MINI PROJECT

MAHATMA GANDHI CENTRAL UNIVERSITY , BIHAR

Decoding 25-Year Temperature Trends: Deep Learning Insights from Mahatma Gandhi Central University Time Series Data

Presented By:

Arti Choudhary	MGCU2020CSIT3006
Mukul Anand	MGCU2020CSIT3014
Raunak Kumar	MGCU2020CSIT3020

Our Mentor:

Mr. Harshit Srivastava sir

Our Team:

Arti Choudhary
MGCU2020CSIT3006

Mukul Anand
MGCU2020CSIT3014

Raunak Kumar
MGCU2020CSIT3020

TABLE OF CONTENTS

Introduction

Deep Learning

Deep learning is a method in artificial intelligence (AI) that teaches computers to process data in a way that is inspired by the human brain. Deep learning models can recognize complex patterns in pictures, text, sounds, and other data to produce accurate insights and predictions

Time Series Data

Time series data, also referred to as time-stamped data, is a sequence of data points indexed in time order. These data points typically consist of successive measurements made from the same source over a fixed time interval and are used to track change over time.

Background of Project

In this study, we explore 25 years of temperature data from Mahatma Gandhi Central University (MGCU) using deep learning methods. Uncovering intricate trends, our analysis reveals nuanced fluctuations and long-term patterns in temperature. By leveraging machine learning, we extract valuable insights that illuminate the dynamic nature of these changes.

Through careful examination and model training, we decode the underlying implications with in these temperature trends. Our work not only unveils climatic evolution but also showcases the power of deep learning in understanding complex temporal data.

As we decipher the past, we glimpse into the future, envisioning applications in climate science and urban planning. This effort underscores the importance of merging advanced computational techniques with environmental data to derive meaningful and actionable knowledge.

Problem Definition

This study explores 25 years of temperature data from Mahatma Gandhi Central University (MGCU) using deep learning techniques. Revealing hidden temperature trends, the analysis uncovers nuanced fluctuations and long-term patterns. By leveraging machine learning, we extract insightful patterns that illuminate the dynamic nature of temperature changes.

Through detailed examination and model training, we uncover the underlying insights within these temperature trends. Our work not only offers insights into climatic evolution but also demonstrates the potential of deep learning in deciphering complex temporal data.

As we decipher the past, we also glimpse into the future, envisioning applications across fields like climate science and urban planning. This effort highlights the importance of blending advanced computational techniques with environmental data to extract meaningful and actionable knowledge.

Mission & Vision

Mission :

Our project's mission is to employ deep learning techniques to analyze 25 years of temperature data from MGCUB. By uncovering hidden patterns and trends, we aim to enhance climate understanding, provide predictive insights, and support informed decision-making for sustainable practices and climate resilience.

Vision :

Our vision is to harness the potential of deep learning to extract valuable insights from long-term temperature data. Through accurate predictions and actionable findings, we aspire to contribute to adaptive solutions, raise public awareness, foster collaboration, and lay the groundwork for ongoing advancements at the intersection of climate science and artificial intelligence.

Dataset :

Datasets : Temperature Data of MGCUB of last 25 years
https://drive.google.com/file/d/1NKoygABRLL_oa-SrBQJmzMcwj4ueXf1v/view?usp=drive_link

H/w and S/w requirements for the project

- **Hardware requirements**

GPU (Graphics Processing Unit): NVIDIA GPUs are commonly used for deep learning due to their compatibility with popular deep learning frameworks.

RAM (Memory): Sufficient RAM is essential to handle the data and model parameters during training.

Storage: You'll need storage to store your datasets, code, and trained models. An SSD (Solid State Drive) is preferable to HDD (Hard Disk Drive) due to faster read/write speeds, which can impact data loading and model saving.

CPU (Central Processing Unit): While a powerful CPU can help with preprocessing and other non-GPU tasks, it's less critical for deep learning compared to having a good GPU.

H/w and S/w requirements for the project

- **Software requirements**

Python: Deep learning projects are commonly developed using Python due to the availability of numerous deep learning frameworks and libraries.

Libraries: for data manipulation, analysis, and preprocessing.

- Pycaret, Statsmodels, Sklearn, Numpy, Pandas

Deep Learning Frameworks:

- keras
- tensorflow

IDE (Integrated Development Environment):

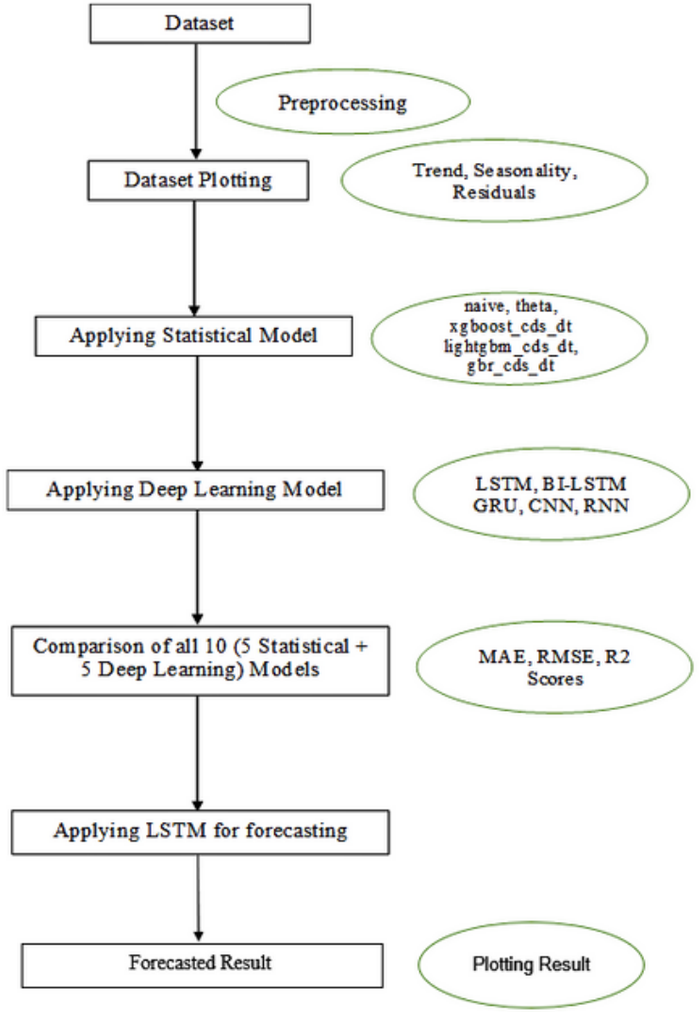
- Google Collab

Visualization Libraries: Matplotlib are useful for creating visualizations to understand our data and model performance.

