JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

A Comprehensive Review on Time Series Forecasting Techniques

Harpinder Kaur[1] Gagandeep Jagdev[2*]
[1]Assistant Professor, Guru Gobind Singh Khalsa College, Bhagta Bhai Ka, Punjab, India.
[2*]Technical Officer, Punjabi University Guru Kashi Campus, Damdama Sahib, Talwandi Sabo, Punjab, India.
harpinder.brar435@gmail.com[1]
drgagan137@pbi.ac.in[2*]

Abstract
Time series forecasting is a crucial task in various domains, ranging from finance and economics to weather prediction and demand forecasting. Over the years, numerous techniques have been developed to address the challenges associated with modeling and predicting time-dependent data. This paper presents a comprehensive review and comparative analysis of different techniques for time series forecasting. The research paper introduces traditional statistical methods, including Autoregressive Integrated Moving Average (ARIMA), Seasonal ARIMA (SARIMA), and Exponential Smoothing. These methods have long been the foundation for time series analysis and provide a solid baseline for forecasting. Their strengths lie in their simplicity, interpretability, and ability to capture different components of time series data. The research paper elaborates on the Neural Networks which excel at capturing complex patterns, non-linear relationships, and interactions among variables. Neural Networks offer the potential for improved accuracy and flexibility in handling large-scale and high-dimensional time series datasets. Bayesian methods approaches incorporate prior knowledge, handle uncertainties, and provide a coherent framework for probabilistic forecasting. They are particularly useful in situations where prior information or expert opinions are available. The advantages and limitations of each technique are discussed, considering factors such as computational complexity, data requirements, interpretability, and the presence of external factors or seasonality. Furthermore, real-world applications across various domains are highlighted to illustrate the practical relevance and effectiveness of these techniques in different contexts. In conclusion, the research paper provides a comprehensive overview of different techniques for time series forecasting, empowering researchers and practitioners to make informed decisions regarding the selection and application of appropriate methods.

Keywords – ARIMA, Bayesian methods, Neural Networks, Prophet, Spectral Analysis, VAR.

I. INTRODUCTION

Time series forecasting is a technique used in statistics and data analysis to predict future values based on historical data points collected at regular intervals over time. It is particularly useful when dealing with data that exhibits a temporal dependence, where the values of the variable being measured are influenced by past observations [1, 2].
The goal of time series forecasting is to capture patterns, trends, and seasonality in historical data and use them to make accurate predictions about future values. The process typically involves the following steps:

- Data collection: Gather historical data points at regular time intervals. The data can be collected from a variety of sources, such as financial markets, weather stations, or manufacturing processes.
- Data preprocessing: Clean the data by handling missing values, outliers, and any other anomalies. Ensure that the data is in a suitable format for analysis.
- Exploratory data analysis: Analyze the data to understand its characteristics, such as trends, seasonality, and any other patterns. Visualizations, such as line plots or autocorrelation plots, can help in identifying these patterns.
- Time series decomposition: Decompose the time series into its constituent components, which typically include trend, seasonality, and residual (or error). This step helps in understanding the individual patterns present in the data.
- Model selection: Choose an appropriate forecasting model based on the characteristics of the data.

There are several models available for time series forecasting, including autoregressive integrated moving averages (ARIMA), exponential smoothing methods (such as Holt-Winters), and more advanced techniques like recurrent neural networks (RNNs) or long short-term memory (LSTM) networks [3].

- Model training: Fit the selected model to the historical data. This involves estimating the model parameters using optimization techniques to minimize the difference between the predicted values and the actual values in the training data [4].
- Model validation: Assess the performance of the trained model by comparing its predictions against a validation dataset or using cross-validation techniques [5, 6]. This step helps in evaluating the model's accuracy and identifying any potential issues, such as overfitting.

- Forecasting: Once the model is validated, use it to make predictions for future periods. The forecasted values can provide insights into potential trends, patterns, or anomalies that may occur in the future.
- Model evaluation: Assess the accuracy and reliability of the forecasting model by comparing the predicted values with the actual values for the test period. Various metrics, such as mean absolute error (MAE), root mean squared error (RMSE), or mean absolute percentage error (MAPE), can be used to quantify the model's performance.
- Model refinement: Iterate the forecasting process by refining the model, adjusting parameters, or considering different techniques if the performance is not satisfactory [7, 8]. Continuous improvement is essential for enhancing the accuracy of predictions over time.

It's important to note that time series forecasting is not a crystal ball that can predict the future with certainty. It relies on historical patterns and assumptions, and the accuracy of the forecasts depends on the quality of the data and the appropriateness of the chosen model. However, with careful analysis and modeling, time series forecasting can provide valuable insights for decision-making, planning, and resource allocation in various domains [9].

II. STATE-OF-THE-ART

Exponential Smoothing Methods:

- Hyndman, R. J., & Khandakar, Y. (2008) [10]. Automatic time series forecasting: the forecast package for R. This paper discusses various exponential smoothing methods and introduces the "forecast" package in R, providing a comprehensive toolset for forecasting.

Machine Learning and Deep Learning Approaches:

- Zhang, G., Patuwo, B. E., & Hu, M. Y. (1998) [11]. Forecasting with artificial neural networks: the state of the art. This paper provides an overview of neural network-based forecasting methods and their applications in time series prediction.
- Lipton, Z. C., Berkowitz, J., & Elkan, C. (2015) [12]. A critical review of recurrent neural networks for sequence learning. This review paper explores the effectiveness of recurrent neural networks, such as LSTM, in modeling and forecasting time series data.

