WAYNE STATE UNIVERSITY

MS Data Science and Business Analytics

IE 7860: Intelligent Analytics

Assignment 5

Instructor:	Teaching Assistant:	Author:
Dr. Ratna Chinnam	Elham Taghizadeh	Rajpal Virk

RECURRENT NEURAL NETWORKS ASSIGNMENT

1. INTRODUCTION

Recurrent neural networks (RNN) are a class of artificial neural networks in which the connections between notes form a direct graph along a temporal sequence. In simple words, recurrent neural networks are used for sequential data modelling.

RNNs can use their internal state (memory) to process variable length sequences of inputs.

The applications of language models are two-fold: First, it allows us to score arbitrary sentences based on how likely they are to occur in the real world. This gives us a measure of grammatical and semantic correctness. Such models are typically used as part of Machine Translation systems. Secondly, a language model allows us to generate new text.

2. PROBLEM STATEMENTS

For this assignment, We are expected to build and compare different types of recurrent neural networks for a challenging time-series or sequence problem.

2.1 DATA DRIVEN PREDICTION MODELS OF ENERGY USE OF APPLIANCES IN A LOW- ENERGY HOUSE

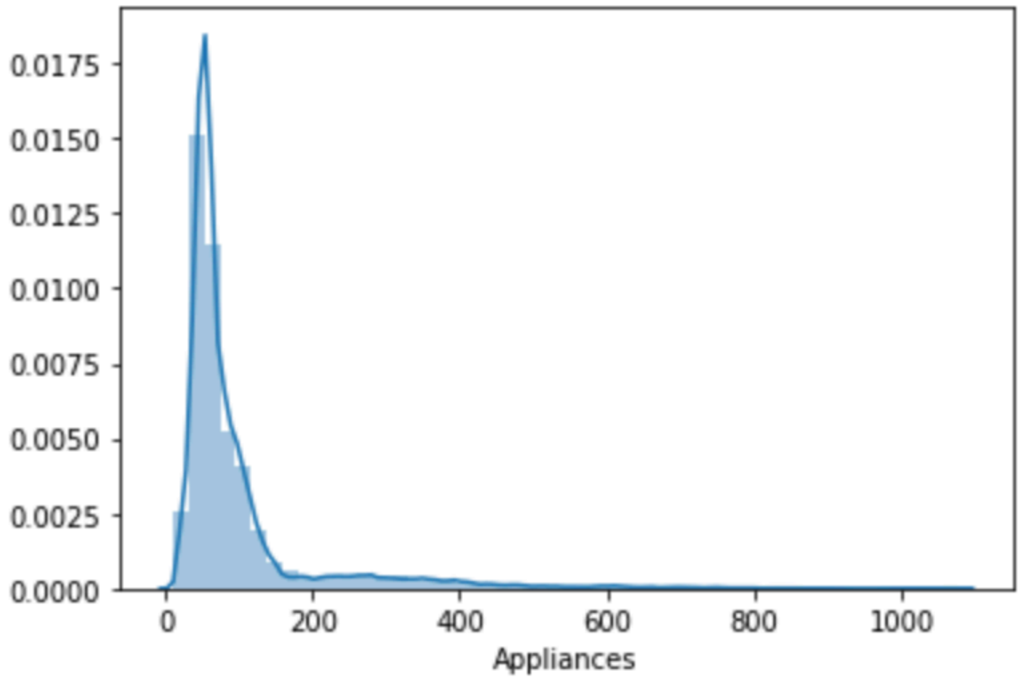
2.1.1 PROBLEM STATEMENT

The data set is at 10 min resolution for about 4.5 months. The house temperature and humidity conditions were monitored with a ZigBee wireless sensor network. Each wireless node transmitted the temperature and humidity conditions around 3.3 min. Then, the wireless data was averaged for 10 minutes periods. The energy data was logged every 10 minutes with m-bus energy meters. Weather from the nearest airport weather station (Chievres Airport, Belgium) was downloaded from a public data set from Reliable Prognosis (rp5.ru), and merged together with the experimental data sets using the date and time column. Two random variables have been included in the data set for testing the forecasting models and to filter out non-predictive attributes (parameters).

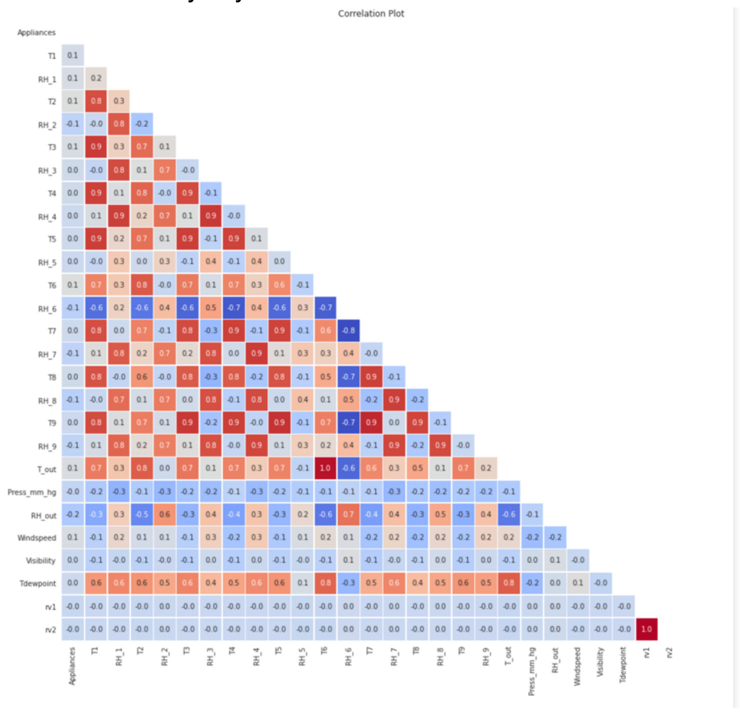
3 SOLUTION

Below is the stepwise explanation of process used to find the solutions for this time series problem.

