Analysis of various Machine Learning Models for Time Series Forecasting

By Aastha Bharill

Problem Statement

The primary goal of this project is to determine which hypothesis classes—linear models, decision trees, or neural networks—are most effective for time series prediction. Specifically, the focus is on predicting the next value in a sequence given the previous k values in a time series dataset. This research aims to identify the hypothesis class that provides the highest accuracy and reliability in forecasting future values based on historical data. Time series prediction in this project is formulated as a supervised regression problem, where the relationship $\boldsymbol{x}_{t-k}, \boldsymbol{x}_{t-k+1}, \cdots, \boldsymbol{x}_{t-1} \rightarrow \boldsymbol{x}_t$ is to be modelled.

The hypothesis for this project is that "Neural networks outperform linear models and decision trees in time series prediction accuracy across diverse datasets characterized by different levels of complexity and noise."

The problem of time series forecasting is more complex than most other Machine Learning problems. Time series forecasting presents unique challenges due to the inherent temporal dependencies and order in the data, where past events significantly influence future outcomes. Unlike other types of data that may assume independence among observations, time series data exhibits autocorrelation, seasonality, and potential non-stationarity—characteristics that require specialized modelling approaches. Models must accurately capture these dynamics and the possibility of trend shifts or distribution changes over time. Additionally, noise, outliers, and missing values add complexity, demanding robust and adaptive algorithms.

We assume this hypothesis because neural networks are known to capture cyclic and seasonal data inherent to time series data while decision trees and linear models are unable to capture these trends properly. Additionally, we have taken special consideration of these factors and explore their impact on prediction accuracies on different hypothesis classes. We have analysed the root cause of errors that are most prevalent in each hypothesis class' predictions and provided suggestions for improvement.

Methodology

We considered two different datasets of varying properties to clearly show the difference between performance, and how those properties impact it. The first dataset under consideration consists of approximately 150,000 data points that span over ten years of *hourly* energy consumption data in Megawatts sourced from PJM and the second dataset consists of 4 years data of *daily* Delhi climate and consists of multiple variables. However, we used the 'meantemp' data to predict the temperature values. The major differences between these datasets is that the first one is 100 times bigger then the second and so provides more information to train on and is also stationary, whereas the second is non-stationary - this will reveal how the models fare on both kinds of data.

In the preprocessing phase, duplicate entries were aggregated to maintain data consistency and integrity. An interquartile range (IQR) method was employed to identify and remove outliers; this was necessary to minimize noise and potential biases that could affect the forecasting accuracy. Values falling outside the bounds (Q1-1.5*IQR) and (Q3+1.5*IQR) were treated as outliers. For the first dataset, there were 3000 missing values after removing outliers but given the substantial size of the dataset, these missing values were deemed insignificant and were interpolated linearly, using the sequential nature of time-series data.
To confirm the applicability of statistical modelling techniques, verifying the stationarity of the time series and understanding the autocorrelation structure was imperative. Stationarity was assessed visually using rolling window statistics and quantitatively via the Dickey-Fuller Test. The latter provides a rigorous measure for the presence of unit roots which indicate non-stationarity. Autocorrelation was examined through the Autocorrelation Function (ACF) and the Partial Autocorrelation Function (PACF). The ACF helps in identifying the overall autocorrelation pattern, whereas the PACF isolates the correlation of a lag with the current value, controlling for the contributions of the intermediate lags. This distinction is essential for distinguishing genuine predictors from spurious ones. Normalization was another critical step undertaken to scale the dataset within the range of [0, 1]. This process is pivotal as it normalizes all variables to a similar scale, reducing bias towards variables with larger magnitudes and facilitating faster convergence during the model training phase.

The forecasting models selected were ElasticNet Linear Regression, XGBoost Decision Trees, and Long Short-Term Memory (LSTM) Networks. Linear Regression was used as a baseline model to capture any linear relationships in the data. ElasticNet was specifically used instead of traditional linear regression to incorporate L1 and L2 regularization and improve generalizability. XGBoost Decision Trees, which are considered an improvement over traditional decision trees through the application of a gradient boosting framework, give us superior handling of non-linear relationships. LSTMs, a type of recurrent neural network, capture cyclical and seasonal trends inherent to time series data with great accuracy and detail, often beating decision trees and linear models.