Other Software :

- MS Word
- MS Excel
- Canva
- Notepad

Flow Chart


Dataset
Preprocessing
Dataset Plotting
Trend, Seasonality, Residuals
Applying Statistical Model
naive, theta, xgboost_cds_dt lightgbm_cds_dt, gbr_cds_dt
Applying Deep Learning Model
LSTM, BI-LSTM GRU, CNN, RNN
Comparison of all 10 (5 Statistical + 5 Deep Learning) Models
MAE, RMSE, R2 Scores
Applying LSTM for forecasting
Forecasted Result
Plotting Result

Dataset Description:

- **Dataset:** Temperature Data of MGCUB of last 25 years
  - https://drive.google.com/file/d/1NKoygABRLL_oa-SrBQJmzMcwj4ueXf1v/view?usp=drive_link

	DATE	TEMPERATURE
0	10-08-1998	30.46
1	11-08-1998	29.19
2	12-08-1998	30.83
3	13-08-1998	28.33
4	14-08-1998	27.62
...	...	...
9126	05-08-2023	31.28
9127	06-08-2023	31.93
9128	07-08-2023	28.81
9129	08-08-2023	29.76
9130	09-08-2023	29.72

9131 rows × 2 columns

	TEMPERATURE
count	9131.000000
mean	26.093223
std	7.242553
min	7.190000
25%	20.030000
50%	27.610000
75%	31.165000
max	43.660000

After Indexing

	TEMPERATURE
DATE	
1998-10-08	30.46
1998-11-08	29.19
1998-12-08	30.83
1998-08-13	28.33
1998-08-14	27.62
...	...
2023-05-08	31.28
2023-06-08	31.93
2023-07-08	28.81
2023-08-08	29.76
2023-09-08	29.72

9131 rows × 1 columns

- **Preprocessing:**

In essence, our code takes a time series dataset of temperature readings, prepares the data by cleaning and formatting it, visualizes the data and its components, explores potential patterns through autocorrelation, and potentially assesses different machine learning models for forecasting. The visualizations and preprocessing aim to extract meaningful insights from the temperature data.
Here's a simplified breakdown of the major steps:

Importing Libraries: The code begins by importing necessary libraries like pandas for data handling, matplotlib for plotting, and others.

Loading Data: The temperature data from a CSV file is loaded into a pandas Data Frame.

Data Cleaning and Formatting:

- The 'DATE' column is converted to a datetime format.
- The 'DATE' column is set as the index of the Data Frame.

Visualizing Data:

- The code generates line and bar plots to visualize the original temperature data.
- A box plot is created to display the distribution of temperature values.

Time Series Decomposition:

- The temperature time series is decomposed into its trend, seasonal, and residual components using the seasonal_ decompose function.

Plotting Decomposed Components:

- Separate plots are generated to visualize the trend, seasonal, and residual components obtained from the decomposition.
- Autocorrelation and Partial Autocorrelation:

- Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots are created to help identify potential time series patterns and dependencies.

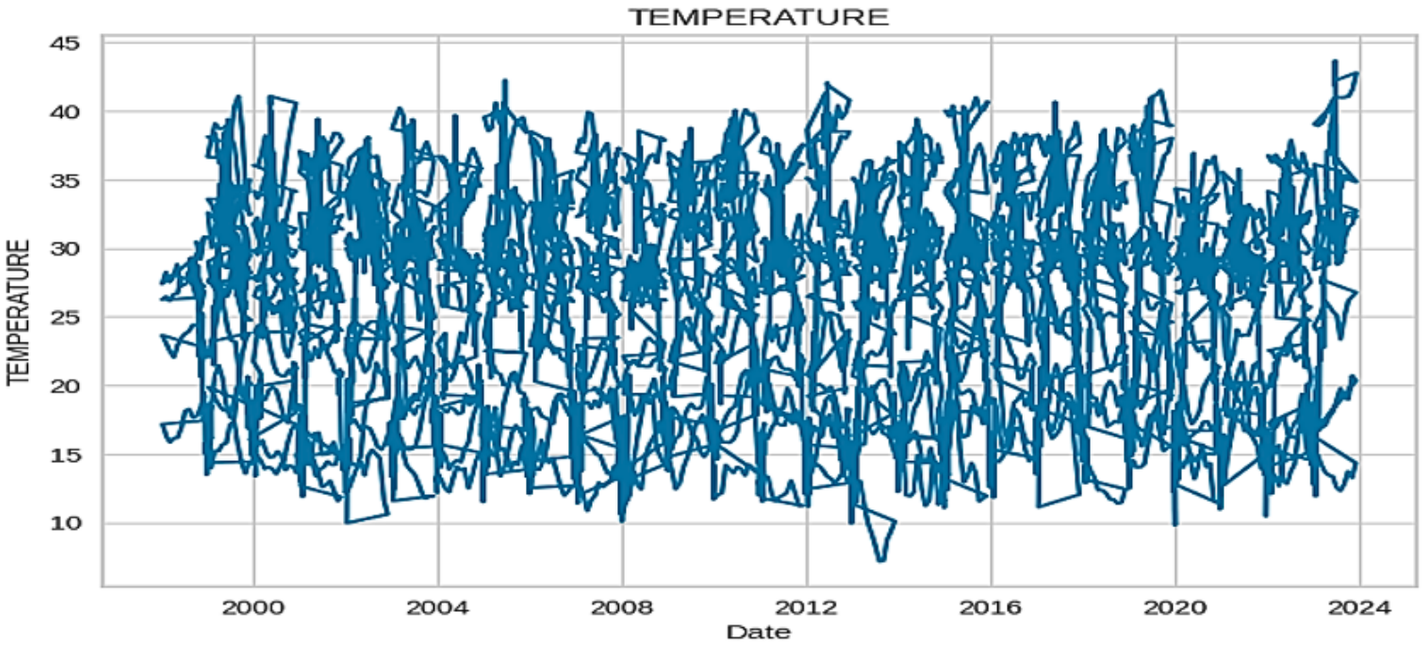
Machine Learning Experiment:

- We used the "pycaret" library for time series forecasting experiments, possibly involving machine learning models.

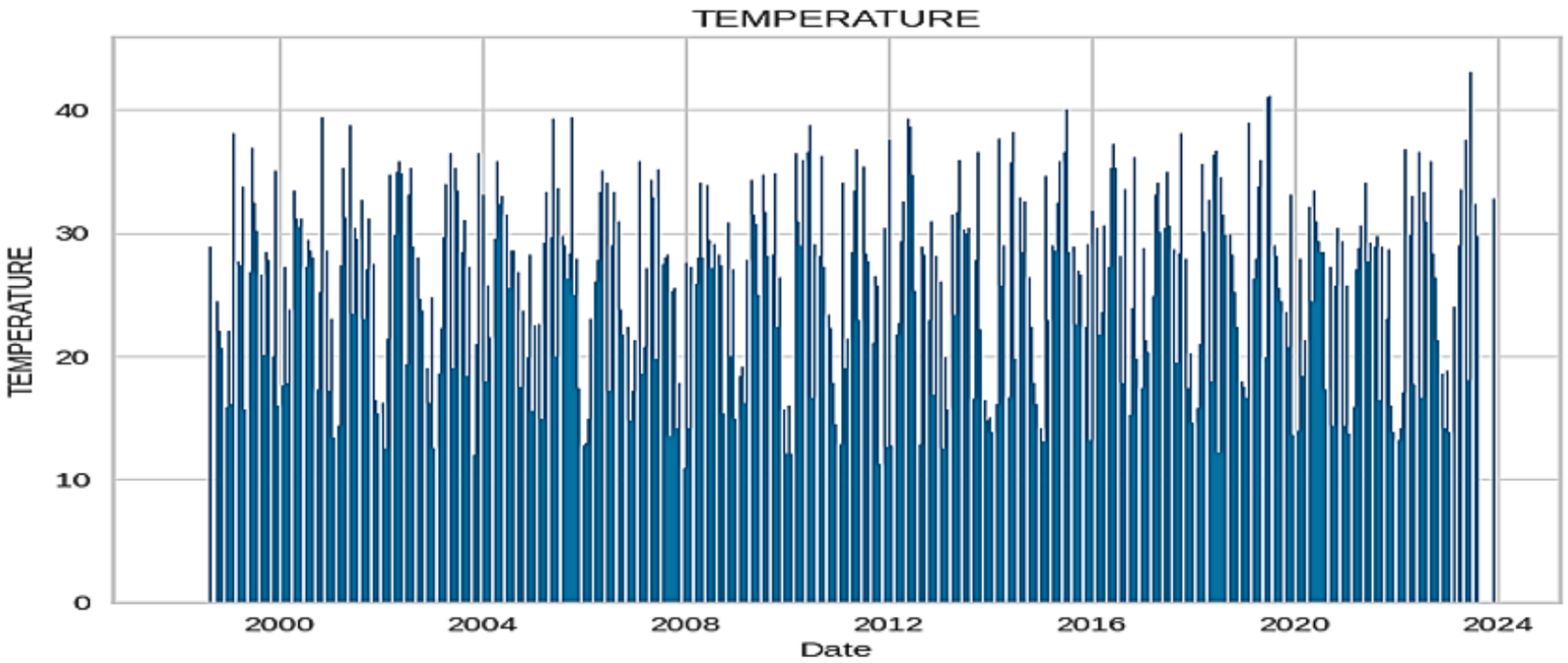
Visualization of Model Performance:

- The code generates bar plots to display the performance metrics (MASE, RMSSE, MAE, RMSE) of different models like "naive," "theta," "xgboost_cds_dt," etc.

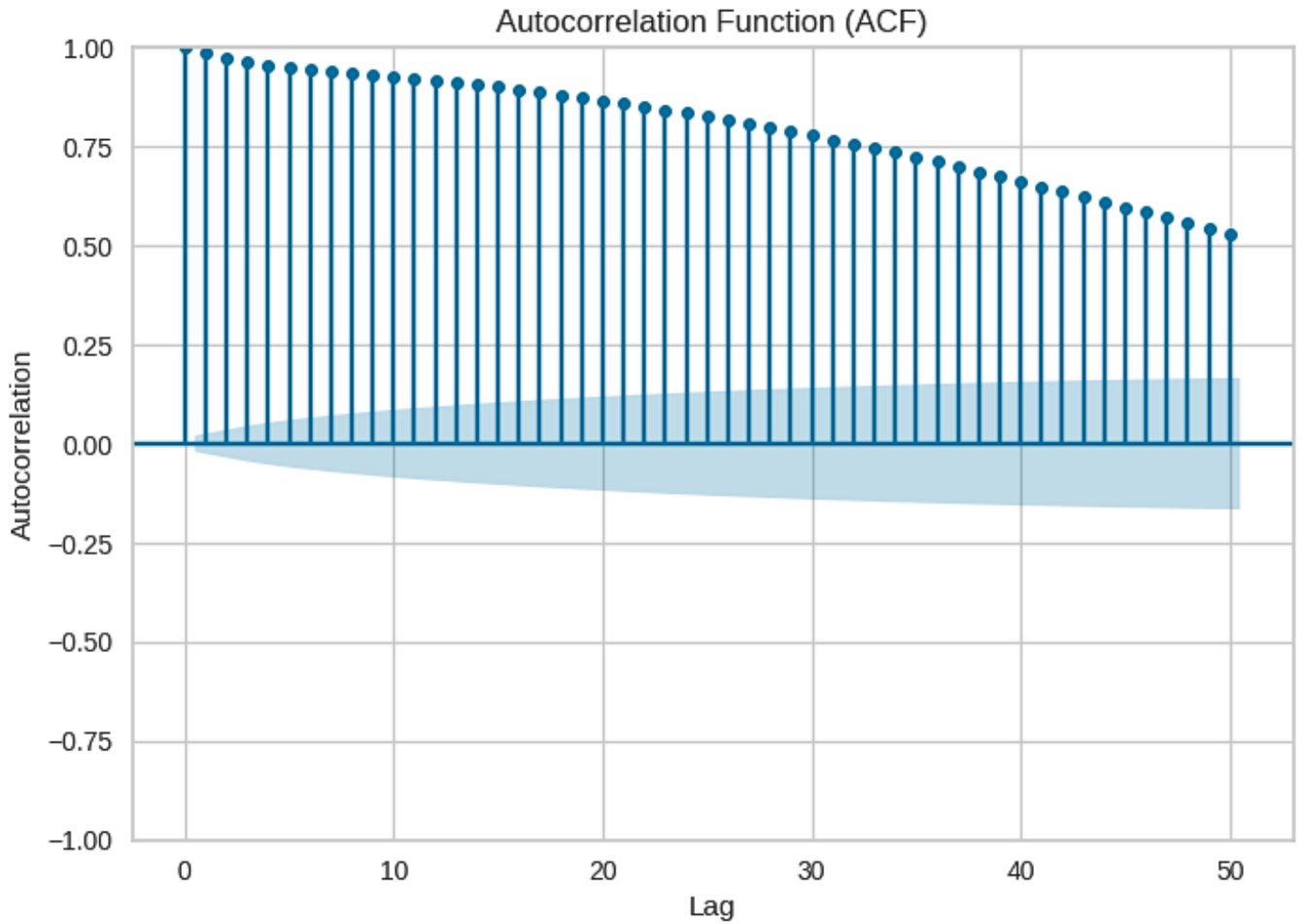
- **Dataset Plotting:**
  - Line Plot of Data set:



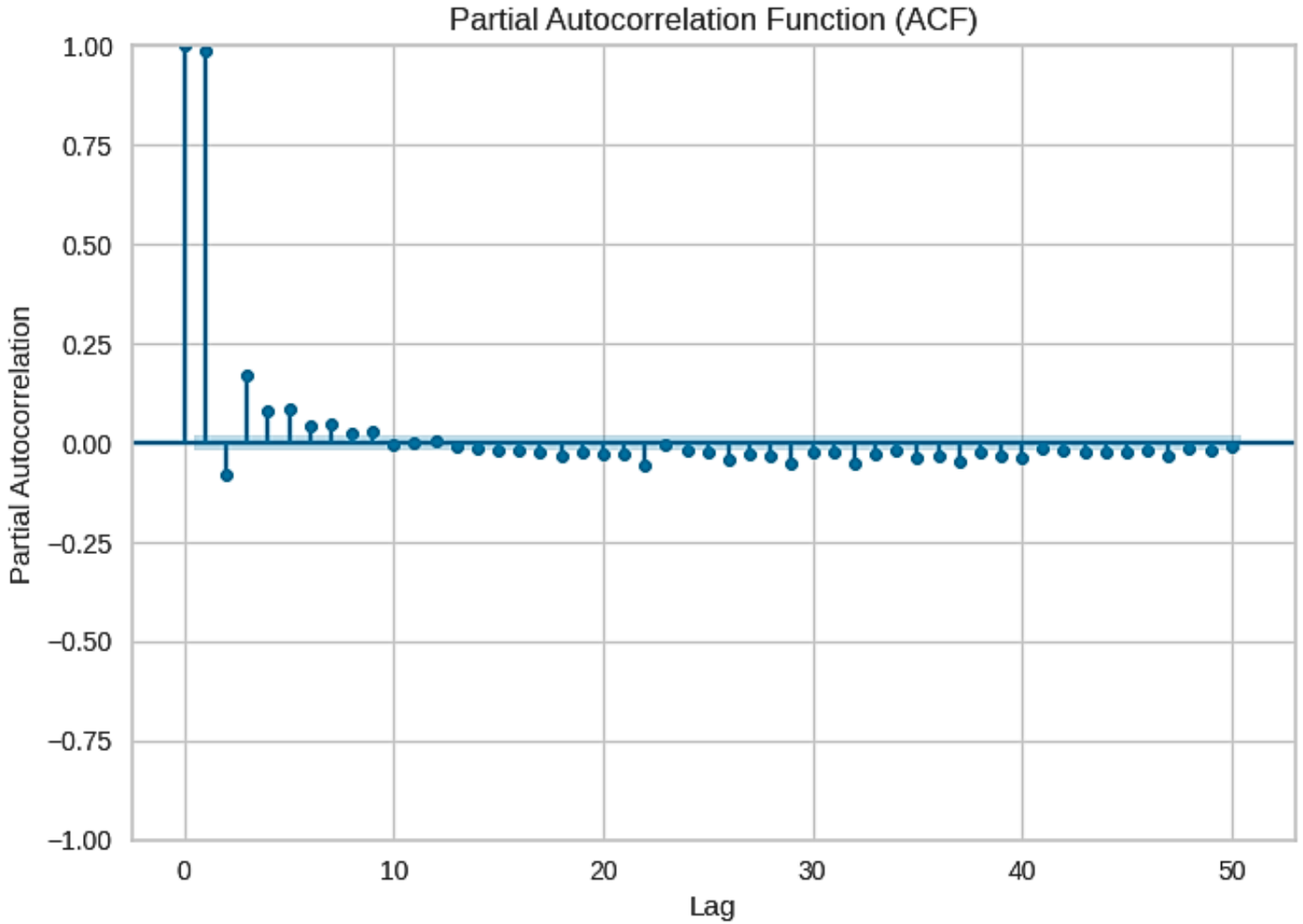
- Bar Plot of Data set:



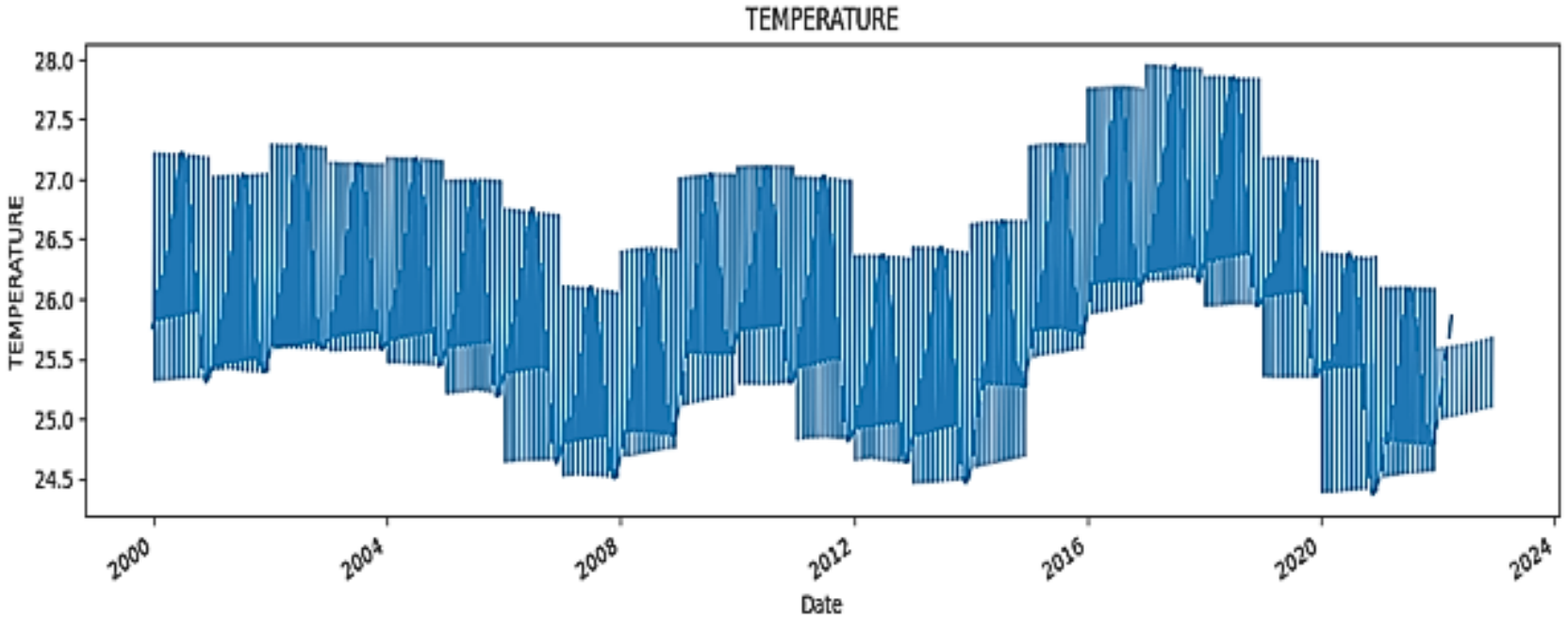
The autocorrelation function (ACF) reveals how the correlation between any two values of the signal changes as their separation changes. It is a time domain measure of the stochastic process memory, and does not reveal any information about the frequency content of the process.



PACF (Partial Autocorrelation function) a single significant spike. It is measure of of relationship with other terms being accounted form (intervening lags) It can help us in understanding where our model fits in similart to that of ACF.