Bayesian Approaches:

- Koopman, S. J., Shephard, N., & Creal, D. (2012) [13]. Dynamic factor models with macro, frailty, and industry effects for U.S. default count the credit crisis of 2008. This paper discusses Bayesian dynamic factor models and their use in forecasting default counts during the 2008 financial crisis.

Ensemble Forecasting:

- Raftery, A. E., Gneiting, T., Balabdaoui, F., & Polakowski, M. (2005) [14]. Using Bayesian model averaging to calibrate forecast ensembles. This paper introduces ensemble forecasting and Bayesian model averaging techniques to improve the accuracy and calibration of forecast ensembles.

Deep Learning for Time Series Forecasting:

- Oreshkin, B. N., et al. (2020) [15]. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. This paper presents the N-BEATS architecture, a deep learning model that offers interpretable time series forecasting while achieving state-of-the-art performance.

Forecast Combination:

- Hong, T., & Pinson, P. (2016) [16]. Probabilistic energy forecasting: Global energy forecasting competition 2014 and beyond. This paper discusses the Global Energy Forecasting Competition (GEFCom) and explores various approaches for combining forecasts to improve accuracy and reliability.

Applications in Finance:

- Chen, C. W. S., et al. (2018) [17]. Financial forecasting using empirical mode decomposition and neural networks. This study demonstrates the application of empirical mode decomposition and neural networks in financial time series forecasting.

Applications in Energy:

- Taylor, J. W. (2008) [18]. An evaluation of methods for very short-term load forecasting using minute-by-minute British data.

III. TIME SERIES TECHNIQUES

There are several techniques for time series analysis, each with its assumptions and strengths. Here are some commonly used techniques:

- Moving Averages: Moving averages involve calculating the average of a fixed number of previous observations to smooth out short-term fluctuations and highlight long-term trends [19, 20]. Simple Moving Average (SMA) uses an equal-weighted average of past observations, while Exponential Moving Average (EMA) assigns exponentially decreasing weights to older observations.
- Autoregressive Integrated Moving Average (ARIMA): ARIMA is a popular model for time series forecasting. It combines autoregressive (AR), differencing (I), and moving average (MA) components to capture the dependencies and patterns in the data. ARIMA models assume that the data is stationary (constant mean and variance) or can be made stationary through differencing [21].
- Seasonal ARIMA (SARIMA): SARIMA extends the ARIMA model to incorporate seasonal patterns in the data. It includes additional seasonal AR, seasonal differencing, and seasonal MA components. SARIMA models are useful for time series with recurring patterns that repeat at regular intervals.
- Exponential Smoothing Methods: Exponential smoothing methods, such as Simple Exponential Smoothing (SES), Holt's Linear Exponential Smoothing, and Holt-Winters' Seasonal Exponential Smoothing, are based on assigning exponentially decreasing weights to past observations [22]. These methods are suitable for data with trends and/or seasonality.
- Prophet: Prophet is a forecasting framework developed by Facebook. It incorporates seasonality, trend, and holiday effects to model time series data. Prophet is known for its flexibility, ease of use, and ability to handle outliers and missing data.

- Vector Autoregression (VAR): VAR models capture the interdependencies between multiple time series variables. They assume that each variable in the system is influenced by its past values as well as the past values of other variables [23]. VAR models are widely used in econometrics and macroeconomic forecasting.
- State Space Models: State-space models represent a time series as a combination of unobserved (latent) states and observed measurements. They can handle complex patterns, nonlinear relationships, and time-varying parameters. Kalman Filter and Particle Filter have commonly used algorithms for estimating the latent states in state space models.
- Neural Networks: Neural networks, such as Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, have gained popularity for time series analysis. These models can capture complex patterns and dependencies in the data and have shown success in various forecasting tasks.
- Spectral Analysis: Spectral analysis techniques, such as Fourier Transform and Wavelet Transform, analyze the frequency components of a time series [24]. They are useful for identifying periodic patterns, detecting seasonality, and extracting signals in the presence of noise.
- Bayesian Methods: Bayesian approaches incorporate prior beliefs and update them based on observed data to estimate the parameters and make predictions. Bayesian Structural Time Series (BSTS) models and Markov Chain Monte Carlo (MCMC) methods are commonly used in Bayesian time series analysis [25].

The choice of technique depends on the specific characteristics of the time series data, such as trend, seasonality, noise, and available historical information. It's often beneficial to try multiple techniques, compare their performance, and select the one that best fits the data and provides accurate forecasts.

ARIMA (Autoregressive Integrated Moving Average)

ARIMA is a popular time series forecasting model that combines autoregressive (AR), differencing (I), and moving average (MA) components. It is widely used for forecasting stationary. The ARIMA model is based on the assumption that future values of a time series can be predicted using a linear combination of its past values and past errors [26].

The ARIMA(p, d, q) model is defined by three parameters:

- p (order of autoregressive term): It represents the number of lagged observations of the dependent variable included in the model. The model uses the linear relationship between past observations and current observations.
- d (order of differencing): It represents the number of times the raw observations are differenced to make the series stationary. Differencing is used to remove trends and make the series stationary by eliminating the non-stationarity caused by trends or seasonality.
- q (order of moving average term): It represents the number of lagged forecast errors included in the model. The model uses the relationship between past forecast errors and current observations.