1. Firstly, we imported into data into Jupyter notebook and reviewe the data.
2. Initially data set has 19,735 observations and 29 features.
3. We check the data and found there are no null values.
4. We set dates as index.
5. We checked the energy consumption on average by appliances and it is below 100 Wh for majority of appliances.

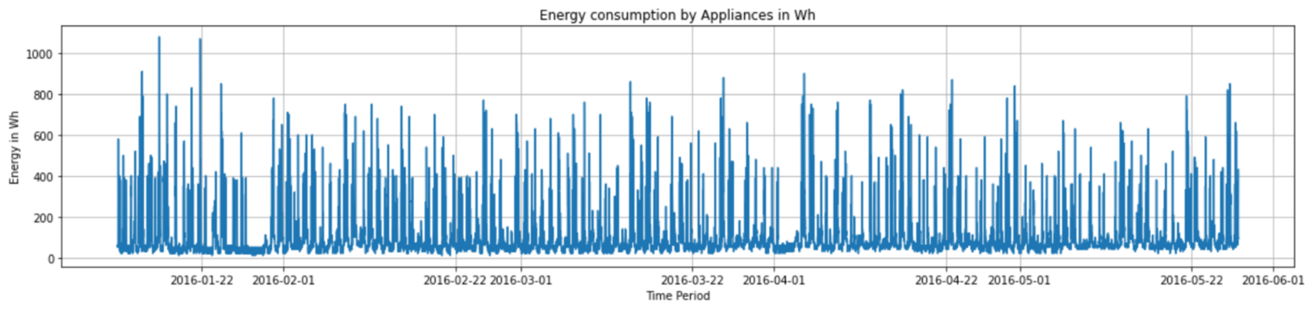


6. Then we looked at if there is a correlation between the available features and found that there are a few features which have a very close correlation with other features. However at this stage, we will try not to reduce the number of features because we only have 29 features and a good machine learning models should be able to work with these features anyway.



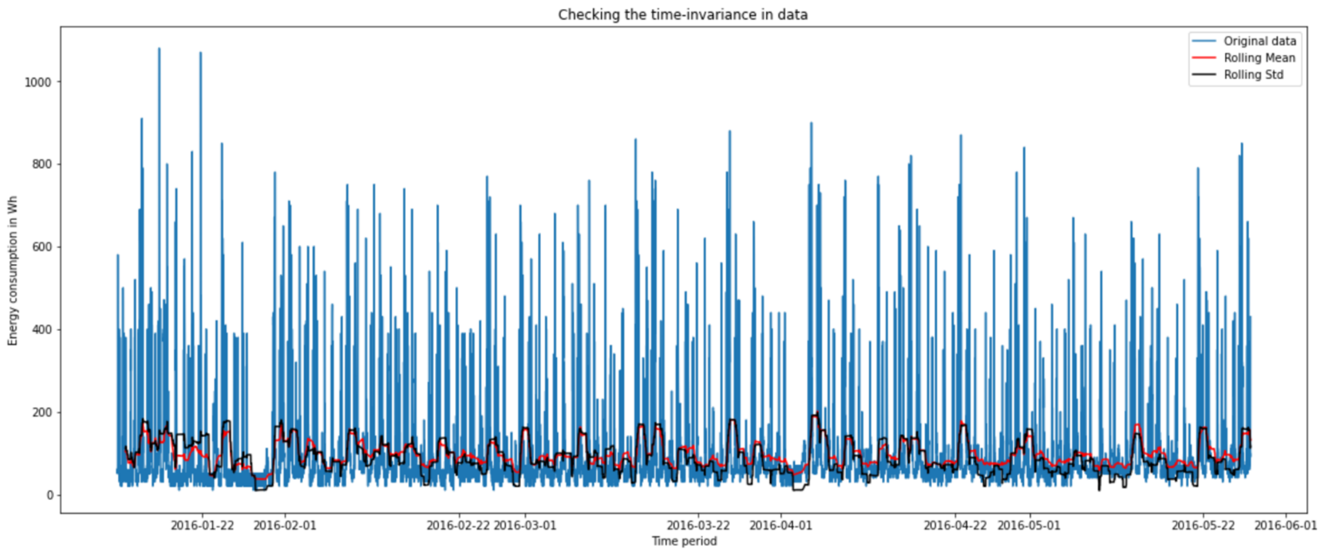
7. Then we plotted data in time series and were able to see the pattern in energy consumption by appliances over time.

```
Time Series Plot of Energy consumption by Appliances
```



