REPORT

TOPIC

Building Energy Load Forecasting using Time-Series techniques and Deep Neural network.

SUBJECT

Regression and Time Series

MENTOR

Dr. Buddhananda Banerjee

GROUP MEMBERS

17BM6JP28 Ankit Jaiswal

17BM6JP35 Puneeth Gandla

17BM6JP36 Teja Ram Kotaprolu

17BM6JP44 Ranadheer V

1. Data Description:

The data set contains power consumption measurements gathered between December 2006 and November 2010 with one-minute resolution which contained aggregate active power load for the whole house and three sub metering for three sections for the house. In this report, only the aggregate active load values for the whole house are used. The dataset contains 2,075,259 measurements. Two types of the data set were considered for modelling: 1). Hourly resolution; and 2). Daily resolution. The hourly resolution data is obtained by averaging the one-minute resolution data and similarly daily data is obtained by aggregate hourly data. ARIMA modelling is done on the daily aggregated data due to lot of noise in the data. And both LSTM architectures were tested on both one minute and one-hour resolution data. The first three years were used to train the model and the last year was used as testing data.

Attribute Information:

1. **date**: Date in format dd/mm/yyyy
2. **time**: time in format hh:mm:ss
3. **global_active_power**: household global minute-averaged active power (in kilowatt)
4. **global_reactive_power**: household global minute-averaged reactive power (in kilowatt)
5. **voltage**: minute-averaged voltage (in volt)
6. **global_intensity**: household global minute-averaged current intensity (in ampere)
7. **sub_metering_1**: energy sub-metering No. 1 (in watt-hour of active energy). It corresponds to the kitchen, containing mainly a dishwasher, an oven and a microwave (hot plates are not electric but gas powered).
8. **sub_metering_2**: energy sub-metering No. 2 (in watt-hour of active energy). It corresponds to the laundry room, containing a washing-machine, a tumble-drier, a refrigerator and a light.
9. **sub_metering_3**: energy sub-metering No. 3 (in watt-hour of active energy). It corresponds to an electric water-heater and an air-conditioner.

2. Source of the Data:

Dataset used for the problem solving has been taken from UCI Machine Learning Repository. Link for the same is given below : https://archive.ics.uci.edu/ml/machine-learning-databases/00235/

3. Problem statement:

Ensuring sustainability demands more efficient energy management with minimized energy wastage. Therefore, the power grid of the future should provide an unprecedented level of flexibility in energy management. To that end, intelligent decision making requires accurate predictions of future energy demand/load, both at aggregate and individual site level. Thus, energy load forecasting have received increased attention in the recent past, however has proven to be a difficult problem.

This report presents a novel energy load forecasting methodology based on some of the time series techniques in addition to ARIMA modelling, Recurrent Neural Networks, specifically Long Short Term Memory (LSTM) algorithms.

4. Introduction:

Energy wastage poses a threat to sustainability; making buildings energy efficient is extremely crucial. Therefore, in making building energy consumption more efficient, it is necessary to have accurate predictions of its future energy consumption.

