

# TIME SERIES FORECASTING



A Project Report  
in partial fulfillment of the degree

**Bachelor of Technology**  
**in**  
**Computer Science & Artificial Intelligence**  
**By**

Roll No	Name of the Student
2203A52057	T. NAVYA LOHITHA

**School of Computer Science & Artificial Intelligence**  
SR University, Ananthasagar, Hasanparthy (M), Warangal,  
Telangana 506371, India

2024-25

# **DATASET DESCRIPTION**

## **DATASET DESCRIPTION**

The dataset used in this time series forecasting project contains daily weather data from Delhi. It includes information like temperature, humidity, wind speed, and air pressure, recorded over several years. The date is used as the index to form a time-based structure.

Before training the models, the data was cleaned and checked for missing values. The values were scaled using standard techniques to help the models learn better. The time series was then turned into input-output sequences using a sliding window method. This prepared dataset was used to train and evaluate deep learning models such as RNN, LSTM, and GRU for predicting future weather conditions.

## **MOTIVATION AND BACKGROUND**

With the increasing availability of historical climate data, predicting future weather conditions has become both important and achievable. Accurate forecasts can help in planning, agriculture, health, and disaster management. Traditional statistical models often struggle to capture complex patterns in time series data. This project explores deep learning techniques—such as RNN, LSTM, and GRU—that are specifically designed to handle sequential data. These models can learn from past patterns and provide more accurate and reliable predictions for future values, making them highly suitable for time series forecasting tasks.

## **CONTENT**

This project begins with data preprocessing, where the daily climate dataset is cleaned, scaled, and prepared for modeling. After preprocessing, the data is transformed into a supervised learning format using a sliding window approach. Three deep learning models—RNN, LSTM, and GRU—are then designed and trained to forecast future climate values. Each model is built using TensorFlow/Keras and includes layers suitable for handling time series data. The models are trained and evaluated using performance metrics such as RMSE, MAE, and  $R^2$  score. A comparison of the results is presented to determine which model performs best for time series forecasting.

## **KEY FEATURES**

### ➤ **Real-World Climate Data:**

The dataset contains daily weather records from Delhi, making it suitable for real-time forecasting tasks.

### ➤ **Time Series Structure:**

The data is indexed by date, allowing for proper sequence modeling and trend analysis.

### ➤ **Deep Learning Models:**

Implements and compares RNN, LSTM, and GRU models to evaluate performance on sequential data.

### ➤ **Comprehensive Preprocessing:**

Includes data cleaning, scaling, and transformation into supervised learning format using a sliding window approach.

➤ **Performance Evaluation:**

Models are assessed using key metrics like RMSE, MAE, and  $R^2$  Score to measure accuracy and generalization.

➤ **Scalability and Adaptability:**

The modeling approach can be extended to other time series datasets and forecasting problems.

## **METHODOLOGY FOR TIME SERIES FORECASTING**

### **1. Data Collection and Exploration**

The dataset used comprises daily weather data from Delhi, including features such as:

- Temperature
- Humidity
- Wind Speed
- Air Pressure

The data spans multiple years and is indexed by date to preserve the temporal nature of the information. Initial exploration was performed to understand trends, seasonal patterns, and potential anomalies.

### **2. Data Preprocessing**

To prepare the raw data for modeling, several preprocessing steps were carried out:

- **Missing Value Treatment:** Checked and imputed or removed any missing or inconsistent records.
- **Feature Scaling:** Applied normalization techniques (e.g., MinMaxScaler or StandardScaler) to standardize the feature values for better convergence in deep learning models.
- **Sliding Window Technique:** The time series was converted into a supervised learning problem using a sliding window approach. This allowed the model to use a fixed-size sequence of past observations to predict future values.

### **3. Model Development**

Three deep learning architectures were implemented using TensorFlow/Keras to model and forecast future climate conditions:

- **Recurrent Neural Network (RNN):** A simple model to capture sequential dependencies.
- **Long Short-Term Memory (LSTM):** Designed to handle long-term dependencies and mitigate the vanishing gradient problem common in standard RNNs.
- **Gated Recurrent Unit (GRU):** A lightweight alternative to LSTM, with fewer parameters but competitive performance.

