**GATED RECURRENT UNIT:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Units** | **Look Back** | **Learning** | **Drop-out** | **Opti** | **Loss** | **Batch size** | **Patience** | **Epoch** | **Stop** | **MAE** | **RMSE** | **Train**  **Vali loss** |
|  | 1 | 30 |  | 0.2 | Adam | MSE | 16 | 10 | 100 |  | 1.6412 | 1.7711 |  |
|  | 1 | 48 |  | 0.2 | Adam | MSE | 16 | 10 | 100 | 63 | 0.7013 | 1.1176 | 2.2800e04  1.256e04 |
|  | 1 | 48 |  | 0.2 | Adam | MSE | 16 | 5 | 100 |  | 3.5646 | 3.6119 | 3.386e04  2.701e04 |
|  | 1 | 48 |  | 0.2 | Adam | MSE | 16 | 12 |  | 100 | 1.2584 | 1.4304 | 2.199e04 1.128e04 |
|  | 1 | 48 |  | 0.2 | Adam | MSE | 16 | 10 | 100 | 60 | 150.6249 | 216.7622 | 10.9876  6.1569 |
| **1** | New data | 48 |  | 0.2 | Adam | MSE | 16 | 10 | 100 | 66 | 1.0349 | 1.3535 | 2.246e04 1.285e04 |
| **2** | New data | 24 |  | 0.2 | Adam | MSE | 16 | 10 | 100 | 46 | 1.7580 | 1.8770 | 2.352e04  1.353e04 |
| **3** | New data | 48 | 0.0001 | 0.2 | Adam | MSE | 16 | 10 | 100 | 52 | 2.5794 | 2.9961 | 0.0103  0.0147 |
| **4** | New data | 48 | 0.0001 | 0.2 | Adam | MSE | 16 | 7 | 100 | 19 | 1.2702 | 1.5968 | 0.0137  0.0187 |
| **5** | New data | 48 | 0.01 | 0.2 | Adam | MSE | 16 | 7 | 100 | 35 | 1.4409 | 1.7368 | 0.0117  0.0137 |
| **6** | New data | 7 | 0.01 | 0.2 | Adam | MSE | 16 | 7 | 100 |  |  |  |  |

**LONG SHORT-TERM MEMORY**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Units** | **Look Back** | **Learning** | **Drop-out** | **Opti** | **Loss** | **Batch size** | **Patience** | **Epoch** | **Stop** | **MAE** | **MSE** | **Train Vali loss** |
| **1** | New data | 48 |  | 0.2 | Adam | MSE | 16 | 10 | 100 | 18 | 1.7109 | 1.8793 | 2.569e04 2.173e04 |
| **2** | New data | 24 |  | 0.2 | Adam | MSE | 16 | 10 | 100 | 46 | 1.2175 | 1.3828 | 2.251e04 1.194e04 |
| **3** | New data | 48 | 0.0001 | 0.2 | Adam | MSE | 16 | 10 | 100 | 19 | 3.6921 | 3.7830 | 0.0122 0.0182 |
| **4** | New data | 48 | 0.0001 | 0.2 | Adam | MSE | 16 | 7 | 100 | 14 | 5.5323 | 5.6720 | 0.0125 0.0198 |
| **5** | New data | 48 | 0.01 | 0.2 | Adam | MSE | 16 | 7 | 100 | 13 | 0.7993 | 1.1206 | 0.0135 0.0171 |
| **6** | New data | 7 | 0.01 | 0.2 | Adam | MSE | 16 | 7 | 100 |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Let’s decompose this time series by viewing the PACF (Partial Auto Correlation Function) plot, which measures how much the y variable, in our case, air passengers, is correlated to past values of itself and how far back a statistically significant correlation exists. The PACF plot is different from the ACF plot in that PACF controls for correlation between past terms.

omponents of LSTMs

So the LSTM cell contains the following components

* Forget Gate “f” ( a neural network with sigmoid)
* Candidate layer “C"(a NN with Tanh)
* Input Gate “I” ( a NN with sigmoid )
* Output Gate “O”( a NN with sigmoid)
* Hidden state “H” ( a vector )
* Memory state “C” ( a vector)
* Inputs to the LSTM cell at any step are Xt (current input) , Ht-1 (previous hidden state ) and Ct-1 (previous memory state).
* Outputs from the LSTM cell are Ht (current hidden state ) and Ct (current memory state)

linkcode

Working of gates in LSTMs

First, LSTM cell takes the previous memory state Ct-1 and does element wise multiplication with forget gate (f) to decide if present memory state Ct. If forget gate value is 0 then previous memory state is completely forgotten else f forget gate value is 1 then previous memory state is completely passed to the cell ( Remember f gate gives values between 0 and 1 ).

