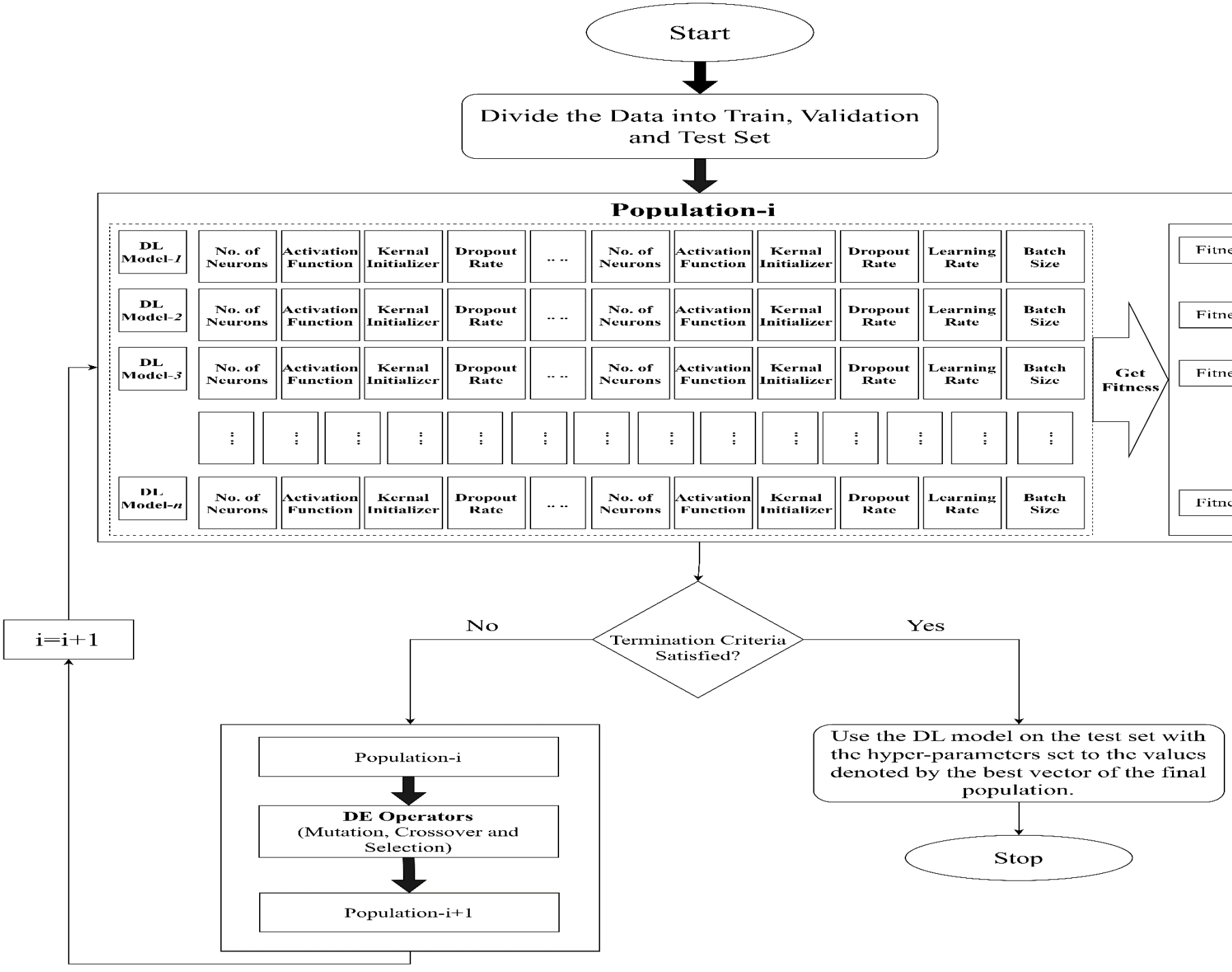
- **Hyper-parameters of DL models used for optimization**
  - Number of neurons in each layer [9, 11, 12, 15-16]
  - Activation function [1, 5, 9-10, 13-14]
  - Learning rate [4-5, 7-8, 10, 13, 15, 18-19]
  - Batch size [1-2, 4-5, 7-8, 10, 13, 18]
  - Dropout rate [1, 3-7, 10, 14, 20]
  - Kernel initializer (weight initialization scheme) [3, 5, 14, 16].
- **Encoding Scheme**
  - Fixed-length (**Drawback:** Can't encode DL models with varying layers and not flexible)
    - Binary [2, 4-5, 11, 18]
    - Real [6-9, 17-18]
    - Block [13]
  - Variable-length **(Drawback:** Complex implementation and high computational overhead**)**
    - Binary [19]
    - Real [1, 3, 10, 12, 16, 18, 20]
    - Block [14]
- **Number of Layers are not optimized**
- **Applicable to a specific DL model**

Overcoming the Limitations in Literature of Optimizing DL Models

- **Ref:** Sourav Kumar Purohit and Sibarama Panigrahi, Novel Deterministic and Probabilistic Forecasting Methods for Crude Oil Price Employing Optimized Deep Learning, Statistical and Hybrid Models. Information Sciences Journal, 120021, 2023. **(Elsevier, Impact Factor: 8.1)**

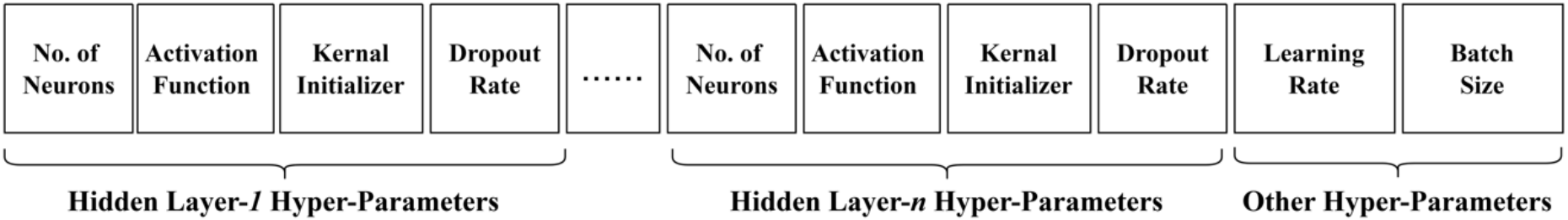
  DOI: https://doi.org/10.1016/j.ins.2023.120021

- **Hyper-parameters of DL models used for optimization**
  - Number of neurons in each layer
  - Activation function
  - Learning rate
  - Batch size
  - Dropout rate
  - Kernel initializer (weight initialization scheme)
- **Encoding Scheme**
  - Proposed fixed-length Real Encoding Scheme with pruning method which has the simplicity of fixed-length encoding scheme with flexibility of variable-length encoding scheme.
- **Optimization Algorithm:** Differential Evolution (DE)
- **Applicable to a optimize different DL model** **(LSTM, CNN, GRU, BiLSTM, ConvLSTM)**


Start
Divide the Data into Train, Validation and Test Set
Population-i
DL Model-1
DL Model-2
DL Model-3
DL Model-n
No. of Neurons
Activation Function
Kernal Initializer
Dropout Rate
.. ..
Learning Rate
Batch Size
Get Fitness
Fitness-1
Fitness-2
Fitness-3
Fitness-n
Termination Criteria Satisfied?
No
Yes
i=i+1
Population-i
DE Operators (Mutation, Crossover and Selection)
Population-i+1
Use the DL model on the test set with the hyper-parameters set to the values denoted by the best vector of the final population.
Stop

Optimizing DL Models using Swarm and Evolutionary Algos

- **Encoding a DL Model to a Chromosome** (C_i)



- **Objective Function:**

$$\min_{C_i} f(C_i)$$

$$\text{Subjected to } L_j \leq C_{i,j} \leq U_j$$

where $f(\cdot)$ *denotes the mean square error using the DL Model with hyper-parameters encoded in* C_i.

OPTIMIZING DL MODELS USING SWARM AND EVOLUTIONARY ALGOS

- **Decoding a Chromosome**

Algorithm 1

Get the activation function: get_activation_function(v).

Input: Decision variable v

Output: Activation function

```
1: gene← round (v)
2: if gene equals to 0 then
3:     return "relu"
4: else if gene equals to 1 then
5:     return "sigmoid"
6: else if gene equals to 2 then
7:     return "tanh"
8: else
9:     return "elu"
10: endif
```

OPTIMIZING DL MODELS USING SWARM AND EVOLUTIONARY ALGOS

- **Decoding a Chromosome**

Algorithm 2
Get the kernel initializer: get_kernel_initializer(v).

