PHM-MAX-XGBoost-LSTM-SVM: A Hybrid Model for Time Series Forecasting – A Comparative Analysis with Traditional and Modern Approaches

Bishwajit Prasad Gond
Department of Computer Science and Engineering,
National Institute of Technology Rourkela, Odisha, India
0000-0003-3640-0463

***Abstract*—This paper presents a comparative analysis of the PHM-MAX-XGBoost-LSTM-SVM hybrid model for time series forecasting, evaluated against traditional statistical, machine learning, and deep learning methods. It addresses key challenges in time series data, including noise, missing values, and non-stationarity, and explores preprocessing techniques such as data cleaning, transformation, and dimensionality reduction, with mathematical foundations and domain-specific relevance. The study benchmarks the hybrid model on various time series types—univariate, multivariate, seasonal, and irregular—across sectors like finance, energy, and healthcare. Strengths and limitations of each approach are discussed, with an emphasis on model performance, adaptability, and interpretability. Results highlight the potential of hybrid architectures to outperform standalone models. Future research directions include enhancing preprocessing integration, handling sparse data, and advancing real-time forecasting capabilities**

***Index Terms*—Time Series, Data Preprocessing, Forecasting Models, Predictive Analytics, Statistical Models, Machine Learning, Deep Learning**

I. INTRODUCTION

Time series data, characterized by sequential observations over time, is prevalent in domains such as finance (e.g., stock prices), energy (e.g., load forecasting), healthcare (e.g., patient monitoring), and meteorology (e.g., weather prediction). Accurate forecasting relies on effective preprocessing to handle issues like noise, missing values, and non-stationarity, followed by robust modeling techniques. This review synthesizes preprocessing methods and forecasting models, focusing on their applicability to diverse time series data types.

The paper is structured as follows: Section II discusses the basic concepts related to time series. Section III discusses the different types of time series. Section IV discusses preprocessing techniques with mathematical foundations. Section V reviews forecasting models and their applications. Section VI discusses the proposed architecture. Section VII compares forecasting techniques. Section VIII concludes with future research directions.

II. BASIC CONCEPTS

This section introduces the fundamental concepts of time series analysis, which are essential for understanding and modeling time-dependent data.

A. Trend

The trend represents the long-term movement or direction in a time series, indicating a consistent increase, decrease, or stability over time.

- **Definition**: A persistent upward or downward pattern in the data over an extended period.
- **Examples**: Population growth, economic expansion, or gradual climate changes.
- **Analysis**: Trends are often modeled using linear regression, moving averages, or polynomial fits.

B. Seasonality

Seasonality refers to regular, predictable patterns that repeat at fixed intervals, often tied to calendar periods.

- **Definition**: Periodic fluctuations in a time series that occur at consistent intervals, such as daily, monthly, or yearly.
- **Examples**: Retail sales peaking during holidays, temperature variations across seasons.
- **Analysis**: Seasonal components can be isolated using decomposition methods or modeled with seasonal ARIMA (SARIMA).

C. Cyclicality

Cyclicality involves fluctuations that occur over longer, irregular periods, often driven by economic or external factors.

- **Definition**: Long-term oscillations that are not fixed in duration, typically spanning multiple years.
- **Examples**: Business cycles with periods of growth and recession.
- **Analysis**: Cyclical patterns are analyzed using spectral analysis or econometric models.

D. Stationarity

Stationarity is a property of time series where statistical characteristics remain constant over time.

- **Definition**: A time series is stationary if its mean, variance, and autocorrelation structure do not change over time.
- **Types**: Strict stationarity (all statistical properties constant) and weak stationarity (constant mean and variance).

- **Importance**: Many forecasting models, like ARIMA, assume stationarity, achieved via differencing or transformations.

E. Noise

Noise represents random, unpredictable variations in a time series that cannot be attributed to trend, seasonality, or cyclicality.

- **Definition**: Irregular, stochastic fluctuations often modeled as white noise with zero mean and constant variance.
- **Examples**: Random market fluctuations, measurement errors.
- **Analysis**: Noise is typically modeled as the residual component after extracting trend, seasonality, and cyclical effects.

III. TYPES OF TIME SERIES

Types of Time Series by Variable Count

A. Univariate Time Series

A univariate time series involves a single variable observed over time. For example, monthly temperature readings in a city represent a univariate series. Analysis focuses on identifying patterns such as trends, seasonality, or cycles within this single variable. Univariate models, such as ARIMA, are commonly used for forecasting [1].

B. Multivariate Time Series

In contrast, a multivariate time series involves multiple variables observed simultaneously over time. An example is the daily recording of temperature, humidity, and wind speed. These series require models that account for interdependencies between variables, such as Vector Autoregression (VAR) models [2].

Types of Time Series by Statistical Properties

C. Stationary Time Series

A stationary time series exhibits a constant mean, variance, and autocorrelation over time. White noise constitutes an ideal stationary example. Any differenced economic indicator is also a stationary time series. Modelling stationary series is relatively easy due to these properties [3].

D. Non-stationary Time Series

Non-stationary time series feature statistical parameters that vary over time, mainly due to underlying trends or seasonal effects. Stock prices, for instance, are examples. Applying a transformation to a time series, usually differencing or detrending, makes it stationary so it can be studied further [1].

E. Seasonal Time Series

Seasonal time series exhibit repeated patterns at precise time intervals, either daily, monthly, or yearly. Retail sales would be a good example since they peak during the holidays. Seasonal differencing or decomposition methods can be implemented in such cases [2].

F. Cyclic Time Series

Cyclic time series show patterns that repeat at irregular intervals, often tied to economic or business cycles. For instance, business cycles with booms and recessions fall into this category. Unlike seasonal series, the periodicity is not fixed [3].

Types of Time Series by Data Collection

G. Continuous Time Series

Continuous time series are recorded continuously over time, often at fine intervals. Examples include heart rate monitoring or real-time sensor data. These series typically require signal processing techniques for analysis [2].

H. Discrete Time Series

Discrete time series are recorded at fixed, discrete intervals, such as hourly or daily. Daily stock prices or monthly unemployment rates are examples. These are the most common in statistical modeling [1].

I. Panel Time Series

Panel time series combine data across multiple entities over time, such as GDP across several countries. Analysis must account for both temporal and cross-sectional variations, often using specialized econometric models [3].

HIERARCHICAL TIME SERIES

Hierarchical time series involve data organized in a hierarchical structure, where observations at different levels (e.g., total, regional, or individual) are aggregated or disaggregated. These are common in business and retail, such as sales data across countries, regions, and stores. Two primary approaches for forecasting hierarchical time series are bottom-up and top-down [4].

J. Bottom-Up Approach

In the bottom-up approach, forecasts are first generated for the lowest level of the hierarchy (e.g., individual stores). These forecasts are then aggregated to produce forecasts for higher levels (e.g., regions or total sales). This method preserves detailed information at the lowest level but may struggle with noise in disaggregated data [4].

K. Top-Down Approach

The top-down approach starts with forecasting at the highest level of the hierarchy (say, total sales), with higher-level forecasts being disaggregated into lower levels based on historical proportions or any other allocation methods. This is a more straightforward forecasting method and will be less noisy, but it will potentially sacrifice accuracy at lower levels because of forecasting on aggregated data [4].

FUZZY TIME SERIES

Fuzzy time series methods extend the traditional time series method by involving fuzzy set theory to handle uncertainty and imprecision in the data. Unlike crisp time series where values are exact numbers, the fuzzy time series assign observations-fuzzy sets acting as linguistic variables (e.g., "high" or "low"). For example, if some observations of a system were vague or incomplete in finance or weather forecasting, the method would prove beneficial. The Forecasting process includes defining fuzzy sets and relations and the defuzzification of results to yield crisp forecasts [5].

IV. DATA PREPROCESSING TECHNIQUES

Preprocessing ensures high-quality time series data by removing noise, filling up missing data, and addressing issues such as non-stationarity [6]. This section deals with cleaning, transformation, and reduction methods. Due to size constraints, some data preprocessing techniques are given in Appendix A.

A. Data Cleaning

Another crucial preprocessing step in data analysis is data cleaning, which includes various procedures that upgrade the data quality and badly influence the analysis. Noise constitutes random errors or unwanted fluctuations in the data. In contrast, outliers are the extreme points of data that are away from probable patterns on account of times arising from erroneous measurements, mistakes in data entry, or uncommon incidences. Missing values exist when information is incomplete, which may influence dependent analysis if it is not treated accordingly. Cleaning trains the data for implementing any further analysis, such as time series forecasting, wherein dirty data will reduce the efficiency and robustness of the model. The following sections present the noise and outlier removal techniques, with a slight emphasis on their methods, application areas, and drawbacks.