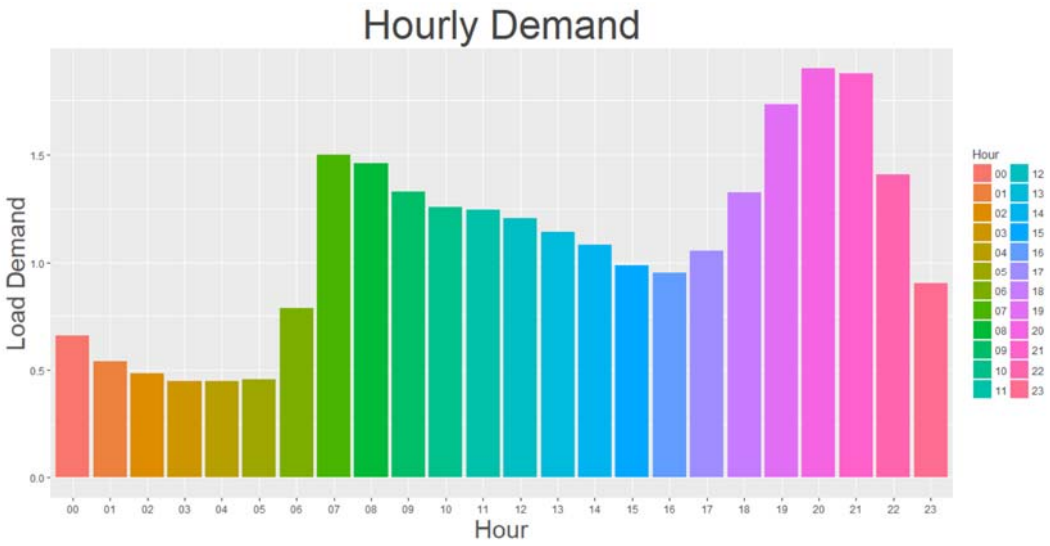
The future of the power grid is moving to a new paradigm of smart grids. Smart grids are promising, providing unprecedented flexibility in energy generation and distribution. In order to provide that flexibility, the power grid has to be able to dynamically adapt to the changes in demand and efficiently distribute the generated energy from the various sources such as renewables. In order to provide that flexibility, the power grid has to be able to dynamically adapt to the changes in demand and efficiently distribute the generated energy from the various sources such as renewables. Therefore, intelligent control decisions should be made continuously at aggregate level as well as modular level in the grid. In achieving that goal and ensuring the reliability of the grid, the ability of forecasting the future demands is important.

In terms of demand response, building level forecasting helps carry out demand response locally since the smart grids incorporate distributed energy generation. The advent of smart meters have made the acquisition of energy consumption data at building and individual site level feasible. Thus data driven and statistical forecasting models are made possible.

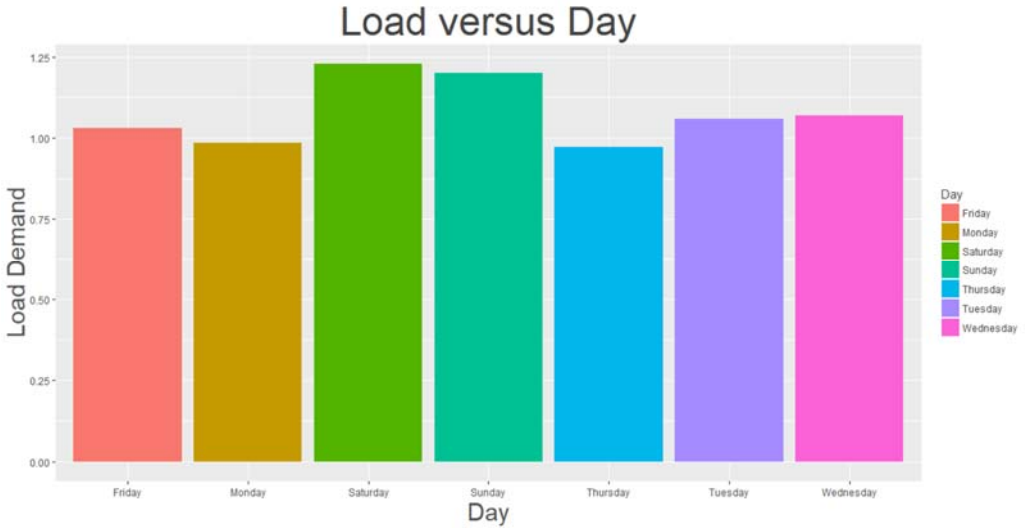
Aggregate level and building level load forecasting can be viewed in three different categories: 1) Short-term 2) Medium term and 3) Long-term. It has been determined that the load forecasting is a hard problem and in that, individual building level load forecasting is even harder than aggregate load forecasting. Thus, it has received increased attention from researchers. In literature, two main methods can be found for performing energy load forecasting: 1) Physics principles based models and 2) Statistical and machine learning based models. Focus of this report is on the second category of statistical load forecasting. This report discusses some of the time series techniques for forecasting along with Recurrent Neural Network (ANN) ensembles to perform the building level load forecasting. Use a support vector machines based regression model coupled with empirical mode decomposition for long-term load forecasting. Despite the extensive research carried out in the area, individual site level load forecasting remains to be a difficult problem.