The inherent sequential nature of time series data makes traditional random or stratified splits inappropriate due to the potential for leakage and bias. To counteract this, an Expanding Window strategy was employed for model training. This technique not only preserves the temporal order of data but also ensures that each successive training set is an extension of the previous set, thus accommodating the cumulative nature of time-bound data and capturing seasonal patterns and trends more effectively. We preferred using this over Rolling Window as due to the strong seasonal cycles present in the data, using as many past historical data benefits the training.

Given the complexity and size of the dataset, Bayesian Optimization was preferred over GridSearchCV and RandomizedSearch CV for hyperparameter tuning. This method capitalizes on prior knowledge of hyperparameter performance, iteratively refining this knowledge, thus optimizing the selection process in a more efficient manner than exhaustive search methods. We also added time-based lag features such as hour, day of the week, quarter, month, year, day of the year, day of the month, and week of the year using feature engineering to capture cyclical behaviors in energy consumption.

The models were trained using the root mean squared loss function to minimize prediction errors. However, the Mean Absolute Percentage Error (MAPE) was utilized as the primary metric for evaluating model performance. MAPE offers a clear, percentage-based measure that facilitates straightforward comparisons across different models and datasets, proving particularly useful in contexts requiring interpretability.

Experimental Results and Validation

The preprocessing stage of the data revealed a lack of trend, strong seasonality, presence of autocorrelation and a pattern in the residuals after trend and seasonality were removed for both the datasets. The Power Consumption Dataset (Dataset 1) is stationary and the Delhi Weather is non-stationary as proved by the ADF Test. For both the datasets, the observed plot shows the actual time series data. (See Fig. 1 and Fig. 3) They exhibit clear cyclic patterns that suggest seasonality within the data. In the trend plots, there's no clear increasing or decreasing linear trend. The seasonal plots are constant over the entire time series, indicating a stable and consistent seasonal pattern. (For the first dataset, since the plot of the seasonal component is condensed due to the long time frame, the details of the seasonal pattern are not visible and a more detailed view is shown beside separately (See Fig. 2)). The residual plots (See Fig. 2 and 3) shows what remains after the seasonal and trend components have been removed from the observed data. Ideally, this would look like white noise if the seasonal and trend components have been fully captured. However, there seems to be some structure left in the residuals, indicating that all the systematic information may not have been fully accounted for by the trend and seasonal components alone. The second dataset also exhibits non-stationarity that is shown in the Rolling Mean and Standard Deviation Plot (See Fig. 4) and was confirmed by running the ADF Test, These factors will impact the performance of the models in their predictions.

