Farbod pourreza
S.No : 610399211
Data mining project

Comparison of Model Performance and Techniques

Models Evaluated:

1. **Long Short-Term Memory (LSTM)**
2. **Gated Recurrent Unit (GRU)**
3. **Simple Recurrent Neural Network (RNN)**

Evaluation Metrics:

- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- R-squared (R^2)

Long Short-Term Memory (LSTM)

Overview: LSTM networks are a special kind of RNN capable of learning long-term dependencies. They were introduced to mitigate the vanishing gradient problem.

Key Components:

- **Cell State:** Maintains information over long periods.
- **Gates:** Control the flow of information into and out of the cell state.
  - **Forget Gate:** Decides what information to discard from the cell state.
  - **Input Gate:** Determines what new information to store in the cell state.
  - **Output Gate:** Controls the output based on the cell state.

Gated Recurrent Unit (GRU)

Overview: GRU is a variant of the LSTM that combines the forget and input gates into a single update gate. It simplifies the LSTM architecture while retaining its advantages.

Key Components:

- **Update Gate:** Combines the functionalities of the forget and input gates.
- **Reset Gate:** Controls how much of the previous hidden state to forget.

Advantages:

- **Simpler Architecture:** Fewer gates and parameters make GRUs faster and more efficient to train compared to LSTMs.
- **Performance:** Often performs comparably to LSTMs on various tasks with less computational cost.

Recurrent Neural Networks (RNN)

Overview: Recurrent Neural Networks (RNNs) are a type of neural network designed for sequential data. They are used in applications where the data points are dependent on previous ones, such as time series prediction, natural language processing, and speech recognition.

Key Features:

- **Sequential Processing:** RNNs process sequences of data by maintaining a hidden state that captures information about previous elements in the sequence.
- **Backpropagation Through Time (BPTT):** The training process involves back propagating errors through time to update weights, making it computationally intensive.

Loss Functions

Loss functions are used to evaluate how well the model's predictions match the actual data. They measure the difference between the predicted output and the actual output.

1. Mean Absolute Error (MAE)

Definition: MAE is the average of the absolute differences between the predicted values and the actual values. It gives a measure of how close the predictions are to the actual outcomes.

Formula:

$$\frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right|$$

- n is the number of observations
- yi is the actual value
- y^i is the predicted value

Characteristics:

- MAE is straightforward to interpret.
- It gives equal weight to all errors, without considering their direction (positive or negative).

2. Mean Squared Error (MSE)

Definition: MSE is the average of the squared differences between the predicted values and the actual values. It emphasizes larger errors due to squaring the differences.

Formula:

$$: \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Characteristics:

- MSE penalizes larger errors more than smaller ones, making it sensitive to outliers.
- It provides a good measure of overall model accuracy.

3. Root Mean Squared Error (RMSE)

Definition: RMSE is the square root of the average of the squared differences between the predicted values and the actual values. It provides an error metric in the same units as the original data.

Formula:

$$\sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$$

Characteristics:

- RMSE is useful for understanding the typical magnitude of errors in a model's predictions.
- Like MSE, it penalizes larger errors more heavily.

4. R-squared (R²)

Definition: R-squared, also known as the coefficient of determination, is a statistical measure that indicates the proportion of the variance in the dependent variable that is predictable from the independent variable(s).

Formula:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

Where:

SS_{res} (Residual Sum of Squares) = $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

SS_{tot} (Total Sum of Squares) = $\sum_{i=1}^{n}(y_i - \bar{y})^2$

$\bar{y}$ is the mean of the actual values

Characteristics:

- R² ranges from 0 to 1, where 0 indicates that the model explains none of the variability of the response data around its mean, and 1 indicates that it explains all the variability.
- Higher R² values indicate a better fit of the model to the data.

Optimizers

Optimizers are algorithms or methods used to adjust the weights of the neural network to minimize the loss function. They play a crucial role in the training process by improving the performance and accuracy of the model. Here are some common optimizers used in neural networks:

1. Stochastic Gradient Descent (SGD)

SGD updates the model's parameters using the gradient of the loss function with respect to the parameters. It updates the parameters after each training example.

Formula:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)})$$

Where:

- θ are the parameters (weights) of the model
- η is the learning rate
- $\nabla_\theta J(\theta; x^{(i)}; y^{(i)})$ is the gradient of the loss function with respect to the parameters

2. Adam (Adaptive Moment Estimation)

Adam combines the benefits of two other extensions of SGD, AdaGrad and RMSProp. It computes adaptive learning rates for each parameter.

Formulas:

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$$
$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$$
$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$
$$\theta = \theta - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon}$$

Where:

- m_t and v_t are the first and second moment estimates
- β_1 and β_2 are the decay rates for the moment estimates
- g_t is the gradient at time step t
- ϵ is a small constant to prevent division by zero

3. RMSProp (Root Mean Square Propagation)

RMSProp adjusts the learning rate for each parameter based on the average of recent magnitudes of the gradients for that parameter.

Formula:

$$E[g^2]_t = \beta E[g^2]_{t-1} + (1 - \beta) g_t^2$$
$$\theta = \theta - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

Where:

- $E[g^2]_t$ is the exponentially decaying average of past squared gradients
- β is the decay rate
- g_t is the gradient at time step t
- ϵ is a small constant

Feature Processing

Feature processing is a critical part of data preparation for modeling. This section covers some of the feature processing techniques used in the training process.