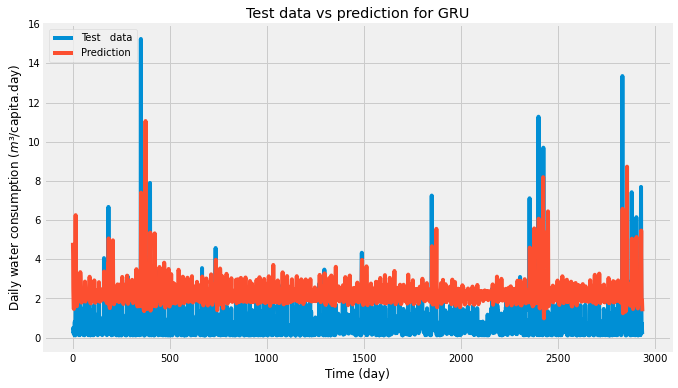
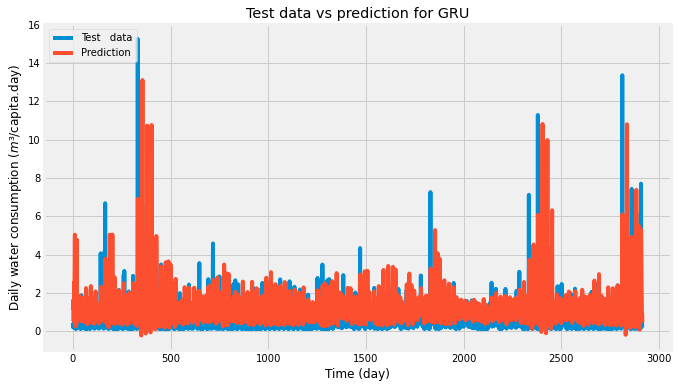
**Ct = Ct-1 \* ft**

Calculating the new memory state:

**Ct = Ct + (It \* C`t)**

Now, we calculate the output:

**Ht = tanh(Ct)**



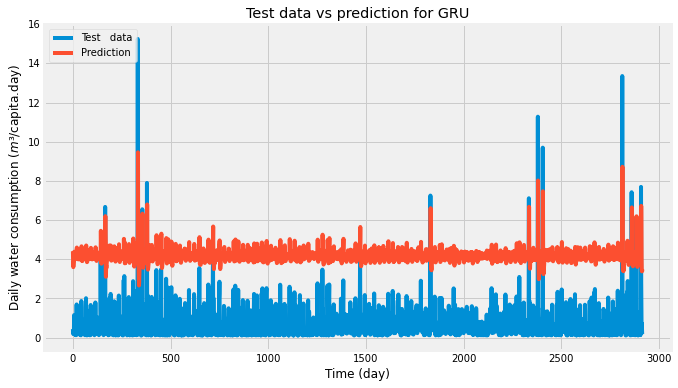


Figure EPOCH 5

Lookback / lags

Different models – feature engineering

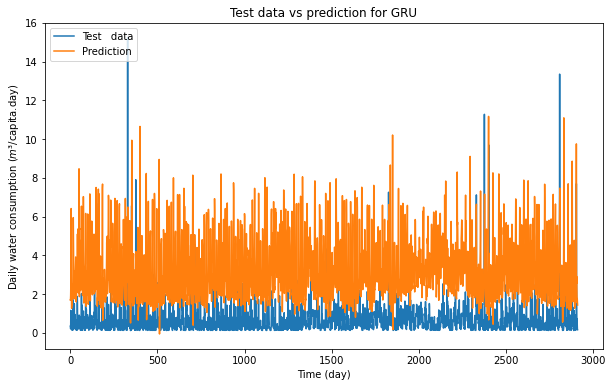
Overall complexity – important !

