1 Experiment

The current section introduces the dataset used for the prediction task and reports the outline and results of the experiments conducted. A specific focus is given to the evaluation of results, which are compared with each other. The proposed LSTM model was implemented and executed on Deep learning library, Keras (version 2.4.3) in Programming language, Python (version 3.8.3) with python libraries such as Pandas(version 1.0.5), Numpy(version 1.18.5), for data visualization Matplotlib(version 3.2.2) and Machine learning libraries such as Scipy(version 1.5.0), Statsmodels(version 0.11.1), and Sklearn(version 0.23.1) in Anaconda software(version 4.9.1). Experiments were run on a computer with an Intel® core™ i3-6006U processor running at 2.00GHz 4.00 GB of RAM, running windows version 10.

For forecasting the household electricity consumption in France, we have collected real data of approximately 4 years i.e. from December 2006 to December 2010 from the open source platform (*1*). The accumulated data was normalized to use only the following attributes (to be used as variables): date time, Global active power, Global reactive power, voltage, Global Intensity, Sub metering 1, sub metering 2, sub metering 3. Once normalized the collected data, it was used to create two different data sets. The first data set was created by dividing the filtered data into training data (data from December 2006 to December 2009) and test data (December 2010) shown in the figure 1 below.

On the other hand, the second data set was created by dividing the normalized data as per standard weeks and separating them into two groups: one for training purposes (Only data of 159 weeks, starting from December 17, 2006(first Sunday) till end of the week in 2009) and another for performance testing purposes (Only data of 46 weeks, Starting from Jan 3, 2010(first Sunday) till end of the week in 2010) shown in the figure 2 below.