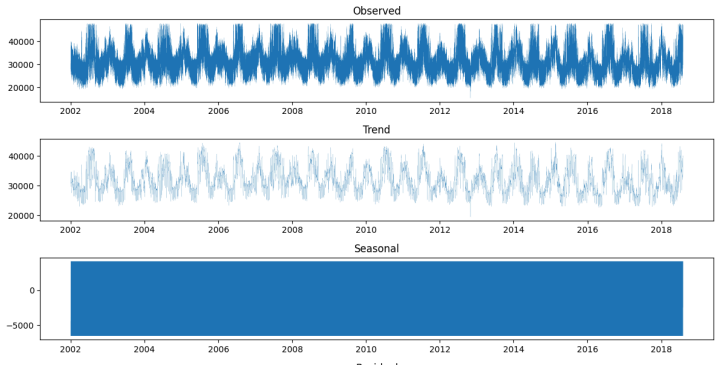


Fig. 1: Observed, Trend and Seasonal Components

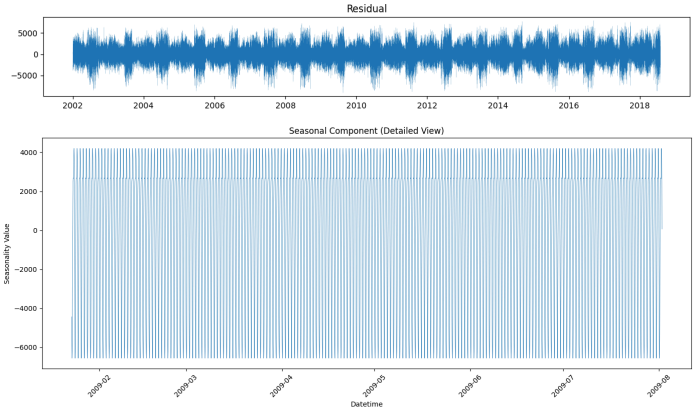


Fig. 2: Residual and Seasonal (Detailed) Plots

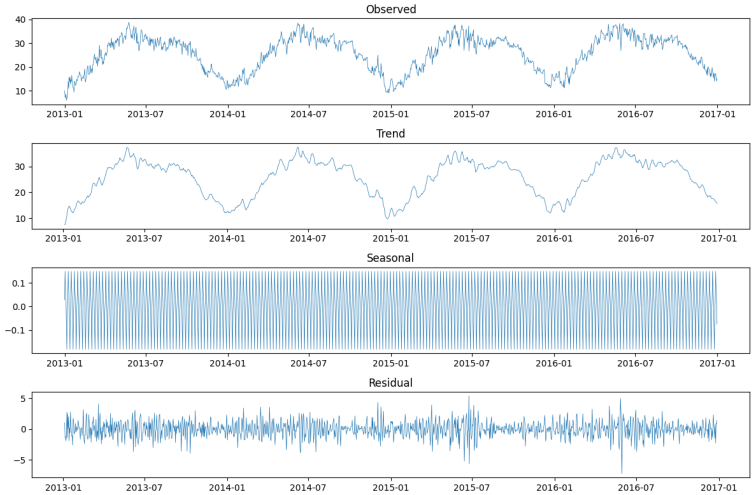


Fig. 3: Observed, Trend and Seasonal Components

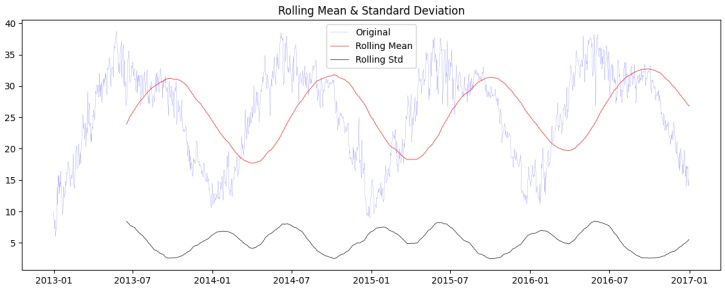


Fig. 4: Rolling Mean and Std. Dev.

For Autocorrelation in the first dataset, the ACF and PACF plots show spikes for lag values ‘k’ that are multiples of 24 that is in line with strong correlation and seasonality at those lags. This can be attributed to the fact that our dataset is an hourly time series dataset and the daily seasonality is being shown. The spikes seem to decrease in magnitude as the lags increase before rising again, which might indicate a dampening effect over time on the influence of past values. We have used this value of 24 as the lag value ‘k’ to be taken while feeding the models. Similarly, for dataset 2 the ACF plot ‘elbows’ or reaches the confidence level of 95% around 60, which is what we used for training.

In our model, we tailored our approach to data normalization based on the specific requirements of each dataset and model. For dataset one, we normalized all features across all models—XGBoost, linear regression, and LSTMs—due to challenges with optimization convergence. However, for dataset two, we adopted a more selective strategy. While we normalized the time series data for LSTMs, we did not normalize features for other models. This decision was informed by considerations such as the distinct characteristics of each dataset, the suitability of each model to handle scaled data, and the optimization of overall performance. By selectively applying normalization, we aimed to balance model convergence and performance optimization while preserving the unique characteristics of the dataset and modelling techniques employed.

In our time series prediction efforts, we encountered a challenge where the model excessively relied on the last lag, resulting in minimal loss values and potential overfitting. To address this issue, we introduced ‘gaps’ in both training and testing datasets. This approach aimed to encourage the model to generalize across a wider temporal context by disrupting its dependence on immediate past values. Specifically, we employed a gap of 7 between consecutive data points in the first dataset, alongside utilizing 25 lags for the second dataset. These gap values were chosen after observing the seasonal patterns in the datasets and ensuring the gap is such that predictions using just the last lag wouldn’t give good results. The first dataset showed hourly patterns and so we wanted to avoid multiples of 12, and we employed a similar approach for the second. Introducing gaps enhanced the generalizing capabilities of our model, as opposed to the previous performance, that even though gave MAPE values of as low as 1% was extremely unreliable and misleading. Specifically in XGBoost, adopting a forecasting approach for unseen data and calculating around a 50% loss provides a more realistic assessment of predictive performance on new data points, enhancing understanding of real-world applicability and robustness.

We now present the comparison and analysis of the prediction accuracy of three different models—LSTM, XGBoost, and Linear Regression—using the Mean Absolute Percentage Error (MAPE) as the evaluation metric on two datasets.