Input: Decision variable v
Output: Kernel Initializer

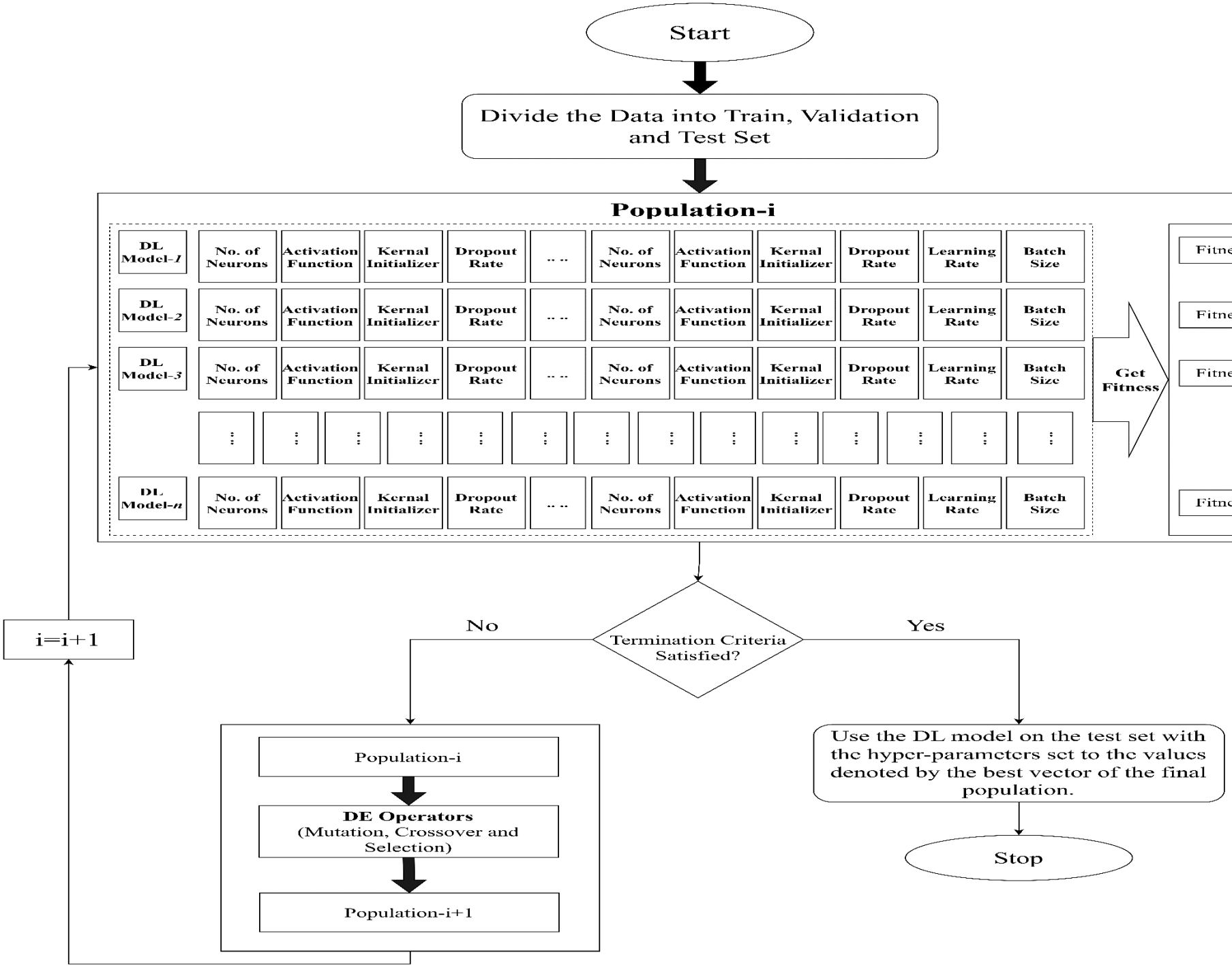
```
1: gene← round (v)
2: if gene equals to 0 then
3:     return "glorot_uniform"
4: else if gene equals to 1 then
5:     return "glorot_normal"
6: else if gene equals to 2 then
7:     return "he_uniform"
8: else
9:     return "he_normal"
10: endif
```

DECODING A CHROMOSOME (DECISION VECTOR) TO A DL MODEL

Algorithm 3
Decode a Decision Vector to a DL Model.

Input: Decision vector $V = [v_1, v_2, \ldots, v_d]$
Output: DL Model

1: model DL← empty
2: add layer input to DL
3: **for** each hidden layer hyper-parameter in decision vector V
4: **if** v_{i+4}not equal to 1 **then** // dropout !=1
5: nn=round(v_i) // number of neuron
6: af=get_activation_function(v_{i+1})
7: ki=get_kernel_initializer(v_{i+2})
8: **add** a layer to DL model with nn neurons, af as activation function, ki as kernel initializer and v_{i+4} as dropout rate.
9: **endif**
10: **endfor**
11: **add** a layer with 1 neuron and linear activation function. // Output layer
12: batch_size=$2^{round(v_d)}$
13: learning_rate=v_{d-1}
14: Compile the DL model with mean square error as the loss, adam as the optimizer with learning rate set to learning_rate.
15: **return** DL


Start
Divide the Data into Train, Validation and Test Set
Population-i
DL Model-1
DL Model-2
DL Model-3
DL Model-n
No. of Neurons
Activation Function
Kernal Initializer
Dropout Rate
.. ..
Learning Rate
Batch Size
Get Fitness
Fitness-1
Fitness-2
Fitness-3
Fitness-n
Termination Criteria Satisfied?
No
Yes
i=i+1
Population-i
DE Operators (Mutation, Crossover and Selection)
Population-i+1
Use the DL model on the test set with the hyper-parameters set to the values denoted by the best vector of the final population.
Stop

Time Series Forecasting using Optimized Deep Learning Models

The number of significant inputs *k* of DL models will be determined by analyzing the autocorrelation and partial autocorrelation function of the time series.

The time series will be pre-processed (Normalization, Treatment of Trend and Seasonal Components).

The time series of length *n* will be transformed to *n-k* patterns using sliding window technique. Then the patterns will be splitted into Train, Validation and Test Sets.

Using the Train and Validation set determine the DL model parameters. Once the model parameters are determined, the forecasts on Test set are computed using the obtained model parameters.

Denormalize, Detrend and Deseasonalize the computed forecasts to obtain the true forecasts.

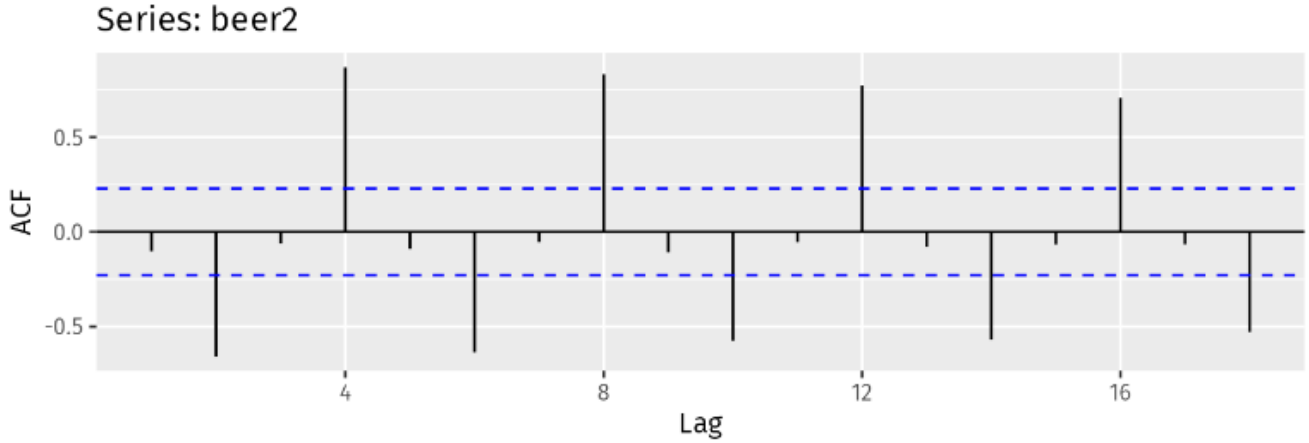
Measure the forecasting accuracy.