CHECKING FOR TIME INVARIANCE

8. We then analyze the data to see if the date is stationary or in other words if the data is time invariant.



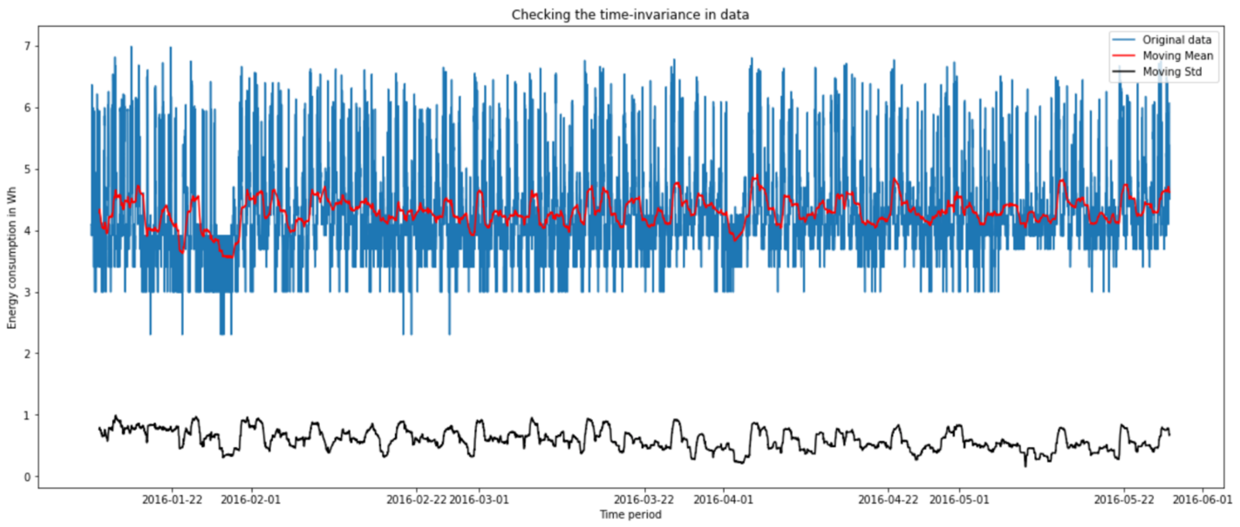
Since there is a clear trend of change in rolling mean and rolling standard deviation so we can conclude that this data is not stationary and has time-variance.

The rolling mean and rolling standard deviation is changing and that indicates us that our data is not stationary hence it is time invariant.

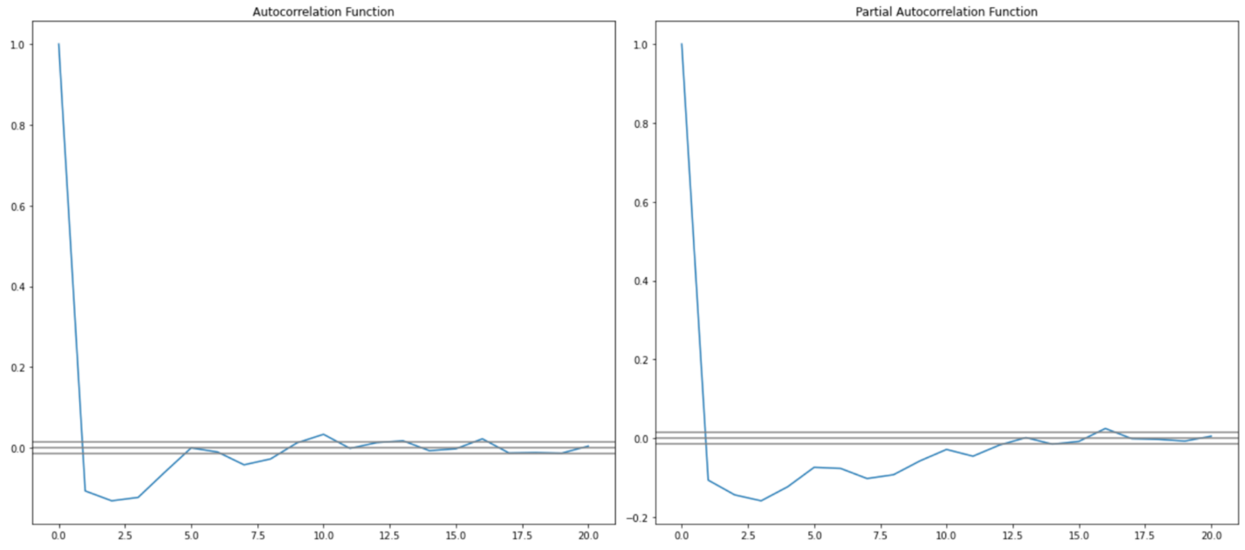
Focused Time Lagged FeedForward Networks (TLFN)