Each model includes:

- Input layers to accept sliding window sequences.
- One or more recurrent layers (RNN, LSTM, or GRU).
- Dense output layers to produce predictions.

### **4. Model Training**

- **Loss Function:** Mean Squared Error (MSE) was used as the loss function to optimize during training.

- **Optimizer:** Adaptive optimizers like Adam were employed for efficient convergence.
- **Validation:** A portion of the dataset was held out as a validation set to monitor overfitting and model generalization.

## 5. Model Evaluation

To assess the accuracy and performance of each model, the following metrics were calculated:

- **Root Mean Squared Error (RMSE)**
- **Mean Absolute Error (MAE)**
- **R<sup>2</sup> Score (Coefficient of Determination)**

These metrics helped compare the predictive capability and robustness of each model.

## 6. Result Analysis and Comparison

After training, predictions were visualized and compared against the actual values to interpret model performance. A comparative analysis was conducted to determine which deep learning model (RNN, LSTM, or GRU) provided the most accurate forecast for the dataset.

## 7. Generalization and Adaptability

The methodology developed is designed to be generalizable. It can be adapted to other geographical regions or time series forecasting tasks with minimal adjustments.

## IMPLEMENTATION

Based on the provided description, the sentiment analysis system utilizing deep learning models (CNN, LSTM, GRU) is implemented in the Jupyter Notebook titled "sentiment-analysis-cnn-lstm-gru.ipynb" within the GitHub repository: [Sentiment Analysis GitHub Repository](#).

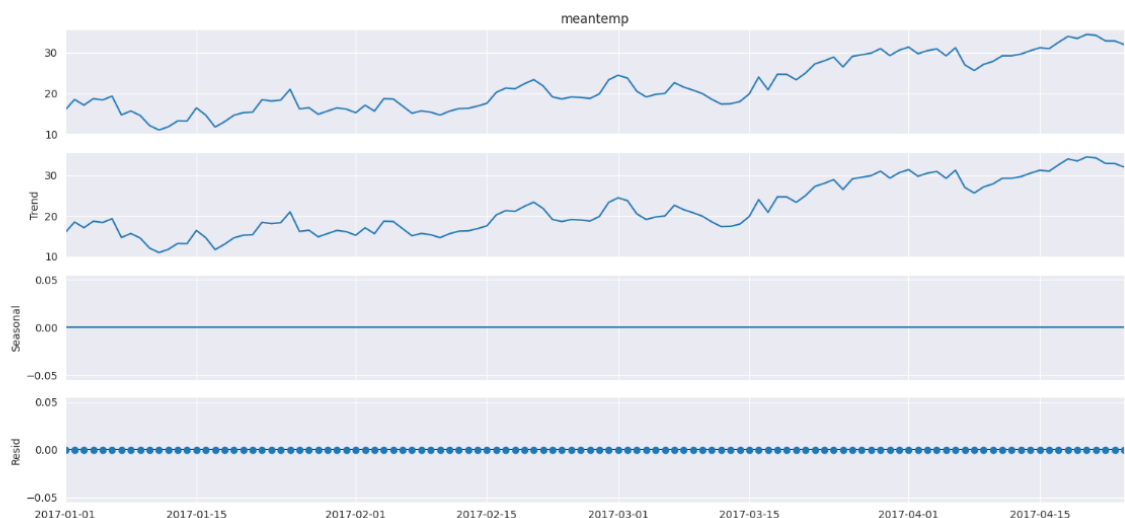
This notebook demonstrates the complete implementation, including:

- **Dataset Preprocessing:** Cleaning and processing Amazon product review text data for training sentiment classification models.
- **Model Architecture:** Building CNN, LSTM, and GRU models using TensorFlow/Keras with embedding layers for word representation and suitable layers for learning sequence patterns.
- **Training and Validation:** Training each model with the prepared dataset, applying dropout for regularization, and monitoring training/validation performance over multiple epochs.
- **Evaluation and Inference:** Evaluating model performance using accuracy, precision, recall, and F1-score, and testing the models on unseen data to assess generalization.