1) Box Plot Method: **Description and Methodology**: The box plot method works to identify outliers and was devised by Tukey (1977), with outliers identified by the interquartile range (IQR). IQR denotes the difference between third quartile (Q_3) and first quartile (Q_1) values, i.e., IQR $= Q_3 - Q_1$. Outliers are considered every data point falling outside of

$$[Q_1 - 1.5 \times \text{IQR}, Q_3 + 1.5 \times \text{IQR}]. \tag{1}$$

Data points identified outside the expected range are potential outliers, which may be removed or flagged for further investigation. The box plot method constructs a box plot, where the "box" shows the interquartile range (IQR), and a line inside it represents the median; whiskers extend to $Q1 - 1.5 \times$ IQR and $Q3 + 1.5 \times$ IQR.

Applications: This method is most suitable for datasets from the financial sector, such as stock prices and trading volumes, where data distributions tend to be skewed but can still benefit from the quartile-based analysis. It is the most famous method in practical applications due to its straightforward interpretation, thus forming its presence in exploratory data analysis.

Limitations: The box plot method assumes a relatively symmetric data distribution, which may not hold for highly skewed or multimodal datasets. Additionally, the fixed multiplier (1.5) may not be optimal for all datasets, potentially leading to false positives or missed outliers in complex time series [7].

2) Z-Score Method: **Description and Methodology**: The Z-score method identifies outliers by measuring how far a data point deviates from the mean in terms of standard deviations. The Z-score for a data point x is calculated as:

$$Z = \frac{x - \mu}{\sigma}, \tag{2}$$

where μ is the mean and σ is the standard deviation of the dataset. Data points with $|Z| > 3$ are typically flagged as outliers, as they lie beyond three standard deviations from the mean, corresponding to the tails of a normal distribution where less than 0.3% of data points are expected.

Applications: This method is effective for datasets that approximate a normal distribution, such as temperature readings or sensor data in environmental monitoring. It is computationally efficient and widely used in statistical quality control and anomaly detection.

Limitations: The Z-score method assumes that the data follows a normal distribution, which may not be true for non-Gaussian or time-varying data. In such cases, the method may incorrectly flag valid data points as outliers or fail to detect true anomalies. Additionally, it is sensitive to the presence of outliers in the dataset, as they can inflate the standard deviation, reducing the method's effectiveness [8].

3) Moving Interquartile Range (MIQR) Method: **Description and Methodology**: The Moving Interquartile Range (MIQR) method adapts the box plot approach to non-stationary time series by applying a sliding window to compute local IQR thresholds. For each window of data, the IQR is calculated as IQR $= Q3 - Q1$, and outliers are identified as points outside the range $[Q1 - 1.5 \times \text{IQR}, Q3 + 1.5 \times \text{IQR}]$. The window slides across the time series, allowing the method to adapt to local trends and variations in the data.

Applications: MIQR is particularly suited for non-stationary datasets, such as energy consumption or network traffic, where statistical properties change over time. By using a sliding window, MIQR captures local patterns, making it more robust to trends and seasonality compared to the global box plot method [9].

Limitations: The effectiveness of MIQR depends on the choice of window size, which must balance sensitivity to local changes with stability in statistical estimates. A small window may lead to overfitting to noise, while a large window may miss local anomalies. Additionally, MIQR requires more computational resources than static methods due to the repeated calculation of IQR across windows.

Missing Value Handling

Missing values in time series data can arise from various sources, such as sensor failures, data transmission errors, or incomplete data collection. These gaps can distort statistical

analyses and degrade the performance of forecasting models if not addressed properly. Imputation methods estimate missing values based on available data, ensuring continuity and consistency in the dataset. The following subsubsections detail two common imputation techniques: Linear Interpolation and K-Nearest Neighbors (KNN).

4) Linear Interpolation: **Description and Methodology**: Linear interpolation estimates missing values by assuming a linear trend between known data points. For a missing value at time x, with known values y_1 at time x_1 and y_2 at time x_2, the imputed value y is calculated as:

$$y = y_1 + \frac{(x - x_1)(y_2 - y_1)}{x_2 - x_1}. \tag{3}$$

This method connects two adjacent known points with a straight line and estimates the missing value based on its position along that line. Linear interpolation is computationally simple and effective when the data exhibits smooth, linear trends.

Applications: Linear interpolation is particularly suitable for time series with gradual changes, such as weather data (e.g., temperature or humidity readings), where short-term trends are often approximately linear. It is widely used in environmental monitoring and other domains where data gaps are small and trends are stable [1].

Limitations: The method assumes linearity between data points, which may not hold for datasets with abrupt changes or non-linear patterns. In such cases, linear interpolation can introduce bias, especially for larger gaps or highly dynamic time series.

5) K-Nearest Neighbors (KNN): **Description and Methodology**: The K-Nearest Neighbors (KNN) imputation method estimates missing values by averaging the values of the K nearest neighbors in the dataset. For a missing value at time t, the imputed value y_t is computed as:

$$y_t = \frac{1}{K} \sum_{i \in N_K(t)} y_i, \tag{4}$$

where $N_K(t)$ represents the set of K nearest neighbors based on a distance metric (e.g., Euclidean distance) in the feature space. Neighbors are typically selected based on temporal proximity or similarity in feature values.

Applications: KNN imputation is effective for datasets with non-linear patterns, such as healthcare data (e.g., patient vital signs), where local patterns can be leveraged to estimate missing values. It is particularly useful when the data has complex, non-linear relationships that linear methods cannot capture [10].

Limitations: KNN imputation requires careful selection of K, as a small K may lead to overfitting to noise, while a large K may smooth out important variations. The method is also computationally intensive for large datasets, as it requires calculating distances to identify neighbors.

Smoothing

Smoothing techniques help eliminate noise in time-series data while preserving underlying trends, resulting in data ready for analysis and forecasting. Noise tends to cover the presence of patterns and causes models to overfit. The smoothing technique transforms the data to filter out high-frequency fluctuations and allow low-frequency trends to pass. The following are descriptions of the two commonly used three smoothing techniques: Simple Moving Average (SMA) and Exponential Moving Averages (EMA).

6) Simple Moving Average (SMA): **Description and Methodology**: The Simple Moving Average (SMA) smooths a time series by averaging the data points within a fixed-size window. For a time series x_t and a window size n, the SMA at time t is calculated as:

$$\text{SMA}_t = \frac{x_{t-n+1} + \cdots + x_t}{n}. \tag{5}$$

This method computes the arithmetic mean of the previous n observations, producing a smoothed value that reduces the impact of short-term fluctuations.

Applications: SMA is commonly used in financial data analysis, such as smoothing stock prices or trading volumes, to identify long-term trends. Its simplicity makes it a popular choice for initial data exploration and visualization [4].

Limitations: SMA treats all observations in the window equally, which can lead to a lag in responding to rapid changes in the data. This makes it less effective for time series with sudden shifts or high volatility, as the smoothed series may not reflect recent trends accurately.

7) Exponential Moving Average (EMA): **Description and Methodology**: The Exponential Moving Average (EMA) smooths a time series by applying exponentially decreasing weights to older observations, giving more importance to recent data. The EMA at time t is calculated recursively as:

$$\text{EMA}_t = \alpha x_t + (1 - \alpha)\text{EMA}_{t-1}, \tag{6}$$

where α (0 ¡ α ¡ 1) is the smoothing factor that controls the weight of the current observation. A higher α makes the EMA more responsive to recent changes.

Applications: EMA is well-suited for time series with dynamic trends, such as energy consumption data, where recent changes are more relevant for forecasting. It is widely used in algorithmic trading and real-time monitoring systems due to its responsiveness [4].

Limitations: The choice of α significantly affects the EMA's performance. A high α may make the series overly sensitive to noise, while a low α may result in excessive smoothing, missing important short-term variations. Tuning α requires domain knowledge or optimization techniques.

B. Data Transformation

Data transformation techniques modify the scale, distribution, or structure of time series data to make it more suitable for analysis and modeling. These methods address issues such

as varying scales across features, non-normal distributions, and non-stationarity, which can affect the performance of machine learning algorithms. Transformations ensure that data meets the assumptions of statistical models and improves their interpretability and predictive power. The following subsubsections detail three common transformation techniques: Normalization, Log Transformation, and Differencing.

1) Normalization: **Description and Methodology**: Normalization scales data to a fixed range, typically $[0, 1]$, to ensure that features with different scales contribute equally to the analysis. For a data point x, the normalized value x' is calculated as:

$$x' = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}, \tag{7}$$

where x_{min} and x_{max} are the minimum and maximum values in the dataset, respectively. This linear transformation preserves the relative relationships between data points while mapping them to a standardized range.

Applications: Normalization is essential for multivariate datasets, such as sensor readings from IoT devices, where features like temperature, pressure, and humidity may have different units and scales. It is widely used in machine learning algorithms, such as neural networks and support vector machines, which are sensitive to feature scales [10].

Limitations: Normalization assumes that the minimum and maximum values are representative of the data's range. Outliers can skew the normalization process, compressing most data points into a narrow range. Additionally, the method is sensitive to the choice of x_{min} and x_{max}, which may not be stable in dynamic datasets.