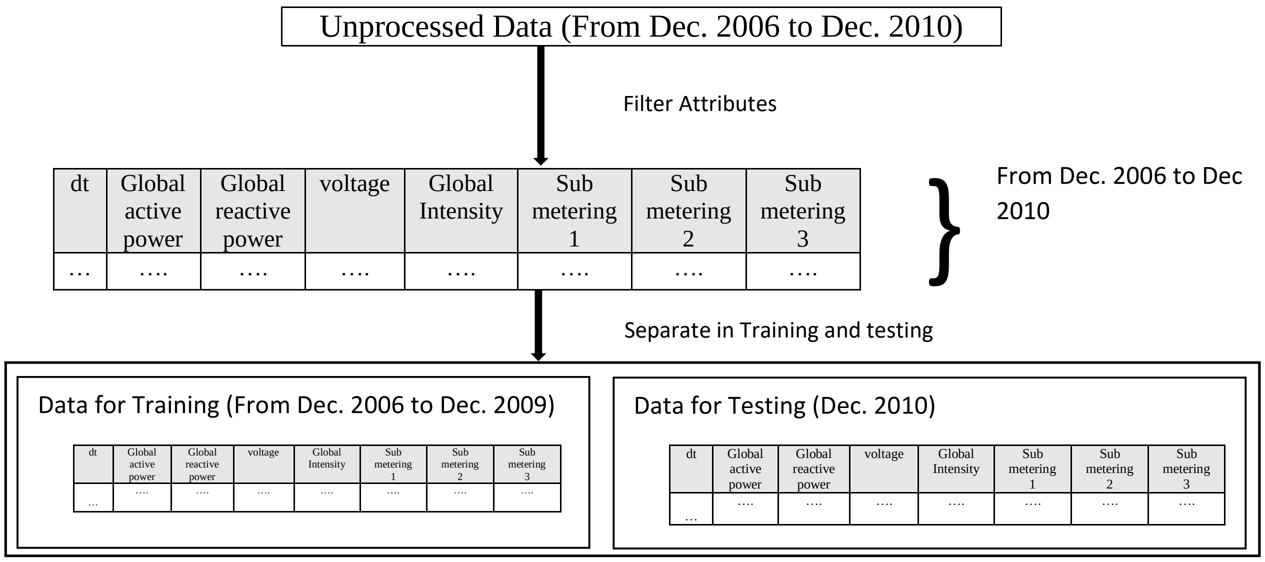
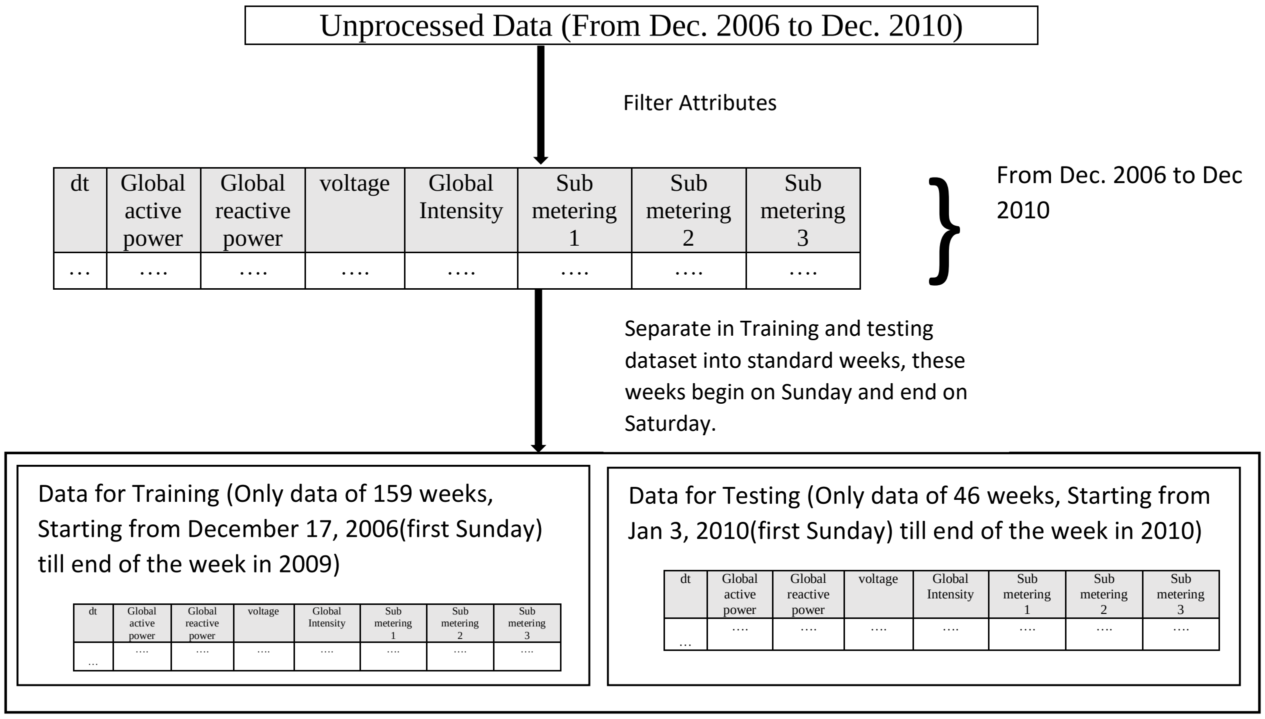

Unprocessed Data (From Dec. 2006 to Dec. 2010)
Filter Attributes
dt
Global active power
Global reactive power
voltage
Global Intensity
Sub metering 1
Sub metering 2
Sub metering 3
From Dec. 2006 to Dec 2010
Separate in Training and testing
Data for Training (From Dec. 2006 to Dec. 2009)
Data for Testing (Dec. 2010)