Model	Dataset1	Dataset2
LSTM	3.27%	7.576%
XGBOOST	18.029%	20.367%
Linear Regression	23.176%	40.970%

ElasticNet Linear Regression

Linear Regression exhibited the highest MAPE values among the three models, indicating the poorest prediction performance. This can be attributed to several factors inherent to the nature of linear regression: Limited Flexibility and Linearity Assumption: Given that Dataset 1 is stationary, linear regression still performed poorly due to its inability to model any non-linear relationships that might exist

even in stationary data. For Dataset 2, which is non-stationary, the performance worsened as the assumption of linearity and constant variance (homoscedasticity) fails, leading to inaccurate and unreliable predictions. Sensitivity to Outliers and Multicollinearity: The higher error in Dataset 2 could also be due to outliers influencing the regression line significantly, compounded by potential multicollinearity among predictors.

XGBoost

XGBoost performed better than linear regression but was not as effective as LSTM. The key advantages of XGBoost include: Handling Non-Linear Relationships and Regularization: These features likely contributed to its better performance on both datasets compared to linear regression. However, XGBoost's limitations in capturing long-term dependencies and handling irregular data could explain why it did not perform as well as LSTM, especially in Dataset 2 where the non-stationarity introduces complex patterns and dependencies that extend beyond short-term lags.

LSTM

LSTM showed the best performance, particularly on Dataset 2, as evidenced by the lowest MAPE score. The architecture of LSTMs offers several advantages for time series data: Handling Long-Term Dependencies and Temporal Order: LSTMs are particularly suited for time series data like Dataset 2, where understanding long-term historical context is crucial for accurate forecasting. Their ability to remember and forget information selectively allows them to perform well even in non-stationary environments. Robustness to Noisy Data: This characteristic makes LSTM superior in handling the irregularities and potential noise in non-stationary data.

To conclude, the evaluation of model performances using MAPE highlights the critical importance of choosing the right model based on the characteristics of the dataset. LSTM's superior ability to handle complex and long-term dependencies in time series data makes it the most suitable model for non-stationary datasets. In contrast, models like Linear Regression are less effective due to their restrictive assumptions and inability to manage non-linearities and outliers. XGBoost, while robust in handling various data issues, falls short in capturing longer-term dependencies as effectively as LSTM. This analysis underscores the necessity of aligning model capabilities with data characteristics for optimal forecasting accuracy.

Conclusion and Future Work

This study meticulously evaluated three hypothesis classes— ElasticNet Linear Regression, XGBoost Decision Trees, and LSTM networks—across two distinct datasets to ascertain their efficacy in time series forecasting. The results highlighted the superior performance of LSTM models, particularly in handling non-stationary data and capturing long-term dependencies, which are crucial in time series analysis. Linear Regression, while serving as a baseline, demonstrated significant limitations due to its inability to model non-linear relationships and its sensitivity to outliers, which were pronounced in the non-stationary Delhi Climate dataset. XGBoost, though better than Linear Regression, still could not match the LSTM's performance in datasets with complex patterns and long-term dependencies.

The study proved that neural networks, specifically LSTM, have a distinct advantage in forecasting time series data, especially when the data exhibits non-linearity, seasonality, and substantial historical

dependencies. The robustness of LSTM models to noisy and complex datasets underscores their suitability for detailed and nuanced time series analysis over more traditional models. Despite these findings, the study did not conclusively prove the absolute efficiency of LSTMs in all time series scenarios. Particularly, the models' performance in the presence of extremely noisy data or data with irregular cyclic patterns was not fully ascertained. Furthermore, the adaptation of models to sudden shifts in trends or distribution, which often occur in real-world data, remains partially unexplored.