## RESULTS

### 1. Plotting Seasonal Decompositions

- **Observed:** The original temperature data over time, showing a clear upward trend from January to April 2017.
- **Trend:** This component highlights the general increase in temperature over time, possibly due to seasonal warming as the region transitions from winter to summer.
- **Seasonal:** Interestingly, the seasonal component is flat at zero, indicating that no significant repeating seasonal pattern was detected within the provided timeframe.
- **Residual:** The residual (noise) component is minimal, suggesting the trend accounts for most of the variation in the dataset.



### 2. Monthly Weather Trends (January to April)

#### Temperature

- The **average temperature increased** each month — from about 16°C in January to over 30°C in April.

- This shows the shift from winter to summer.

### Wind Speed

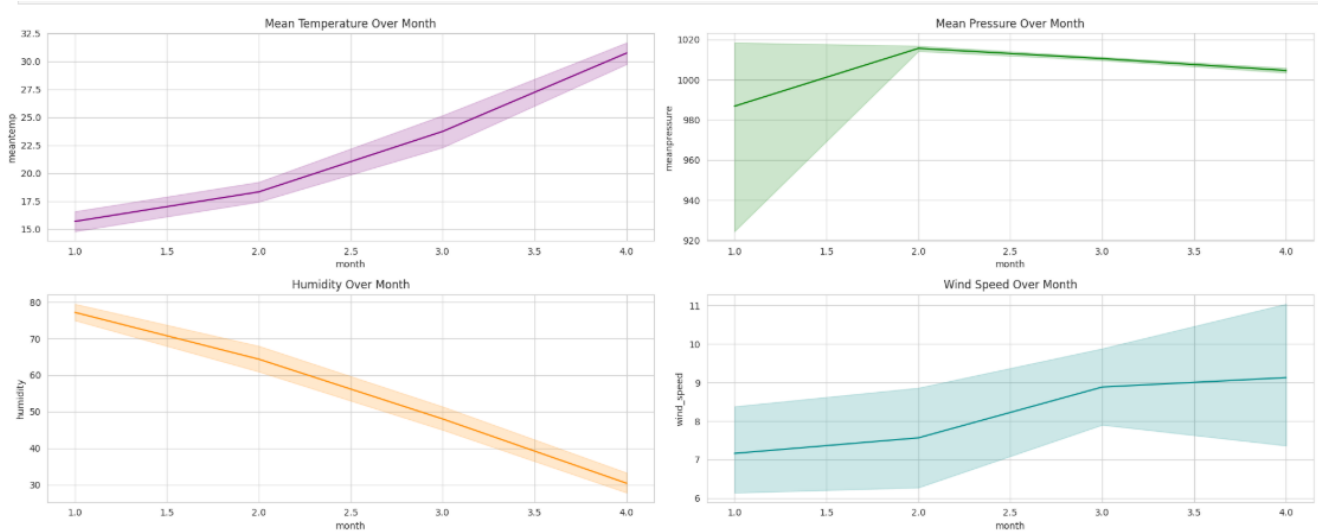
- **Wind speed also increased** a bit over the months, especially after February.
- This could be due to seasonal changes.

### Humidity

- **Humidity dropped** a lot — from around 75% in January to about 30% in April.
- As the temperature goes up, the air gets drier.