The general steps to apply ARIMA for time series forecasting are as follows:

- Data Preparation: Collect the time series data and preprocess it by handling missing values, outliers, and any other anomalies. Ensure that the data is in a suitable format for analysis.
- Stationarity Check: Assess the stationarity of the time series using statistical tests or visual inspection. Stationarity is a key assumption for ARIMA models. If the series is non-stationary, apply differencing to make it stationary [27].
- Parameter Selection: Determine the values of p, d, and q for the ARIMA model. This can be done by analyzing the autocorrelation function (ACF) and partial autocorrelation function (PACF) plots of the differenced series. These plots help identify the appropriate lag values for the autoregressive and moving average terms.
- Model Fitting: Fit the ARIMA model to the data by estimating the model parameters. This involves using optimization techniques to minimize the difference between the predicted values and the actual values in the training data.
- Model Diagnostic Checking: Assess the adequacy of the model by examining diagnostic plots of the residuals. Residuals should exhibit no significant patterns and should be approximately normally distributed with constant variance. If there are issues, consider refining the model or trying alternative models [28].
- Forecasting: Once the ARIMA model is validated, use it to make predictions for future periods. Generate forecasts based on the estimated model parameters and the available historical data.
- Model Evaluation: Assess the accuracy of the ARIMA model's forecasts by comparing them against the actual values for a validation dataset. Measure performance using metrics such as mean absolute error (MAE), root mean squared error (RMSE), or mean absolute percentage error (MAPE).
- Refinement and Iteration: Refine the ARIMA model by adjusting the model parameters, considering different lag values, or incorporating other techniques if the performance is not satisfactory. Iterate on the forecasting process to improve the accuracy of predictions [29].

ARIMA models have been widely used in various fields, including finance, economics, meteorology, and demand forecasting. They provide a flexible framework for time series analysis and forecasting, capturing both short-term and long-term dependencies in the data. However, it's important to note that ARIMA models assume linearity and may not perform well in the presence of complex non-linear relationships or irregular patterns in the data. In such cases, more advanced techniques like machine learning or deep learning models may be worth exploring.

SARIMA (Seasonal ARIMA)

SARIMA is an extension of the ARIMA model that incorporates seasonal patterns in the time series data. It is particularly useful when dealing with data that exhibits recurring patterns or seasonality [30].

The SARIMA model incorporates additional seasonal components in addition to the autoregressive (AR), differencing (I), and moving average (MA) components of the regular ARIMA model. It is denoted as SARIMA (p, d, q)(P, D, Q, s), where:

- p, d, q represent the non-seasonal AR, differencing, and MA orders, respectively.
- P, D, and Q represent the seasonal AR, differencing, and MA orders, respectively.

- s represents the seasonal period or the number of observations in one season.

The general steps to apply SARIMA for time series forecasting are similar to those of ARIMA:

- Data Preparation: Collect the time series data and preprocess it by handling missing values, outliers, and any other anomalies. Ensure that the data is in a suitable format for analysis.
- Stationarity Check: Assess the stationarity of the time series using statistical tests or visual inspection. If the series is non-stationary, apply differencing to make it stationary.
- Seasonal Differencing: Analyze the seasonal patterns in the data by differencing the series at the seasonal periods. If needed, apply seasonal differencing to remove seasonality and achieve stationarity.
- Parameter Selection: Determine the values of the non-seasonal and seasonal orders (p, d, q, P, D, Q) by analyzing the ACF and PACF plots of the differenced and seasonally differenced series.
- Model Fitting: Fit the SARIMA model to the data by estimating the model parameters. This involves using optimization techniques to minimize the difference between the predicted values and the actual values in the training data.
- Model Diagnostic Checking: Assess the adequacy of the model by examining diagnostic plots of the residuals, ensuring they exhibit no significant patterns and are approximately normally distributed with constant variance.
- Forecasting: Once the SARIMA model is validated, use it to make predictions for future periods. Generate forecasts based on the estimated model parameters and the available historical data.
- Model Evaluation: Assess the accuracy of the SARIMA model's forecasts by comparing them against the actual values for a validation dataset. Measure performance using metrics such as mean absolute error (MAE), root mean squared error (RMSE), or mean absolute percentage error (MAPE).
- Refinement and Iteration: Refine the SARIMA model by adjusting the model parameters, considering different lag values, or incorporating other techniques if the performance is not satisfactory.

SARIMA models can capture both non-seasonal and seasonal patterns in time series data, making them suitable for forecasting data with both short-term and long-term dependencies. They have been widely used in domains such as retail, finance, demand forecasting, and weather forecasting, where seasonal patterns play a significant role. However, as with any forecasting model, it is essential to interpret the results critically and validate the forecasts against real-world data to ensure their accuracy and reliability.

Exponential Smoothing Methods

Exponential smoothing methods are a family of forecasting techniques that assign exponentially decreasing weights to past observations to forecast future values. These methods are based on the principle that recent observations should carry more weight in the forecast than older observations. Exponential smoothing methods are known for their simplicity, computational efficiency, and ability to handle various types of time series data. Some commonly used exponential smoothing methods are mentioned as under [31].