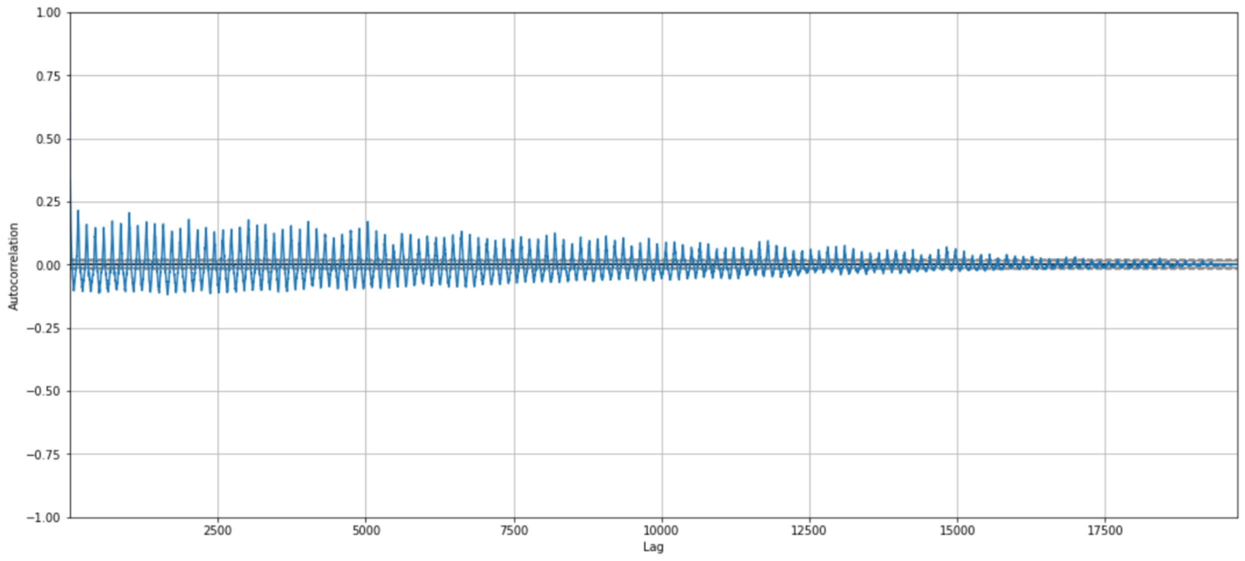
9. We can try to address the issue of time-invariance by converting the data to a stationary data.
10. After converting this we checked results both visually and by using Dickey Fuller analysis.

```
Dickey-Fuller Results:
Test Statistics              -21.616378
p-value                        0.000000
#Lags used                    11.000000
Number of observations     19723.000000
Critical Value (1%)           -3.430682
Critical Value (5%)           -2.861687
Critical Value (10%)          -2.566848
dtype: float64
```



11. Then we ploted autocorrelation and partial autocorrelation plots.





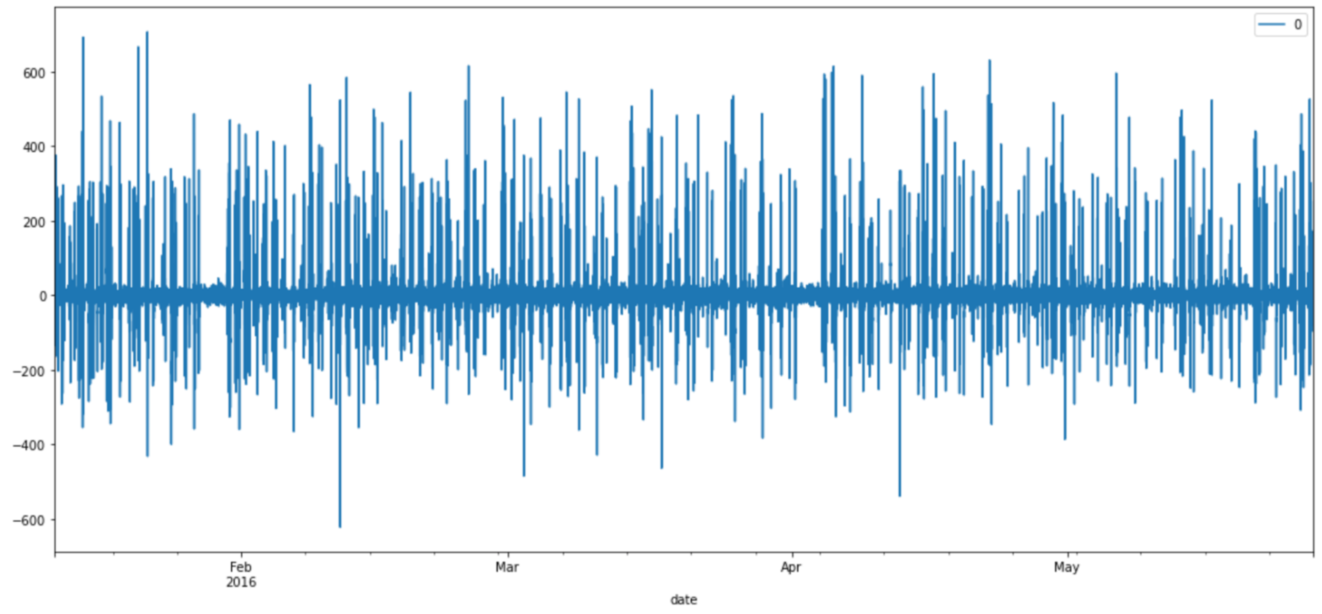
12. For this focused time lagged neural network, we decided to use ARIMA model to make predictions.