2) Log Transformation: **Description and Methodology**: Log transformation stabilizes variance and reduces skewness in data by applying a logarithmic function. For a data point x, the transformed value x' is computed as:

$$x' = \log(x + c), \tag{8}$$

where c is a small constant (e.g., 1) added to avoid issues with zero or negative values. This transformation compresses large values and expands small ones, making the data distribution more symmetric and closer to normal.

Applications: Log transformation is particularly useful for skewed datasets, such as sales figures, income distributions, or biological measurements, where extreme values are common. It is widely applied in time series analysis to stabilize variance in financial or economic data, improving the performance of models like ARIMA [1].

Limitations: Log transformation is not applicable to datasets with zero or negative values unless an appropriate constant c is chosen. It may also distort linear relationships in the data, making interpretation more complex. The choice of c can affect results and requires careful consideration.

3) Differencing: **Description and Methodology**: Differencing removes trends in a time series to achieve stationarity, a property where the mean, variance, and autocorrelation structure are constant over time. The first-order differenced series x'_t is calculated as:

$$x'_t = x_t - x_{t-1}, \tag{9}$$

where x_t is the value at time t and x_{t-1} is the value at the previous time step. Higher-order differencing can be applied for more complex trends.

Applications: Differencing is commonly used in economic time series, such as GDP or stock indices, where trends and seasonality can obscure underlying patterns. It is a standard preprocessing step for models like ARIMA, which require stationarity [4].

Limitations: Differencing reduces the length of the time series by one observation per order, which can be problematic for short datasets. It may also amplify noise in datasets with low signal-to-noise ratios, requiring additional smoothing techniques.

C. Data Reduction

Data reduction methods aim to simplify the data by reducing dimensionality or complexity while preserving relevant information. The processing time for such high-dimensional datasets is high, so these types of data bring about inefficiency in computation, overfitting, and noise in the models created for prediction. Data reduction brings in better performance and lower computational costs as it is based on just those features or representations that matter. The next subsubsections will discuss two standard data reduction techniques: Principal Component Analysis and Feature Selection.

1) Principal Component Analysis (PCA): **Description and Methodology:** The PCA reduces dimensionality by projecting the data onto a lower-dimensional space defined by the orthogonal principal components. For a data matrix X, the transformed data Z is obtained as:

$$Z = XW, \tag{10}$$

In this context, W represents the matrix of eigenvectors corresponding to the largest eigenvalues of the data's covariance matrix. These eigenvectors define the principal components, which indicate the directions of maximum variance in the data.

Applications: PCA is applied in numerous low- and high-dimensional datasets, such as sensor data in industrial systems or imaging applications, where numerous features could be correlated. It reduces the data dimensionality while preserving most of its variability desired preprocessing technique for most machine learning tasks [11].

Limitations: PCA assumes linear relationships between variables and may not capture non-linear patterns effectively. The principal components are linear combinations of original features, which can make interpretation challenging. Additionally, PCA is sensitive to outliers, which can distort the covariance matrix.

2) Feature Selection: **Description and Methodology**: Feature selection identifies and retains the most relevant features in a dataset, discarding redundant or irrelevant ones. Common methods include correlation analysis, where features with low correlation to the target variable or high correlation with other features are removed, and wrapper methods, which evaluate feature subsets based on model performance.

Applications: Feature selection is critical in financial forecasting, where datasets may include numerous indicators (e.g., stock prices, trading volumes, economic indices), but only a subset is predictive. By reducing the number of features, it decreases model complexity and training time while improving interpretability [12].

Limitations: Feature selection methods like correlation analysis may overlook complex interactions between features. Wrapper methods can be computationally expensive, especially for large datasets, as they require training multiple models to evaluate feature subsets. The choice of selection criteria can also introduce bias.

V. TIME SERIES FORECASTING MODELS

This section presents a comprehensive overview of forecasting models, organized by their methodological approach and applicability to different types of time series data (univariate, multivariate, seasonal, and irregular). Each model is discussed in terms of its methodology, strengths, limitations, and appropriate use cases. For implementation details and source code, please refer to the GitHub[1] repository.

A. Traditional Statistical Models

Statistical models are widely used for their interpretability and computational efficiency, particularly for linear and stationary time series [4].

1) ARIMA: **Description**: Autoregressive Integrated Moving Average (ARIMA) models are designed for stationary univariate time series data. ARIMA combines three components: autoregression (AR), differencing (I) to achieve stationarity, and moving average (MA). The model is expressed as:

$$\phi(B)(1-B)^d x_t = \theta(B)\epsilon_t, \tag{11}$$

where $\phi(B)$ and $\theta(B)$ are the AR and MA polynomials, B is the backshift operator, d is the differencing order, x_t is the time series, and ϵ_t is white noise. ARIMA is widely used for forecasting stock prices and economic indicators [1].

Advantages:

- **Interpretable**: Parameters directly relate to the time series' autocorrelation structure.
- **Efficient**: Computationally lightweight, suitable for small datasets.
- **Robust**: Performs well on stationary, linear data with clear patterns.

Limitations:

- **Stationarity Requirement**: Non-stationary data requires preprocessing (e.g., differencing), which can be complex.
- **Linear Assumption**: Struggles with non-linear patterns or complex dependencies.
- **Univariate Focus**: Not suitable for multivariate time series without extensions.

Suitable Datasets:

- Univariate, stationary time series (e.g., stock prices, temperature records).
- Small to medium-sized datasets with linear patterns.
- Example: Daily stock price data or monthly economic indicators.

2) SARIMA: **Description**: Seasonal ARIMA (SARIMA) extends ARIMA to handle seasonal patterns in univariate time series. It incorporates seasonal AR, differencing, and MA components, expressed as:

$$\phi(B)\Phi(B^s)(1-B)^d(1-B^s)^D x_t = \theta(B)\Theta(B^s)\epsilon_t, \tag{12}$$

where $\Phi(B^s)$ and $\Theta(B^s)$ are seasonal AR and MA polynomials, s is the seasonal period, and D is the seasonal differencing order. SARIMA is effective for data with recurring seasonal patterns, such as retail sales [4].

Advantages:

- **Seasonality Handling**: Explicitly models seasonal patterns, improving accuracy for periodic data.
- **Flexible**: Can be tuned for various seasonal periods (e.g., daily, monthly).
- **Interpretable**: Like ARIMA, parameters are directly interpretable.

Limitations:

- **Complexity**: Requires identifying both non-seasonal and seasonal parameters, increasing model complexity.
- **Stationarity**: Assumes stationarity after differencing, which may not always hold.
- **Univariate**: Limited to single time series without extensions.

Suitable Datasets:

- Univariate time series with clear seasonality (e.g., monthly retail sales, quarterly GDP).
- Medium-sized datasets with consistent seasonal patterns.
- Example: Monthly airline passenger data or seasonal electricity consumption.

3) Exponential Smoothing (ETS): **Description**: Exponential Smoothing (ETS) models forecast time series by assigning exponentially decreasing weights to past observations. The simple ETS model is:

$$\hat{x}_{t+1} = \alpha x_t + (1-\alpha)\hat{x}_t, \tag{13}$$

where α is the smoothing parameter (0 ¡ α ¡ 1), x_t is the observed value, and $\hat{x}_t$ is the forecast. ETS can be extended to include trends and seasonality, making it versatile for short-term forecasting, such as energy demand [13].