- Simple Exponential Smoothing (SES): SES is the most basic form of exponential smoothing. It forecasts future values based on a weighted average of past observations, giving more weight to recent observations. SES uses a single smoothing parameter (alpha) that determines the weight assigned to the most recent observation. It is suitable for time series data without trend or seasonality.
- Holt's Linear Exponential Smoothing: Holt's linear method extends simple exponential smoothing by incorporating a trend component in the forecast. It includes two smoothing parameters: alpha for the level (the most recent observation) and beta for the trend (the rate of change). Holt's linear method is effective for time series data with a linear trend but without seasonality.
- Holt-Winters' Exponential Smoothing: Holt-Winters' method further extends exponential smoothing to handle time series data with both trend and seasonality. It includes three components: level (alpha), trend (beta), and seasonality (gamma). Holt-Winters' method uses a seasonal adjustment to account for repeated patterns at regular intervals. It can capture both short-term fluctuations and long-term trends in the data.
- Seasonal Exponential Smoothing: Seasonal exponential smoothing methods are variations of Holt-Winters' method that focus on capturing seasonality in the data. These methods employ seasonal adjustment factors or smoothing parameters to account for seasonal patterns. Examples include additive seasonal exponential smoothing and multiplicative seasonal exponential smoothing.
- Double and Triple Exponential Smoothing: Double exponential smoothing, also known as Brown's linear exponential smoothing, extends Holt's linear method to handle time series data with both level and trend components. Triple exponential smoothing, also called Holt-Winters' triple exponential smoothing, incorporates seasonality in addition to the level and trend components. These methods are useful when the data exhibit trend and seasonality but no clear patterns in the seasonal adjustment factors.

Exponential smoothing methods are relatively easy to implement and can provide accurate forecasts for time series data with appropriate parameter tuning. However, they assume that the future behavior of the series is solely influenced by its past values and do not consider other external factors. Thus, they may not be suitable for data with complex dependencies or significant external influences. It is essential to evaluate the performance of exponential smoothing methods using appropriate error metrics and consider alternative techniques if the data exhibit unique characteristics that are not adequately captured by exponential smoothing.

Prophet

Prophet is a time series forecasting framework developed by Facebook's Core Data Science team. It is designed to provide accurate and intuitive forecasting on various types of time series data, including those with trend changes, seasonality, and holiday effects. Prophet combines the flexibility of additive and multiplicative models with a component-wise approach to capture different patterns in the data. Some key features and characteristics of the Prophet are mentioned as under [32].

- Automatic Seasonality Detection: Prophet automatically detects and models various types of seasonality present in the data, including weekly, monthly, yearly, and sub-daily patterns. It can handle irregularly spaced time series and missing data.
- Trend Modeling: Prophet incorporates a flexible trend model that can capture both linear and non-linear trends. It uses a piecewise linear function to model trend changes and adapts to trend shifts in the data.

- Holiday Effects: Prophet allows the inclusion of holiday effects to account for the impact of holidays or special events on the time series. Users can provide a custom list of holidays or use built-in country-specific holiday calendars.
- Uncertainty Estimation: Prophet provides uncertainty intervals for the forecasted values, allowing users to assess the uncertainty associated with the predictions. These intervals are generated by incorporating the historical forecast error patterns into the model.
- Outliers and Change Points: Prophet can detect and handle outliers and change points in the data, enabling it to capture sudden shifts or anomalies in the time series. Change points are automatically selected based on a sparse Bayesian change point detection algorithm.
- Scalability and Ease of Use: Prophet is implemented in Python and offers a user-friendly API, making it accessible to users with varying levels of expertise. It is designed to handle large-scale forecasting problems efficiently and can be easily integrated into existing workflows.

Prophet has gained popularity due to its ability to produce accurate and interpretable forecasts while being relatively straightforward to use. It has found applications in various domains, including retail, finance, demand forecasting, and web analytics. However, it's important to note that while Prophet performs well in many scenarios, it may not capture more complex relationships or irregular patterns that require more advanced modeling techniques.

To use Prophet, one needs to install the Prophet package in Python and follow the API documentation and examples provided by the Prophet team. The package offers functions to fit the model, make forecasts, visualize the results, and access additional features like cross-validation and parameter tuning. Overall, Prophet is a powerful tool for time series forecasting, particularly when dealing with data that exhibit trends, seasonality, and holiday effects. It provides an accessible and reliable approach for generating accurate forecasts, making it a valuable asset for analysts and practitioners working with time series data.

VAR (Vector Auto Regression)

VAR is a statistical model used to analyze the interdependencies and dynamic relationships among multiple time series variables. Unlike univariate models that focus on a single variable, VAR models consider multiple variables simultaneously, allowing for the examination of their mutual influence over time. In a VAR model, each variable in the system is regressed on its own lagged values as well as the lagged values of all other variables in the system. The order of the VAR model, denoted as p, specifies the number of lagged observations included in the model. Thus, a VAR(p) model captures the relationships between variables based on past p observations [33].

The main steps involved in implementing a VAR model are as follows:

- Data Preparation: Gather the time series data for the variables of interest and ensure that it is stationary or transformed to achieve stationarity if required. Stationarity is essential for VAR models to provide meaningful results.
- Lag Selection: Determine the appropriate lag order, p, for the VAR model. This can be done by analyzing autocorrelation and partial autocorrelation functions (ACF and PACF) of the variables and considering information criteria such as Akaike Information Criterion (AIC) or Bayesian Information Criterion (BIC).
- Model Estimation: Estimate the coefficients of the VAR model using methods such as least squares, maximum likelihood estimation, or Bayesian estimation. This involves solving a system of equations that describe the relationships between the variables.
- Model Diagnostic Checking: Evaluate the adequacy of the VAR model by examining diagnostic tests and residual analysis. Common diagnostics include testing for autocorrelation in the residuals, assessing the normality of residuals, and conducting stability tests.
- Impulse Response Analysis: Analyze the dynamic responses of the variables to shocks in the system. Impulse response analysis helps understand the short-term and long-term effects of shocks on the variables of interest.
- Forecasting: Generate forecasts for the variables using the estimated VAR model. Forecasting can be done by recursively applying the estimated model to obtain future values of the variables.