1. Adding New Features

Initially, some new features were defined but ultimately not used because they did not significantly improve the results. These features included the interaction between temperature and humidity (`temp_humidity_interaction`) and the interaction between wind speed and mean pressure (`wind_pressure_interaction`).

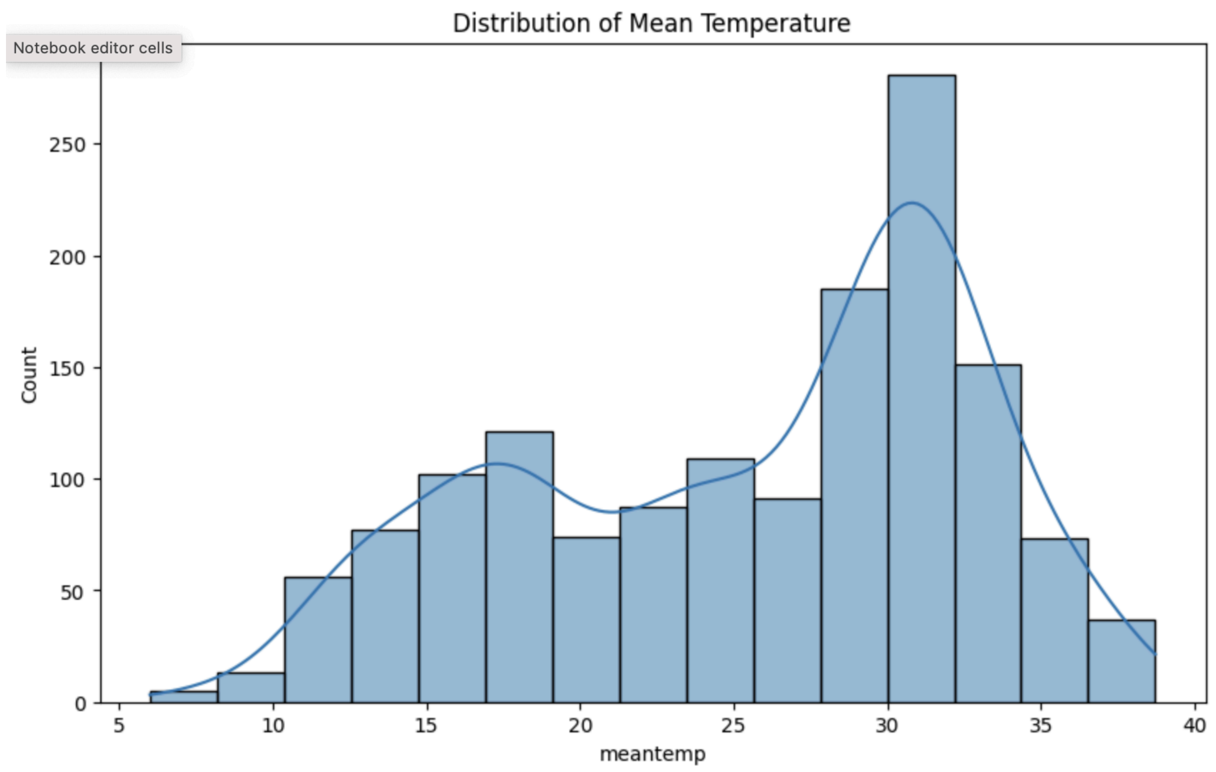
2. Handling missing values

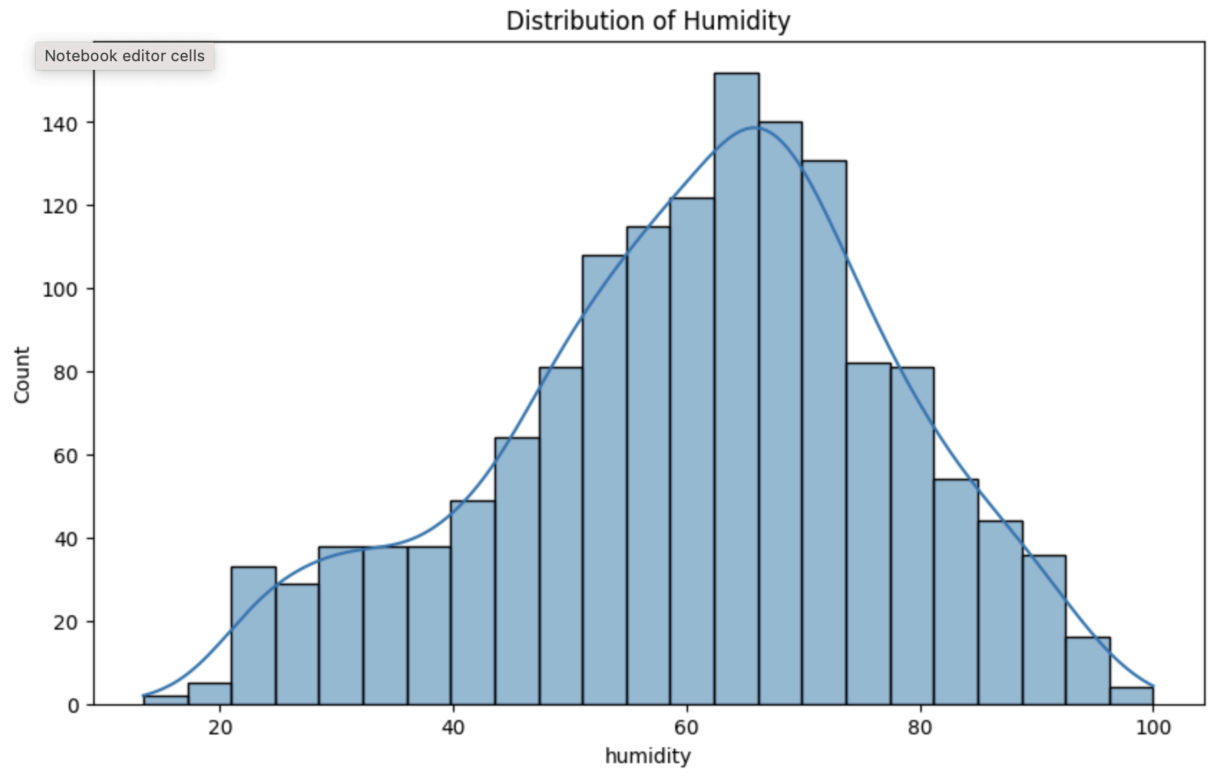
Since we don't have missing values in the dataset we can skip this part.