Time Series Forecasting using Optimized Deep Learning Models

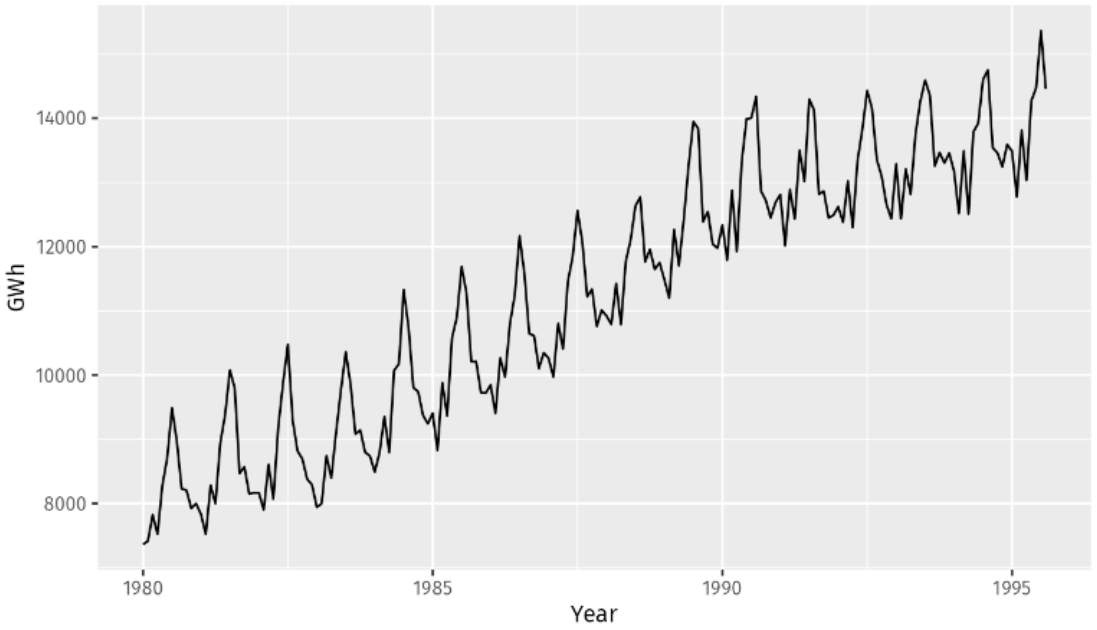
- Just as correlation measures the extent of a linear relationship between two variables, autocorrelation measures the linear relationship between *lagged values* of a time series.

- There are several autocorrelation coefficients, corresponding to each panel in the lag plot. For example, r_1 measures the relationship between y_t and y_{t-1}, r_2 measures the relationship between y_t and y_{t-2}, and so on.

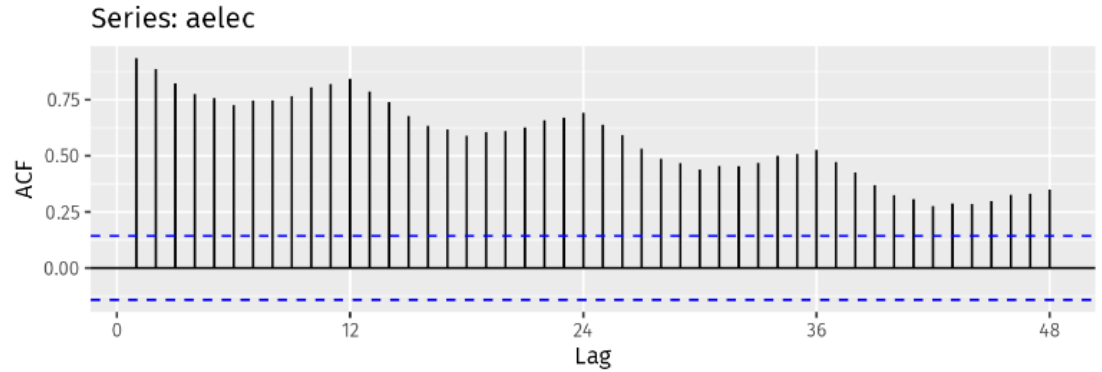
- The value of r_k can be written as $r_k = \frac{\sum_{i=k+1}^{n}(y_i-\bar{y})(y_{i-k}-\bar{y})}{\sum_{i=k+1}^{n}(y_i-\bar{y})^2}$



Time Series Forecasting using Optimized Deep Learning Models



Monthly Australian electricity demand from 1980–1995.



ACF of monthly Australian electricity demand.

Time Series Forecasting using Optimized Deep Learning Models

The number of significant inputs *k* of DL models will be determined by analyzing the autocorrelation and partial autocorrelation function of the time series.

The time series will be pre-processed (Normalization, Treatment of Trend and Seasonal Components).

The time series of length *n* will be transformed to *n-k* patterns using sliding window technique. Then the patterns will be splitted into Train, Validation and Test Sets.

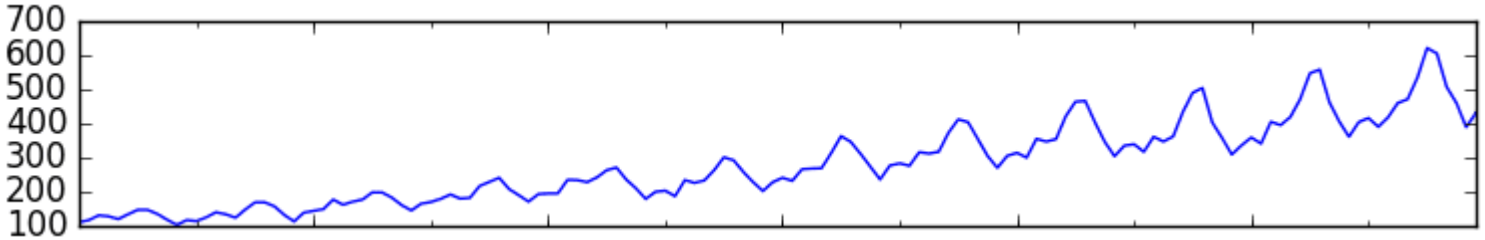
Using the Train and Validation set determine the DL model parameters. Once the model parameters are determined, the forecasts on Test set are computed using the obtained model parameters.

Denormalize, Detrend and Deseasonalize the computed forecasts to obtain the true forecasts.

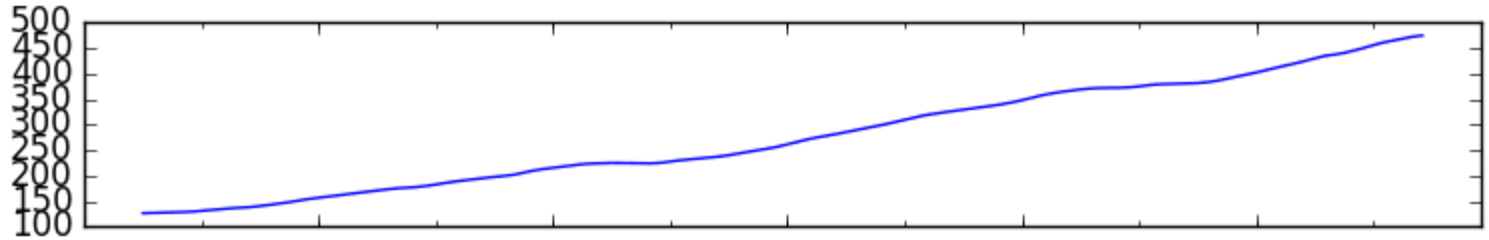
Measure the forecasting accuracy.

Motivation for Hybrid Models

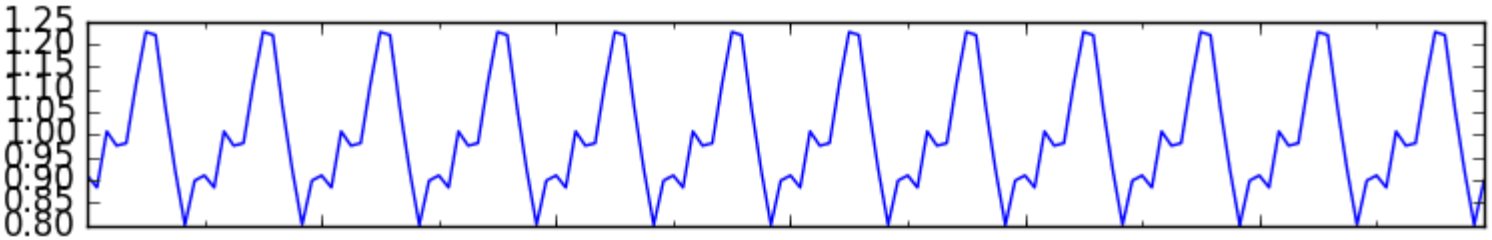
Real-world time series may not be purely linear or purely nonlinear. Rather it contains a combination of linear and nonlinear components.