5. Exploratory Data Analysis:

Hourly Demand: Mean Load Demand vs. hour of the day is shown in the plot below. Presence of two peaks, one at 7:00 AM and the other at 8:00 PM can be inferred from the plot below. The reason for these peaks is self-explanatory that as they are peak timings for electrical usage.



Weekly Demand: Similarly, mean Load demand was plotted for each day of the week. It can be observed that Saturdays and Sundays have peak which are usually off days for office and school and people tend to spend more time at home increasing electrical usage.



Monthly Demand: We can be observed that there is clear ‘U’ shaped trend in the data with minimum occurring during month of September. It can be inferred that there will high usage of electricity during winters and drops as summer approaches and further dips till September.

6. Methodology:

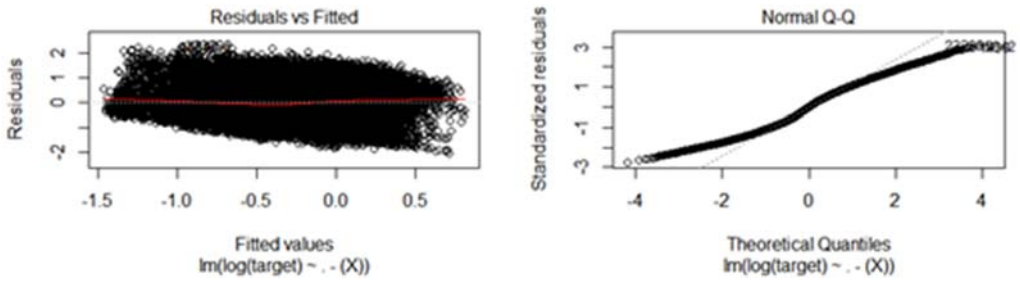
- Linear Regression + Boosting
- Seasonalities + ARIMA modelling
- Recurrent Neural Network using:
    - Standard LSTM architecture
    - Sequence to sequence LSTM architecture

A). Linear Regression + Boosting:

1Using insights from the exploratory data analysis, a linear regression model was fitted to predict log of target variable (Load Demand) using features like day of the month, day of the week, month of the year etc. Results of this model are shown below.

```
Residual standard error: 0.7455 on 34122 degrees of freedom
Multiple R-squared:  0.2215,    Adjusted R-squared:  0.2205
F-statistic: 215.8 on 45 and 34122 DF,  p-value: < 2.2e-16
```

From the above results it can be seen that the Adjusted R-squared for this model is very low.

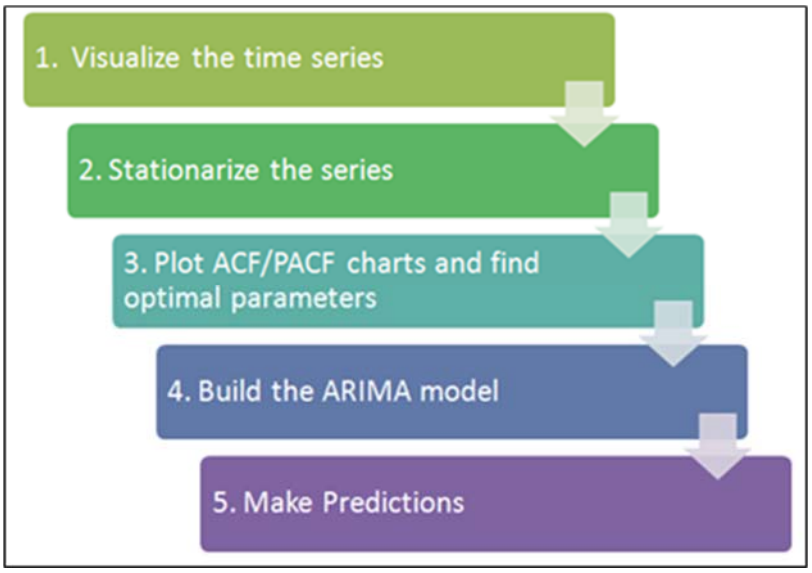


To improve the accuracy of the regression model, boosting is performed (using XG boosting).The following parameters were used for XGBoost algorithm.