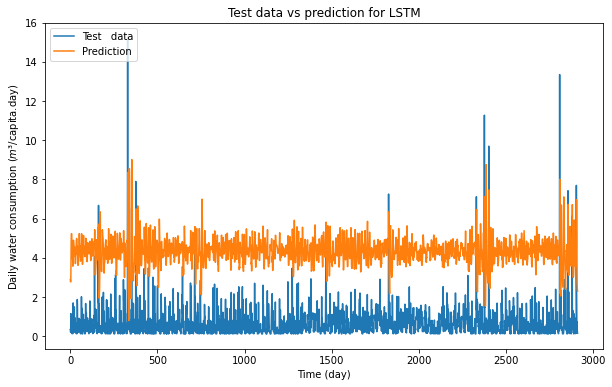
Nr models

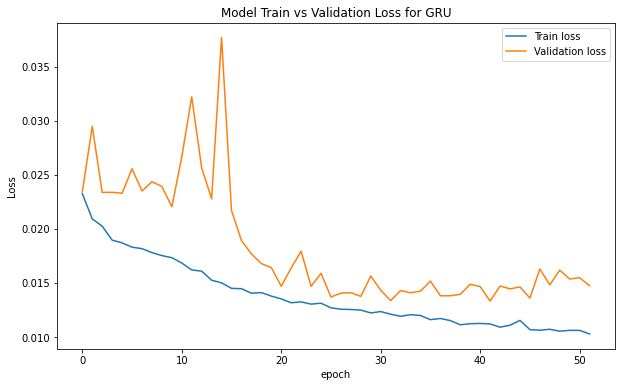
Hyperparameter

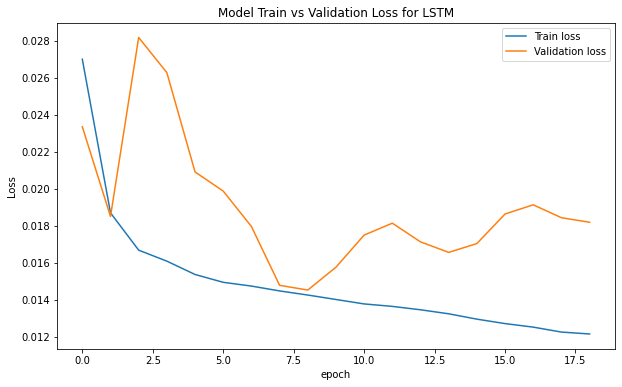
Evaluation

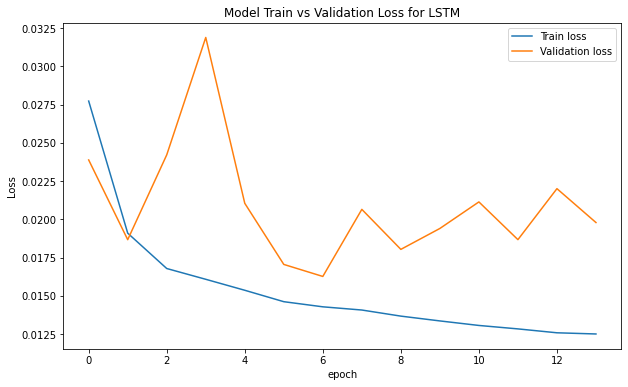
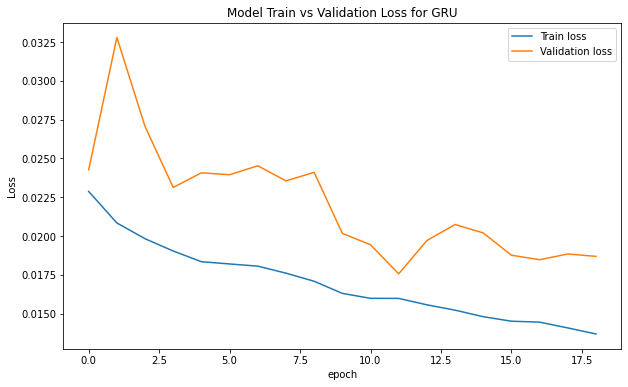
Run newdata 3