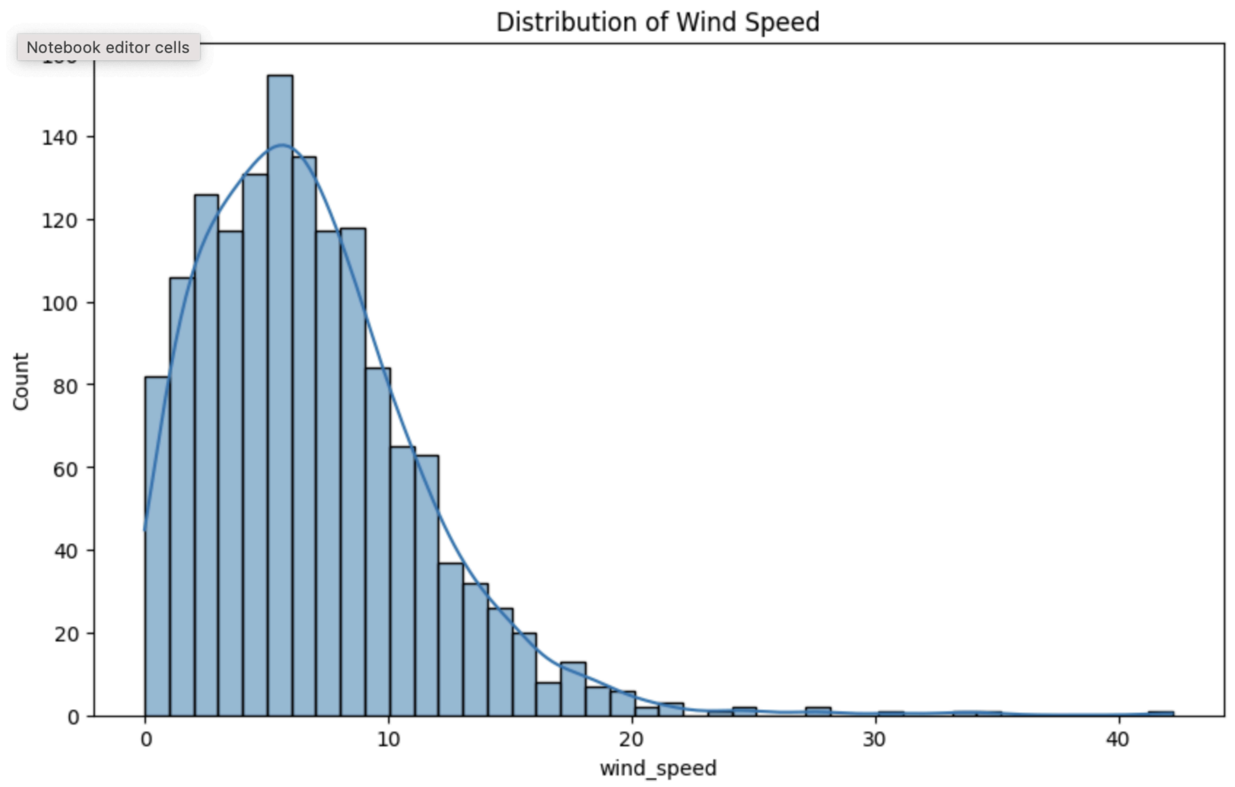
3. Normalization

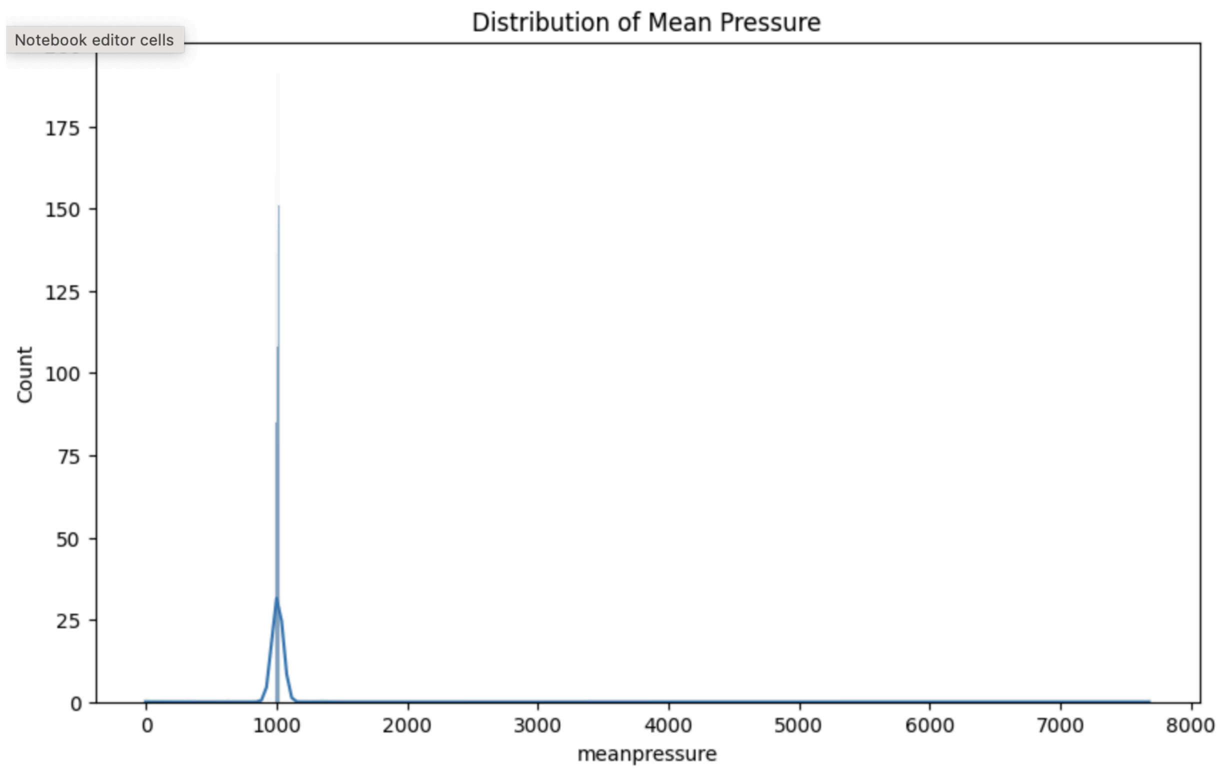
To ensure that the features are within a suitable range, normalization was performed. Features that needed normalization included humidity, mean temperature, wind speed, and the defined