Original Time Series



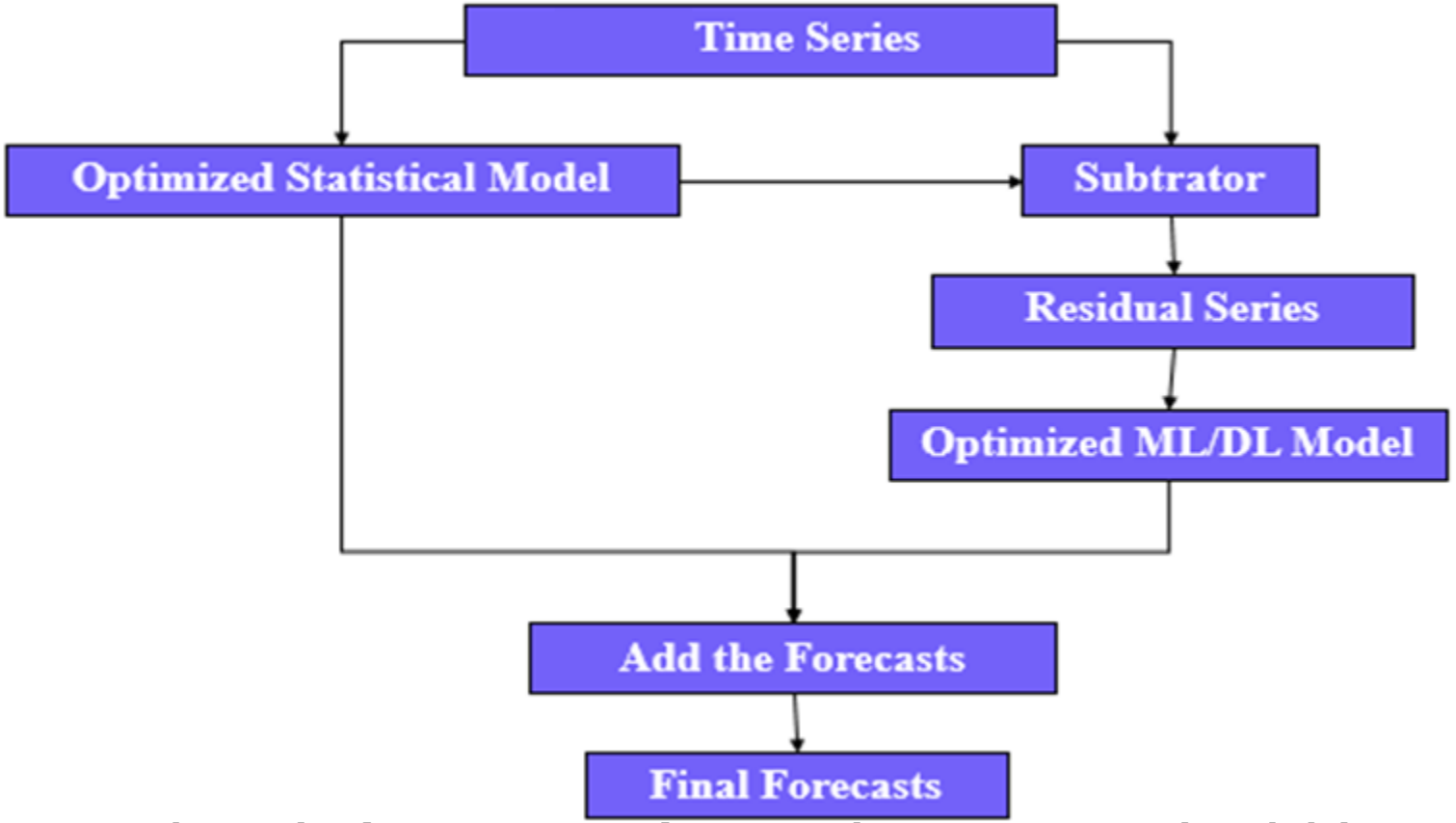
Linear Component



Nonlinear Component

Time Series Forecasting using Optimized Series Hybrid Models

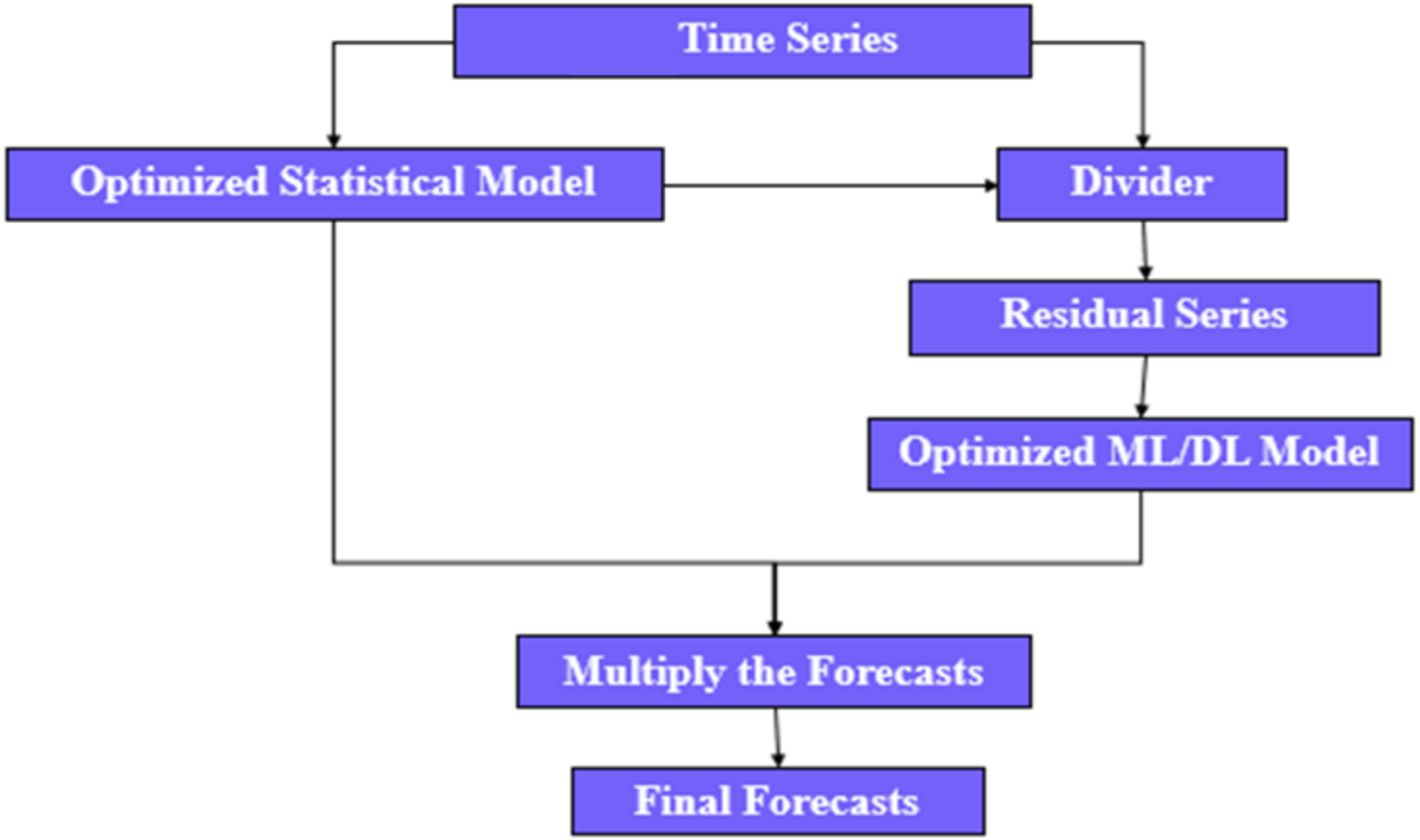
Additive Hybrid Model



Sourav Kumar Purohit, and **Sibarama Panigrahi***. "Novel Deterministic and Probabilistic Forecasting Methods for Crude Oil Price employing Optimized Deep Learning, Statistical and Hybrid Models." *Information Sciences*, vol. 658, 120021 (2024). **[Elsevier, IF:8.1, SCI]**

Time Series Forecasting using Optimized Series Hybrid Models

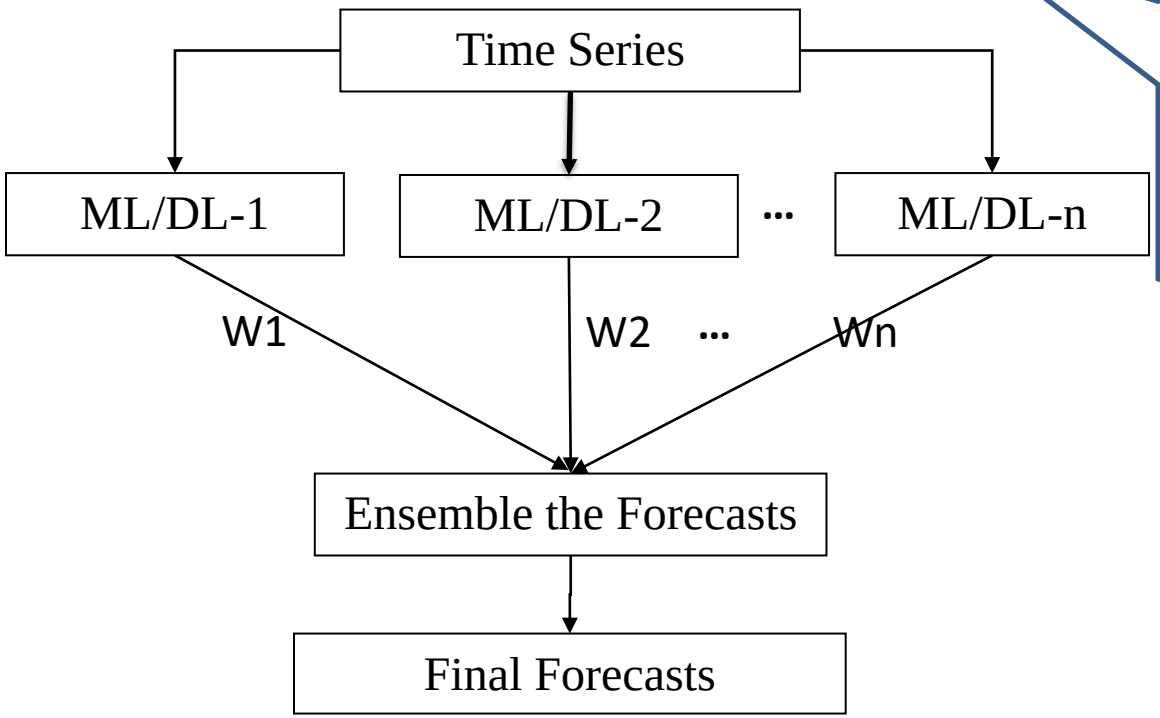
Multiplicative Hybrid Model



Sourav Kumar Purohit, and **Sibarama Panigrahi***. "Novel Deterministic and Probabilistic Forecasting Methods for Crude Oil Price employing Optimized Deep Learning, Statistical and Hybrid Models." *Information Sciences*, vol. 658, 120021 (2024). **[Elsevier, IF:8.1, SCI]**