Limitations and Future Work

To advance the research presented, we propose several potential directions for further investigation. First, exploring more complex models or configurations of LSTM, such as incorporating attention mechanisms or bidirectional structures, could significantly enhance prediction accuracy and model adaptability. More advanced neural networks have been introduced for this purpose that might work better than LSTMs - like GRUs (Gated Recurrent Unit Networks) and TCNs (Temporal Convolutional Networks). Econometric methods for time series forecasting like ARIMA can also be explored as they are proven to give good performance. Moreover, for time series forecasting data pre-processing particularly holds a lot of value, and even though this report tried to compare different models on minimally pre-processed data, ideally to derive better results pre-processing should be improved. For instance, the second dataset was non-stationary which decreases model performance. This dataset can be made stationary by ‘differencing’ the data i.e. forming lagged values and using those as features instead of original values. The second dataset became stationary just after differencing it once and this could have shown better performance. Adopting more sophisticated methods for outlier detection, such as utilizing the Median Absolute Deviation (MAD), could enhance the model's resilience against extreme values and improve overall robustness. Finally, incorporating additional statistical tests for autocorrelation and stationarity, including the Ljung-Box test or enhanced Augmented Dickey-Fuller (ADF) tests, could offer deeper insights into the data structure and assist in guiding more tailored model adjustments. Additionally, there is considerable scope for improvement in feature engineering; specifically, deriving and integrating new predictors that effectively capture underlying patterns observed in the residuals might substantially boost model performance. Another valuable area of development involves implementing and comparing different methods for deseasonalizing and detrending the data, which could provide critical insights into optimizing preprocessing steps to enhance model accuracy. Analyzing the data over shorter time spans or at a more granular level might also help in identifying intraday or intraweek patterns, thus refining models to be more responsive to immediate changes in the dataset.These proposed explorations aim to build on the current research foundations, potentially leading to more accurate and robust forecasting models.

In conclusion, while this study has made significant strides in understanding and demonstrating the capabilities of various modeling techniques in time series forecasting, the dynamic and complex nature of time series data necessitates ongoing research. Continuous refinement of models, along with an adaptive approach to feature engineering and data preprocessing, will be crucial in advancing the field of time series analysis.

Appendix

1. Factors to be considered while designing a Machine Learning Model for Time Series Forecasting:

Factor	Linear Regression	Decision Trees	Neural Networks
Autocorrelation	Assumes independence among observations; inefficiencies and biases in estimates if present.	Treats each instance as independent; doesn't naturally leverage temporal dependencies.	Assumes independence in standard form; sequence dependencies not captured unless specifically designed.
Handling Autocorrelation	Add lagged variables, use ARIMA models for time series data.	Include lagged variables as features to enable capturing temporal dependencies.	Use RNNs, LSTMs, or GRUs, which are designed to handle sequence data and can capture temporal dependencies effectively.
Seasonality	Can mislead the model if not explicitly accounted for due to periodic fluctuations.	Can handle if seasonal patterns affect the target in identifiable splits but requires explicit feature engineering.	Basic architectures might not inherently account for it unless represented in features.
Handling Seasonality	Incorporate seasonal dummies, sinusoidal transformations to model seasonal effects.	Include features that represent seasonality (e.g., day of week, month of year) to allow the model to utilize these patterns in its splits.	Complex architectures like CNNs or LSTMs can learn seasonal patterns through hierarchical structures and memory cells.
Non-Stationarity	Leads to misleading results due to changing variance or mean over time.	Can manage non-stationary data if changes affect the distribution in detectable ways in splits.	Can adapt to non-stationary data, especially with architectures that allow for dynamic learning.
Handling Non-Stationarity	Use differencing, detrending, or transformations to stabilize variance and mean.	Use tree ensembles like Random Forests or Gradient Boosting for better robustness to non-stationarity due to their aggregative and boosting approaches.	Employ techniques like rolling windows or continual learning setups to adapt to non-stationarity.
Noise/Outliers/Missing Values	Very sensitive to outliers; cannot handle missing values without prior processing.	Relatively robust to noise and outliers; can handle missing values using surrogate splits.	Sensitive to noise and outliers; typically requires complete data, necessitating imputation for missing values.
Handling Noise/Outliers/Missing	Use robust regression techniques, outlier removal, and imputation strategies for missing data.	Less sensitive; adjustments often not needed as critically as in linear models, but outlier removal and imputation can still improve performance.	Implement regularization, robust scaling, dropout for noise and outliers; use advanced imputation techniques for missing data.