Figure 1 Schematic overview for An hour prediction



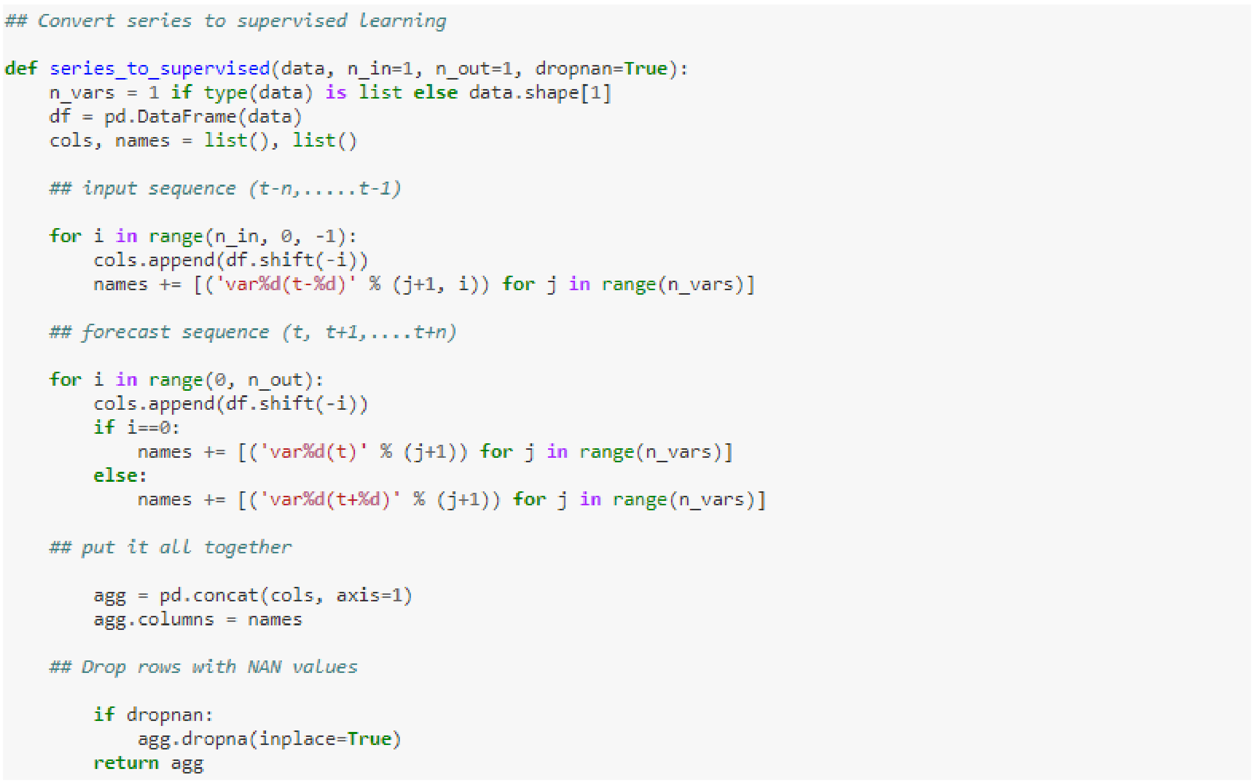
Unprocessed Data (From Dec. 2006 to Dec. 2010)
Filter Attributes
dt
Global active power
Global reactive power
voltage
Global Intensity
Sub metering 1
Sub metering 2
Sub metering 3
From Dec. 2006 to Dec 2010
Separate in Training and testing dataset into standard weeks, these weeks begin on Sunday and end on Saturday.
Data for Training (Only data of 159 weeks, Starting from December 17, 2006(first Sunday) till end of the week in 2009)
Data for Testing (Only data of 46 weeks, Starting from Jan 3, 2010(first Sunday) till end of the week in 2010)

Figure 2 Schematic overview for a week prediction

1.1 An hour Prediction

Preparing the LSTM Model for an hour power consumption prediction, the following steps involved as shown below:

After the data processing and data cleaning, the dataset is ready to make the LSTM model. At first, the data set is framed as a supervised learning problem and normalizing the input variables. Framing the supervised learning problem as predicting the power consumption at the current hour (t). Predicting the power consumption for the next hour based on the power consumption over the last 24hrs. (*2*)



```python
## Convert series to supervised learning

def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
    n_vars = 1 if type(data) is list else data.shape[1]
    df = pd.DataFrame(data)
    cols, names = list(), list()

    ## input sequence (t-n,.....t-1)

    for i in range(n_in, 0, -1):
        cols.append(df.shift(-i))
        names += [('var%d(t-%d)' % (j+1, i)) for j in range(n_vars)]

    ## forecast sequence (t, t+1,....t+n)

    for i in range(0, n_out):
        cols.append(df.shift(-i))
        if i==0:
            names += [('var%d(t)' % (j+1)) for j in range(n_vars)]
        else:
            names += [('var%d(t+%d)' % (j+1)) for j in range(n_vars)]

    ## put it all together

        agg = pd.concat(cols, axis=1)
        agg.columns = names

    ## Drop rows with NAN values

        if dropnan:
            agg.dropna(inplace=True)
        return agg
```