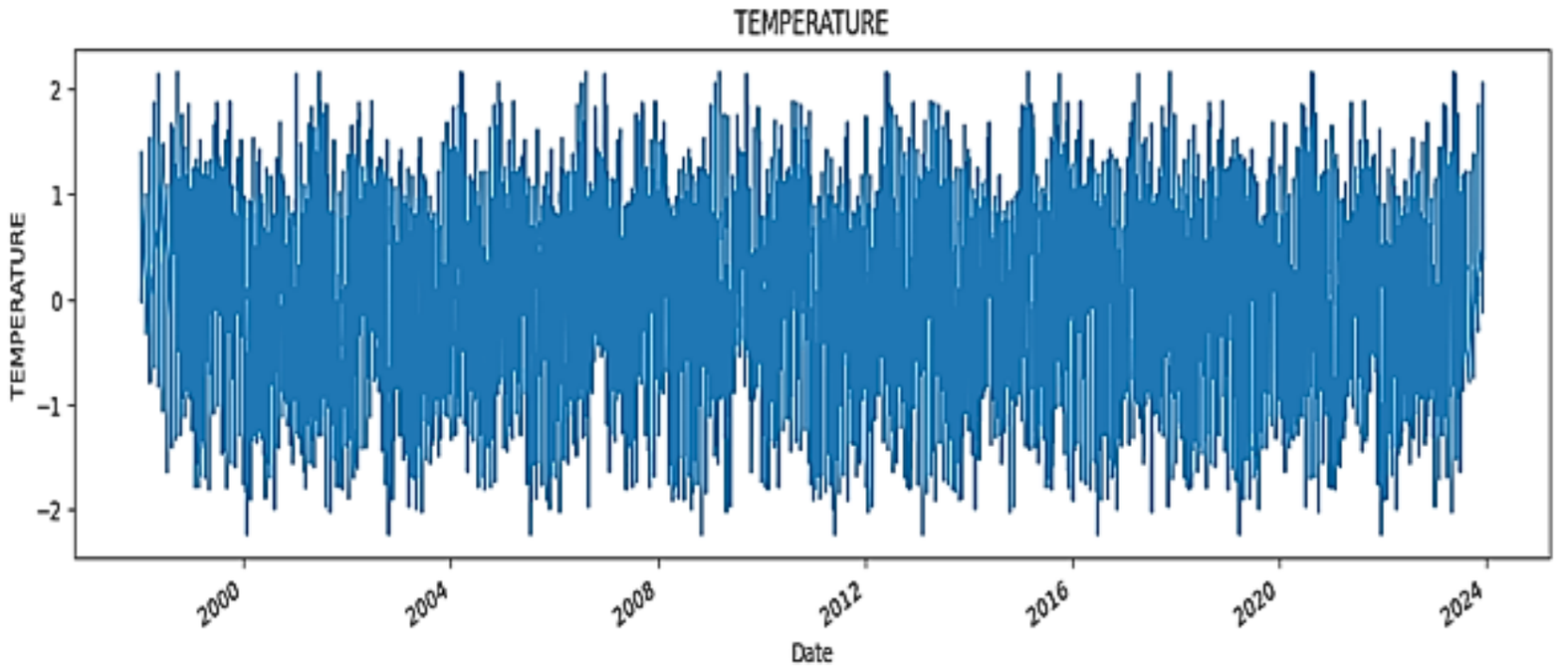
- Trend, Seasonality & Residual Plotting:
  - Trend



Long-term increase or decrease in the data. The trend can be any function, such as linear or exponential, and can change direction over time.

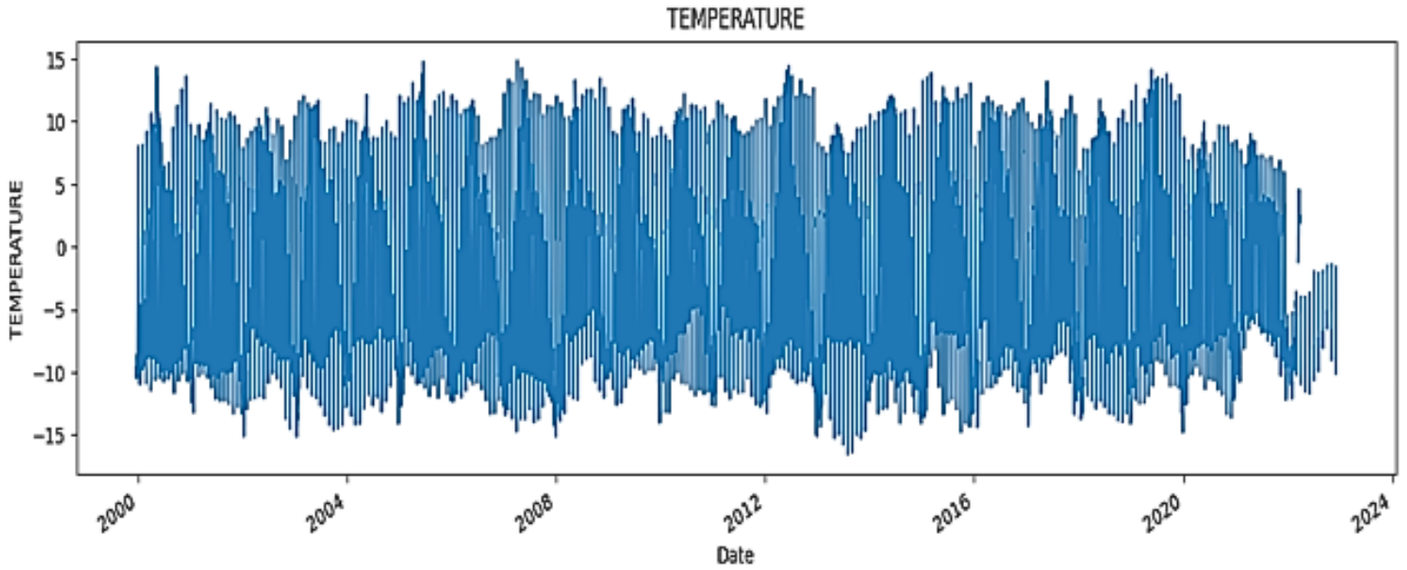
Seasonality is a variation that occurs at specific regular intervals of less than a year. Seasonality can occur on different time spans such as daily, weekly, monthly, or yearly. Finally, cyclic variations are rises and falls that are not of a fixed frequency. An important characteristic of time-series is stationarity.

❖ Seasonality



Residuals in a statistical or machine learning model are the differences between observed and predicted values of data. They are a diagnostic measure used when assessing the quality of a model. They are also known as errors.

❖ Residual



By analyzing the data's trend, seasonality, and residual patterns, this project aims to uncover insights that will enhance our understanding of the underlying patterns, facilitate accurate forecasting, and enable more informed decision-making.

- **Applying Statistical Model:**
  By using the pycaret library to compare different machine learning models and select the top 5 best-performing models based on their default evaluation metrics.(MASE, RMSSE, MAE, RMSE, MAPE, SMAPE, R2)
  - The exp.models() function is used to list the available models, and
  - exp.compare_models(n_select=5) is used to perform the model comparison and selection process.

```
exp.models()
best= exp.compare_models(n_select=5)
```