[1] https://github.com/bishwajitprasadgond/timeseries

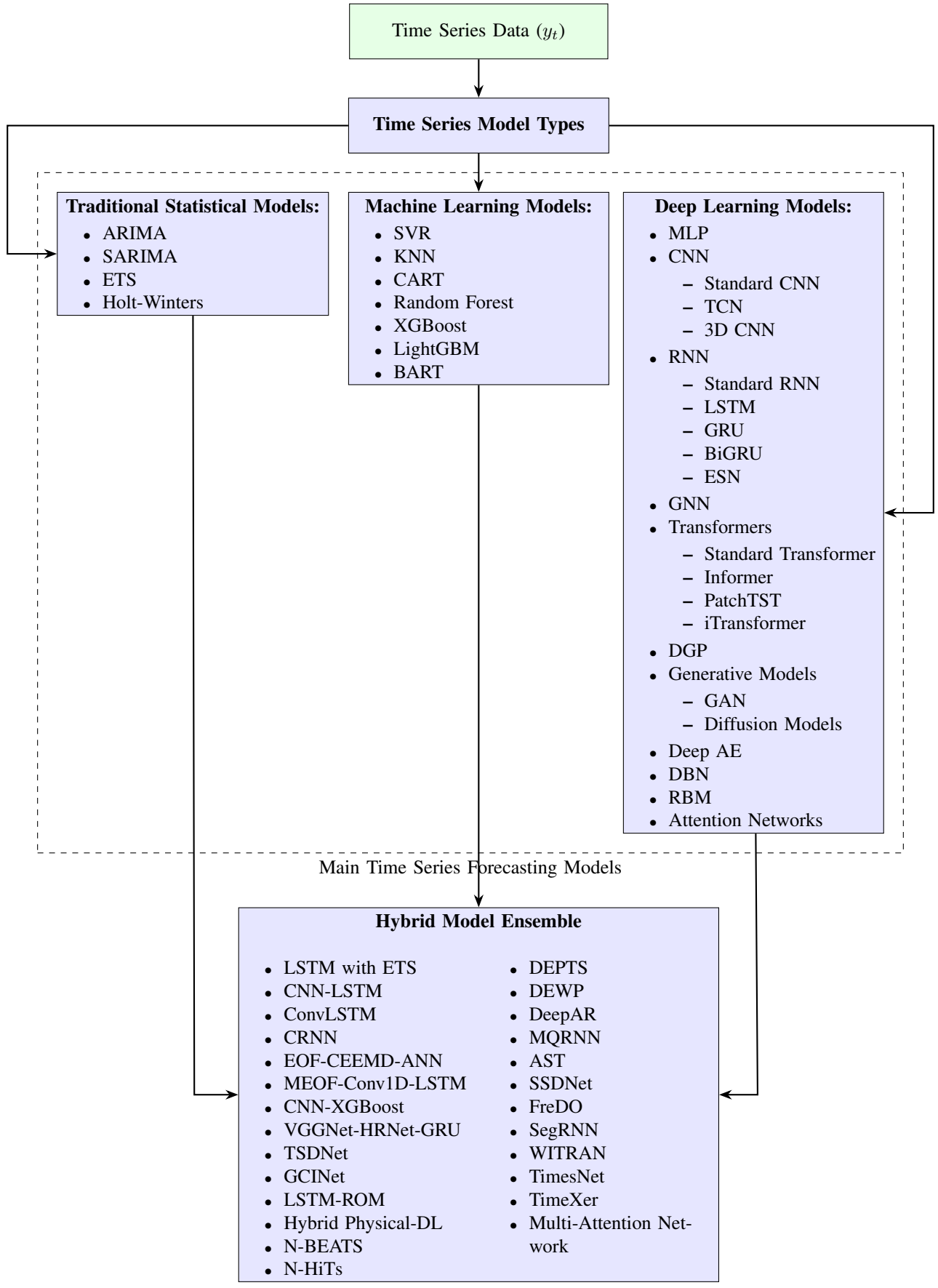

Fig. 1: Different Types of Time Series Forecasting Models: Statistical, Machine Learning, Deep Learning, and Hybrid

Advantages:

- **Simplicity**: Easy to implement and computationally efficient.
- **Flexible**: Variants handle trends and seasonality (e.g., double or triple exponential smoothing).
- **Robust**: Performs well for short-term forecasts with stable patterns.

Limitations:

- **Short-Term Focus**: Less effective for long-term forecasting due to reliance on recent data.
- **Linear Assumption**: Struggles with non-linear or complex patterns.
- **Parameter Tuning**: Requires careful selection of smoothing parameters.

Suitable Datasets:

- Univariate time series with or without trends and seasonality.
- Short-term forecasting tasks (e.g., daily energy consumption, weekly sales).
- Example: Hourly electricity load data or short-term retail demand.

4) Holt-Winters: **Description**: The Holt-Winters method extends exponential smoothing to explicitly model level, trend, and seasonality. It uses three smoothing equations for level, trend, and seasonal components, making it suitable for time series with both trends and seasonal patterns, such as retail demand forecasting [13].

Advantages:

- **Comprehensive**: Captures level, trend, and seasonality simultaneously.
- **Practical**: Widely used in business applications due to its balance of simplicity and accuracy.
- **Adaptable**: Can handle additive or multiplicative seasonality.

Limitations:

- **Seasonal Period**: Requires a fixed, known seasonal period.
- **Complexity**: More parameters to tune than simple ETS, increasing computational cost.
- **Linear Patterns**: Limited in capturing non-linear dynamics.

Suitable Datasets:

- Univariate time series with trends and seasonality (e.g., retail sales, inventory levels).
- Medium-term forecasting tasks with consistent seasonal patterns.
- Example: Monthly retail sales or quarterly demand for consumer goods.

B. Machine Learning Models

Machine learning models excel at capturing non-linear patterns and are suitable for complex time series [10].

1) Support Vector Regression (SVR): **Description**: SVR applies support vector machines to regression tasks, finding a function that predicts continuous values within a margin of tolerance. It is defined as:

$$f(x) = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) K(x, x_i) + b, \tag{14}$$

where $K(x, x_i)$ is a kernel function (e.g., RBF), α_i, α_i^* are Lagrange multipliers, and b is the bias. SVR is effective for energy load forecasting due to its ability to handle non-linear relationships [14].

Advantages:

- **Non-Linearity**: Captures complex, non-linear patterns via kernel functions.
- **Robustness**: Insensitive to outliers due to the margin-based approach.
- **Flexible**: Kernel choice allows adaptation to various data types.

Limitations:

- **Scalability**: Computationally expensive for large datasets due to kernel computations.
- **Parameter Tuning**: Requires careful selection of kernel and regularization parameters.
- **Interpretability**: Less interpretable than statistical models.

Suitable Datasets:

- Univariate or multivariate time series with non-linear patterns.
- Medium-sized datasets (e.g., energy load, financial metrics).
- Example: Hourly electricity demand or stock volatility data.

2) Random Forest: **Description**: Random Forest is an ensemble method that builds multiple decision trees and aggregates their predictions. It handles non-linear relationships and feature interactions, making it suitable for financial forecasting [15].

Advantages:

- **Robust**: Reduces overfitting through ensemble averaging.
- **Feature Handling**: Effectively processes multivariate data with many predictors.
- **Non-Linear**: Captures complex patterns without assuming linearity.

Limitations:

- **Computationally Intensive**: Requires significant resources for large datasets.
- **Black-Box**: Limited interpretability compared to statistical models.
- **Time Series Adaptation**: Requires feature engineering (e.g., lagged variables) for time series.

Suitable Datasets:

- Multivariate time series with non-linear patterns.
- Financial or economic datasets with multiple predictors.
- Example: Stock market forecasting with technical indicators or macroeconomic data.

3) XGBoost: **Description**: XGBoost is a gradient boosting framework that builds an ensemble of decision trees optimized for performance. It minimizes a loss function through iterative tree construction, making it effective for large-scale time series like traffic flow [16].

Advantages:

- **High Performance**: Optimized for speed and accuracy, suitable for large datasets.
- **Flexible**: Handles multivariate data and missing values effectively.
- **Feature Importance**: Provides insights into predictor importance.

Limitations:

- **Complexity**: Requires significant tuning of hyperparameters (e.g., learning rate, tree depth).
- **Overfitting Risk**: Can overfit without proper regularization.
- **Interpretability**: Less interpretable than statistical models.

Suitable Datasets:

- Large-scale multivariate time series with complex patterns.
- Datasets with multiple predictors (e.g., traffic flow, energy consumption).
- Example: Real-time traffic data or large-scale sensor networks.

C. Deep Learning Models

Deep learning models are designed to capture complex, nonlinear dependencies in large datasets [17].

1) LSTM: **Description**: Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) designed to model long-term dependencies in sequential data. The LSTM cell is defined as:

$$h_t = o_t \odot \tanh(c_t), \tag{15}$$

where h_t is the hidden state, o_t is the output gate, and c_t is the cell state. LSTMs are widely used in healthcare monitoring for their ability to capture temporal dependencies [18].

Advantages:

- **Long-Term Dependencies**: Effectively models sequences with long lags.
- **Flexible**: Suitable for both univariate and multivariate time series.
- **Robust**: Handles noisy data with proper training.

Limitations:

- **Computational Cost**: Requires significant resources for training.
- **Overfitting**: Risk of overfitting on small datasets without regularization.
- **Complexity**: Difficult to tune and interpret compared to statistical models.

Suitable Datasets:

- Univariate or multivariate time series with long-term dependencies.
- Sequential data in healthcare or finance (e.g., patient vitals, stock prices).
- Example: ECG signals or daily financial time series.

2) Transformers: **Description**: Transformers use attention mechanisms to weigh the importance of different time steps, making them effective for multivariate time series. They rely on self-attention to capture dependencies across long sequences, widely used in natural language processing and time series forecasting [19].

Advantages:

- **Attention Mechanism**: Captures complex dependencies across long sequences.
- **Scalable**: Handles multivariate data efficiently with parallel processing.
- **Flexible**: Adapts to various time series patterns.

Limitations:

- **Resource Intensive**: Requires large computational resources and data.
- **Interpretability**: Complex architecture reduces interpretability.
- **Training Complexity**: Requires careful tuning of attention layers and parameters.

Suitable Datasets:

- Multivariate time series with complex dependencies.
- Large datasets with multiple features (e.g., sensor networks, financial markets).
- Example: Multi-sensor industrial data or multivariate economic indicators.