The Next step is to define and fit the LSTM Model. First, Splitting the data sets into the train and test data sets $(train, test)$ as shown in the fig. To speed up the training model, fit the model on the first year of data, then evaluate on the remaining 3 years of data. Then split the $(train, test)$ into input and output as $(train_x, train_y)$ and $(test_x, test_y)$. Now the $(train_x, train_y)$ and $(test_x, test_y)$ are in 2D format. The LSTM expects the data has to be in 3D format. So, convert $(train_x, train_y)$ and $(test_x, test_y)$ into 3D format by reshaping function.

```
## values returns a list of all the values in the data frame
values = df_resample.values
```

```
## Normalize features
scaler = MinMaxScaler(feature_range=(0, 1))
```

```
## calculating scaler using fit_transform fn for values and assigning it as scaled
scaled = scaler.fit_transform(values)
```

```
## Called series_to_supervised fn to make it as supervised learning
reframed = series_to_supervised(scaled, 1, 1)
```

```
## Droping the other columns because we are Predicting only for Global active power
reframed.drop(reframed.columns[[8,9,10,11,12,13]], axis=1, inplace=True)
```

```
## Reframed head
## This prints the first five rows of transformed dataset. We can see the 8 input variables( input series )
## The 1 output variable at the current hour that is var1(t)

reframed.head()
```

	var1(t-1)	var2(t-1)	var3(t-1)	var4(t-1)	var5(t-1)	var6(t-1)	var7(t-1)	var1(t)
0	0.545045	0.103358	0.335501	0.541487	0.0	0.144652	0.782676	0.636816
1	0.509006	0.110073	0.283802	0.502152	0.0	0.030869	0.774169	0.545045
2	0.488550	0.096987	0.315987	0.481110	0.0	0.000000	0.778809	0.509006
3	0.455597	0.099010	0.434417	0.449904	0.0	0.008973	0.798917	0.488550
4	0.322555	0.072536	0.495847	0.323529	0.0	0.002872	0.205723	0.455597

The next step is to fit LSTM model, So, define the LSTM with 100 neurons in the first hidden layer and 1 neuron in the output layer for predicting the power consumption. The input shape will be 1-time step with 8 features. Using Mean Absolute error(MAE) loss function and the optimizer adam as stochastic gradient descent. The model is fit for 20 training epchos with a batch size of 70. To track the training and test loss during training, added the validation data argument in the $fit()$ function. Which is already explained in section 2.