Time Series Forecasting using Optimized Parallel Hybrid Models

Parallel Hybrid Model



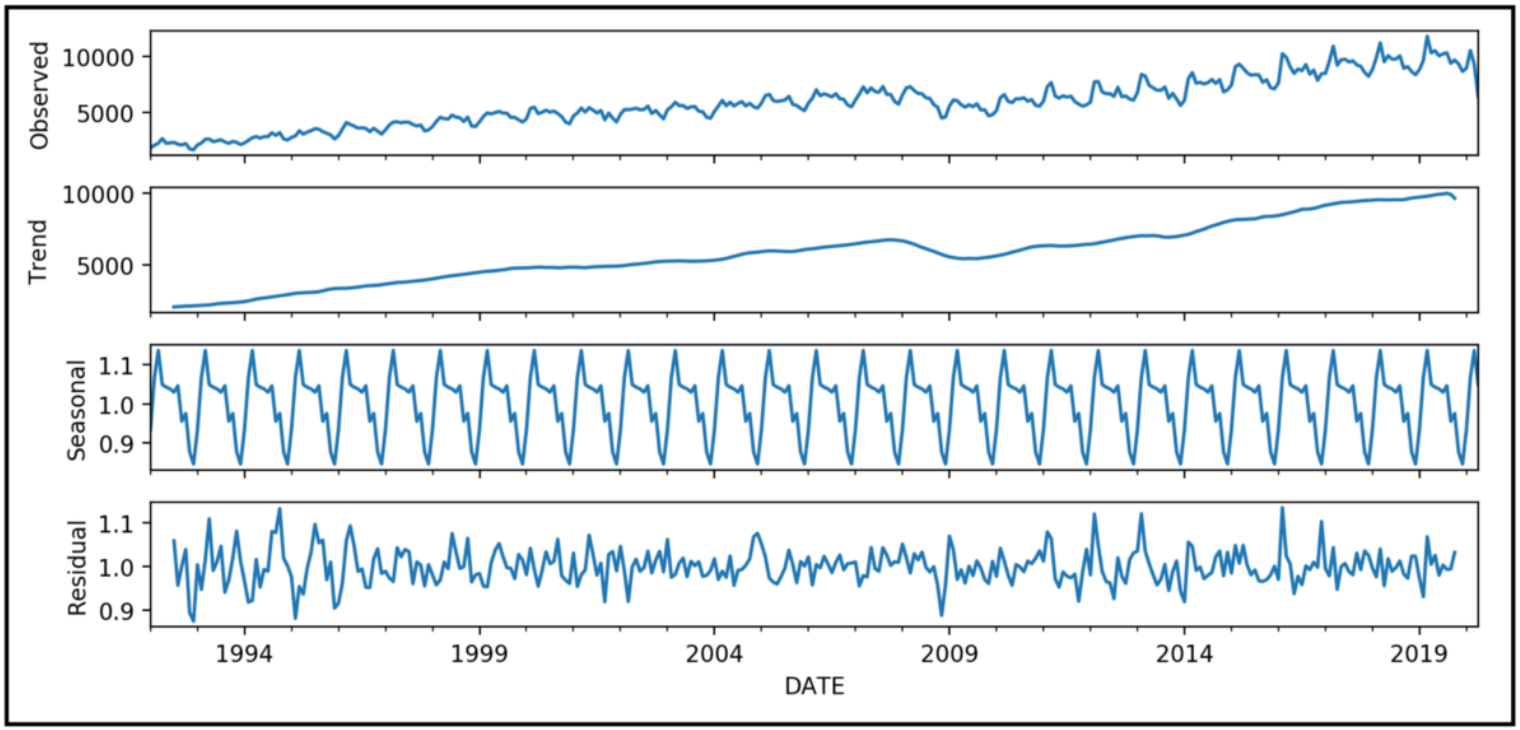
1. Equal Weight
2. Variable Weight
3. Time-Varying Weight

Motivation for Decomposition Based Hybrid Models

Blind application of a linear (statistical) or nonlinear (ML/DL) model to a time series containing both linear and/or nonlinear patterns may not be appropriate.

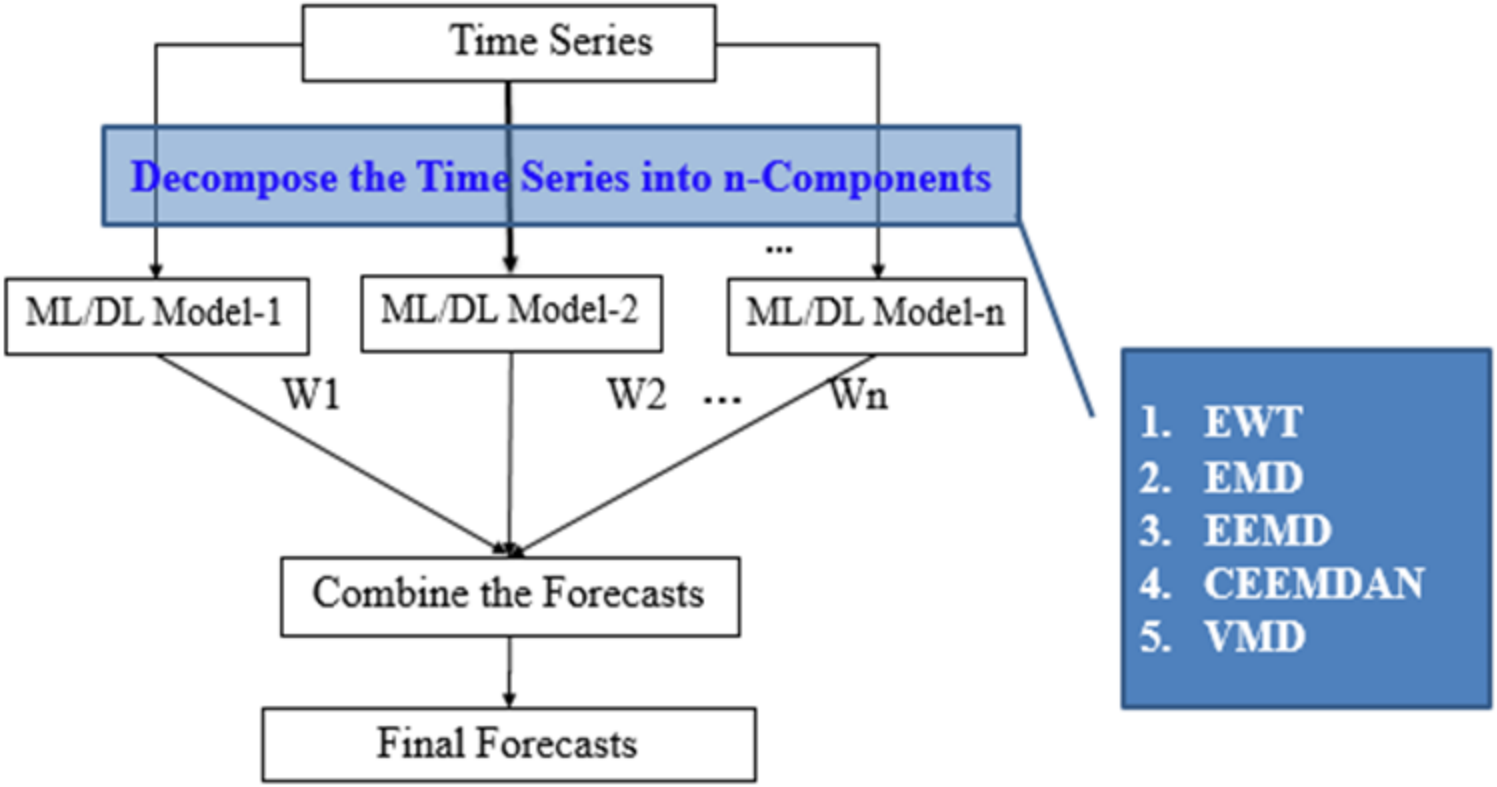
First Decompose the time series and then model individual components based on its characteristics.