ARIMA MODEL

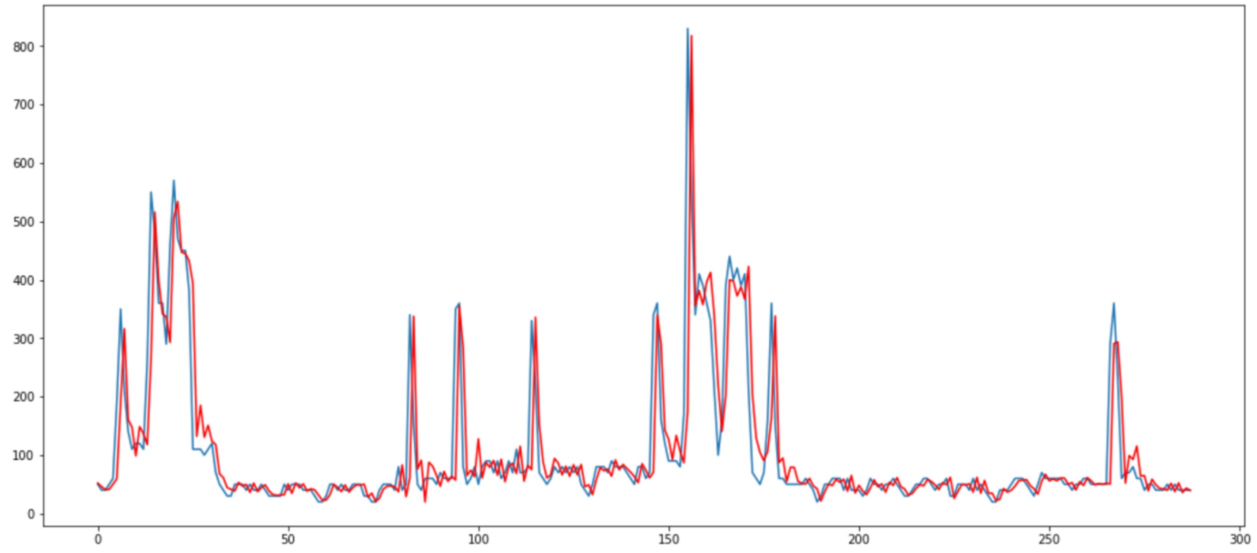
We build a ARIMA model by using p,q,d values: 5, 1 and 0 respectively.

13. We then established the results of our model.

```
                              ARIMA Model Results
==============================================================================
Dep. Variable:           D.Appliances   No. Observations:                19734
Model:                 ARIMA(5, 1, 0)   Log Likelihood             -111306.882
Method:                       css-mle   S.D. of innovations             68.130
Date:                Sat, 11 Apr 2020   AIC                         222627.764
Time:                        21:53:34   BIC                         222682.995
Sample:                    01-11-2016   HQIC                        222645.846
                         - 05-27-2016
======================================================================================
                         coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------------
const                  0.0140      0.264      0.053      0.958      -0.503       0.531
ar.L1.D.Appliances    -0.1342      0.007    -18.931      0.000      -0.148      -0.120
ar.L2.D.Appliances    -0.3070      0.007    -43.258      0.000      -0.321      -0.293
ar.L3.D.Appliances    -0.1804      0.007    -24.664      0.000      -0.195      -0.166
ar.L4.D.Appliances    -0.1304      0.007    -18.378      0.000      -0.144      -0.117
ar.L5.D.Appliances    -0.0875      0.007    -12.337      0.000      -0.101      -0.074
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            0.8233           -1.1770j            1.4364           -0.1529
AR.2            0.8233           +1.1770j            1.4364            0.1529
AR.3           -0.5955           -1.5785j            1.6871           -0.3074
AR.4           -0.5955           +1.5785j            1.6871            0.3074
AR.5           -1.9460           -0.0000j            1.9460           -0.5000
-----------------------------------------------------------------------------
```



14. Once the model was ready we implemented the model on first 10 days data to find predictions. The results of predictions are plotted below:



15. Though our model was able to show a very good prediction patterns, performed poorly on mean square error metrics.

Since we didn't find very good results, we decided to try a Distributed_Time Lagged Feed Forward Networks (TLFN).

16. **Distributed_Time Lagged Feed Forward Networks (TLFN)**
    We used PyTorch to make a LSTM model and find predictions for the energy consumption by appliances in future.
17. First, we build the model and prepare of a data set.

```
# Define function to create seq/label tuples
def input_data(seq,ws):  # ws is the window size
    out = []
    L = len(seq)
    for i in range(L-ws):
        window = seq[i:i+ws]
        label = seq[i+ws:i+ws+1]
        out.append((window,label))
    return out
```