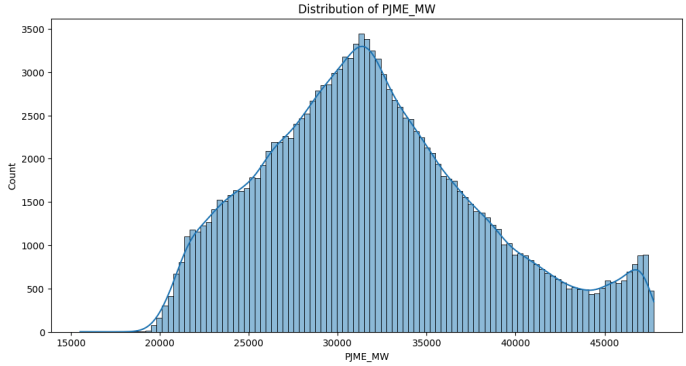
External Variability	Can include external factors as additional regressors but may be limited by the model's linearity.	Naturally incorporates external factors through splits, testing their impact effectively.	Can incorporate external factors, though how effectively depends on the architecture and feature engineering.
Handling External Variability	Incorporate interaction terms or non-linear transformations; add external variables directly into the model.	Natural handling through model design; no specific adjustments usually needed unless for enhancing the model's sensitivity to subtle variations.	Depending on the architecture, use feature engineering to enhance the model's ability to adapt to and learn from external variability; use architectures like CNNs for spatial data or recurrent layers for sequential data to process influences from external factors dynamically and effectively.

2. Dataset 1 - Power Consumption Dataset

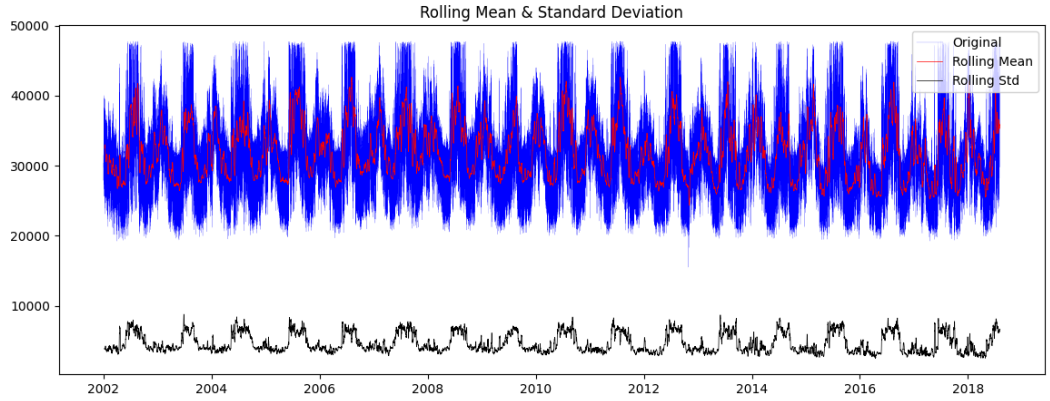
1. Data Analysis:

frequency	hourly
count	145362
mean	32080.507722
Standard deviation	6463.870519
Minimum	14544
25% Quartile	27573
50% Quartile	31421
75% Quartile	35650
Maximum	62009
Skewness	0.4616981486871183
Kurtosis	-0.2036298381480175

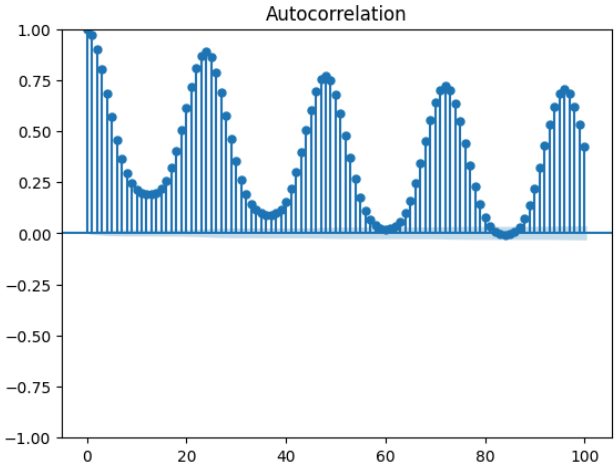
2. Histogram and Density Plot:



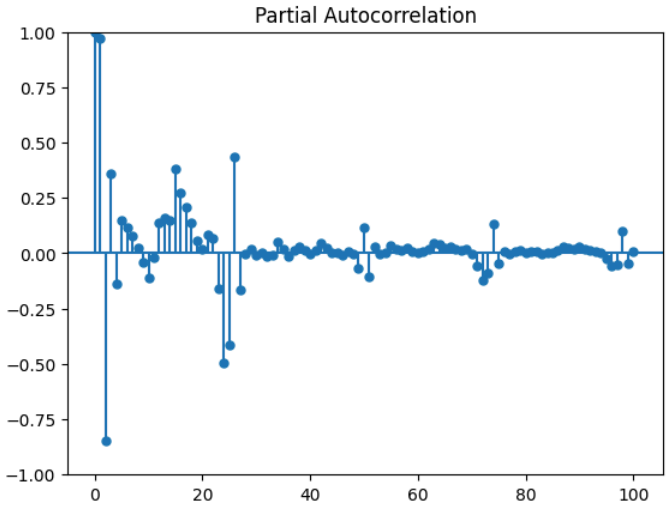
3. Rolling Mean and Standard Deviation:



4. ACF Plot:



5. PACF Plot:

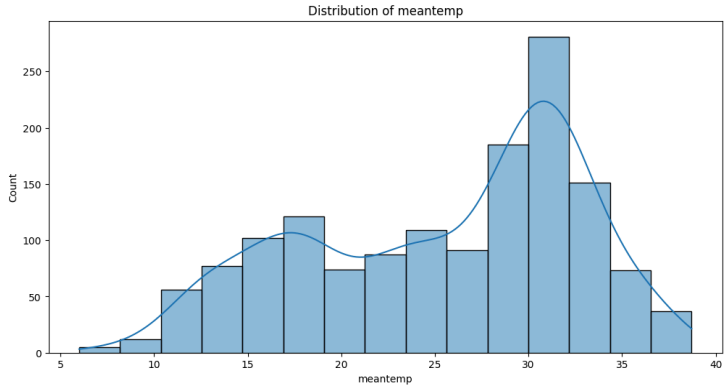


3. Dataset 2- Delhi Climate Dataset

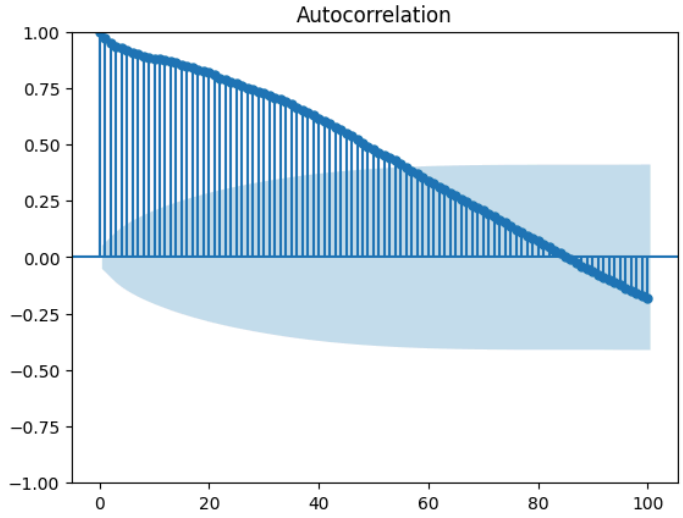
1. Data Analysis:

frequency	daily
count	1461
mean	25.506127
Standard deviation	7.339416
Minimum	6
25% Quartile	18.857143
50% Quartile	27.714286
75% Quartile	31.312500
Maximum	38.714286
Skewness	-0.4461322636404364
Kurtosis	-0.9379486547218705

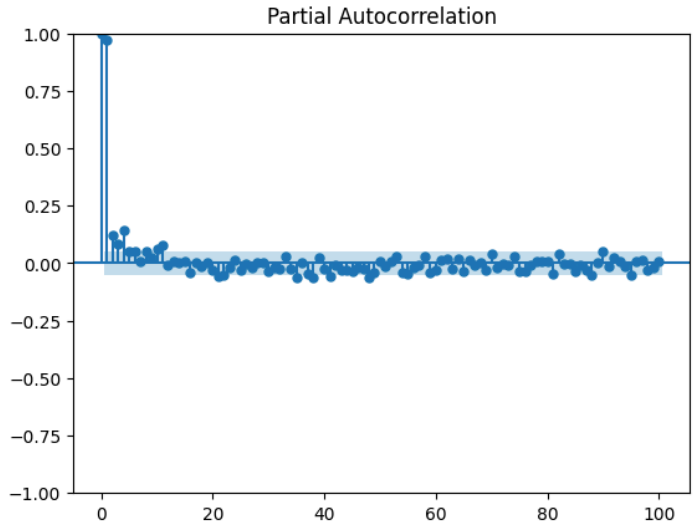
2. Histogram and Density Plot:



3. ACF Plot:



4. PACF Plot:


Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
20
40
60
80
100