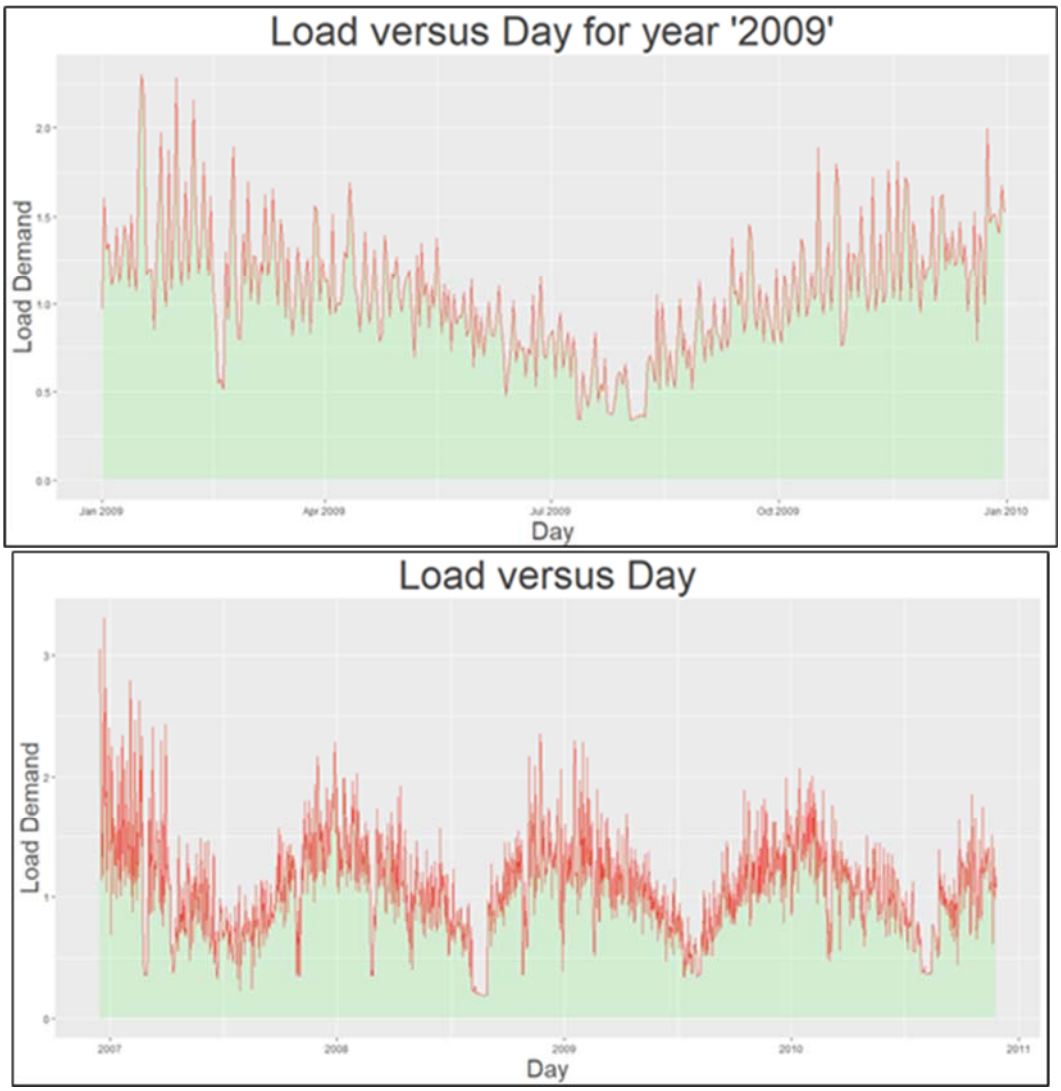
Parameter	Value
booster	gbtree
nrounds	2000
max_depth	6
eta	0.005
lambda	0.001
metric	rmse

The model achieved train-rmse of 0.66 and test-rmse:0.69 which can be further improved with the following time series models discussed below.

B). Seasonalities + ARIMA:



As discussed above, using regression, lower RMSE values couldn't be achieved. Hence, time series modelling technique ARIMA is implemented in this section. The procedure mentioned in the above image is used for modelling. Data is visualized to get idea about trend and seasonality of the data. Once the trend and seasonalities are identified, they are removed from the data to make the stationary data. Once the data is stationary, ARIMA model is used to fit the data using ACF, PACF and other diagnostic tools. Finally predictions are made using the ARIMA model and features.