Decomposition can also be based on Error(Residual), Trend and Seasonal Components. It may be again additive or multiplicative.



Time Series Forecasting using Optimized Decomposition Based Hybrid Model

Decomposition Based Hybrid Model



Published Papers

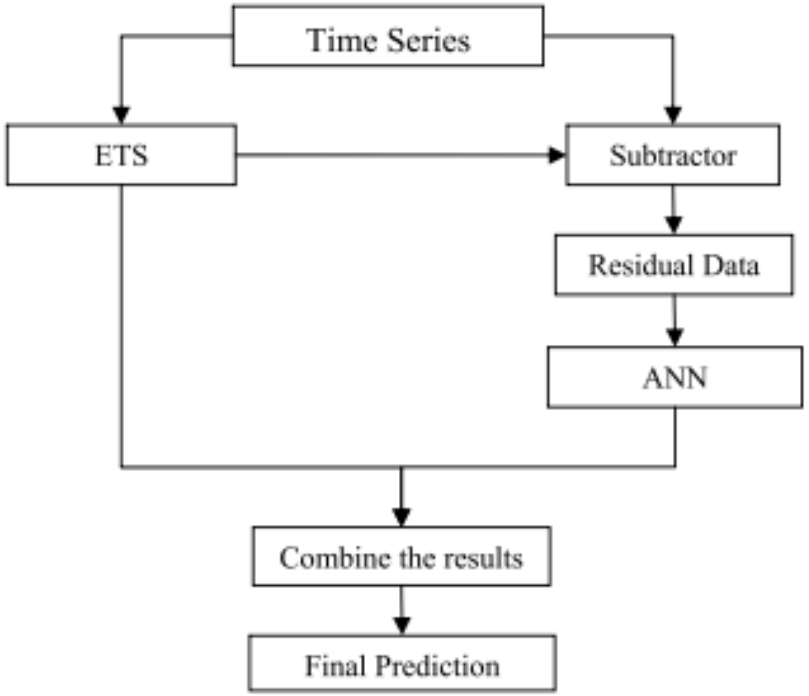
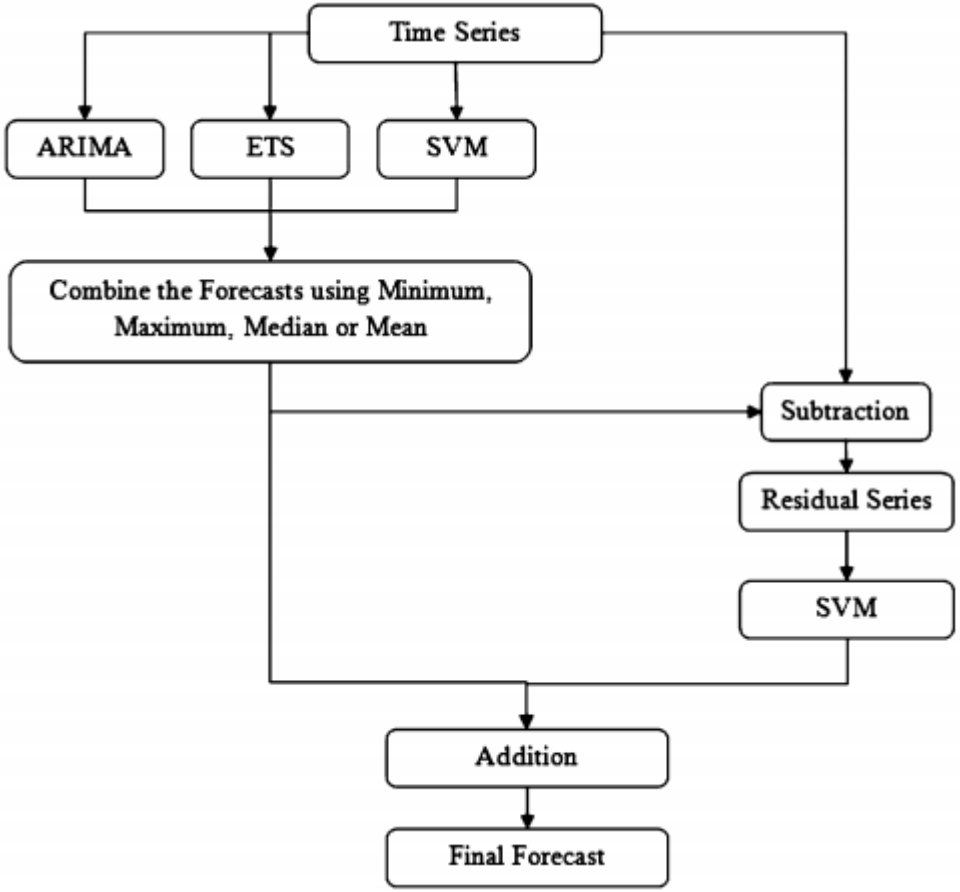


Fig. 2. Proposed methodology.

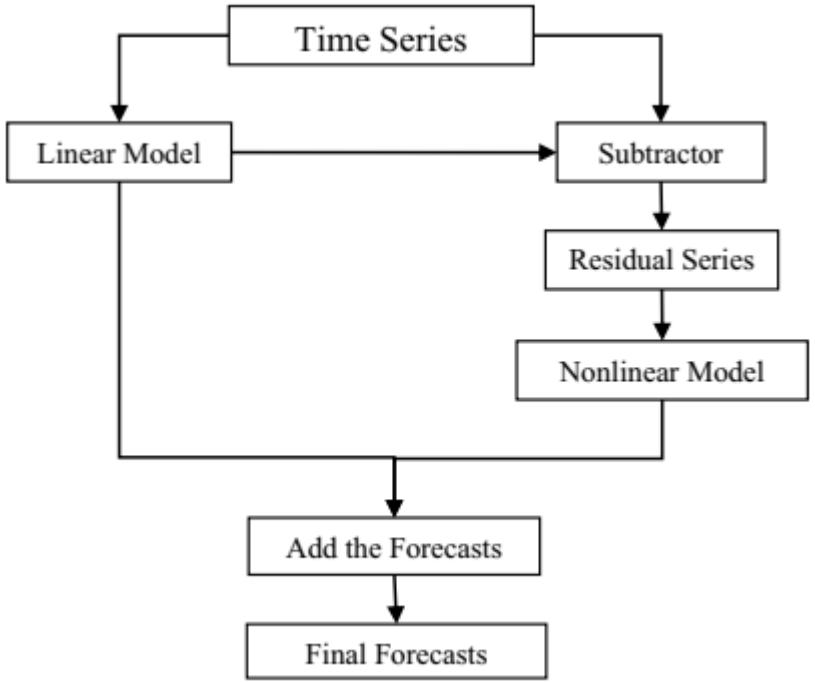
Sibarama Panigrahi, and Himansu Sekhar Behera. "A hybrid ETS–ANN model for time series forecasting." Engineering Applications of Artificial Intelligence 66 (2017): 49-59. ISSN: 0952-1976 **[Elsevier, IF:7.80, SCI]**

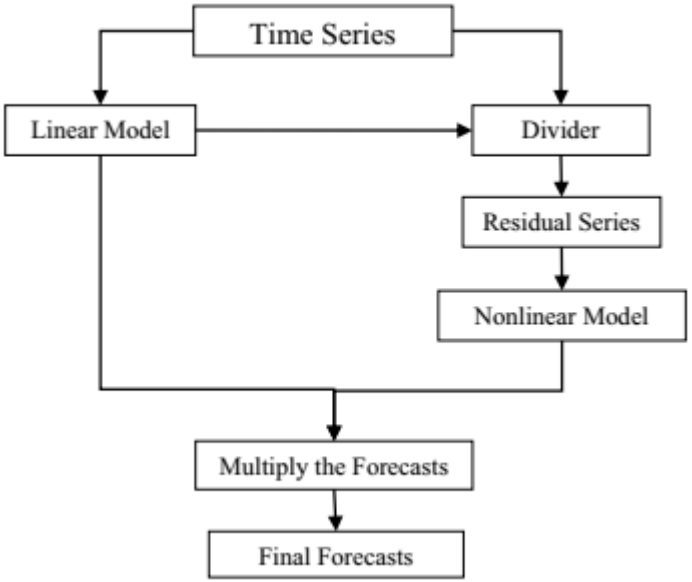
Published Papers



Sibarama Panigrahi*, Radha Mohan Pattanayak, Prabira Kumar Sethy, Santi Kumari Behera. "Forecasting of Sunspot Time Series Using a Hybridization of ARIMA, ETS and SVM Methods." Solar Physics 296.1 (2021): 1-19. **[Springer, IF:2.96, SCI]**