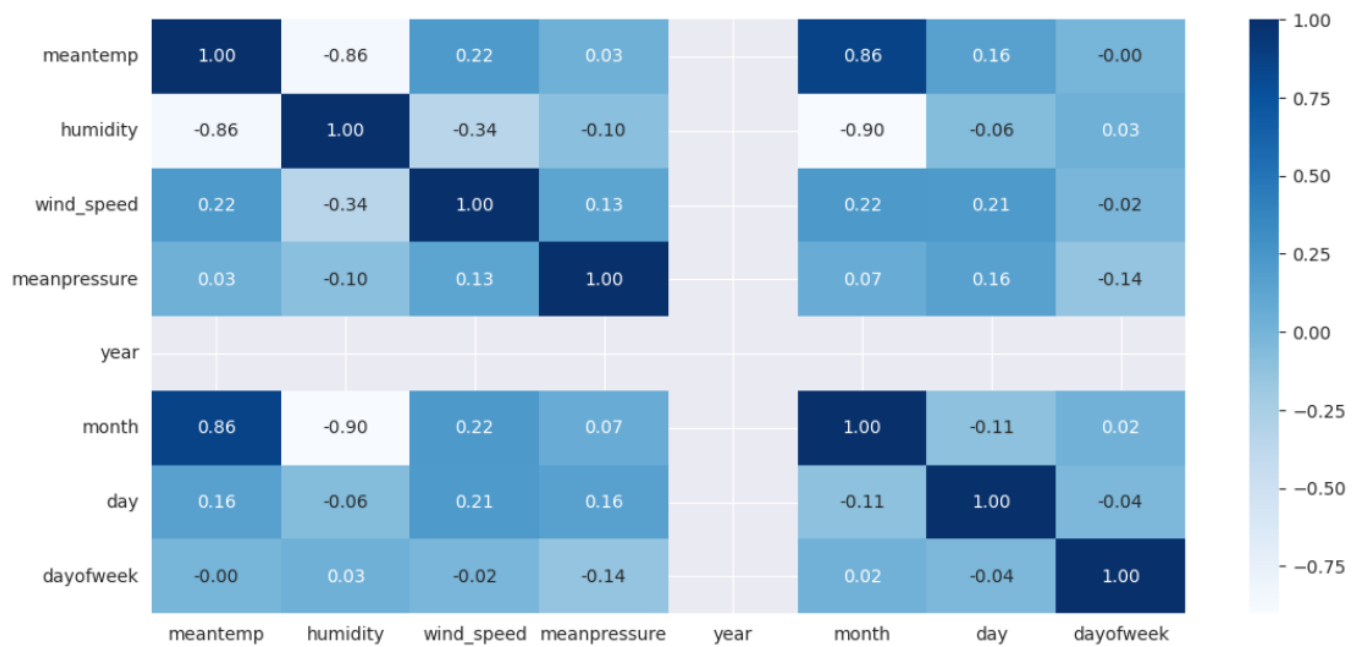
### Air Pressure

- Air pressure was highest in February, then slowly decreased.
- January had more variation in pressure.



## 3. Plotting Correlation and Heatmaps

- This heatmap shows how different weather features are related to each other. The values range from -1 to 1:  
**+1 means strong positive correlation** (both increase together).  
**-1 means strong negative correlation** (one increases, the other decreases).  
**0 means no correlation.**
- **Key Observations:**
- **Temperature & Humidity:** Strong **negative correlation** (-0.86). As temperature increases, humidity drops — this makes sense in dry seasons.
- **Temperature & Month:** Strong **positive correlation** (+0.86). As months go from January to April, temperature increases.
- **Humidity & Month:** Strong **negative correlation** (-0.90). Humidity decreases over the months.
- **Wind Speed:** Has a **weak positive correlation** with temperature and day — not strongly linked to other features.
- **Pressure:** Shows very low correlation with other features, meaning it changes more independently.