VAR models have several advantages. They allow for the examination of complex interactions among variables, capture the dynamic nature of relationships, and provide flexibility in forecasting multiple variables simultaneously. VAR models have been widely used in various fields, including macroeconomics, finance, and econometrics. However, VAR models also have some limitations. They assume linearity, stationarity, and constant coefficients over time, which may not hold in all cases. Additionally, VAR models can be sensitive to the choice of lag order, and their interpretation can be challenging when dealing with a large number of variables.

State Space Models

State Space Models (SSMs) are a flexible and powerful class of statistical models used to represent and analyze complex time series data. SSMs consist of two main components: a state equation and an observation equation. The state equation describes the underlying dynamics of the system by specifying how the unobserved states evolve. It is typically represented as a linear or nonlinear stochastic equation, where the current state depends on the previous state and some random noise. The state equation can capture various patterns, such as trends, seasonality, and dynamic relationships between variables. The observation equation links the unobserved states to the observed data. It defines the relationship between the observed variables and the underlying states. The observation equation can be linear or nonlinear, and it incorporates measurement errors or disturbances that account for the discrepancy between the observed data and the true states. SSMs have several advantages that make them suitable for time series analysis [34]:

- Flexibility: SSMs can handle a wide range of time series data, including univariate, multivariate, continuous, discrete, and irregularly spaced observations. They provide a flexible framework to model complex dependencies and relationships among variables.
- Missing Data: SSMs can effectively handle missing data and irregular observation times. By jointly modeling the states and observations, SSMs can incorporate available information from the observed data and make predictions or imputations for the missing values.

- Forecasting and Filtering: SSMs enable accurate forecasting and filtering of the underlying states based on observed data. They can provide estimates of the current state based on all available data up to that point, incorporating information from both past observations and current measurements.
- Parameter Estimation: SSMs allow for the estimation of model parameters using various estimation techniques, such as maximum likelihood estimation, Bayesian inference, or filtering-based approaches like the Kalman filter. This enables the extraction of meaningful insights from the data and the estimation of unknown parameters in the model.
- Model Diagnostics: SSMs provide diagnostic tools to assess the goodness of fit and identify model deficiencies. Residual analysis and model comparison techniques help evaluate the adequacy of the chosen model and identify areas for improvement.

SSMs have applications in various fields, including economics, finance, engineering, biology, and environmental sciences. They have been used for tasks such as time series forecasting, anomaly detection, signal processing, and system identification. Popular methods and algorithms for working with SSMs include the Kalman filter, extended Kalman filter (EKF), unscented Kalman filter (UKF), particle filters (PF), and Markov Chain Monte Carlo (MCMC) methods for Bayesian inference. Implementing SSMs often requires specialized software libraries or packages, such as the *statsmodels* or *pykalman* libraries in Python, or the *dlm* package in R. These tools provide functions and algorithms specifically designed for modeling and analyzing state space models.

Neural Networks

Neural networks are a class of machine learning models inspired by the structure and functionality of the human brain. They are widely used for a variety of tasks, including pattern recognition, classification, regression, and time series analysis. A neural network consists of interconnected nodes called neurons, organized in layers. The most common type of neural network is a feedforward neural network, where information flows in one direction, from the input layer to the output layer. Each neuron receives inputs, performs a computation, and produces an output, which is passed to the next layer. The layers between the input and output layers are called hidden layers. The key components of a neural network are:

- Input Layer: The input layer receives the initial data or features to be processed by the network. Each neuron in the input layer represents a specific feature or input variable.
- Hidden Layers: The hidden layers, located between the input and output layers, perform computations and transform the input data. Each neuron in a hidden layer receives inputs from the previous layer, applies an activation function, and passes the output to the next layer.
- Output Layer: The output layer produces the final output or prediction of the neural network. The number of neurons in the output layer depends on the task at hand. For example, in a classification problem with multiple classes, the output layer may have one neuron per class.
- Neuron Activation Function: Each neuron applies an activation function to the weighted sum of its inputs. Common activation functions include the sigmoid function, the hyperbolic tangent function, and the rectified linear unit (ReLU) function. Activation functions introduce non-linearities and enable the network to learn complex relationships in the data.
- Connections and Weights: Neurons are connected by weighted connections. Each connection between neurons has an associated weight, which determines the strength of the connection. During the training process, these weights are adjusted to minimize the difference between the network's predictions and the true values.
- Loss Function: The loss function quantifies the difference between the predicted output and the true output. The network aims to minimize this loss function during training by adjusting the weights to improve its predictions.
- Training and Optimization: Neural networks are trained using optimization algorithms such as stochastic gradient descent (SGD) or its variants. The training process involves forward propagation of inputs, computing the output, comparing it with the true output, and then backpropagating the error to update the weights to minimize the loss.
- Regularization Techniques: To prevent overfitting and improve generalization, various regularization techniques are used in neural networks. Common techniques include dropout, L1 and L2 regularization, early stopping, and batch normalization.

Neural networks can be implemented using various deep learning frameworks, such as TensorFlow, PyTorch, or Keras, which provide high-level APIs and tools to build, train, and deploy neural network models efficiently. These frameworks offer a wide range of neural network architectures, including convolutional neural networks (CNNs) for image data, recurrent neural networks (RNNs) for sequential data, and transformer networks for natural language processing tasks. Neural networks have achieved remarkable success in many areas, including computer vision, natural language processing, speech recognition, and time series analysis. They can learn complex patterns and relationships in data and provide highly accurate predictions when trained on large and diverse datasets. However, neural networks can be computationally intensive and require substantial amounts of training data and computational resources for optimal performance.