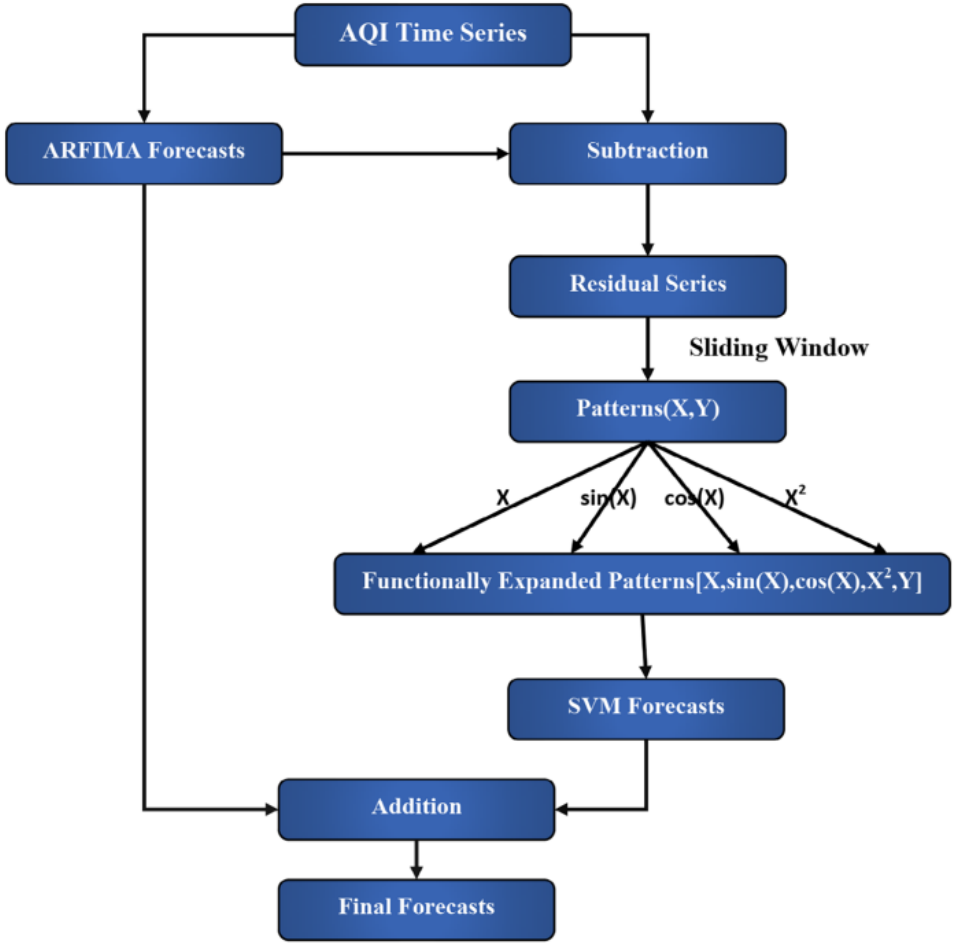
Published Papers





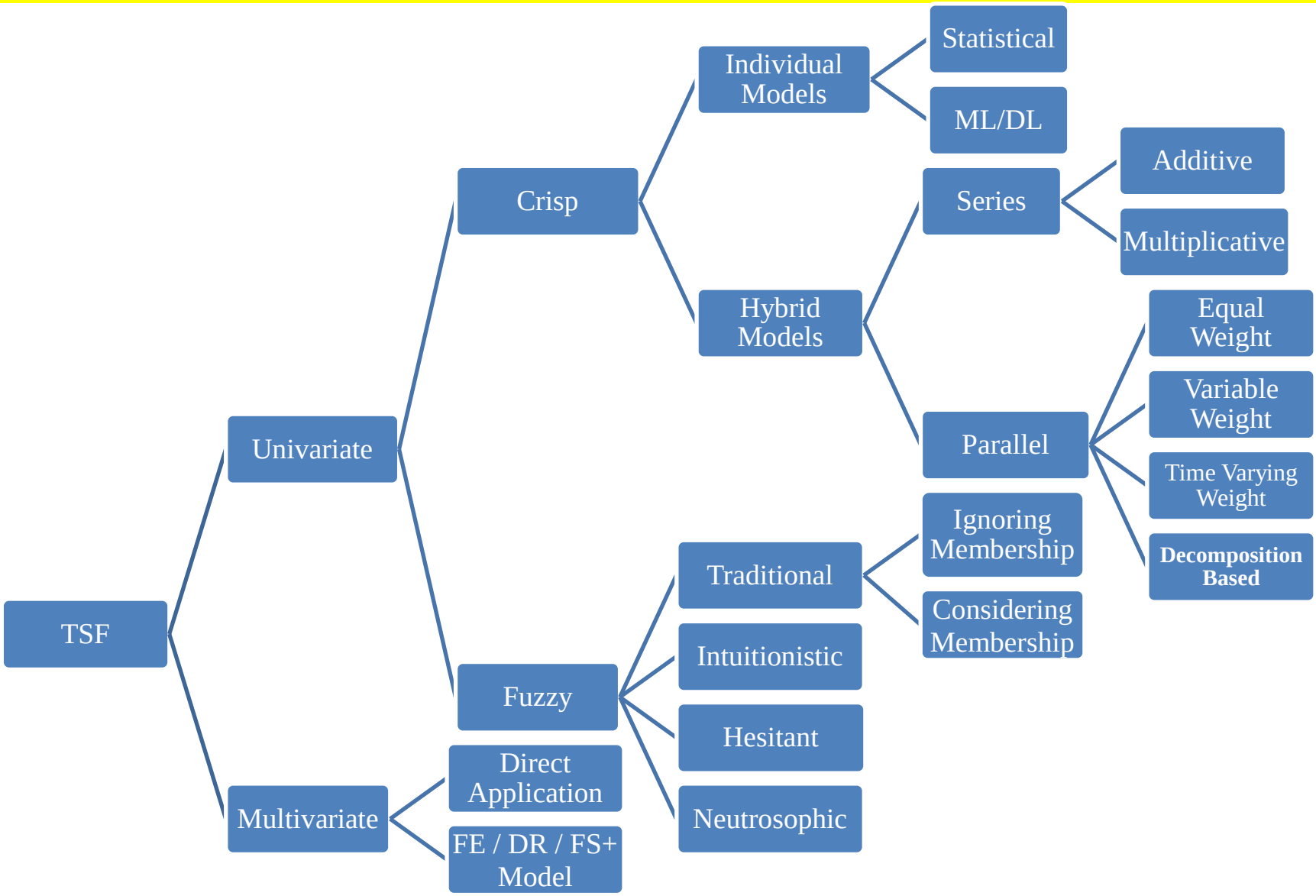
Sourav Kumar Purohit, **Sibarama Panigrahi***, Prabira Kumar Sethy, and Santi Kumari Behera. "Time series forecasting of price of agricultural products using hybrid methods." Applied Artificial Intelligence (2021), vol. 35, no. 15, pp. 1388-1406. **[Taylor & Francis, IF:2.78, SCI]**

Published Papers



S. S. Pradhan, **Sibarama Panigrahi***, S. K. Purohit, and J. Dash. "Study and development of hybrid and ensemble forecasting models for air quality index forecasting." Expert Syatems (2023). **[Wiley, IF: 3.3, SCI]**

Time Series Forecasting (TSF) Techniques [Classification]