## 4. RNN

```

Epoch 1/50
88/88 ————— 2s 6ms/step - loss: 0.0871 - val_loss: 0.0439
Epoch 2/50
88/88 ————— 0s 3ms/step - loss: 0.0163 - val_loss: 0.0133
Epoch 3/50
88/88 ————— 0s 3ms/step - loss: 0.0186 - val_loss: 0.0148
Epoch 4/50
88/88 ————— 0s 3ms/step - loss: 0.0139 - val_loss: 0.0055
Epoch 5/50
88/88 ————— 0s 3ms/step - loss: 0.0182 - val_loss: 0.0084
Epoch 6/50
88/88 ————— 0s 3ms/step - loss: 0.0174 - val_loss: 0.0195
Epoch 7/50
88/88 ————— 0s 3ms/step - loss: 0.0161 - val_loss: 0.0191
Epoch 8/50
88/88 ————— 0s 3ms/step - loss: 0.0100 - val_loss: 0.0219
Epoch 9/50
88/88 ————— 0s 3ms/step - loss: 0.0092 - val_loss: 0.0170
Epoch 10/50
88/88 ————— 1s 3ms/step - loss: 0.0103 - val_loss: 0.0257
Epoch 11/50
88/88 ————— 0s 3ms/step - loss: 0.0117 - val_loss: 0.0138
Epoch 12/50
88/88 ————— 1s 3ms/step - loss: 0.0091 - val_loss: 0.0321
Epoch 13/50
88/88 ————— 0s 3ms/step - loss: 0.0134 - val_loss: 0.0250
Epoch 14/50
88/88 ————— 0s 3ms/step - loss: 0.0124 - val_loss: 0.0185
1/1 ————— 0s 301ms/step - loss: 0.0055
Validation Loss: 0.0054764519445598125

```

```

1/1 ————— 0s 157ms/step

```

```

RMSE: 1.507813170326015

```

```

R2 Score: 0.6469011420623805

```

## Weather Data Insights

- **Temperature is rising** month by month.
- **Humidity is falling** as the months go on.
- **Pressure and wind speed** change a little but not too much.

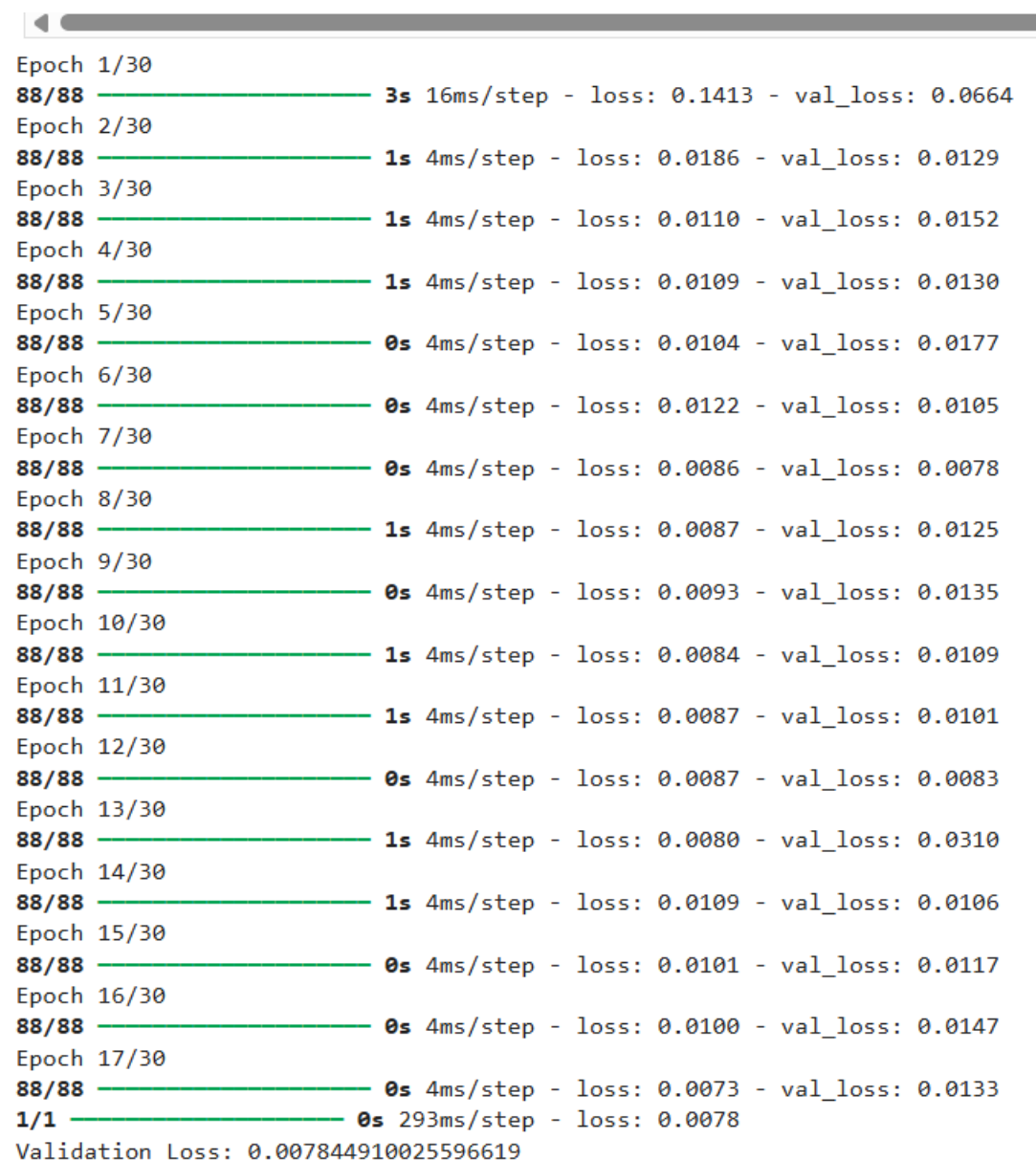
## Correlations (How things are related)