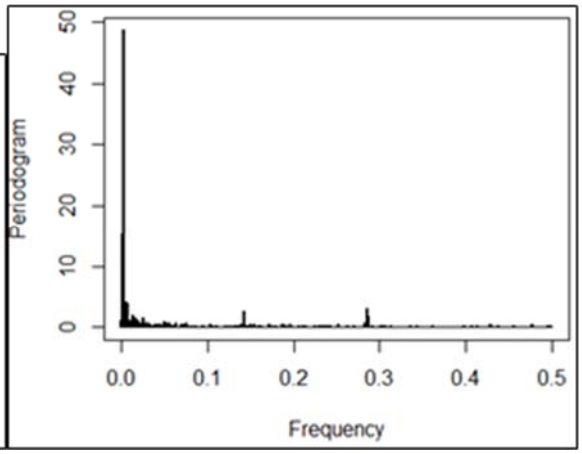


From the above plots yearly and weekly seasonalities can be clearly seen. As this data is for a single household, lot of challenges like missing data, high noise in the dataset etc. are present in the data. Hence hourly aggregate of the data is taken first and various modelling techniques are used to model the data. Later due to high noise even in the hourly data, daily aggregate of demand is taken.

Usually, multiple seasonalities like daily, weekly and yearly is present in electric load demand data. Hence the data is checked for seasonality using periodogram approach and also later this seasonality is confirmed using '**tbats'** package.

Once the periodogram for the data is extracted, the plot and the table corresponding to the periodicity of the data is shown.

frequency	amplitude	time
0.002962963	48.659897	337.500000
0.002222222	15.410657	450.000000
0.001481481	5.737415	675.000000
0.005925926	4.275972	168.750000
0.008148148	3.859439	122.727273
0.285925926	3.019477	3.497409
0.142962963	2.754549	6.994819
0.005185185	2.603021	192.857143
0.006666667	2.264385	150.000000
0.014074074	1.934961	71.052632



From the above periodicity table and plot it can be seen that there is a peak close to 365, which confirms the presence of yearly seasonality. Also, the presence of yearly seasonality can be confirmed used the code snippet below. This implementation uses **tbats** package in R.

"ts.total_power.yearly <- ts(total_power, frequency = 365)
fit <- tbats(y = ts.tot_power.yearly, use.parallel = T, num.cores = 8)
seasonal <- !is.null(fit$seasonal)
seasonal # if TRUE, seasonality is confirmed "

Hence after confirming the presence of yearly seasonality using exploratory data analysis, periodogram and other methods, it is removed from the data by decomposing using STL (Seasonal Decomposition of Time Series by Loess) in R. Code snippet shown below is used to remove yearly seasonality present in the data.

"decomposed.yearly <- decompose(ts.tot_power.yearly, "additive")
ts.tot_power.yearly <- ts.tot_power.yearly - decomposed.yearly$seasonal"

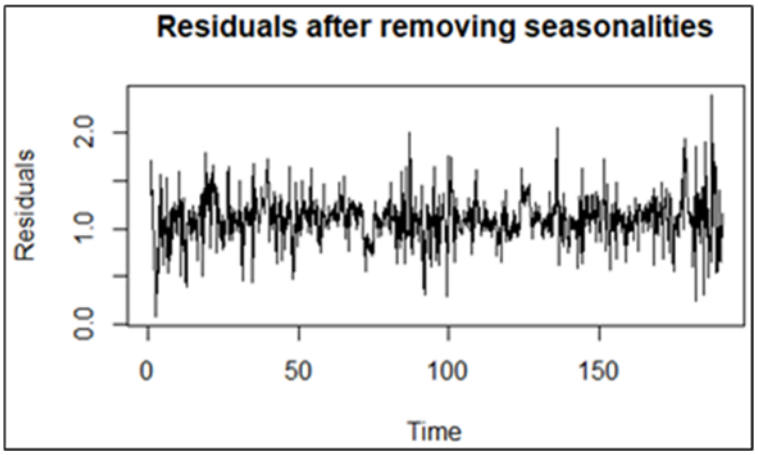
Once the yearly seasonality is removed, it can be confirmed by looking at the periodogram again. It can be seen from the above table that there is no peak close to 365. Hence the yearly seasonality is not present in the data anymore.