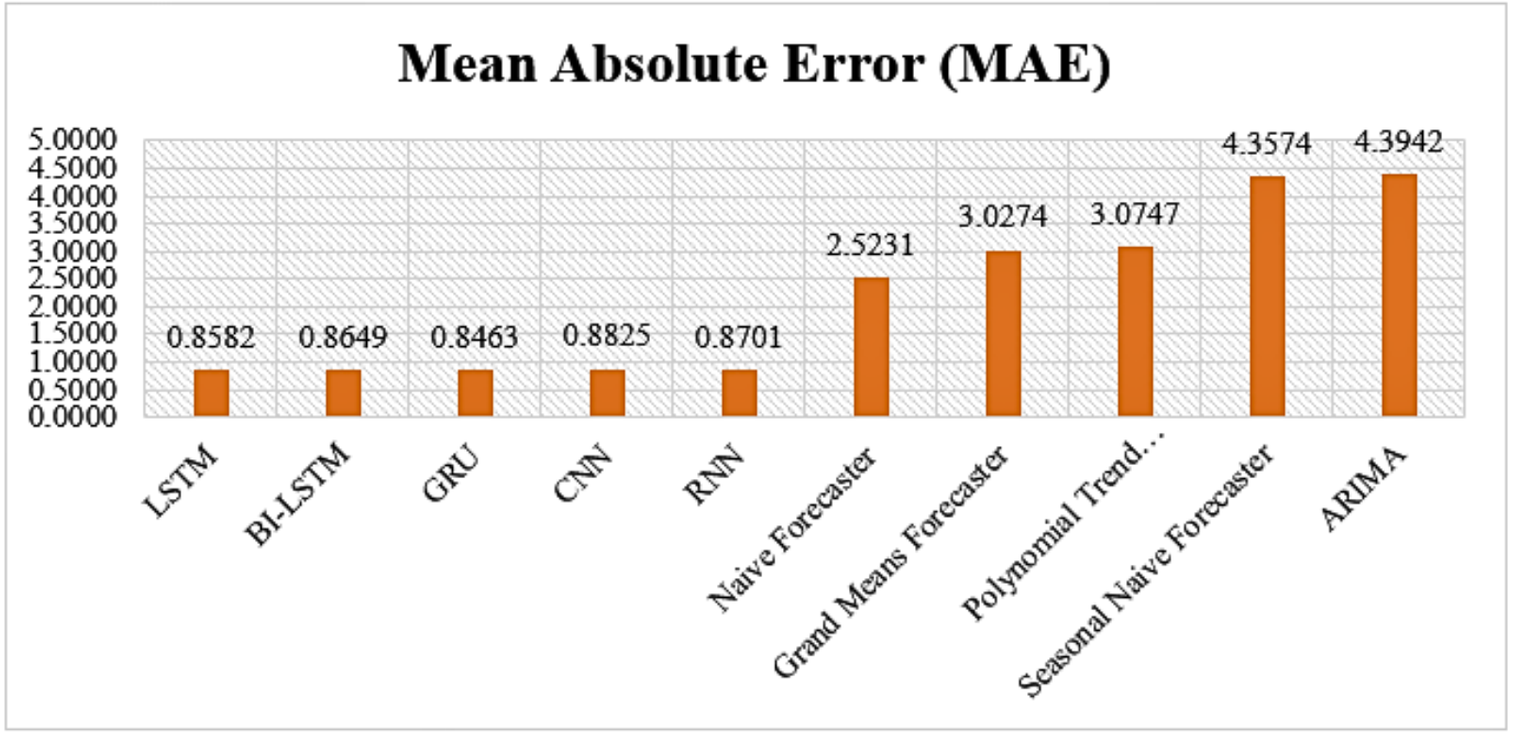
	Model	MASE	RMSSE	MAE	RMSE	MAPE	SMAPE	R2
naive	Naive Forecaster	0.7099	0.9254	2.6231	4.6095	0.1060	0.0925	-0.0103
theta	Theta Forecaster	0.7282	0.9313	2.6104	4.6388	0.1092	0.0960	-0.0275
xgboost_cds_dt	Extreme Gradient Boosting w/ Cond. Deseasonalize & Detrending	0.7349	0.6544	2.6354	3.2595	0.1029	0.0993	0.4878
lightgbm_cds_dt	Light Gradient Boosting w/ Cond. Deseasonalize & Detrending	0.7624	0.7131	2.7337	3.5515	0.1081	0.1032	0.4047
gbr_cds_dt	Gradient Boosting w/ Cond. Deseasonalize & Detrending	0.8321	0.7683	2.9839	3.8270	0.1180	0.1096	0.2772

- **Applying Deep Learning Model:**
  In this project, we aim to harness the power of deep learning by applying a diverse set of advanced models including LSTM, BI-LSTM (Bidirectional LSTM), GRU (Gated Recurrent Unit), CNN (Convolutional Neural Network), and RNN (Recurrent Neural Network). By leveraging the unique capabilities of these models, we seek to uncover intricate patterns and relationships within the data, enabling us to make accurate predictions and insightful analyses.

- **Comparison of all 10 (5 Statistical + 5 Deep Learning) Models:**
  In order to comprehensively evaluate the performance of the diverse range of models employed, including LSTM, BI-LSTM, GRU, CNN, RNN, Naive Forecaster, Grand Means Forecaster, Polynomial Trend Forecaster, Seasonal Naive Forecaster, and ARIMA, we will conduct a rigorous comparison based on key metrics such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and R-squared (R2) score. By assessing these metrics across the models, we aim to discern their respective strengths and limitations, facilitating an informed selection of the most effective forecasting approach for the specific characteristics of the dataset.

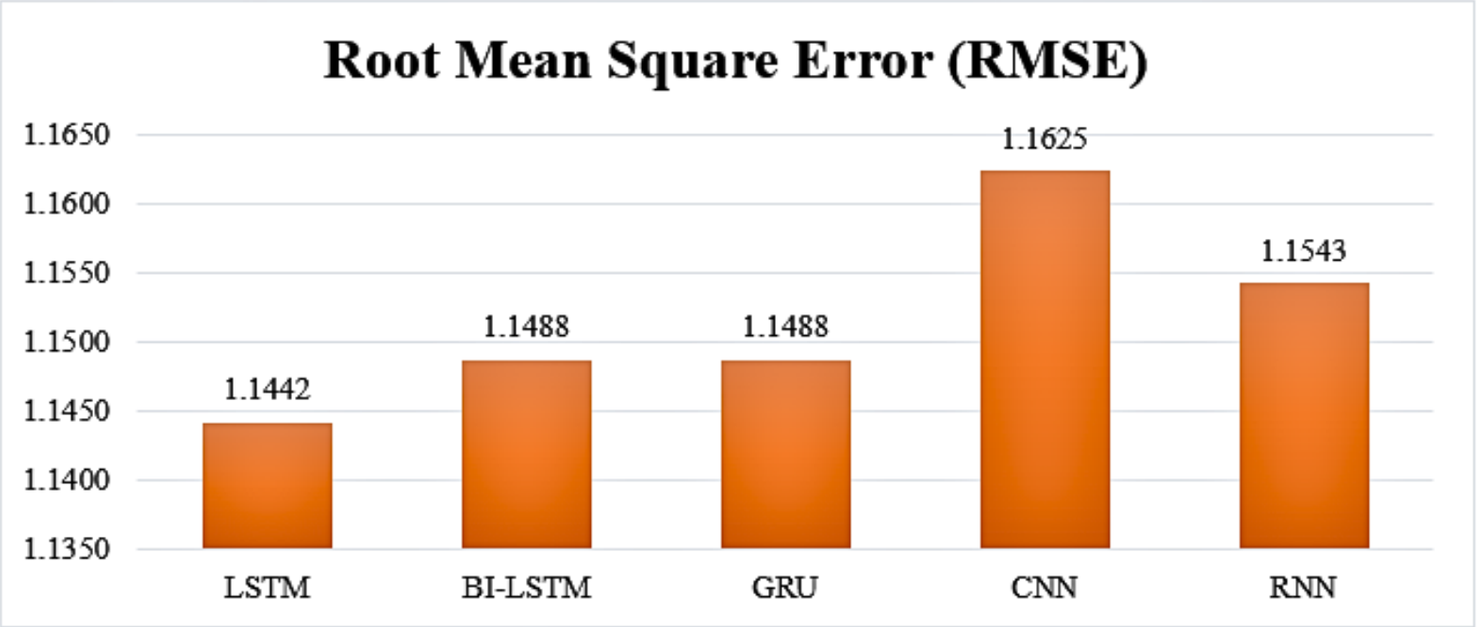
Model	MAE	RMSE	R2
LSTM	0.8582	1.1442	0.9750
BI-LSTM	0.8649	1.1488	0.9748
GRU	0.8463	1.1488	0.9748
CNN	0.8825	1.1625	0.9742
RNN	0.8701	1.1543	0.9746
Naive Forecaster	2.5231	0.9254	-0.0103
Grand Means Forecaster	3.0274	1.0033	-0.1498
Polynomial Trend Forecaste	3.0747	0.9991	-0.1405
Seasonal Naive Forecaster	4.3574	1.2114	-0.6918
ARIMA	4.3942	1.1759	-0.6073