3) TCN: **Description**: Temporal Convolutional Networks (TCNs) use dilated convolutions to capture long-term patterns in time series, particularly irregular ones. TCNs are designed to be computationally efficient while modeling temporal dependencies [20].

Advantages:

- **Efficiency**: Faster training than RNNs due to convolutional architecture.
- **Long-Range Dependencies**: Dilated convolutions capture extended temporal patterns.
- **Robust**: Effective for irregular or noisy time series.

Limitations:

- **Fixed Receptive Field**: Limited by the size of the convolutional kernel.
- **Complexity**: Requires careful design of network architecture.
- **Data Requirements**: Performs best with large datasets.

Suitable Datasets:

- Irregular or noisy time series with long-term patterns.
- Univariate or multivariate data (e.g., IoT sensor data, event-based time series).
- Example: Network traffic data or irregular sensor readings.

D. Hybrid Models

Hybrid models combine multiple approaches to leverage their strengths for improved accuracy.

1) CNN-LSTM: **Description**: CNN-LSTM combines convolutional neural networks (CNNs) for feature extraction with LSTMs for sequence modeling. CNNs extract spatial or temporal features, which are then fed into LSTMs to capture sequential dependencies, making it suitable for energy forecasting [21].

Advantages:

- **Feature Extraction**: CNNs reduce dimensionality and extract relevant patterns.
- **Sequential Modeling**: LSTMs capture temporal dependencies effectively.
- **Versatile**: Handles complex, multivariate time series.

Limitations:

- **Computational Cost**: Combines the resource demands of CNNs and LSTMs.
- **Complexity**: Requires tuning of both CNN and LSTM layers.
- **Data Hungry**: Needs large datasets for optimal performance.

Suitable Datasets:

- Multivariate time series with spatial and temporal patterns.
- Energy or environmental data with complex dependencies.
- Example: Smart grid energy consumption or weather-related time series.

2) ETS-LSTM: **Description**: ETS-LSTM combines the statistical robustness of exponential smoothing with the sequence modeling capabilities of LSTMs. ETS preprocesses the data to capture trends and seasonality, while LSTMs model residual non-linear patterns, often used in sales forecasting [22].

Advantages:

- **Hybrid Strength**: Combines interpretability of ETS with LSTM's non-linear modeling.
- **Improved Accuracy**: Captures both linear and non-linear patterns.
- **Flexible**: Suitable for seasonal and non-linear time series.

Limitations:

- **Complexity**: Requires expertise to integrate and tune both components.
- **Computational Cost**: Higher than standalone ETS or LSTM models.
- **Data Requirements**: Needs sufficient data for LSTM training.

Suitable Datasets:

- Univariate or multivariate time series with seasonality and non-linear patterns.
- Sales or demand forecasting datasets.
- Example: Retail sales with seasonal trends and promotional effects.

E. Domain-Specific Techniques

Domain-specific techniques incorporate expert knowledge to enhance forecasting performance.

1) Physically Guided Deep Learning: **Description**: Physically Guided Deep Learning integrates domain-specific physical models (e.g., weather equations) into deep learning architectures. This approach constrains neural networks with physical laws, improving accuracy in domains like weather forecasting [23].

Advantages:

- **Domain Knowledge**: Incorporates physical constraints for realistic predictions.
- **Improved Accuracy**: Reduces errors in domains with known physical models.
- **Robust**: Performs well in data-scarce scenarios by leveraging domain expertise.

Limitations:

- **Domain Dependency**: Requires accurate physical models, which may not always exist.
- **Complexity**: Integrating physical models with neural networks is challenging.
- **Computational Cost**: Can be resource-intensive due to hybrid nature.

Suitable Datasets:

- Time series governed by physical laws (e.g., weather, climate, fluid dynamics).
- Datasets with limited observations but strong domain knowledge.
- Example: Weather forecasting or hydrological time series.

2) Bayesian Networks: **Description**: Bayesian Networks model probabilistic relationships among variables using directed acyclic graphs. They are used in healthcare to forecast outcomes based on probabilistic dependencies, such as patient health metrics [24].

Advantages:

- **Probabilistic Modeling**: Captures uncertainty and dependencies effectively.
- **Interpretable**: Graphical structure provides insights into relationships.
- **Flexible**: Suitable for multivariate and causal forecasting tasks.

Limitations:

- **Scalability**: Computationally expensive for large networks.
- **Data Requirements**: Needs sufficient data to estimate probabilities accurately.
- **Complexity**: Designing and learning network structures can be challenging.

Suitable Datasets:

- Multivariate time series with probabilistic relationships.
- Healthcare or social science datasets with causal dependencies.
- Example: Patient health monitoring or epidemiological time series.

VI. PROPOSED ARCHITECTURE

The proposed PHM-MAX-XGBoost-LSTM-SVM hybrid model is designed for forecasting univariate time series data,

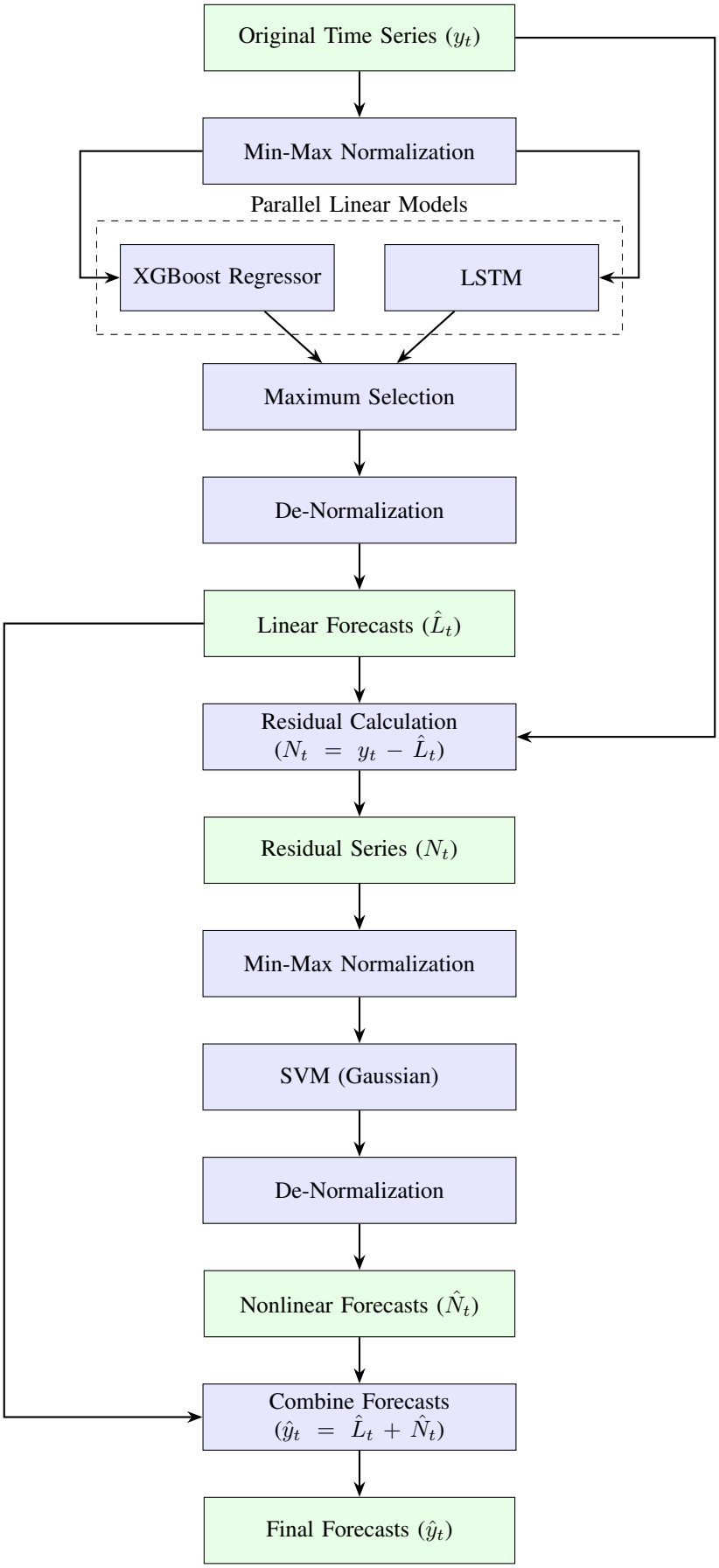

Fig. 2: Architecture of the PHM-MAX-XGBoost-LSTM-SVM Hybrid Model for Time Series Forecasting

adapting the methodology from [25] by using XGBoost Regressor and Long Short-Term Memory (LSTM) neural networks to model the linear component, and a Support Vector Machine (SVM) with a Gaussian (RBF) kernel for the nonlinear component. The architecture assumes the time series y_t is a sum of a linear component L_t and a nonlinear component N_t, as defined by $y_t = L_t + N_t$. The model processes the data through a series of steps, including normalization, parallel modeling, maximum selection, residual calculation, and nonlinear modeling, to produce final forecasts $\hat{y}_t = \hat{L}_t + \hat{N}_t$. The architecture is detailed below and visualized in Figure 2.