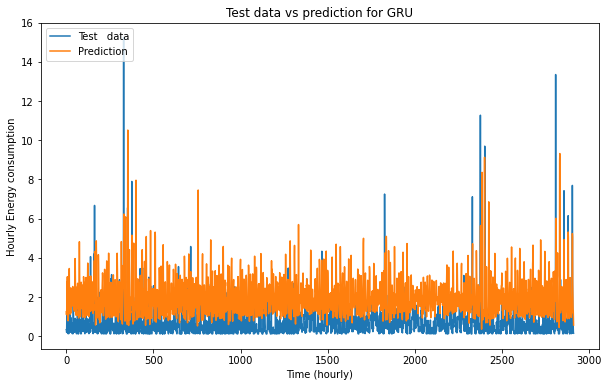


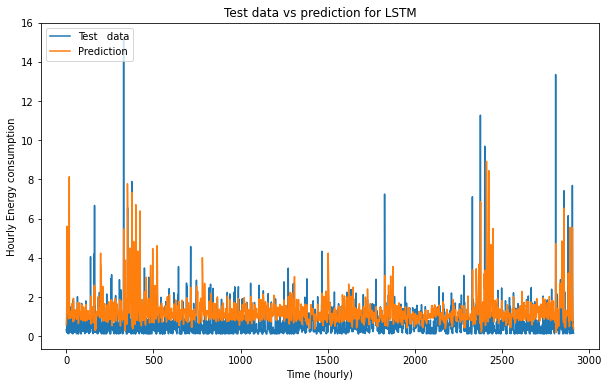


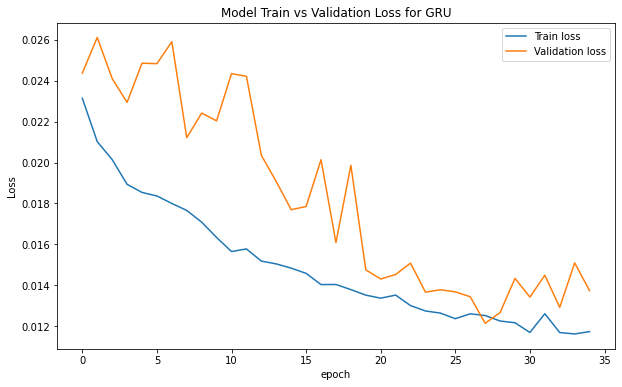


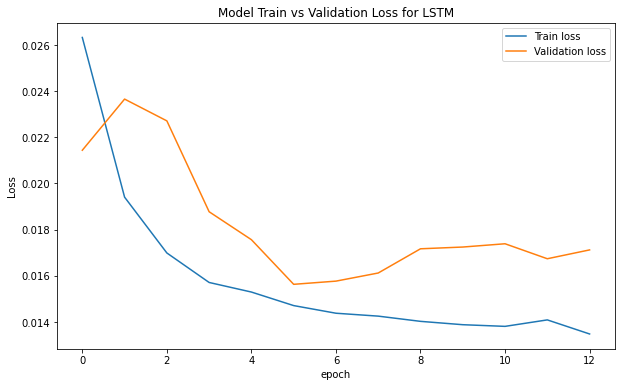
Run 4

Run5









## LSTM Model With Univariate Input and Vector Output

We will start off by developing a simple or vanilla LSTM model that reads in a sequence of days of total daily power consumption and predicts a vector output of the next standard week of daily power consumption.

This will provide the foundation for the more elaborate models developed in subsequent sections.

The number of prior days used as input defines the one-dimensional (1D) subsequence of data that the LSTM will read and learn to extract features. Some ideas on the size and nature of this input include:

* All prior days, up to years worth of data.
* The prior seven days.
* The prior two weeks.
* The prior one month.
* The prior one year.
* The prior week and the week to be predicted from one year ago.

There is no right answer; instead, each approach and more can be tested and the performance of the model can be used to choose the nature of the input that results in the best model performance.

These choices define a few things:

* How the training data must be prepared in order to fit the model.
* How the test data must be prepared in order to evaluate the model.
* How to use the model to make predictions with a final model in the future.

A good starting point would be to use the prior seven days.