interactions. `StandardScaler` was used for this purpose.

We used Standardscaler because features distributions are close to normal:

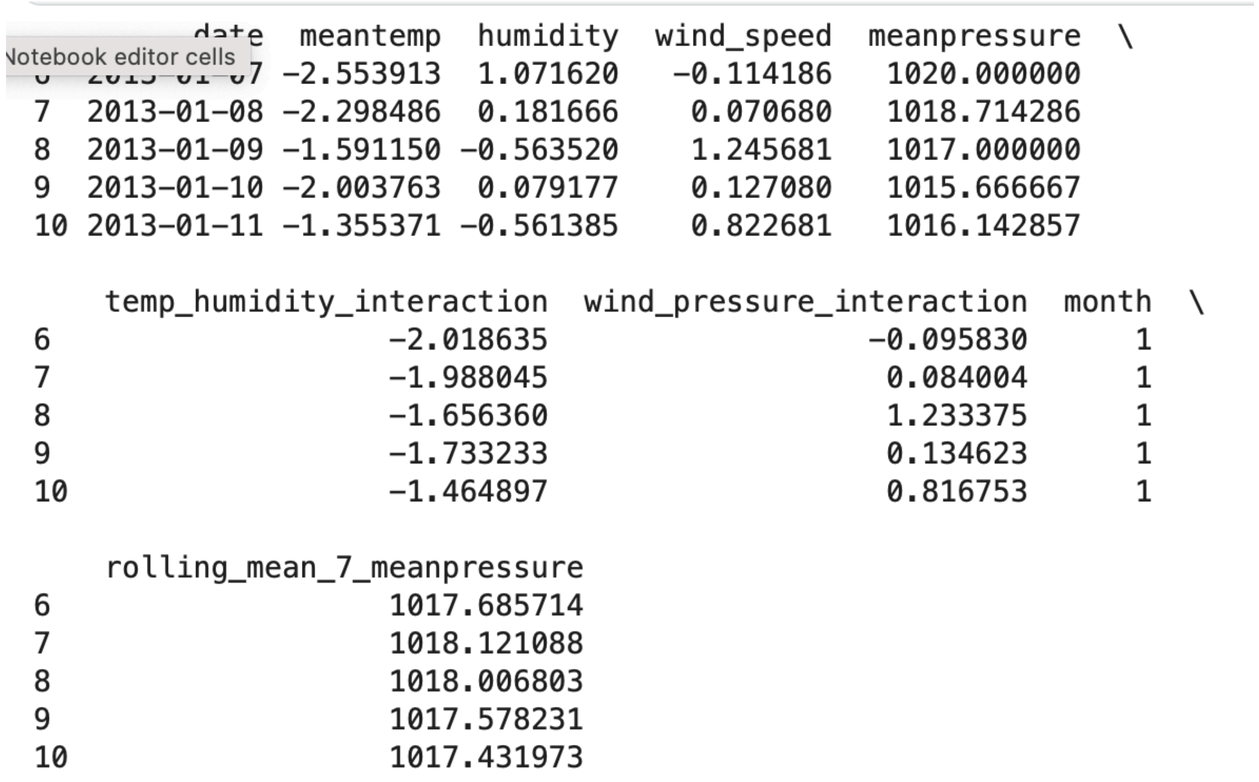

Distribution of Mean Temperature
Notebook editor cells
Count
0
50
100
150
200
250
5
10
15
20
25
30
35
40
meantemp