The time series data is loaded from a CSV file, with the target variable specified in a user-defined column (e.g., 'value'), and split into training (60%), validation (20%), and test (20%) sets. The in-sample data (training + validation) is normalized using the min-max normalization technique to scale values between 0 and 1. The normalized data is processed by two parallel models: XGBoost Regressor and LSTM. The XGBoost model is configured with 100 trees, a learning rate of 0.05, and a maximum depth of 3, using 25 lagged features (adjustable, e.g., 11 for yearly data). The LSTM model consists of a single LSTM layer with 50 units (ReLU activation) followed by a dense output layer, trained for 100 epochs with the Adam optimizer, using 25 lagged features reshaped into a 3D tensor (samples, timesteps, features).

The forecasts from XGBoost and LSTM are combined by selecting the maximum value at each time step to form the normalized linear component forecasts $\hat{L}'_t$. These are denormalized to obtain the linear forecasts $\hat{L}_t$. The residual series, representing the nonlinear component, is computed as $N_t = y_t - \hat{L}_t$. The residuals are normalized using min-max normalization and modeled using an SVM with a Gaussian kernel to capture nonlinear patterns. The resulting nonlinear forecasts are denormalized to obtain $\hat{N}_t$. Finally, the linear and nonlinear forecasts are combined to produce the final forecasts $\hat{y}_t = \hat{L}_t + \hat{N}_t$.

VII. COMPARISON OF FORECASTING TECHNIQUES

The term "time series forecasting" comprises a wide range of techniques, each suited to particular data configurations and application areas, with a trade-off existing among interpretability, computational complexity, and predictive capability. For instance, statistical models of the ARIMA type are suited for stationary time series in a univariate sense. They provide explanatory results and are computationally inexpensive, which is very useful for stock price forecasting. Due to their linear limitations, they fail to identify complex patterns [1]. SARIMA is an extended form of the ARIMA model that adds a seasonal factor so that data with repeat period fluctuations, such as retail sales, can be modelled while remaining limited to linear dynamics [4]. In the same way, Exponential Smoothing (ETS) presents a simple yet sufficient method for univariate time series. Given its capacity to predict level and trend components, it is commonly adopted for very short-term energy forecasting. Again, its short-time horizons might compromise its capacity for longer-term forecasting [13]. Holt-Winters, another statistical method, enhances ETS by explicitly capturing level, trend, and seasonality, making it suitable for datasets like retail sales or inventory levels with consistent seasonal patterns. However, it requires a predefined seasonal period and struggles with non-linear trends, restricting its flexibility [13].

Machine learning models offer robust solutions for nonlinear and multivariate time series. Support Vector Regression (SVR) excels in capturing non-linear relationships through kernel functions, such as the radial basis function (RBF), making it effective for load forecasting in energy systems. Its robustness to outliers is a key strength, though scalability issues and the need for careful kernel selection pose challenges [14]. Random Forest, an ensemble method, handles multivariate data and non-linear patterns with high robustness, ideal for financial forecasting with multiple predictors. Its computational intensity and need for feature engineering, such as lagged variables, are notable drawbacks [15]. XGBoost, a gradient boosting framework, delivers high performance for large-scale multivariate datasets, such as traffic flow or energy consumption, by optimizing tree-based models and providing feature importance insights. However, its complex hyperparameter tuning and potential for overfitting require careful configuration [16].

Deep learning models are designed for complex sequential data. Long Short-Term Memory (LSTM) networks capture long-term dependencies in sequential time series, making them suitable for applications like healthcare monitoring and financial forecasting. Their ability to handle noisy data is a strength, but high computational costs and data-intensive requirements limit their use in smaller datasets [18]. Transformers, leveraging attention mechanisms, excel in multivariate time series with long-range dependencies, such as sensor network data, due to their scalable parallel processing. Their high computational demands and reduced interpretability are significant challenges [19]. Temporal Convolutional Networks (TCNs) use dilated convolutions to efficiently model long-range patterns in irregular time series, like network traffic or IoT sensor data, offering faster training than recurrent networks. Their fixed receptive fields and need for large datasets are limitations [20].

Hybrid models combine multiple approaches for enhanced accuracy. CNN-LSTM integrates convolutional feature extraction with LSTM's sequential modeling, ideal for energy forecasting with complex spatial-temporal patterns, though it requires complex training and large datasets [21]. ETS-LSTM combines ETS's interpretability for trends and seasonality with LSTM's non-linear modeling, making it effective for sales forecasting with seasonal and promotional effects, but integration complexity increases computational costs [22]. Physically Guided Deep Learning incorporates domain-specific physical constraints, improving accuracy in data-scarce scenarios like weather forecasting, but depends on accurate physical models and is computationally intensive [23]. Bayesian Networks model probabilistic relationships in multivariate data, offering

TABLE I: Comparison of time series forecasting techniques

Model	**Type**	**Data Type**	**Strengths**	**Limitations**	**Applications**
ARIMA	Statistical	Univariate, Stationary	Interpretable, efficient	Linear assumptions	Stock prices [1]
SARIMA	Statistical	Seasonal	Handles seasonality	Limited to linear patterns	Retail sales [4]
ETS	Statistical	Univariate	Simple, effective	Short-term focus	Energy forecasting [13]
Holt-Winters	Statistical	Univariate, Seasonal	Captures level, trend, seasonality; adaptable to additive/multiplicative seasonality	Fixed seasonal period; limited to linear patterns	Retail sales, inventory levels [13]
SVR	Machine Learning	Multivariate	Non-linear modeling; robust to outliers	Kernel choice critical; less scalable	Load forecasting [14]
Random Forest	Machine Learning	Multivariate	Robust, non-linear; handles multiple predictors	Computationally intensive; needs feature engineering	Financial forecasting [15]
XGBoost	Machine Learning	Multivariate	High performance; handles missing values; feature importance	Complex tuning; overfitting risk	Traffic flow, energy consumption [16]
LSTM	Deep Learning	Sequential	Long-term dependencies; robust to noise	Data-intensive; high computational cost	Healthcare, financial time series [18]
Transformers	Deep Learning	Multivariate	Long-range dependencies; scalable for multivariate data	High computational cost; less interpretable	Multivariate forecasting [19]
TCN	Deep Learning	Sequential, Irregular	Efficient training; captures long-range patterns	Fixed receptive field; needs large datasets	Network traffic, IoT sensor data [20]
CNN-LSTM	Hybrid	Sequential	Feature extraction; sequential modeling	Complex training; data-intensive	Energy forecasting [21]
ETS-LSTM	Hybrid	Univariate, Seasonal	Combines ETS interpretability with LSTM non-linear modeling	Complex integration; high computational cost	Sales forecasting [22]
Physically Guided Deep Learning	Hybrid	Multivariate, Physical	Incorporates physical constraints; accurate in data-scarce scenarios	Requires accurate physical models; complex integration	Weather forecasting [23]
Bayesian Networks	Probabilistic	Multivariate	Probabilistic modeling; interpretable dependencies	Computationally expensive; needs sufficient data	Healthcare monitoring [24]

interpretable dependencies for healthcare monitoring. Their scalability issues and data requirements limit their applicability in large systems [24]. Each technique offers unique strengths, with choices driven by data type, computational resources, and forecasting objectives, ensuring flexibility across diverse domains.

VIII. CONCLUSIONS AND FUTURE WORK

The comparison of time series forecasting techniques reveals a diverse landscape of statistical, machine learning, deep learning, hybrid, and probabilistic models, each offering unique strengths and trade-offs. Statistical models like ARIMA, SARIMA, ETS, and Holt-Winters provide interpretable and efficient solutions for univariate and seasonal data but are limited by linear assumptions, making them less suitable for complex, non-linear patterns. Machine learning models, including SVR, Random Forest, and XGBoost, excel in capturing non-linear relationships and handling multivariate data, though they require careful tuning and significant computational resources. Deep learning approaches, such as LSTM, Transformers, and TCNs, are powerful for sequential and irregular time series with long-term dependencies, but their data and computational demands can be prohibitive. Hybrid models like CNN-LSTM, ETS-LSTM, and Physically Guided Deep Learning combine complementary strengths, improving accuracy for complex datasets, yet introduce integration challenges. Bayesian Networks offer probabilistic insights for multivariate data but face scalability issues. Selecting the appropriate model depends on the dataset's characteristics—such as stationarity, seasonality, or multivariate complexity—and the forecasting context, including available computational resources and interpretability needs. For instance, financial forecasting may favor Random Forest or XGBoost, while healthcare applications benefit from LSTM or Bayesian Networks. Future advancements may focus on optimizing hybrid models, such as the PHM-MAX-XGBoost-LSTM-SVM approach, to balance accuracy and efficiency, leveraging domain knowledge to enhance performance across diverse applications.

REFERENCES

[1] George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel, and Greta M. Ljung. *Time Series Analysis: Forecasting and Control*. Wiley, 2015.