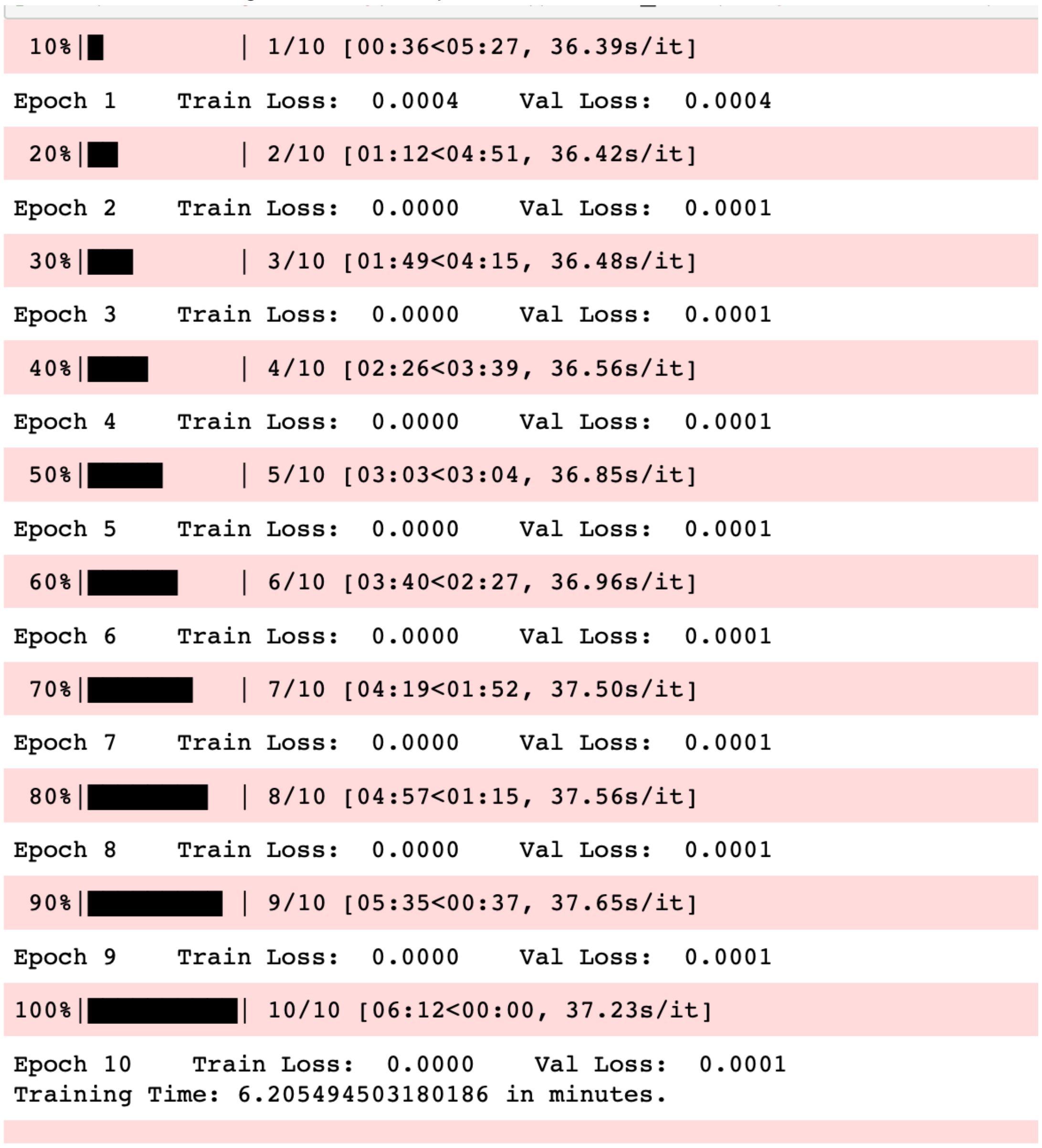
```
# Define the model
class LSTMnetwork(nn.Module):
    def __init__(self,input_size=1,hidden_size=2,output_size=1):
        super().__init__()
        self.hidden_size = hidden_size

        # Add an LSTM layer:
        self.lstm = nn.LSTM(input_size,hidden_size)

        # Add a fully-connected layer:
        self.linear = nn.Linear(hidden_size,output_size)

        # Initialize h0 and c0:
        self.hidden = (torch.zeros(1,1,self.hidden_size),
                       torch.zeros(1,1,self.hidden_size))

    def forward(self,seq):
        lstm_out, self.hidden = self.lstm(
            seq.view(len(seq),1,-1), self.hidden)
        pred = self.linear(lstm_out.view(len(seq),-1))
        return pred[-1]  # we only want the last value
```

18. After preparing the model and data set, we trained our data set on 10 days of data.
    We used all the entries from first 10 days of energy consumption by appliances and fed through our LSTM model.

19. LSTM Model Structure
    Our model has one input and one output size but we have used 10 hidden layers to build this model.

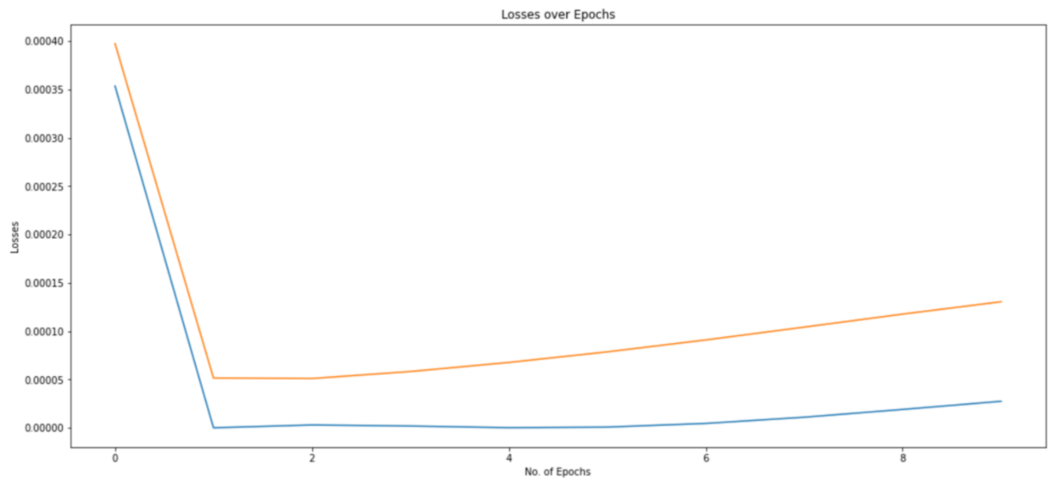
```
LSTMnetwork(
  (lstm): LSTM(1, 2)
  (linear): Linear(in_features=2, out_features=1, bias=True)
)
```