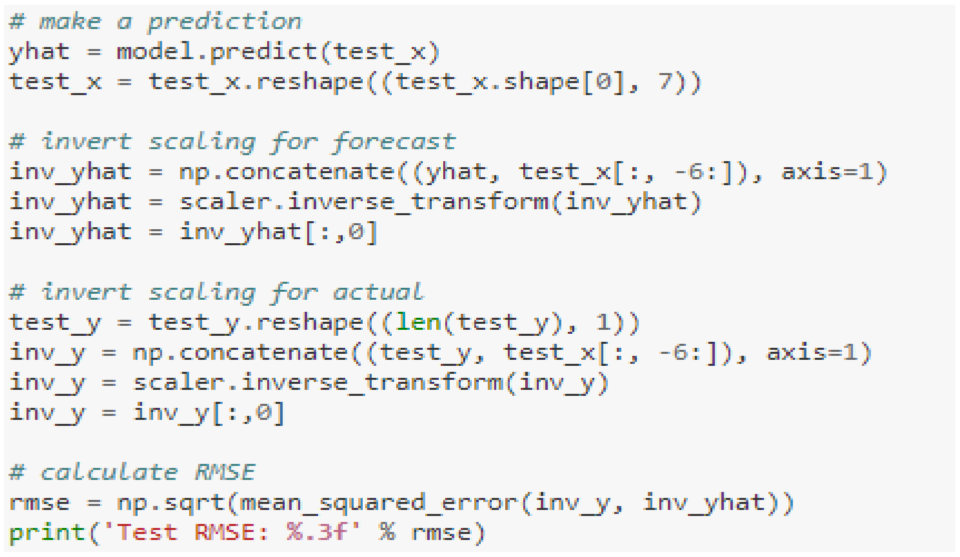
```
## design network
model = Sequential()
model.add(LSTM(100, input_shape=(train_x.shape[1], train_x.shape[2])))
model.add(Dropout(0.2))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')

# fit network
history = model.fit(train_x, train_y, epochs=20, batch_size=70, validation_data=(test_x, test_y), verbose=2, shuffle=False)

# plot history
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right')
plt.show()
```

The next step is to evaluate the error, which is defined below. (*3*)

Mean Squared Error - Average squared difference between the actual and predicted value.

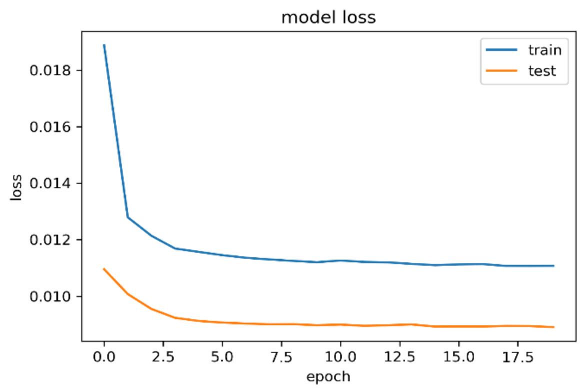
$$\sum_{n=1}^{k} = \frac{(Actual - Predicted)^2}{k}$$

Root Mean Squared Error - $\sum_{n=1}^{k} = \sqrt{\frac{(Actual-Predicted)^2}{k}}$



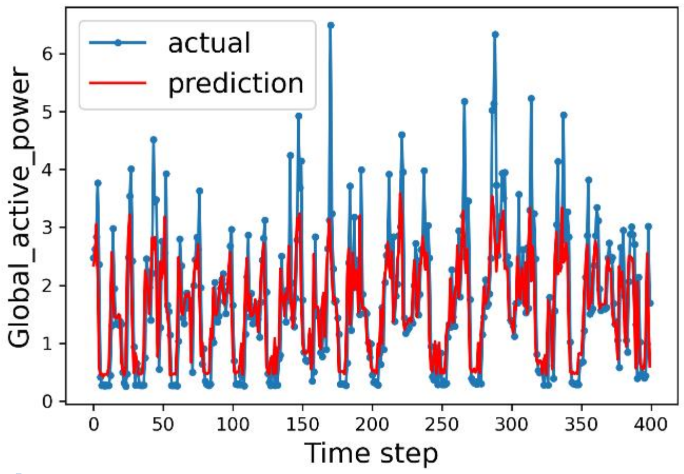
```
# make a prediction
yhat = model.predict(test_x)
test_x = test_x.reshape((test_x.shape[0], 7))

# invert scaling for forecast
inv_yhat = np.concatenate((yhat, test_x[:, -6:]), axis=1)
inv_yhat = scaler.inverse_transform(inv_yhat)
inv_yhat = inv_yhat[:,0]

# invert scaling for actual
test_y = test_y.reshape((len(test_y), 1))
inv_y = np.concatenate((test_y, test_x[:, -6:]), axis=1)
inv_y = scaler.inverse_transform(inv_y)
inv_y = inv_y[:,0]

# calculate RMSE
rmse = np.sqrt(mean_squared_error(inv_y, inv_yhat))
print('Test RMSE: %.3f' % rmse)
```