[2] Ruey S. Tsay. *Analysis of Financial Time Series*. Wiley, 2010.

[3] Peter J. Brockwell and Richard A. Davis. *Introduction to Time Series and Forecasting*. Springer, 2002.

[4] Rob J Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, 2018.

[5] Q. Song and B. S. Chissom. Fuzzy time series and its models. *Fuzzy Sets and Systems*, 54(3):269–277, 1993.

[6] Peter J Brockwell and Richard A Davis. *Introduction to Time Series and Forecasting*. Springer, 2016.

[7] John W Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977.

[8] Robert H Shumway and David S Stoffer. Time series analysis and its applications. *Springer Texts in Statistics*, 2017.

[9] Li Zhang and Wei Wang. Outlier detection in time series data using local iqr. *Journal of Data Science*, 18(3):456–472, 2020.

[10] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.

[11] Ian T Jolliffe and Jorge Cadima. Principal component analysis: A review and recent developments. *Philosophical Transactions of the Royal Society A*, 374(2065), 2016.

[12] Isabelle Guyon and André Elisseeff. *An Introduction to Variable and Feature Selection*, volume 3. 2003.

[13] Rob J Hyndman, Anne B Koehler, J Keith Ord, and Ralph D Snyder. *Exponential Smoothing: The State of the Art*, volume 24. 2008.

[14] Vladimir N Vapnik. The nature of statistical learning theory. *Springer*, 1995.

[15] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[16] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD*, pages 785–794, 2016.

[17] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. Attention is all you need. *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[20] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[21] Yuxuan Liu, Wei Zhang, Jie Yang, et al. Deep learning for time series forecasting: A survey. *Big Data*, 8(5):321–339, 2020.

[22] Wei Zhang and Yuxuan Liu. Hybrid models for time series forecasting: A review. *Journal of Forecasting*, 39(7):1023–1042, 2020.

[23] Markus Reichstein, Gustau Camps-Valls, Bjorn Stevens, et al. Deep learning and process understanding for data-driven earth system science. *Nature*, 566(7743):195–204, 2019.

[24] Judea Pearl. Probabilistic reasoning in intelligent systems. *Morgan Kaufmann*, 1988.

[25] 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*, 35(15):1388–1406, 2021.

APPENDIX

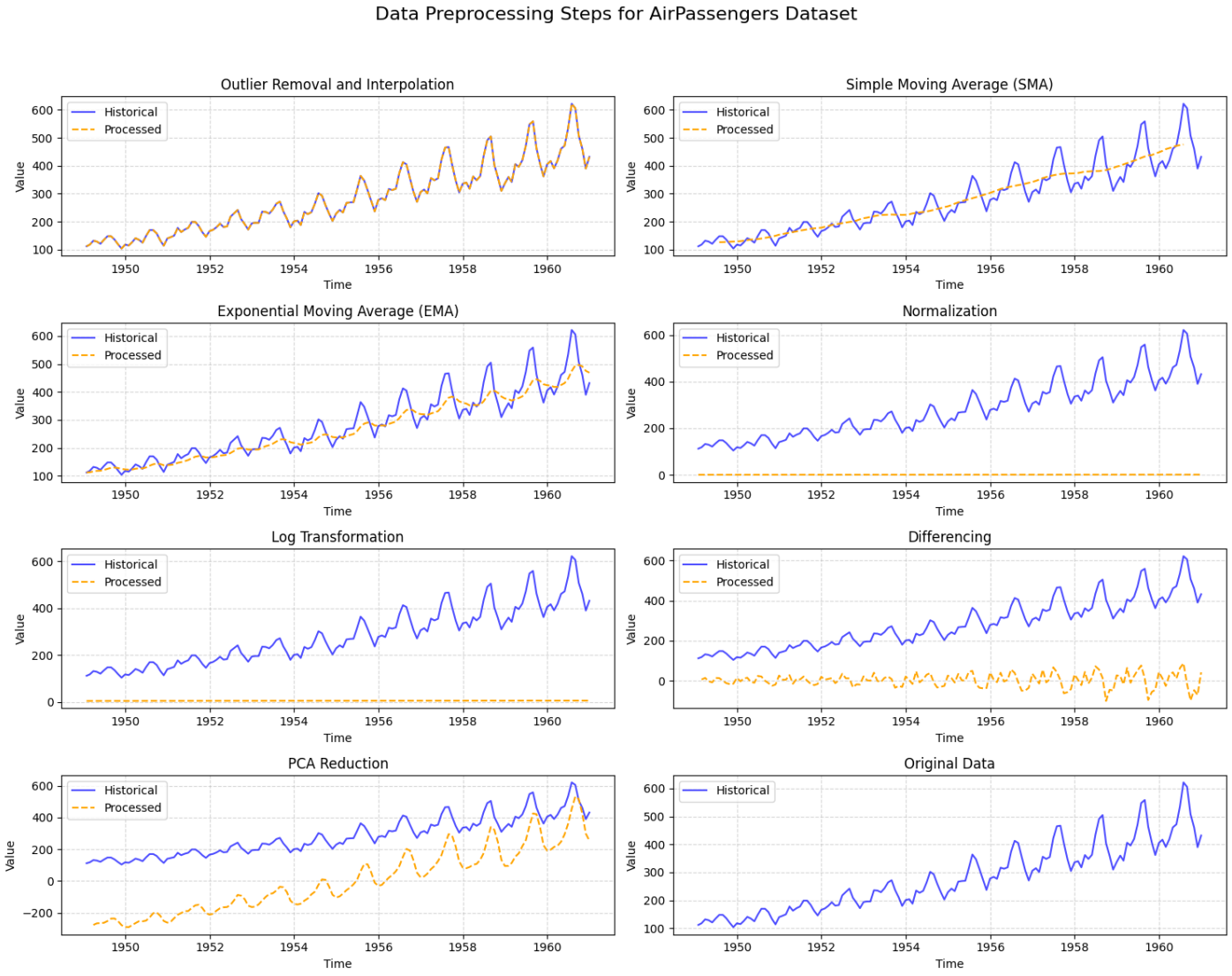

Fig. 3: Preprocessing of AirPassanger Dataset.

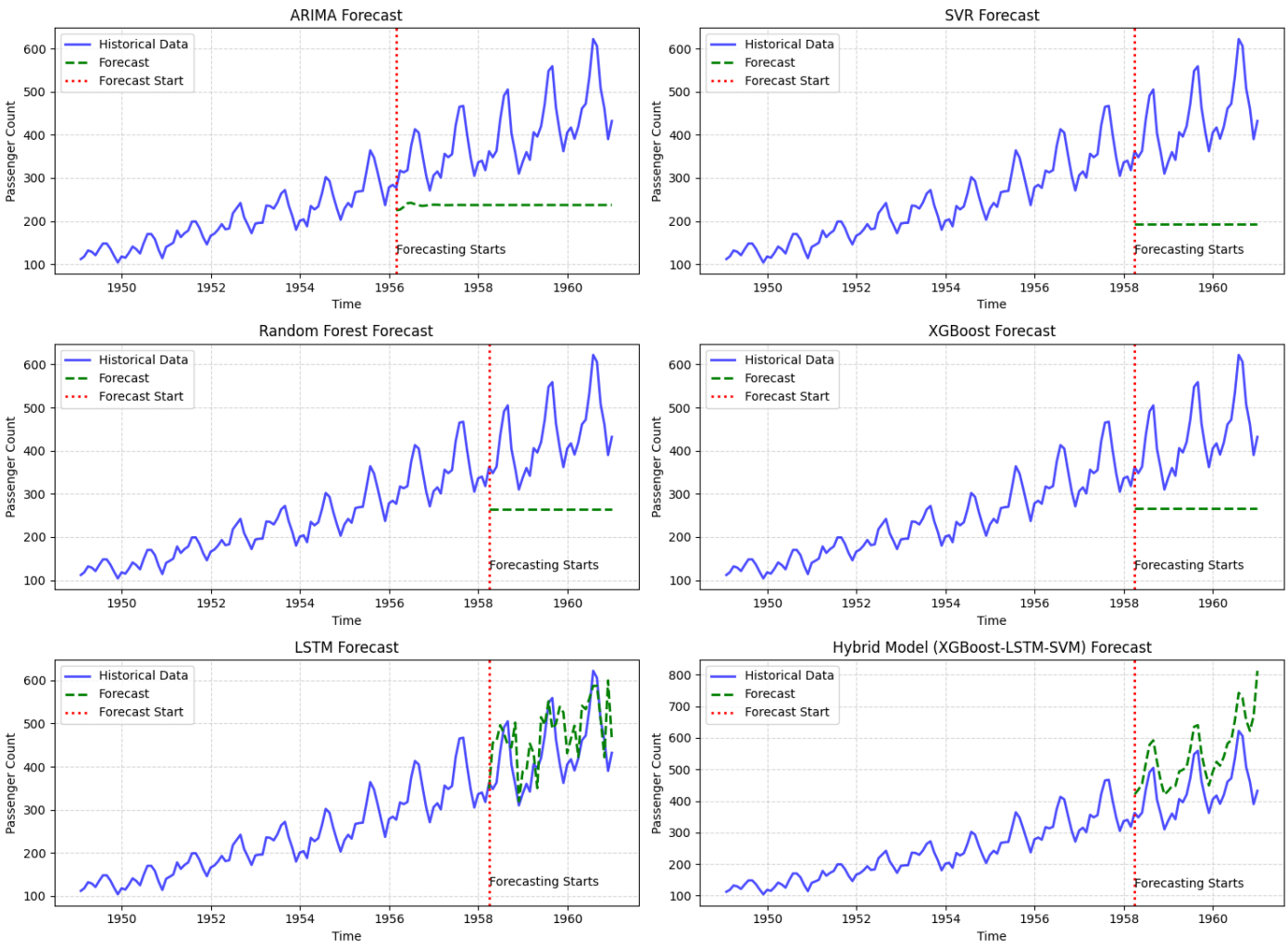

Fig. 4: Models prediction on AirPassanger Dataset.