20. The model trained using 10 number of epochs and results were recorded.

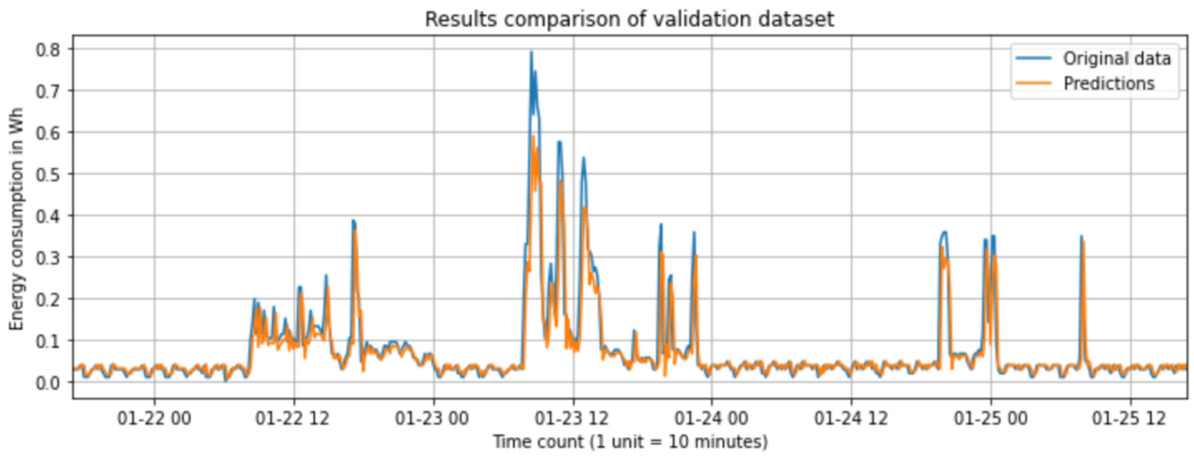


```
 10%|█         | 1/10 [00:36<05:27, 36.39s/it]
Epoch 1    Train Loss:  0.0004    Val Loss:  0.0004
 20%|██        | 2/10 [01:12<04:51, 36.42s/it]
Epoch 2    Train Loss:  0.0000    Val Loss:  0.0001
 30%|███       | 3/10 [01:49<04:15, 36.48s/it]
Epoch 3    Train Loss:  0.0000    Val Loss:  0.0001
 40%|████      | 4/10 [02:26<03:39, 36.56s/it]
Epoch 4    Train Loss:  0.0000    Val Loss:  0.0001
 50%|█████     | 5/10 [03:03<03:04, 36.85s/it]
Epoch 5    Train Loss:  0.0000    Val Loss:  0.0001
 60%|██████    | 6/10 [03:40<02:27, 36.96s/it]
Epoch 6    Train Loss:  0.0000    Val Loss:  0.0001
 70%|███████   | 7/10 [04:19<01:52, 37.50s/it]
Epoch 7    Train Loss:  0.0000    Val Loss:  0.0001
 80%|████████  | 8/10 [04:57<01:15, 37.56s/it]
Epoch 8    Train Loss:  0.0000    Val Loss:  0.0001
 90%|█████████ | 9/10 [05:35<00:37, 37.65s/it]
Epoch 9    Train Loss:  0.0000    Val Loss:  0.0001
100%|██████████| 10/10 [06:12<00:00, 37.23s/it]
Epoch 10    Train Loss:  0.0000    Val Loss:  0.0001
Training Time: 6.205494503180186 in minutes.
```

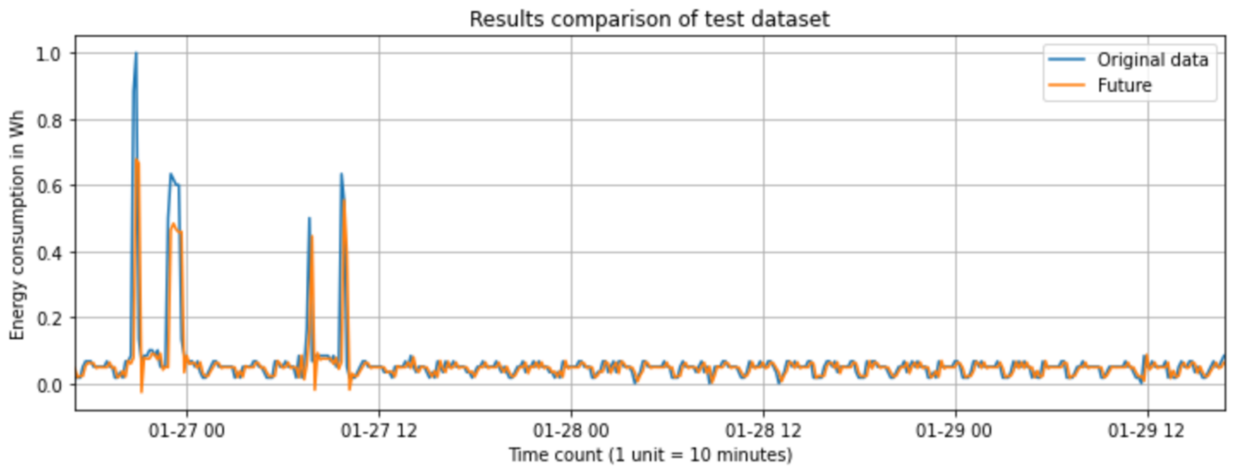
21. Further, we plotted the change in training and validation loss over time.