Spectral Analysis

Spectral analysis is a technique used to analyze the frequency content or spectral characteristics of a signal or time series data. It involves decomposing a signal into its constituent frequencies to understand the underlying patterns and properties. The Fourier Transform is a fundamental tool in spectral analysis. It converts a time-domain signal into its frequency-domain representation, providing information about the amplitude and phase of different frequency components present in the signal. The Fast Fourier Transform (FFT) is an efficient algorithm commonly used to compute the Fourier Transform. Some key concepts and techniques in spectral analysis are mentioned as under.

- Power Spectrum: The power spectrum represents the distribution of power or energy across different frequencies in a signal. It provides information about the strength or magnitude of each frequency component. The power spectrum is often computed by taking the squared magnitude of the Fourier Transform.
- Periodogram: A periodogram is an estimate of the power spectrum of a signal. It is obtained by calculating the squared magnitude of the Fourier Transform or FFT of the signal. The periodogram provides insights into the dominant frequencies and their power in the signal.

- Windowing: Windowing is a technique used to reduce the leakage effect in spectral analysis. When analyzing a finite-length signal, the Fourier Transform assumes that the signal is periodic, leading to spectral leakage. Windowing involves applying a window function to the signal before computing the Fourier Transform. Common window functions include the Hamming, Hanning, and Blackman windows.
- Spectrogram: A spectrogram is a visualization of the time-varying power spectrum of a signal. It represents the changes in frequency content over time. The spectrogram is obtained by applying the Fourier Transform to short segments of the signal, resulting in a time-frequency representation.
- Welch's Method: Welch's method is a commonly used technique to compute the power spectrum or periodogram of a signal. It involves dividing the signal into overlapping segments, computing the periodogram for each segment, and averaging them to obtain a smoother estimate of the power spectrum.
- Autocorrelation Function: The autocorrelation function measures the correlation between a signal and a lagged version of itself at different time lags. It provides information about the repetitive patterns or periodicities in the signal. The Fourier Transform of the autocorrelation function is the spectral density function.
- Spectral analysis has applications in various fields, including signal processing, telecommunications, physics, astronomy, and geophysics. It is used for tasks such as detecting periodicities, identifying dominant frequencies, noise analysis, filtering, and anomaly detection.

Several software packages provide tools for spectral analysis, including MATLAB, Python's NumPy and SciPy libraries, and R's signal and spec packages. These packages offer functions and methods to compute power spectra, periodograms, and spectrograms, and perform various spectral analysis techniques.

Bayesian Methods

Bayesian methods are a class of statistical techniques that utilize Bayesian inference to analyze and make inferences about unknown parameters or variables in a model. These methods are based on Bayes' theorem, which allows for the updating of prior beliefs or knowledge in light of observed data. In Bayesian inference, the probability is assigned to both the unknown parameters (prior distribution) and the observed data (likelihood function). The goal is to obtain the posterior distribution, which represents the updated probability distribution of the parameters given the observed data. The posterior distribution incorporates both the prior beliefs and the information contained in the data, providing a comprehensive and coherent framework for inference. Some key concepts and techniques in Bayesian methods are mentioned as under.

- Prior Distribution: The prior distribution represents the initial beliefs or knowledge about the unknown parameters before observing any data. It encapsulates any information available from previous studies, expert opinions, or prior data. The choice of the prior can have an impact on the posterior distribution and subsequent inferences.
- Likelihood Function: The likelihood function captures the probability of observing the data given the values of the unknown parameters. It quantifies the relationship between the data and the parameters and forms the basis for updating the prior beliefs.
- Posterior Distribution: The posterior distribution is the updated probability distribution of the parameters after incorporating the observed data. It is obtained by applying Bayes' theorem, which involves multiplying the prior distribution by the likelihood function and normalizing it to ensure that the posterior is a valid probability distribution.
- Markov Chain Monte Carlo (MCMC): MCMC methods, such as the Metropolis-Hastings algorithm and the Gibbs sampler, are commonly used in Bayesian inference. MCMC methods generate a sequence of samples from the posterior distribution, allowing for approximate estimation and inference in complex models.
- Model Comparison: Bayesian methods provide a natural framework for model comparison and selection. The posterior probabilities of different models can be computed, taking into account the data and prior beliefs. This allows for the quantification of the evidence in favor of one model over another.
- Bayesian Predictive Inference: Bayesian methods enable predictive inference by integrating the uncertainty in the parameters. The posterior distribution can be used to make predictions for new observations or future outcomes, incorporating both the observed data and the uncertainty in the parameters.

Bayesian methods have several advantages, including their ability to incorporate prior information, handle small sample sizes, and provide a coherent framework for inference. They also offer a flexible approach to dealing with complex models, hierarchical structures, and missing data. Implementing Bayesian methods often involves using software packages or libraries specifically designed for Bayesian inference, such as Stan, PyMC3, or JAGS. These tools provide functionalities for specifying models, sampling from the posterior distribution using MCMC, performing model diagnostics, and conducting inference. Bayesian methods are widely used in various fields, including statistics, machine learning, economics, finance, biology, and engineering. They have applications in parameter estimation, hypothesis testing, model fitting, decision analysis, and prediction.