RMSE found to be, $RMSE = 0.608$ in our experiment, which is lesser than the mean of standard deviation $Mean\ Std = 0.710$. Hence, the prediction model is a good model. The RMSE is the square root of the variance of the residuals. It indicates the absolute fit of the model to the data–how close the observed data points are to the model's predicted values. Whereas R-squared is a relative measure of fit, RMSE is an absolute measure of fit. As the square root of a variance, RMSE can be interpreted as the standard deviation of the unexplained variance, and has the useful property of being in the same units as the response variable. Lower values of RMSE indicate better fit. RMSE is a good measure of how accurately the model predicts the response, and it is the most important criterion for fit if the main purpose of the model is prediction. (*4*)



MSE	0.369
RMSE	0.608
Mean Std	0.710

The next step is to plot the actual and the predicted power consumption. From the Graph the predicted curve is following the actual one. The LSTM model predicted the next 400 time stamps (1-time stamp means one hour) which means the LSTM model learned the data and predicted well.



1.2 A week prediction

Preparing the LSTM Model for a week prediction power consumption, the steps following involved as shown below:

After the data processing and data cleaning, the dataset is ready to make the LSTM model. At first, the data set is framed as a supervised learning problem and normalizing the input variables. Framing the supervised learning problem as predicting the power consumption at the current hour (t). Predicting the power consumption for the next week based on the power consumption over the last week (7 days), last two weeks (14 days), last three weeks(21 days), and the last four weeks (28 days).

The Next step is to define and fit the LSTM Model. First, Splitting the data sets into the standard weeks as train and test data sets $(train, test)$ as shown in the fig. for training set (Only data of 159 weeks, starting from December 17, 2006(first Sunday) till end of the week in 2009) and another for performance testing purposes (Only data of 46 weeks, Starting from Jan 3, 2010(first Sunday) till end of the week in 2010). Now the data set of $(train, test)$ match the specific dates that defined as the boundary on the standard weeks for each data set. This is where a model is required to form a 1-week prediction, then the particular data for that week is formed available to the model so it is often used because the basis for creating a prediction on the next week. This can be both realistic for a way the model could also be utilized in practice and beneficial to the models allowing them to form use of the simplest available data. The idea shown below.

Input,	Predict
[week 1]	week 2
[week 1+ week 2]	week 3
[week 1+ week 2 + week 3]	week 4

[week n-1 + week n-2 + week n-3]	week n (present week)
[--------- + ------------- + week n]	week n+1 (next week)

Then split the $(train, test)$ into input and output as $(train_x, train_y)$ and $(test_x, test_y)$. Now the $(train_x, train_y)$ and $(test_x, test_y)$ are in 2D format. The LSTM expects the data has to be in 3D format. So, convert $(train_x, train_y)$ and $(test_x, test_y)$ into 3D format by reshaping function.

The next step is to fit LSTM model, So, define the LSTM with 200 neurons in the first hidden layer and 1 neuron in the output layer for predicting the power consumption for each day in a week. The input shape is unrelated when giving an input. The LSTM connected with 200 neurons will interpret the features learned by LSTM layer. Using Mean Absolute error(MAE) loss function and the optimizer adam as stochastic gradient descent. The model is fit for 70 training epchos with a batch size of 16.

After the above step, we have the data of the prior week to predict the next week. So, the prior week data is collected in an array of standard weeks. The data obtained will be in the form of standard weeks. So, to predict next week, we need to retrieve the data of last days information for this, we need to use the flatten data function to break weekly readings into individual days reading which gives us eight parallel time series data.