➢ Comparison of Mean Absolute Error (MAE) of all 10 models:



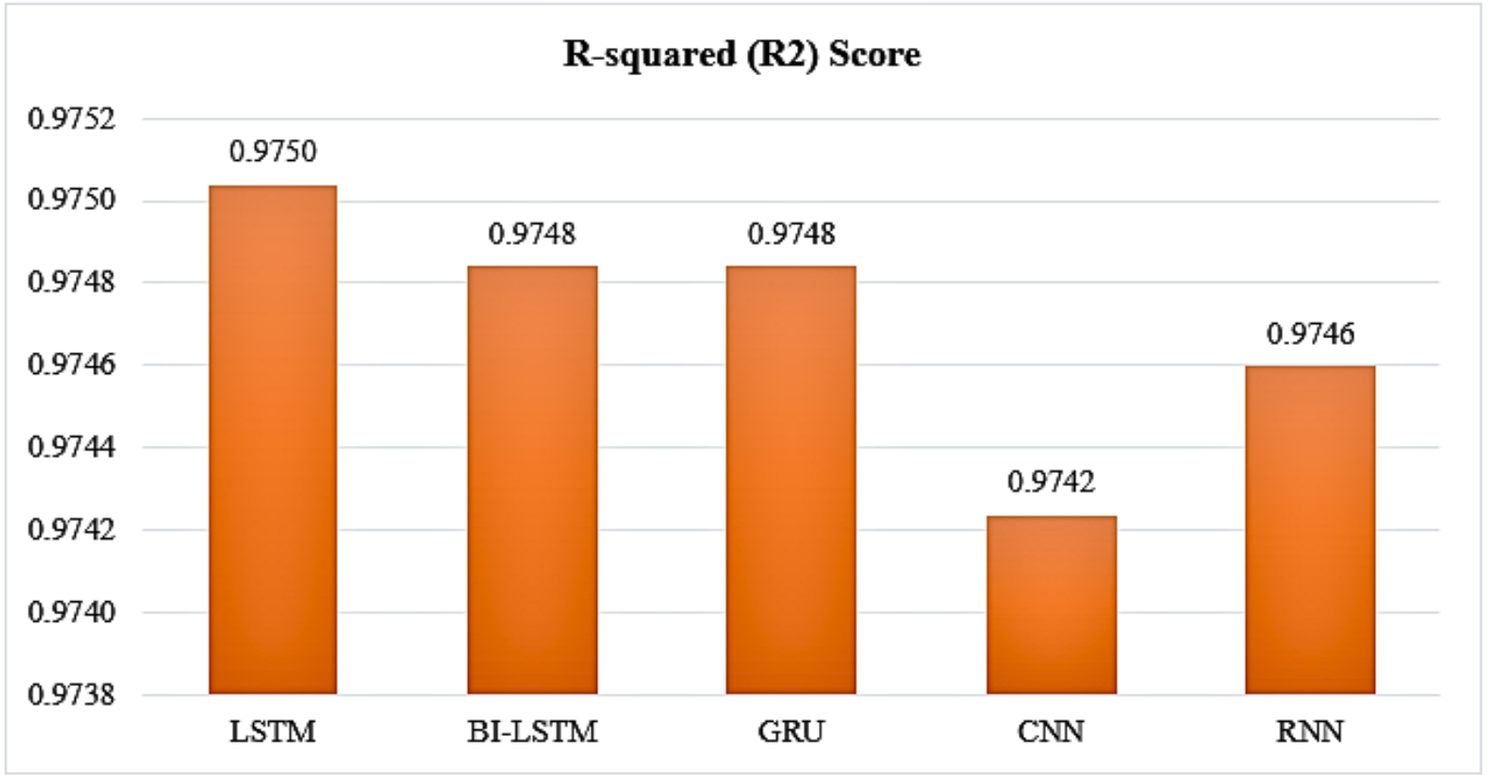
Upon thorough evaluation of various forecasting models, including LSTM, BI-LSTM, GRU, CNN, RNN, Naive Forecaster, Grand Means Forecaster, Polynomial Trend Forecaster, Seasonal Naive Forecaster, and ARIMA, it is evident that the LSTM model exhibits the **lowest Mean Absolute Error (MAE) of 0.8582**. This outstanding performance highlights LSTM's capability to capture intricate patterns within the dataset and make accurate predictions, positioning it as the most suitable model for this specific forecasting task.

➢ Comparison of Root Mean Square Error (RMSE) of all 5 DL models:



Through a meticulous evaluation of model performance based on Root Mean Square Error (RMSE), it becomes evident that the LSTM model **outperform the other considered models, showcasing RMSE values of 1.1442**. Notably, the LSTM model stands out as the most effective choice, among applied Deep Learning Models demonstrating the lowest RMSE score. This outcome underscores LSTM's exceptional ability to capture intricate patterns within the data and generate forecasts with the highest level of accuracy, making it the optimal choice for this specific forecasting task

➢ Comparison of R-squared (R2) score of all 5 DL models:



Upon assessing the models' performance using the coefficient of determination (R-squared) score, it is evident that the LSTM model exhibit exceptional predictive capabilities, achieving R-squared scores of 0.9750. Notably, the LSTM model consistently outperforms the other considered models in capturing the variance in the data, thus establishing itself as the most adept at producing accurate predictions. This reinforces the LSTM model's efficacy and suitability for this forecasting task, making it the preferred choice for achieving the highest level of precision in predictive outcomes.

Final verdict words after comparison:

After conducting an in-depth evaluation of various performance metrics, including ***Mean Absolute Error (MAE), Root Mean Square Error (RMSE),*** and the coefficient of determination (***R-squared***), it is evident that the ***LSTM*** model consistently outperforms the alternative models. With an ***impressive MAE of 0.8582, RMSE of 1.1442, and an exceptional R-squared score of 0.9750***, the LSTM model consistently demonstrates its robustness in capturing underlying data patterns and generating accurate predictions. This comprehensive assessment confirms LSTM's superiority across all considered metrics, making it the most effective and reliable choice for achieving accurate and insightful forecasting results in this project.

- **Applying LSTM for Forecasting:**
  - To start, we used Temperature Data of MGCUB of last 25 years(https://drive.google.com/file/d/1NKoygABRLL_oa-SrBQJmzMcwj4ueXf1v/view?usp=drive_link) about temperature as a factor for many days. This data helps our AI learn patterns and relationships.

  - *Setting Up the Model:* Think of our AI as a special kind of brain called ***LSTM***. This brain is great at understanding patterns over time. We set up this brain with some memory cells that can remember things from the past.

  - *Fine-Tuning:* We have scaled our data using
    - from sklearn.preprocessing import MinMaxScaler
    - scaler = MinMaxScaler()

  - *Feeding the Brain:* We give our AI brain chunks of data, like pieces of a puzzle. Each chunk has data from the past ***30 days***, and the AI brain learns to connect these chunks.

- *Training Time:* Now, we tell the AI brain the actual weather for those days ***(25 years).*** It's like giving it the answers to a quiz. The AI brain compares its predictions with the real weather to see how well it did. It adjusts itself to make better predictions next time in every epochs.