Distribution of Humidity
Notebook editor cells
Count
0
20
40
60
80
100
120
140
20
40
60
80
100
humidity


Distribution of Wind Speed
Notebook editor cells
Count
140
120
100
80
60
40
20
0
0
10
20
30
40
wind_speed


Distribution of Mean Pressure
Notebook editor cells
Count
175
150
125
100
75
50
25
0
0
1000
2000
3000
4000
5000
6000
7000
8000
meanpressure

Data after normalization:



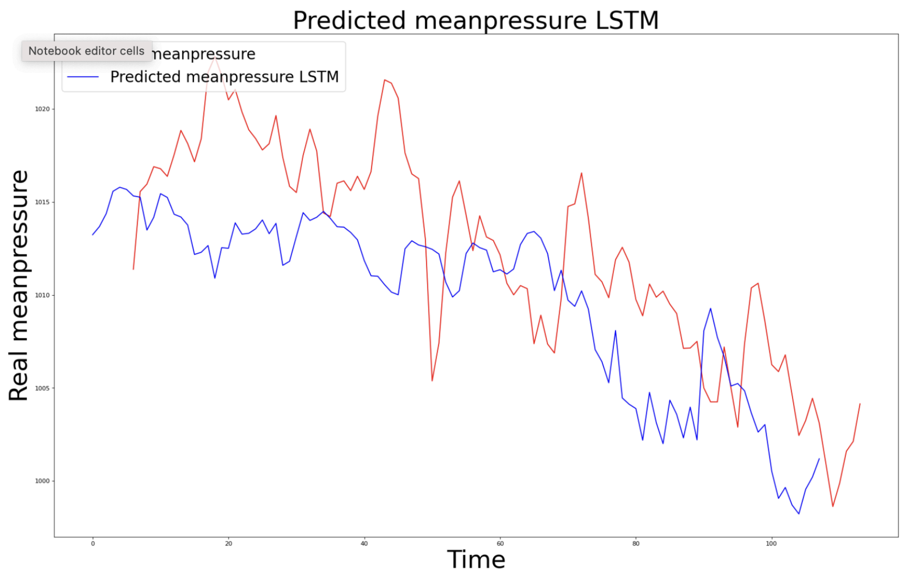
```
          date  meantemp  humidity  wind_speed  meanpressure  \
6   2013-01-07 -2.553913  1.071620   -0.114186   1020.000000
7   2013-01-08 -2.298486  0.181666    0.070680   1018.714286
8   2013-01-09 -1.591150 -0.563520    1.245681   1017.000000
9   2013-01-10 -2.003763  0.079177    0.127080   1015.666667
10  2013-01-11 -1.355371 -0.561385    0.822681   1016.142857

    temp_humidity_interaction  wind_pressure_interaction  month  \
6                   -2.018635                  -0.095830      1
7                   -1.988045                   0.084004      1
8                   -1.656360                   1.233375      1
9                   -1.733233                   0.134623      1
10                  -1.464897                   0.816753      1

    rolling_mean_7_meanpressure
6                   1017.685714
7                   1018.121088
8                   1018.006803
9                   1017.578231
10                  1017.431973
```

Results:

The performance of the three models was evaluated using the aforementioned metrics. They were all on the same settings (2 layers, units = 50, epoch = 50, dense = 25 , batch_size = 20) Here are the key findings:

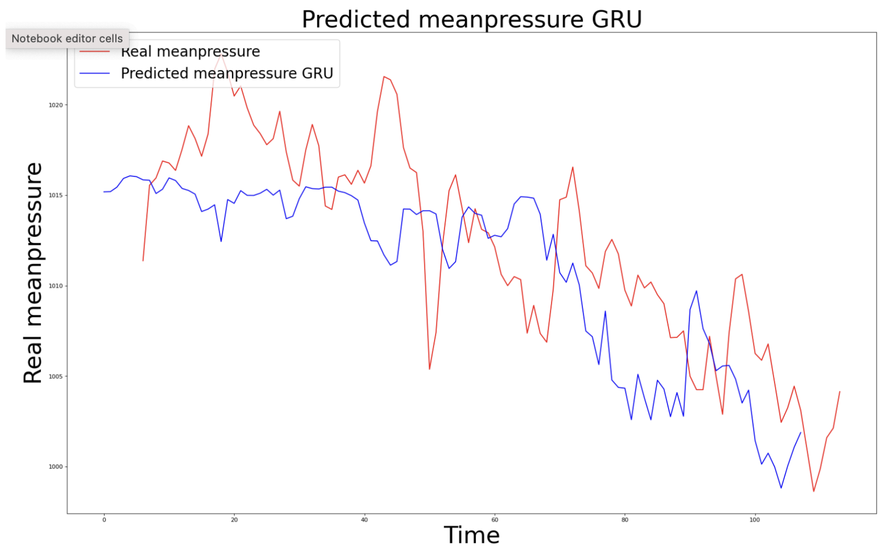
LSTM:

- **MAE:** 2.43
- **MSE:** 9.27
- **RMSE:** 3.04
- **R²:** 0.72



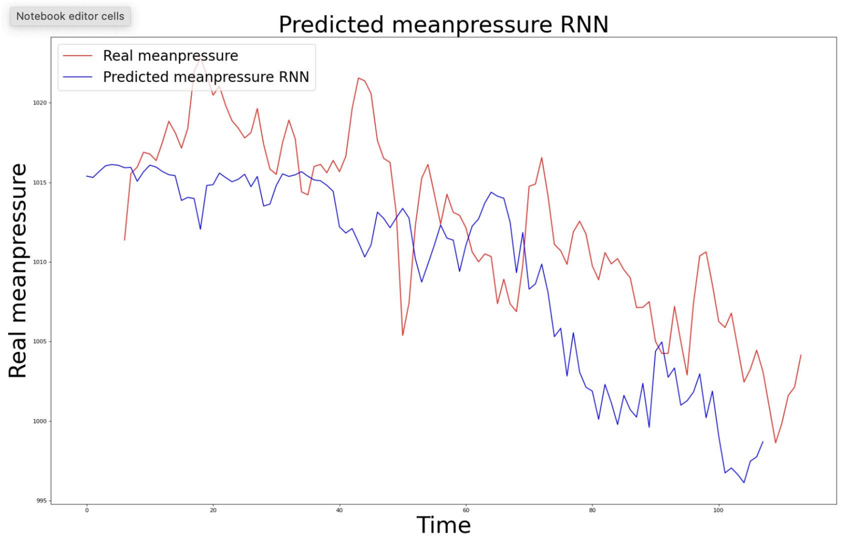
GRU:

- **MAE:** 2.23
- **MSE:** 8.83
- **RMSE:** 2.97
- **R²:** 0.73



RNN:

- **MAE:** 2.58
- **MSE:** 10.25
- **RMSE:** 3.20
- **R²:** 0.69



Analysis:

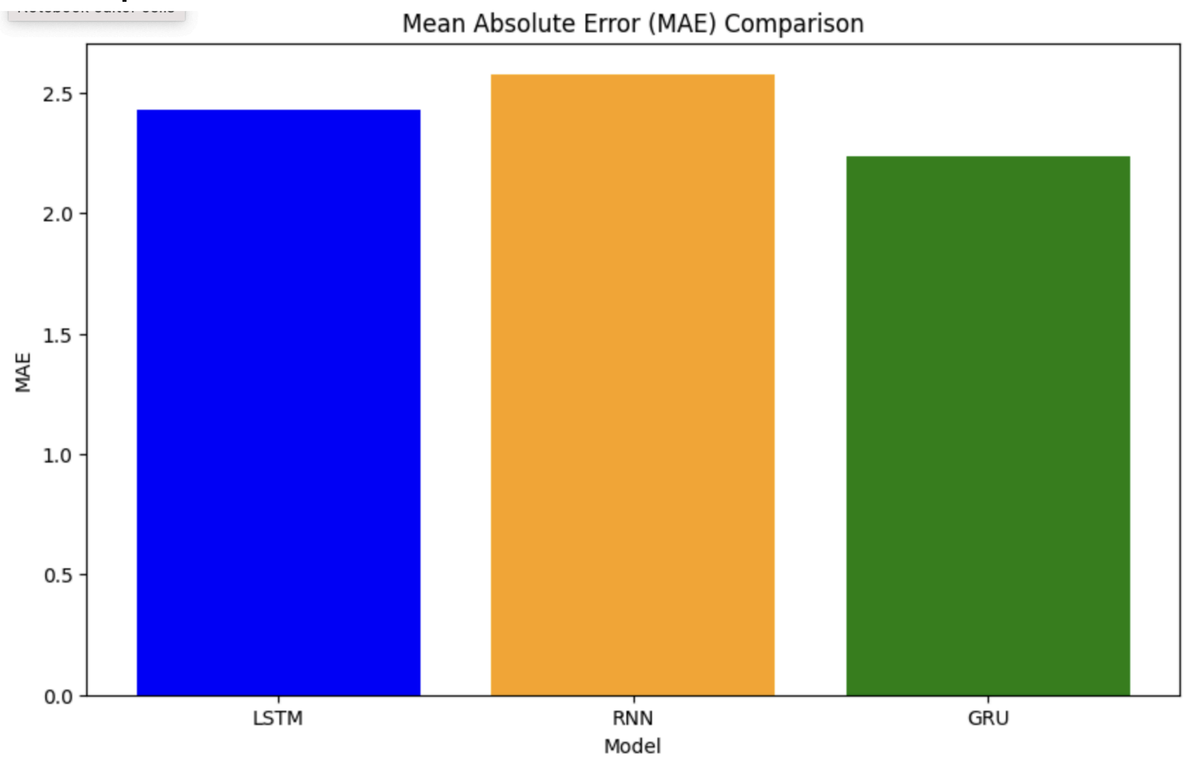
Strengths and Weaknesses:

- **LSTM:**
  - **Strengths:** Effective in capturing long-term dependencies, good performance across various metrics.
  - **Weaknesses:** More computationally intensive due to complex architecture.
- **GRU:**
  - **Strengths:** Comparable performance to LSTM with fewer parameters, making it more efficient.
  - **Weaknesses:** Slightly less capable of capturing very long dependencies compared to LSTM.
- **RNN:**
  - **Strengths:** Simpler and faster to train.
  - **Weaknesses:** Struggles with long-term dependencies due to vanishing gradient problem, leading to lower performance metrics.