IV. CONCLUSION

Each technique has its advantages and limitations, considering factors such as computational complexity, data requirements, interpretability, and the presence of external factors or seasonality. It is crucial to choose an appropriate technique based on the specific requirements and characteristics of the dataset at hand. The research paper discussed time series forecasting techniques in detail. Traditional statistical methods, such as ARIMA, SARIMA, and Exponential Smoothing, offer a solid foundation for time series analysis. They are relatively simple, interpretable, and effective at capturing different patterns in the data. These methods are particularly suitable for datasets with well-defined trends, seasonality, and stationary properties. The Bayesian method of time series forecasting is well-suited for situations where prior knowledge or expert opinions about the data and parameters are available. Spectral analysis of time series data is best suited in scenarios where the frequency content and periodicities in the data are of interest. State space models are particularly suited for time series forecasting in situations where the underlying process is dynamic, complex, and subject to various sources of uncertainty. Neural networks are well-

suited for time series forecasting in various contexts, particularly when dealing with complex and non-linear relationships within the data. Vector Autoregression (VAR) models are well-suited for time series forecasting in situations where multiple interrelated variables influence each other's behavior over time.

REFERENCES

[1]. Hu, Z.; Zhao, Y.; Khushi, M. A Survey of Forex and Stock Price Prediction Using Deep Learning. *Appl. Syst. Innov.* 2021, *4*, 9. [Google Scholar] [CrossRef]

[2]. Hyun, A.; Kyunghee, S.; Kwanghoon, P.K. Comparison of Missing Data Imputation Methods in Time Series Forecasting. *Comput. Mater. Contin.* 2022, *70*, 423–431. [Google Scholar] [CrossRef]

[3]. Wu, X.; Mattingly, S.; Mirjafari, S.; Huang, C.; Chawla, N.V. Personalized Imputation on Wearable Sensory Time Series via Knowledge Transfer. In Proceedings of the 29th ACM International Conference on Information & Knowledge Management, Virtual Event, Ireland, 19–23 October 2020; pp. 1625–1634. [Google Scholar] [CrossRef]

[4]. Ramin Fadaei, F.; Orhan, E.; Emin, A. A DDoS attack detection and defense scheme using time-series analysis for SDN. *J. Inf. Secur. Appl.* 2020, *54*, 102587. [Google Scholar] [CrossRef]

[5]. Dwivedi, S.A.; Attry, A.; Parekh, D.; Singla, K. Analysis and forecasting of Time-Series data using S-ARIMA, CNN, and LSTM. In Proceedings of the 2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), Greater Noida, India, 19–20 February 2021; pp. 131–136. [Google Scholar]

[6]. Bendong, Z.; Lu, H.; Chen, S.; Liu, J.; Wu, D. Convolutional neural networks for time series classification. *J. Syst. Eng. Electron.* 2017, *28*, 162–169. [Google Scholar] [CrossRef]

[7]. Han, Z.; Zhao, J.; Leung, H.; Ma, K.F.; Wang, W. A Review of Deep Learning Models for Time Series Prediction. *IEEE Sens. J.* 2021, *21*, 7833–7848. [Google Scholar] [CrossRef]

[8]. Omer Berat, S.; Mehmet Ugur, G.; Ahmet Murat, O. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Appl. Soft Comput. J.* 2020, *90*, 106181. [Google Scholar] [CrossRef][Green Version]

[9]. Soheila Mehr, M.; Mohammad Reza, K. An analytical review for event prediction system on time series. In Proceedings of the 2nd International Conference on Pattern Recognition and Image Analysis, Rasht, Iran, 11–12 March 2015. [Google Scholar]

[10]. Rob J. Hyndman and Yeasmin Khandakar, "Automatic Time Series Forecasting: The forecast Package for R," *J. Stat. Softw.*, vol. 27, no. 3, p. 22, 2008, [Online]. Available: http://www.jstatsoft.org/%0Ahttp://www.jstatsoft.org/v27/i03/paper.

[11]. G. Zhang, B. Eddy Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks: The state of the art," *Int. J. Forecast.*, vol. 14, no. 1, pp. 35–62, 1998, doi: 10.1016/S0169-2070(97)00044-7.

[12]. Z. C. Lipton, J. Berkowitz, and C. Elkan, "A Critical Review of Recurrent Neural Networks for Sequence Learning," pp. 1–38, 2015, [Online]. Available: http://arxiv.org/abs/1506.00019.

[13]. D. Creal, S. J. Koopman, and A. Lucas, "Generalized autoregressive score models with applications," *J. Appl. Econom.*, vol. 28, no. 5, pp. 777–795, 2013, doi: 10.1002/jae.1279.

[14]. A.E. Raftery, T. Gneiting, F. Balabdaoui, and M. Polakowski, "Using Bayesian model averaging to calibrate forecast ensembles," *Mon. Weather Rev.*, vol. 133, no. 5, pp. 1155–1174, 2005, doi: 10.1175/MWR2906.1.

[15]. A.N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting," pp. 1–31, 2019, [Online]. Available: http://arxiv.org/abs/1905.10437.

[16]. T. Hong, P. Pinson, S. Fan, H. Zareipour, A. Troccoli, and R. J. Hyndman, "Probabilistic energy forecasting: Global Energy Forecasting Competition 2014 and beyond," *Int. J. Forecast.*, vol. 32, no. 3, pp. 896–913, 2016, doi: 10.1016/j.ijforecast.2016.02.001.

[17]. M. C. Dong, S. Tian, and C. W. S. Chen, "Predicting failure risk using financial ratios: Quantile hazard model approach," *North Am. J. Econ. Financ.*, vol. 44, no. August 2017, pp. 204–220, 2018, doi: 10.1016/j.najef.2018.01.005.