REFERENCES

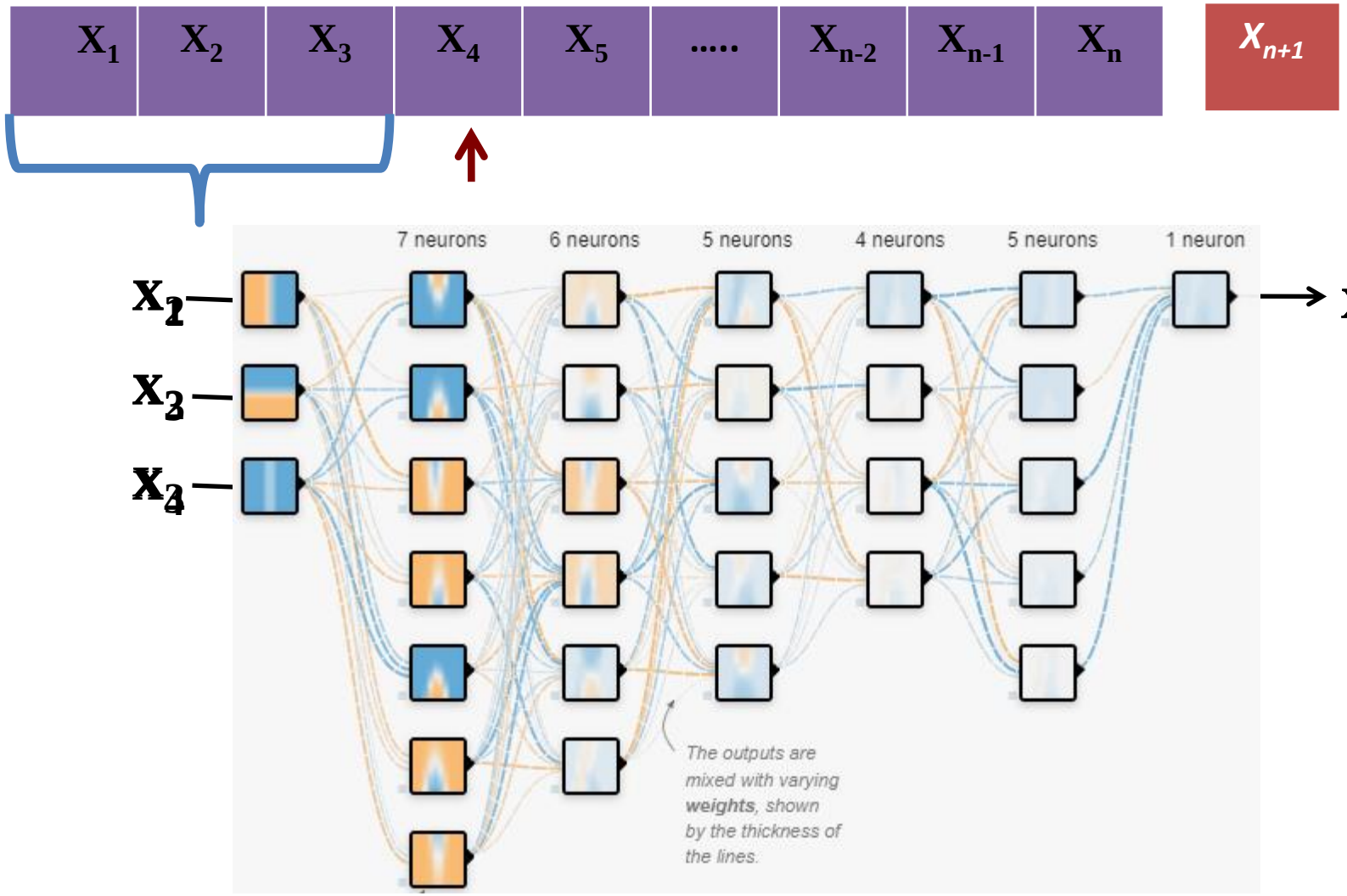
1. Martín, Alejandro, Raúl Lara-Cabrera, Félix Fuentes-Hurtado, Valery Naranjo, and David Camacho. "Evodeep: a new evolutionary approach for automatic deep neural networks parametrization." Journal of Parallel and Distributed Computing 117 (2018): 180-191.
2. Li, Y., Xiao, J., Chen, Y., & Jiao, L. (2019). Evolving deep convolutional neural networks by quantum behaved particle swarm optimization with binary encoding for image classification. Neurocomputing, 362, 156-165.
3. Junior, F. E. F., & Yen, G. G. (2019). Particle swarm optimization of deep neural networks architectures for image classification. Swarm and Evolutionary Computation, 49, 62-74.
4. Elmasry, W., Akbulut, A., & Zaim, A. H. (2020). Evolving deep learning architectures for network intrusion detection using a double PSO metaheuristic. Computer Networks, 168, 107042..
5. Jiang, J., Han, F., Ling, Q., Wang, J., Li, T., & Han, H. (2020). Efficient network architecture search via multiobjective particle swarm optimization based on Decomposition. Neural Networks, 123, 305-316.
6. Darwish, A., Ezzat, D., & Hassanien, A. E. (2020). An optimized model based on convolutional neural networks and orthogonal learning particle swarm optimization algorithm for plant diseases diagnosis. Swarm and evolutionary computation, 52, 100616.
7. Souquet, L., Shvai, N., Llanza, A., & Nakib, A. (2023). Convolutional neural network architecture search based on fractal decomposition optimization algorithm. Expert Systems with Applications, 213, 118947.
8. Wen, L., Gao, L., Li, X., & Li, H. (2022). A new genetic algorithm based evolutionary neural architecture search for image classification. Swarm and Evolutionary Computation, 75, 101191.
9. Laddach, K., Łangowski, R., Rutkowski, T. A., & Puchalski, B. (2022). An automatic selection of optimal recurrent neural network architecture for processes dynamics modelling purposes. Applied Soft Computing, 116, 108375.
10. Jalali, S. M. J., Ahmadian, M., Ahmadian, S., Hedjam, R., Khosravi, A., & Nahavandi, S. (2022). X-ray image based COVID-19 detection using evolutionary deep learning approach. Expert Systems with Applications, 201, 116942.

REFERENCES

11. Nistor, S. C., & Czibula, G. (2022). IntelliSwAS: Optimizing deep neural network architectures using a particle swarm-based approach. Expert Systems with Applications, 187, 115945.
12. Ghosh, A., Jana, N. D., Mallik, S., & Zhao, Z. (2022). Designing optimal convolutional neural network architecture using differential evolution algorithm. Patterns, 3(9), 100567.
13. Hassanzadeh, T., Essam, D., & Sarker, R. (2022). EvoDCNN: An evolutionary deep convolutional neural network for image classification. Neurocomputing, 488, 271-283.
14. Shang, R., Zhu, S., Ren, J., Liu, H., & Jiao, L. (2022). Evolutionary neural architecture search based on evaluation correction and functional units. Knowledge-Based Systems, 109206.
15. Zhang, C., Ma, H., Hua, L., Sun, W., Nazir, M. S., & Peng, T. (2022). An evolutionary deep learning model based on TVFEMD, improved sine cosine algorithm, CNN and BiLSTM for wind speed prediction. Energy, 124250.
16. Li, Y., Liu, J., & Teng, Y. (2022). A decomposition-based memetic neural architecture search algorithm for univariate time series forecasting. Applied Soft Computing, 130, 109714.
17. He, C., Tan, H., Huang, S., & Cheng, R. (2021). Efficient evolutionary neural architecture search by modular inheritable crossover. Swarm and Evolutionary Computation, 64, 100894.
18. Louati, H., Bechikh, S., Louati, A., Hung, C. C., & Said, L. B. (2021). Deep convolutional neural network architecture design as a bi-level optimization problem. Neurocomputing, 439, 44-62.
19. Fernandes Jr, F. E., & Yen, G. G. (2021). Pruning deep convolutional neural networks architectures with evolution strategy. Information Sciences, 552, 29-47.
20. Samir, A. A., Rashwan, A. R., Sallam, K. M., Chakrabortty, R. K., Ryan, M. J., & Abohany, A. A. (2021). Evolutionary algorithm-based convolutional neural network for predicting heart diseases. Computers & Industrial Engineering, 161, 107651.

For Your Valuable Time.

How to use Deep Learning Models in TSF



INTRODUCTION

- **Point Forecasting Accuracy Measures**

Error-metric	Accuracy Measure	Formula
Scale-dependent	Mean Absolute Error (MAE) or Mean Absolute Deviation (MAD)	$\frac{1}{n}\sum_{i=1}^{n}\lvert y_i - \hat{y}_i\rvert$
	Geometric Mean Absolute Error (GMAE)	$\left(\prod_{i=1}^{n}\lvert y_i - \hat{y}_i\rvert\right)^{\frac{1}{n}}$
	Mean Square Error (MSE)	$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
	Root Mean Square Error (RMSE)	$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$
Percentage	Mean Absolute Percentage Error(MAPE)	$\frac{1}{n}\sum_{i=1}^{n}\frac{\lvert y_i - \hat{y}_i\rvert}{y_i} \times 100$
	Symmetric Mean Absolute Percentage Error(SMAPE)	$\frac{1}{n}\sum_{i=1}^{n}\frac{\lvert y_i - \hat{y}_i\rvert}{(\lvert y_i\rvert + \lvert\hat{y}_i\rvert)/2}$
Relative	Median Relative Absolute Error (MdRAE)	$median\left(\left\lvert\frac{y_i - \hat{y}_i}{y_i - \hat{y}_i^*}\right\rvert\right)$
	Geometric Mean Relative Absolute Error (GMRAE)	$\left(\prod_{i=1}^{n}\left\lvert\frac{y_i - \hat{y}_i}{y_i - \hat{y}_i^*}\right\rvert\right)^{\frac{1}{n}}$
Scale-free	Mean Absolute Scaled Error (MASE)	$\frac{1}{n}\sum_{i=1}^{n}\frac{\lvert y_i - \hat{y}_i\rvert}{\frac{1}{n-1}\sum_{j=2}^{n}\lvert y_i - y_{i-1}\rvert}$

where y_i and $\hat{y}_i$ denote the ith actual and forecasted value, n denotes the number of observations. **Lower the value better the forecasts.**

Time Series Forecasting using Optimized Deep Learning Models

The number of significant inputs *k* of DL models will be determined by analyzing the autocorrelation and partial autocorrelation function of the time series.

The time series will be pre-processed (Normalization, Treatment of Trend and Seasonal Components).

The time series of length *n* will be transformed to *n-k* patterns using sliding window technique. Then the patterns will be splitted into Train, Validation and Test Sets.

Using the Train and Validation set determine the DL model parameters. Once the model parameters are determined, the forecasts on Test set are computed using the obtained model parameters.

Denormalize, Detrend and Deseasonalize the computed forecasts to obtain the true forecasts.

Measure the forecasting accuracy.