The next is to obtain the last seven day's power consumption (feature index 0). Then we need to parameterize this as we did for the training data so the number of prior days used as input and then make a prediction using the fit model and therefore the input file and retrieve the vector of seven days of output. The function implements this and to take the model to fit in the training dataset. The history is already stored in the model, and it expects the input time steps. Then the model fits and evaluate the model. It prints the overall RMSE across the number of days given in the input and the per-day RMSE for each lead time. (5)

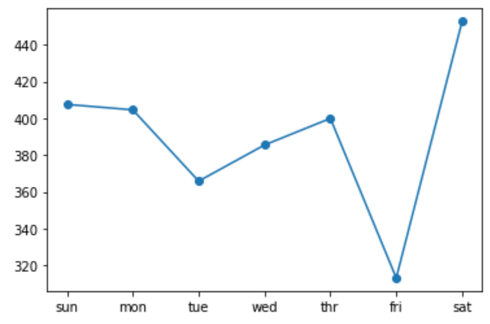


Figure 4 Forecast for a week by taking past 7 days values

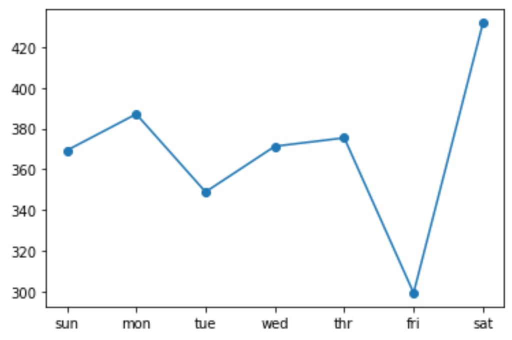


Figure 3 Forecast for a week by taking past 14 days values

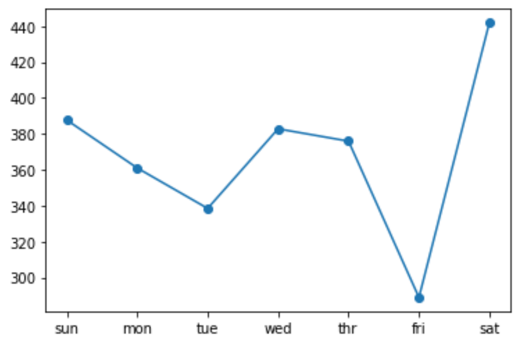


Figure 6 Forecast for a week by taking past 21 days values

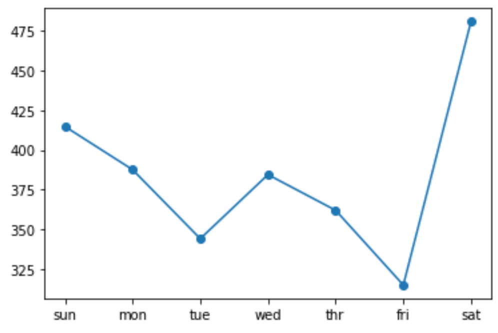


Figure 5 Forecast for a week by taking past 28 days values

LSTM Model Prediction	Predicted (Kilowatt) 7 days	Predicted (Kilowatt) 14 days	Predicted (Kilowatt) 21 days	Predicted (Kilowatt) 28 days
Overall power consumption for next week	391.92	370.91	370.62	387.33
Sunday	407.5	369.3	387.7	414.8
Monday	404.6	387.2	360.9	387.6
Tuesday	365.9	348.3	338.4	344.1
Wednesday	385.5	371.3	382.7	384.5
Thursday	399.9	375.5	375.9	362.0
Friday	313.3	299.2	288.8	314.9
Saturday	452.6	432.0	441.9	481.0

The plot shows that, Tuesdays and Fridays are easier to make forecast than the end day of the standard week, Saturdays (comparing to the overall consumption Saturdays shows more difference).