- When **temperature goes up**, **humidity goes down**.
- As the **month increases**, the **temperature increases**.
- As the **month increases**, the **humidity decreases**.

## Model Training Results

- The model **learned well** — loss got smaller each time.
- Final result shows the model makes **pretty good predictions**.
- **RMSE (error)** is low → Model's predictions are close to real values.
- **R<sup>2</sup> Score is 0.65** → The model explains about **65% of the data's pattern**

## 5.LSTM



```
Epoch 1/30
88/88 ————— 3s 16ms/step - loss: 0.1413 - val_loss: 0.0664
Epoch 2/30
88/88 ————— 1s 4ms/step - loss: 0.0186 - val_loss: 0.0129
Epoch 3/30
88/88 ————— 1s 4ms/step - loss: 0.0110 - val_loss: 0.0152
Epoch 4/30
88/88 ————— 1s 4ms/step - loss: 0.0109 - val_loss: 0.0130
Epoch 5/30
88/88 ————— 0s 4ms/step - loss: 0.0104 - val_loss: 0.0177
Epoch 6/30
88/88 ————— 0s 4ms/step - loss: 0.0122 - val_loss: 0.0105
Epoch 7/30
88/88 ————— 0s 4ms/step - loss: 0.0086 - val_loss: 0.0078
Epoch 8/30
88/88 ————— 1s 4ms/step - loss: 0.0087 - val_loss: 0.0125
Epoch 9/30
88/88 ————— 0s 4ms/step - loss: 0.0093 - val_loss: 0.0135
Epoch 10/30
88/88 ————— 1s 4ms/step - loss: 0.0084 - val_loss: 0.0109
Epoch 11/30
88/88 ————— 1s 4ms/step - loss: 0.0087 - val_loss: 0.0101
Epoch 12/30
88/88 ————— 0s 4ms/step - loss: 0.0087 - val_loss: 0.0083
Epoch 13/30
88/88 ————— 1s 4ms/step - loss: 0.0080 - val_loss: 0.0310
Epoch 14/30
88/88 ————— 1s 4ms/step - loss: 0.0109 - val_loss: 0.0106
Epoch 15/30
88/88 ————— 0s 4ms/step - loss: 0.0101 - val_loss: 0.0117
Epoch 16/30
88/88 ————— 0s 4ms/step - loss: 0.0100 - val_loss: 0.0147
Epoch 17/30
88/88 ————— 0s 4ms/step - loss: 0.0073 - val_loss: 0.0133
1/1 ————— 0s 293ms/step - loss: 0.0078
Validation Loss: 0.007844910025596619
```



---

RMSE: 1.8046444740500398  
R2 Score: 0.4941933206667226


### Metrics:

- **RMSE: 1.8046**  
→ A bit more prediction error compared to the first one.
- **R<sup>2</sup> Score: 0.4941**  
→ Explains less of the data's variation. Still okay, but not as strong.