[18]. J. W. Taylor, "Estimating value at risk and expected shortfall using expectiles," *J. Financ. Econom.*, vol. 6, no. 2, pp. 231–252, 2008, doi: 10.1093/jjfinec/nbn001.

[19]. Kim, K.J. Financial time series forecasting using support vector machines. *Neurocomputing* 2003, *55*, 307–319. [Google Scholar] [CrossRef].

[20]. Lai, R.K.; Fan, C.Y.; Huang, W.H.; Chang, P.C. Evolving and clustering fuzzy decision tree for financial time series data forecasting. *Expert Syst. Appl.* 2009, *4*, 3761–3773. [Google Scholar] [CrossRef]

[21]. Shen, L.; Loh, H.T. Applying rough sets to market timing decisions. *Decis.Support Syst.* 2004, *37*, 583–597. [Google Scholar] [CrossRef]

[22]. Patel, J.; Shah, S.; Thakkar, P.; Kotecha, K. Predicting stock market index using fusion of machine learning techniques. *Expert Syst. Appl.* 2015, *42*, 2162–2172. [Google Scholar] [CrossRef]

[23]. Zhang, X.; Shi, J.; Wang, D.; Fang, B. Exploiting investors social network for stock prediction in China's market. *J. Comput. Sci.* 2018, *28*, 294–303. [Google Scholar] [CrossRef][Green Version]

[24]. Kumar, I.; Dogra, K.; Utreja, C.; Yadav, P. A Comparative Study of Supervised Machine Learning Algorithms for Stock Market Trend Prediction. In Proceedings of the 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), Coimbatore, India, 20–21 April 2018; pp. 1003–1007. [Google Scholar] [CrossRef]

[25]. Zhou, F.; Zhou, H.m.; Yang, Z.; Yang, L. EMD2FNN: A strategy combining empirical mode decomposition and factorization machine based neural network for stock market trend prediction. *Expert Syst. Appl.* 2019, *115*, 136–151. [Google Scholar] [CrossRef]

[26]. Chen, K.; Zhou, Y.; Dai, F. A LSTM-based method for stock returns prediction: A case study of China stock market. In Proceedings of the 2015 IEEE International Conference on Big Data, Santa Clara, CA, USA, 29 October–1 November 2015; pp. 2823–2824. [Google Scholar] [CrossRef]

[27]. Persio, L.D.; Honchar, O. Recurrent neural networks approach to the financial forecast of Google assets. *Int. J. Math. Comput. Simul.* 2017, *11*, 1–7. [Google Scholar]

[28]. Blem, M.; Cristaudo, C.; Moodley, D. Deep Neural Networks For Online Trend Prediction. In Proceedings of the 2022 25th International Conference on Information Fusion (FUSION), Linköping, Sweden, 4–7 July 2022; pp. 1–8. [Google Scholar] [CrossRef]

[29]. Chandola, D.; Mehta, A.; Singh, S.; Tikkiwal, V.A.; Agrawal, H. Forecasting Directional Movement of Stock Prices using Deep Learning. *Ann. Data Sci.* 2022. [Google Scholar] [CrossRef]
[30]. Kittisak, P.; Peerapon, V. Stock Trend Prediction Using Deep Learning Approach on Technical Indicator and Industrial Specific Information. *Information* 2021, *12*, 250. [Google Scholar] [CrossRef]
[31]. Liu, S.; Zhang, C.; Ma, J. CNN LSTM Neural network model for quantitative strategy analysis in stock markets. In *Neural Information Processing*; Springer International Publishing: Berlin, Germany, 2017; pp. 198–206. [Google Scholar] [CrossRef]
[32]. Batres-Estrada, B. Deep Learning for Multivariate Financial Time Series. Master's Thesis, KTH, Mathematical Statistics , Stockholm, Sweden, 2015. [Google Scholar]
[33]. Wang, K.; Li, K.; Zhou, L.; Hu, Y.; Cheng, Z.; Liu, J.; Cen, C. Multiple convolutional neural networks for multivariate time series prediction. *Neurocomputing* 2019, *360*, 107–119. [Google Scholar] [CrossRef]
[34]. Lian, L.; Tian, Z. A novel multivariate time series combination prediction model. *Commun. Stat.-Theory Methods* 2022, 1–32. [Google Scholar] [CrossRef]

About the Authors

Harpinder Kaur is currently serving in the capacity of Assistant Professor at Guru Gobind Khalsa College, Bhagta Bhai Ka, Bathinda, Punjab, India. She has a teaching experience of 12 years and have 3 international research papers published to her credit. Her field of research is Time Series Forecasting.

Dr. Gagandeep Jagdev is currently serving in the capacity of Technical Officer at Punjabi University Guru Kashi Campus, Damdama Sahib (PB). His total teaching and research experience is more than 16 years and has 153 International and National publications in reputed journals and conferences to his credit. He is the guest editor of the Journal of Imaging, MDPI. He is also a member of the editorial board of several reputed International Journals indexed in ESCI, Scopus, ACM, WoS, and Pubmeds and has been an active Technical Program Committee member of several International and National conferences conducted by renowned universities and academic institutions. He has been bestowed with Best Research Paper awards 4 times by NITTTR (Chandigarh), Government College of Engineering and Technology (Jammu), and Guru Nanak College, Budhlada (PB). He has actively participated in more than 80 Webinars and FDPs. His field of expertise is Image Processing, Big Data Analytics, Data Science, Cloud Computing, Cloud Security, Cryptography, and WANETs.