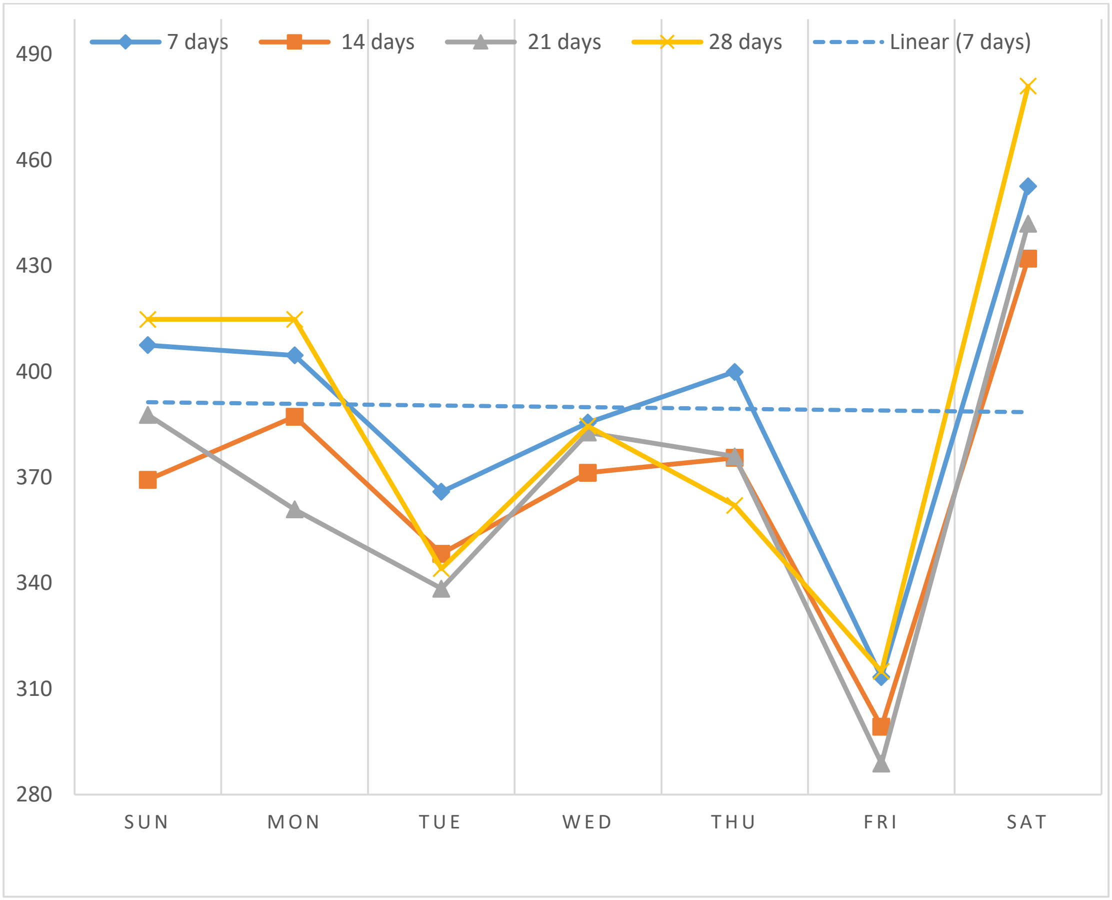


Figure 7 Visualization of overall days

References

1. *UCI Machine Learning Repository: Individual household electric power consumption Data Set*. https://archive.ics.uci.edu/ml/datasets/Individual+household+electric+power+consumption (accessed 2020-11-26).

2. GitHub. *Pushpak2227/Master-Thesis*. https://github.com/Pushpak2227/Master-Thesis/blob/master/predicting%20the%20power%20consumption%20for%20the%20next%20hour%20based%20on%20the%20power%20consumption%20over%20last%2024%20hrs.%20.ipynb (accessed 2020-11-26).

3. Qiu, J.; Wang, B.; Zhou, C. Forecasting stock prices with long-short term memory neural network based on attention mechanism. *PloS one* **2020,** *15* (1), e0227222. DOI: 10.1371/journal.pone.0227222.

4. Grace-martin, K. Assessing the Fit of Regression Models. *The Analysis Factor,* Dec 8, 2008. https://www.theanalysisfactor.com/assessing-the-fit-of-regression-models/ (accessed 2020-11-22).

5. GitHub. *Pushpak2227/Master-Thesis*. https://github.com/Pushpak2227/Master-Thesis/blob/master/Prediction%20for%20a%20week%20by%20taking%20past%207%2C14%2C21%2C28%20days%20values.ipynb (accessed 2020-11-26).