### Training Progress:

- The model trained for **17 epochs**.
- The losses (both training and validation) generally decreased, meaning the model learned over time.
- Final validation loss: **0.0078**, which is slightly higher than the first model's.

## 6.GRU



```
Epoch 1/30
88/88 ————— 3s 8ms/step - loss: 0.0418 - val_loss: 0.0242
Epoch 2/30
88/88 ————— 1s 5ms/step - loss: 0.0086 - val_loss: 0.0088
Epoch 3/30
88/88 ————— 1s 4ms/step - loss: 0.0098 - val_loss: 0.0056
Epoch 4/30
88/88 ————— 1s 4ms/step - loss: 0.0079 - val_loss: 0.0161
Epoch 5/30
88/88 ————— 0s 4ms/step - loss: 0.0109 - val_loss: 0.0093
Epoch 6/30
88/88 ————— 0s 4ms/step - loss: 0.0084 - val_loss: 0.0063
Epoch 7/30
88/88 ————— 0s 4ms/step - loss: 0.0086 - val_loss: 0.0070
Epoch 8/30
88/88 ————— 0s 4ms/step - loss: 0.0088 - val_loss: 0.0152
Epoch 9/30
88/88 ————— 0s 5ms/step - loss: 0.0080 - val_loss: 0.0071
Epoch 10/30
88/88 ————— 0s 5ms/step - loss: 0.0083 - val_loss: 0.0090
Epoch 11/30
88/88 ————— 1s 5ms/step - loss: 0.0087 - val_loss: 0.0117
Epoch 12/30
88/88 ————— 0s 5ms/step - loss: 0.0067 - val_loss: 0.0094
Epoch 13/30
88/88 ————— 1s 5ms/step - loss: 0.0097 - val_loss: 0.0105
1/1 ————— 0s 318ms/step - loss: 0.0056
Validation Loss: 0.005616790149360895
```

---

RMSE: 1.527009485835524  
R2 Score: 0.6378531451240571

### Training Progress (Loss & Validation Loss)

- This shows how your model is learning over **13 epochs**.
- **Loss**: How well your model is doing on the training data.
- **Val\_loss**: How well your model is doing on unseen (validation) data.

### Key Info:

- The losses are **going down**, which means the model is learning.
- Final **Validation Loss: 0.0056** → This is quite low, meaning good performance.

### Evaluation Metrics (RMSE & R<sup>2</sup> Score)

- **RMSE (Root Mean Squared Error): 1.5270**
  - Lower is better. It tells how much error your model makes when predicting.
- **R<sup>2</sup> Score: 0.6378**
  - This tells how well your model explains the data.
  - 1.0 is perfect, and 0 means it's not better than average.