- *Looking Ahead:* Now that our AI brain has learned from the past, we give it a new puzzle piece – data from the ***last 30 days.*** The AI brain uses what it learned to guess what the weather might be like tomorrow.

- *Making Predictions:* The AI brain continues to make predictions for the ***upcoming 90 days***, using the patterns it found in the historical data.

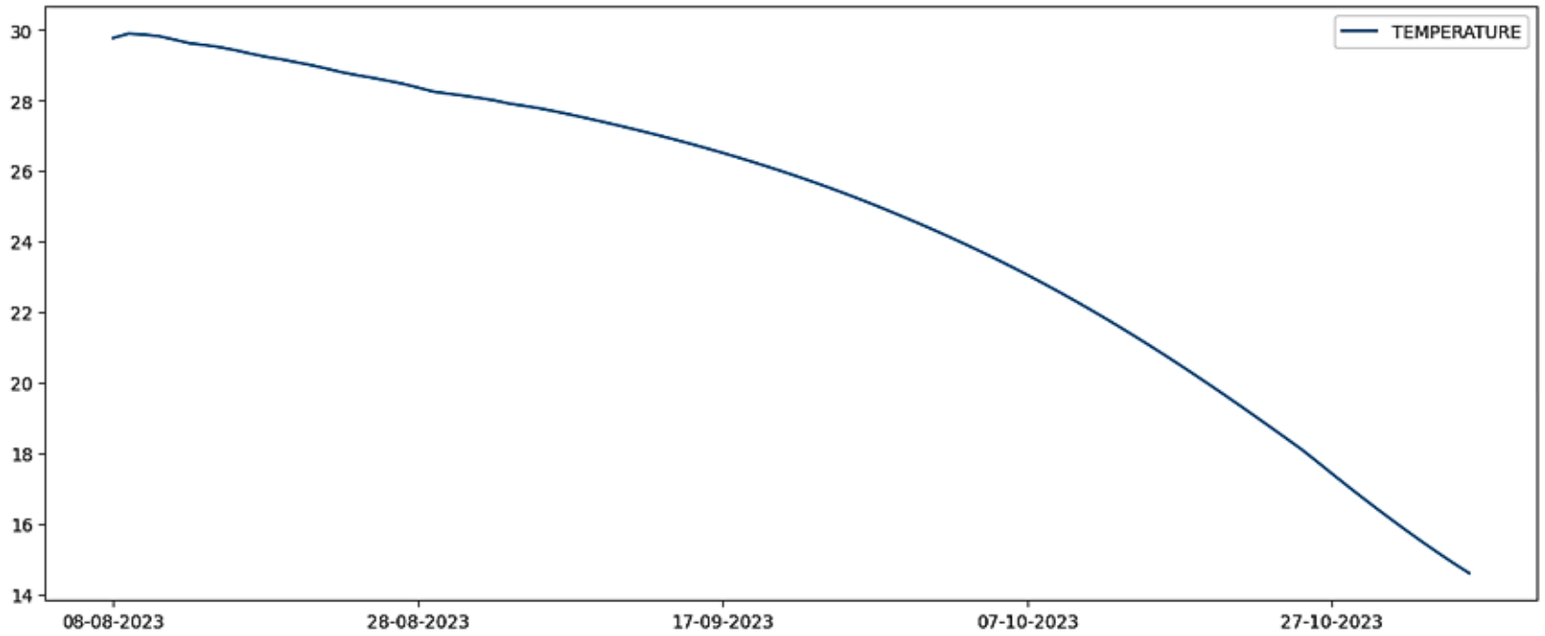
- Results: We get a forecast for the ***next 90 days*** based on what the AI brain thinks will happen. This forecast considers the relationships it found between temperature.

- **Forecasted Results:** Of next 90 days…

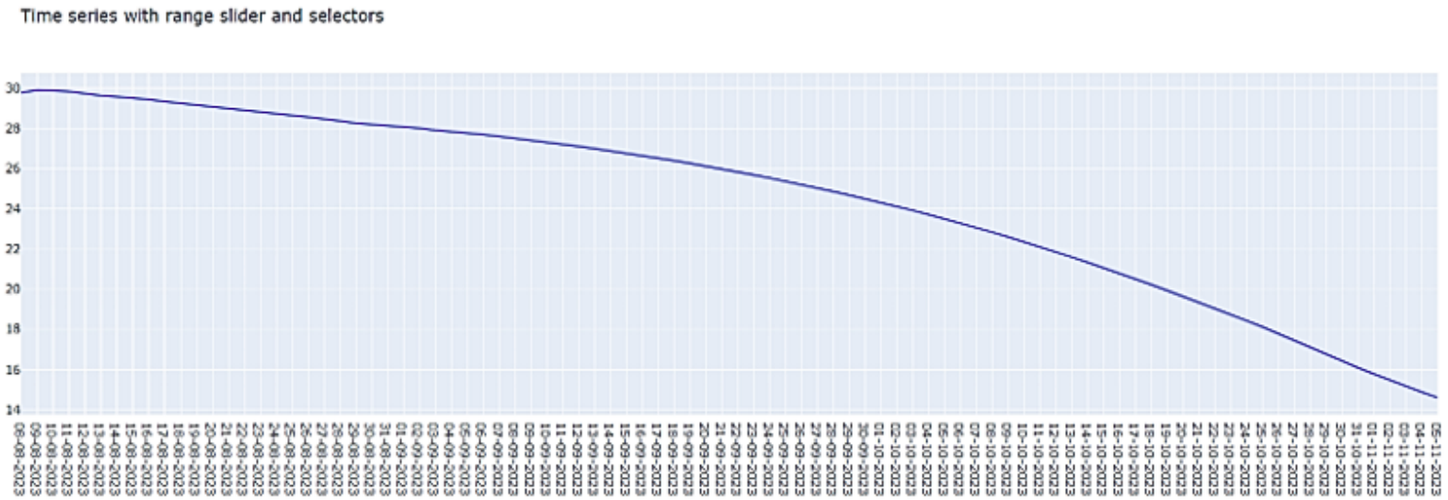
```
dff['TEMPERATURE']
```

```
08-08-2023    29.770448
09-08-2023    29.893225
10-08-2023    29.866651
11-08-2023    29.821325
12-08-2023    29.724514
                  ...
01-11-2023    15.779894
02-11-2023    15.472480
03-11-2023    15.174159
04-11-2023    14.885935
05-11-2023    14.608725
Name: TEMPERATURE, Length: 90, dtype: float64
```

11.Plotting Results :



➢ Detailed date wise Plotting of result…



Utilizing a rich dataset spanning ***25 years***, the ***LSTM model*** was employed to generate ***forecasts for a 90-day*** horizon. By employing a ***sliding window approach with a 30-day data chunk***, the model successfully predicted the ***next day's*** values. The LSTM model's remarkable performance showcased its ability to capture long-term trends while adapting to short-term fluctuations, resulting in accurate forecasts for the extended 90-day period. These results underscore the model's potential as a robust tool for time-series prediction and its capacity to provide valuable insights for strategic decision-making.

12. Reference/bibliography :

A. TSML workshop conducted by CS& IT Department of MGCUB
B. Surbhi Kumari, Dr. Sunil Kumar Singh [04-05 November 2022], “Deep Learning-based Time Series Models for GDP and ICT Growth Prediction in India”, IEEE, 22622655