22. Since our data set is not very complicated and the model is quite good, model was able to learn the pattern after simply 1 epoch. The errors were brought to 0 by the model or near 0 for both training and validation data.
23. Next, we plotted the results on predictions made by model on validation data.



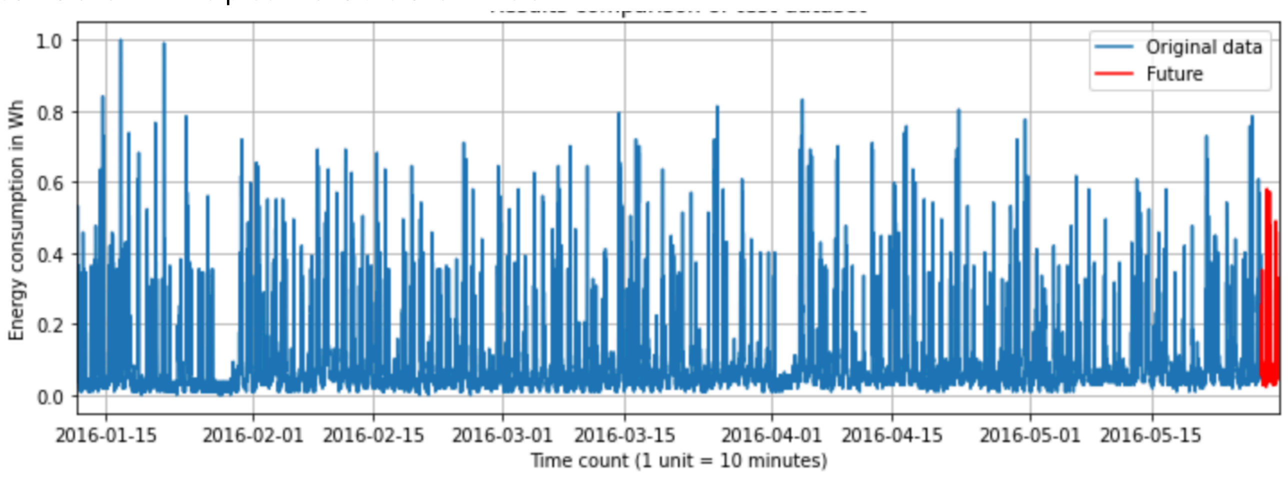
24. Further, we repeated the process for the test data set. For testing, we have used for days of data and made predictions. These predictions were done compared with the original values and the results are shown below.



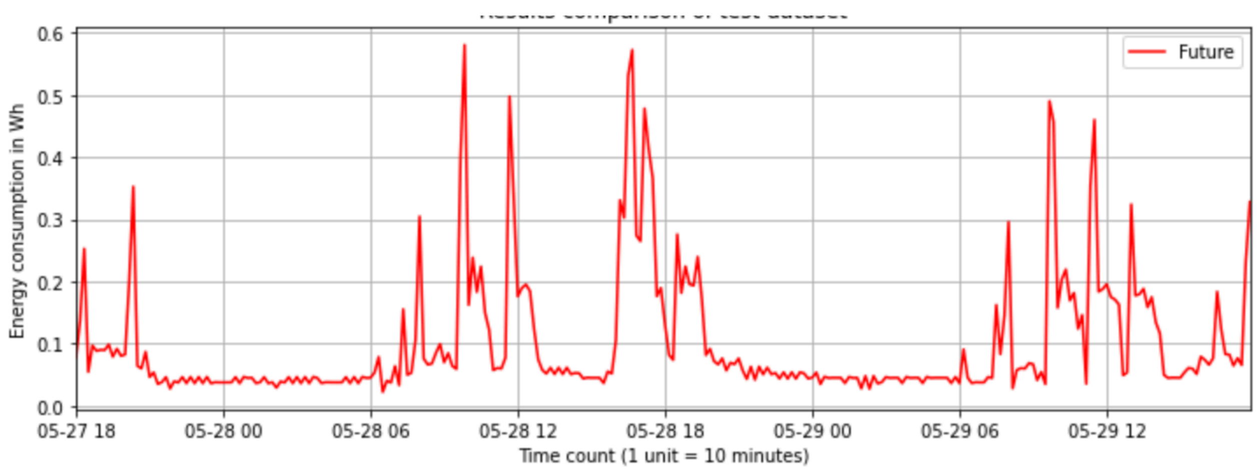
25. Since our model has not seen test data before, still it was able to do fine predictions.
26. Next we checked the metrics and find very good values of MAE = 0.02 and MSE = 0.005.

```
Mean Absolute Error on test data is  0.0242.
Mean Squared Error on test data is  0.0053.
```

27. Now, we have some confidence in our model, we tried to predict values in future. For this we predicted Energy consumption by appliances for next 3 days in future.
28. Results of our future predictions are shown below:



Above plot shows the results of future predictions plotted along with the original data.



Above plot shows the results of future predictions plotted in bigger scale to show the pattern of energy consumption.

Conclusion:

In this assignment, we tried with 2 models – ARIMA model and LSTM model.

The results of LSTM model are much better compared to ARIMA Model and low mean squared error value does bring some confidence in our model to be used for future predictions.

Key notes:

1. Models are built in python.
2. PyTorch is used extensively however due to system capability restrictions and no availability of WSU Grid from last couple of weeks, I was not able to experiment much with the data. The data was taking nearly 8 hours to run on my system using PyTorch. Therefore, I end up using a portion of data to train the model. If we could have used GRID we would have been able to reduce the errors further and improve our predictions.