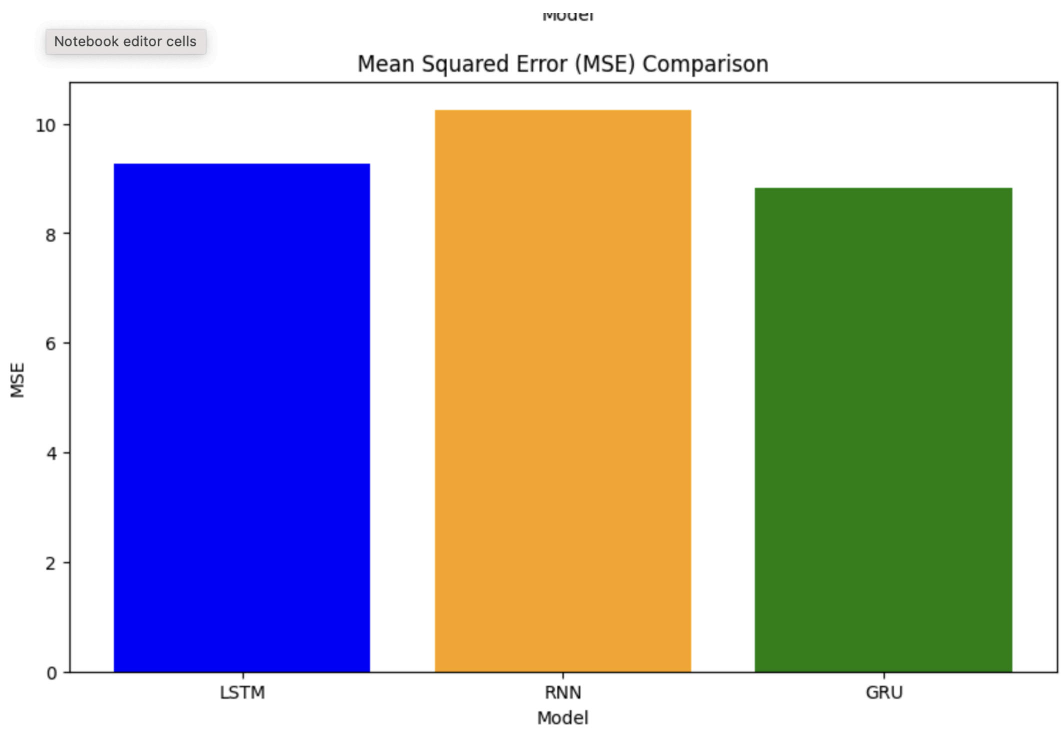
Visual Comparison:

Graphs were used to visually compare the performance of the models across the different metrics:

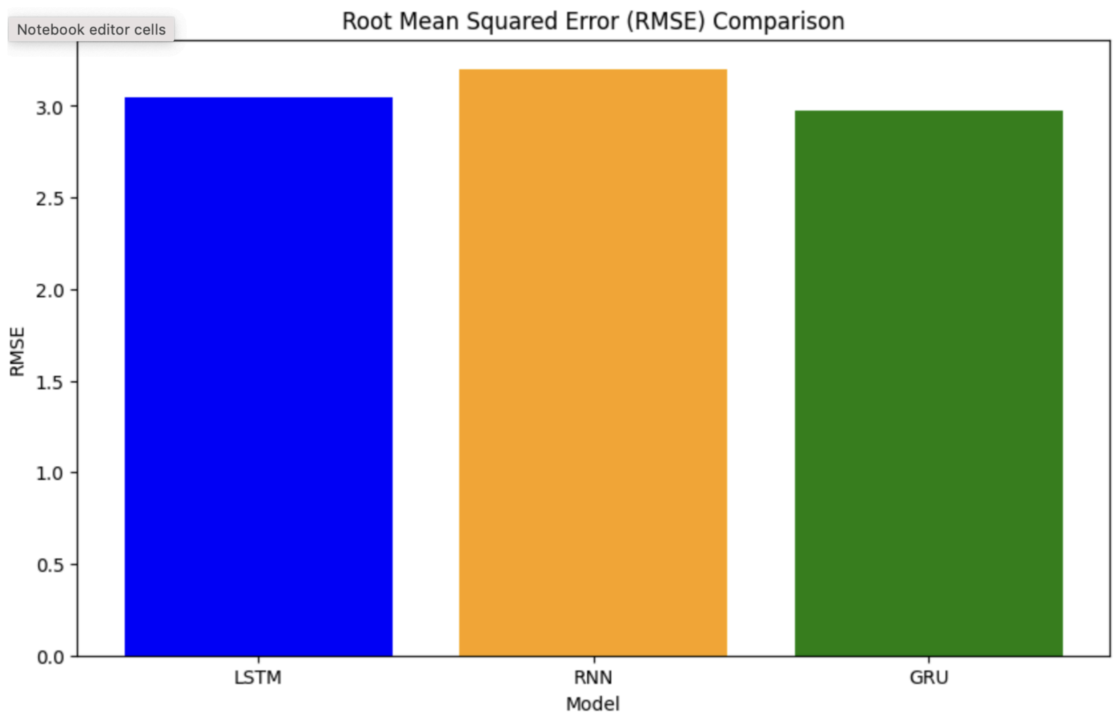
- **MAE Comparison:**



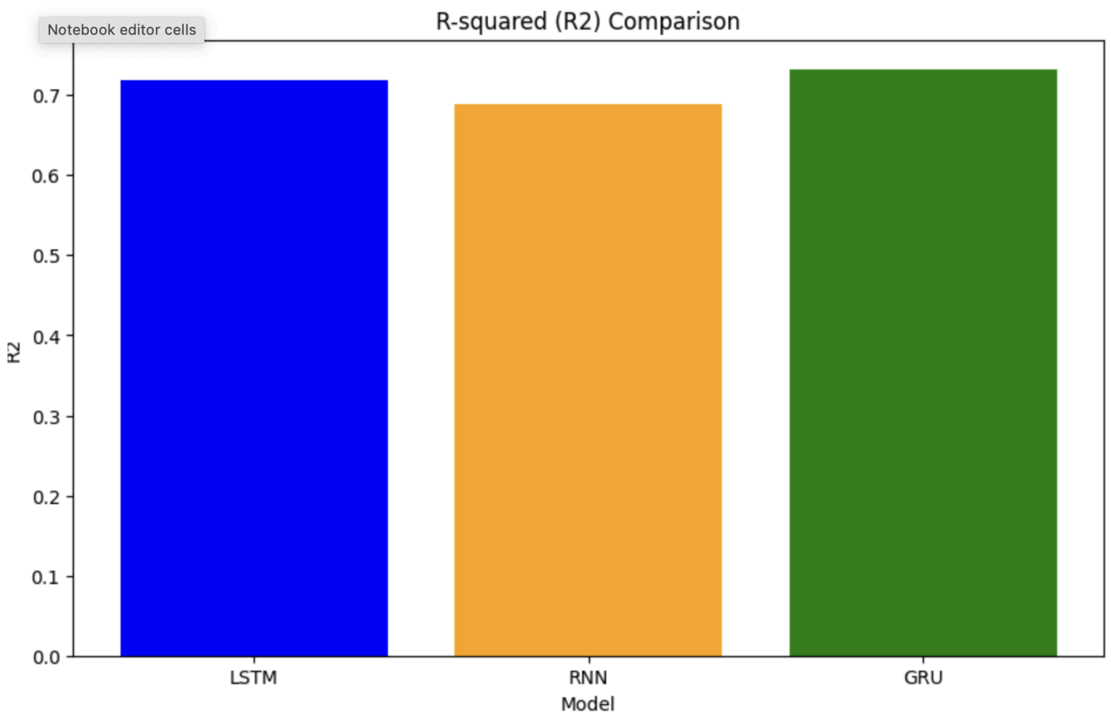
-
- **MSE Comparison:**



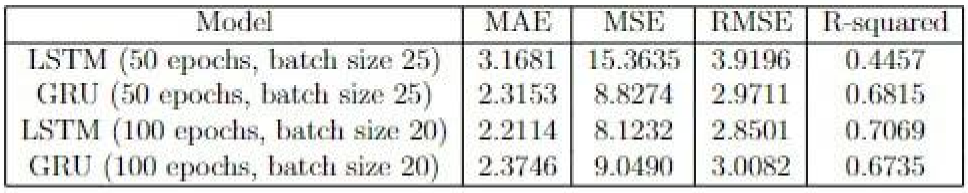
- **RMSE Comparison:**



- **R² Comparison:**



Paper Comparison:



Model	MAE	MSE	RMSE	R-squared
LSTM (50 epochs, batch size 25)	3.1681	15.3635	3.9196	0.4457
GRU (50 epochs, batch size 25)	2.3153	8.8274	2.9711	0.6815
LSTM (100 epochs, batch size 20)	2.2114	8.1232	2.8501	0.7069
GRU (100 epochs, batch size 20)	2.3746	9.0490	3.0082	0.6735

Based on this paper's results, they are close to our results.
The GRU model with 50 epochs and batch size 25 from this paper is very close in performance to our results, with the model slightly outperforming it in MAE and RMSE
Our LSTM with 50 epochs and 20 batch size is doing better than LSTM with 50 epochs and 25 batch size of the paper.

Conclusion:

- **Best Model:** GRU and LSTM, due to their balanced performance and efficiency.
- **Recommendation:** Depending on the computational resources and specific requirements (e.g., need for long-term dependency handling), either GRU or LSTM could be selected. RNNs are generally less effective for complex sequence tasks due to their limitations in handling long-term dependencies.

Note that these results are based on batch_size = 20 , epoch = 50 settings , they can be improved if we change these settings.