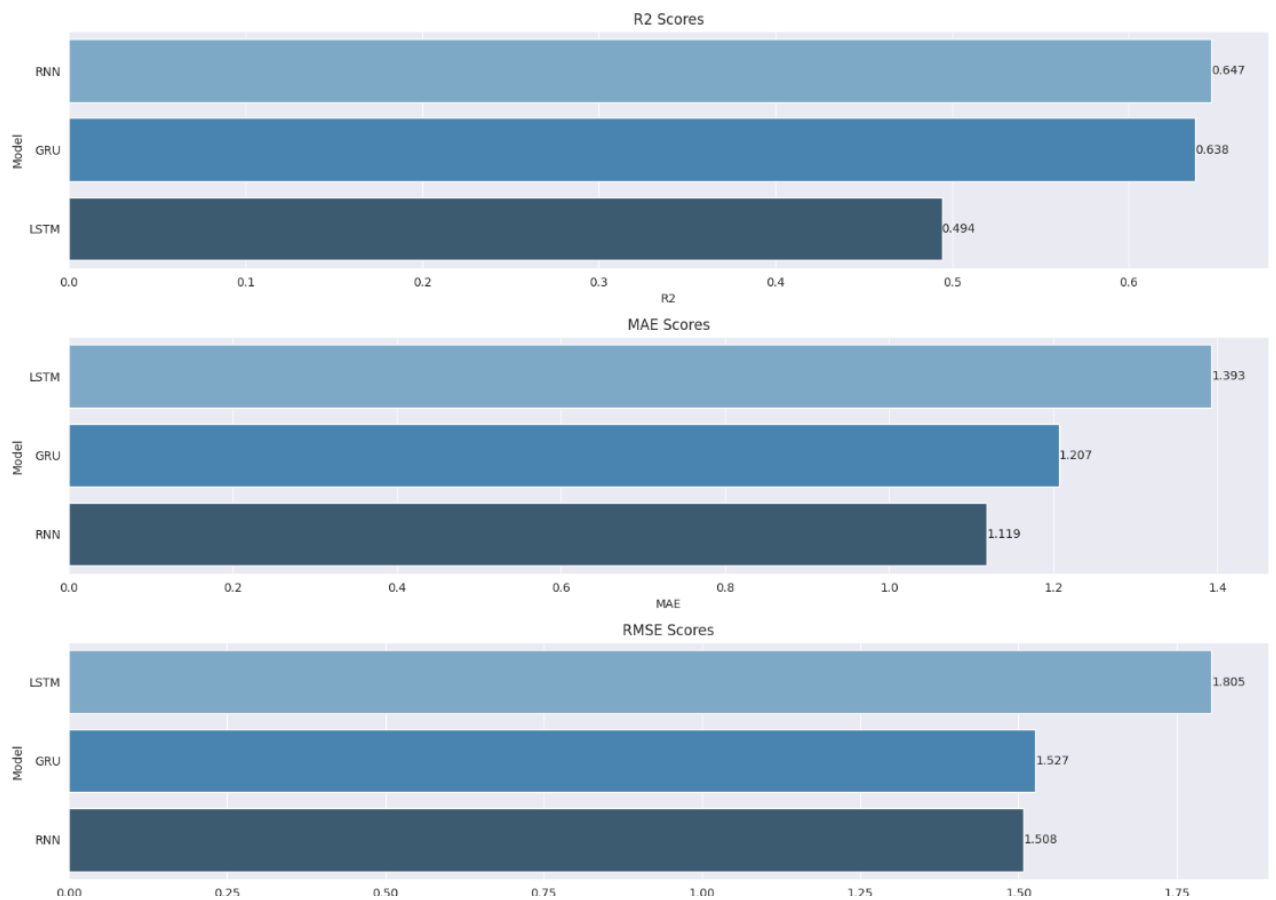
## CONCLUSION

### Key Points:

- **R<sup>2</sup> Score** (Higher is better): Measures how well the model explains the data.
- **MAE (Mean Absolute Error)** (Lower is better): Average absolute difference between predicted and actual values.
- **RMSE (Root Mean Squared Error)** (Lower is better): Penalizes larger errors more than MAE.

## CONCLUSION

- **Best Model Overall: RNN**
  - **Highest R<sup>2</sup> score (0.647)** → Best at explaining data variability.
  - **Lowest MAE (1.119)** → Most accurate on average.
  - **Lowest RMSE (1.508)** → Fewer large errors.
- **Runner-Up: GRU**
  - **Slightly lower R<sup>2</sup> (0.638) and higher errors than RNN.**
  - **Still a strong model.**
- **LSTM Performance:**
  - **Lowest R<sup>2</sup> (0.494)** → Explains less variance.
  - **Highest MAE and RMSE** → Less accurate, more error-prone.



- **RNN is the most effective model** for your current task.
- **LSTM underperformed** in all metrics and may need tuning or reconsideration.
- **GRU is a solid middle ground**, offering good balance between accuracy and complexity.