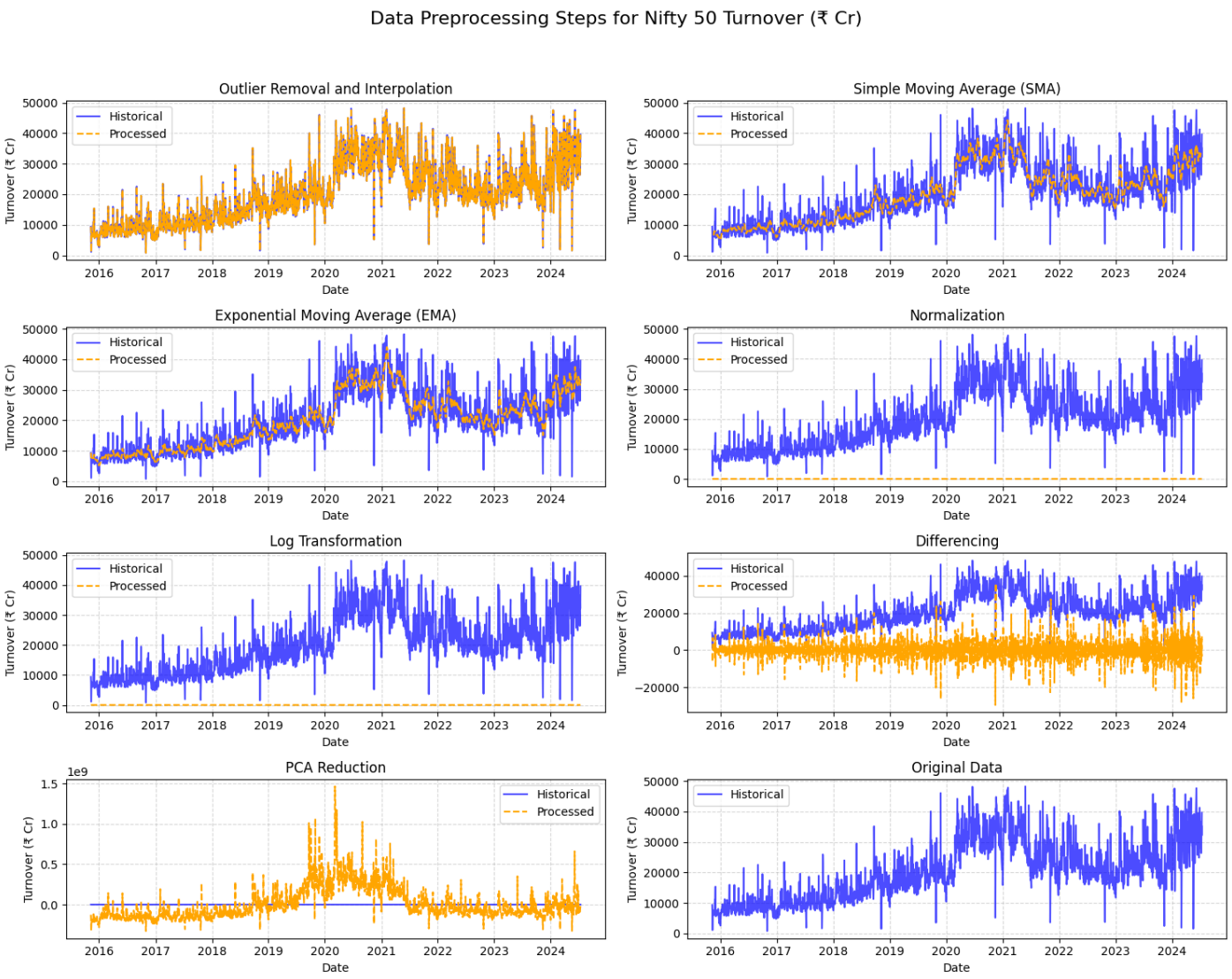

Fig. 5: Preprocessing of Nifty 50 Stock Dataset.

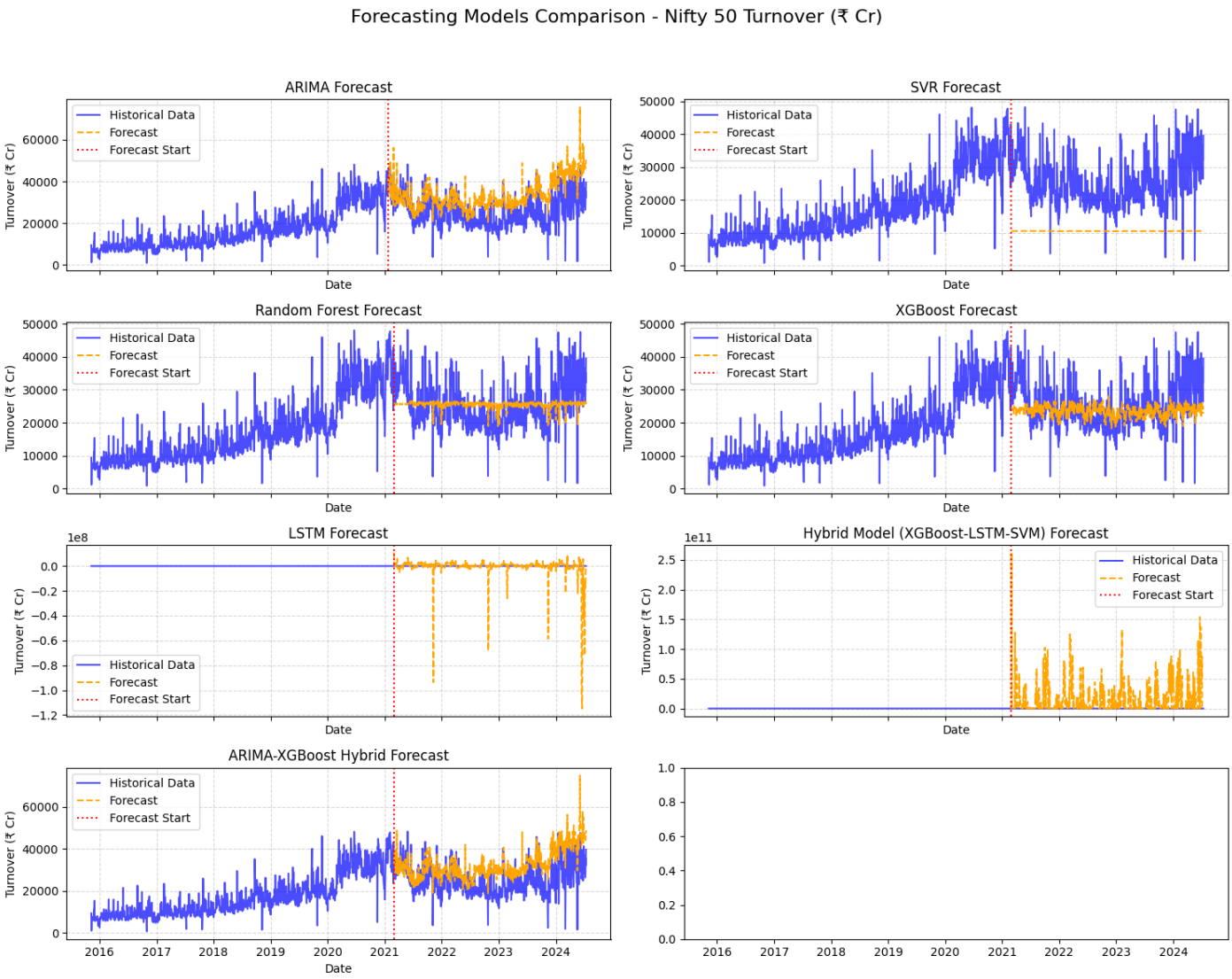

Fig. 6: Models prediction on Nifty 50 Stock Dataset.