frequency	amplitude	time
0.285925926	3.0666053	3.497409
0.006666667	2.0228592	150.000000
0.026666667	1.3167675	37.500000
0.142962963	1.3002475	6.994819
0.012592593	1.0081700	79.411765
0.075555556	0.8816630	13.235294
0.008888889	0.8491778	112.500000
0.028888889	0.7816284	34.615385
0.050370370	0.7569190	19.852941
0.027407407	0.7401363	36.486486

Also in the above plot there is a peak close to 7(6.99) as well. This confirms the presence of weekly seasonality. Seven in the periodogram represents weekly seasonality because our data is aggregated per day. Once weekly seasonality is also removed, get the results of periodogram again to confirm.

freq	spec	time
0.006666667	2.0423043	150.00000
0.026666667	1.3123425	37.50000
0.012592593	1.0105807	79.41176
0.075555556	0.8771080	13.23529
0.008888889	0.8463831	112.50000
0.028888889	0.7773889	34.61538
0.050370370	0.7666346	19.85294
0.027407407	0.7386756	36.48649
0.023703704	0.7166815	42.18750
0.028148148	0.7027580	35.52632

From this table it can be confirmed that yearly and weekly seasonalities are removed as there are no peaks close to 7 or 365. Now data is plotted to see if there is possibility of any other seasonalities.

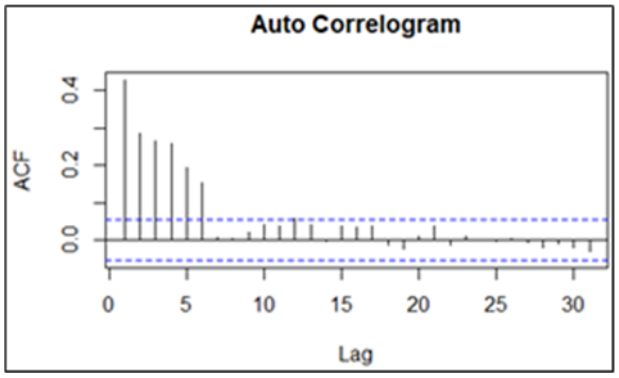


From the above plot, it can be seen that the data is fairly stationary and almost all of the seasonalities are removed. Also, not much trend is present in the data. It can be seen that mean and variance remain almost same w.r.t time. Confirm stationarity of data using Augmented Dickey-Fuller test (ADF test).

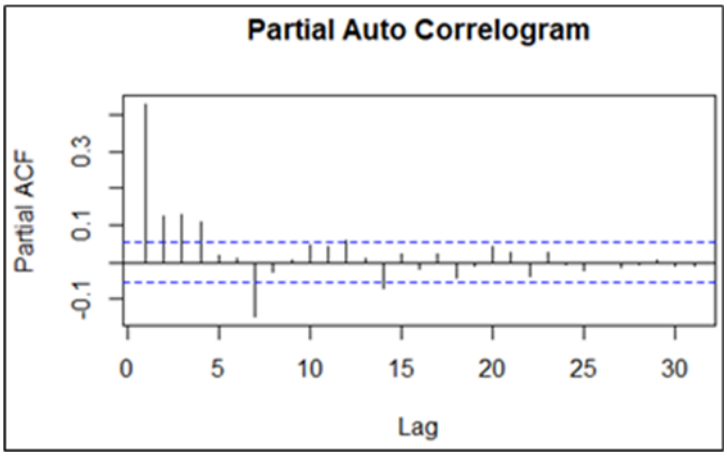
```
> adf.test(residuals) # check for stationary using dicky fuller

        Augmented Dickey-Fuller Test

data:  temp
Dickey-Fuller = -8.7799, Lag order = 10, p-value = 0.01
alternative hypothesis: stationary
```

From the above results it can be seen that the data is stationary. When p-value is less than 0.05, null hypothesis is rejected and alternative hypothesis is accepted. Null hypothesis in Augmented Dickey-Fuller test is that the data is not stationary. And hence, as null hypothesis is rejected in this case, data is stationary.

Now ARIMA model to fit to this stationary data. Correlogram and partial Correlogram for the stationary data are shown below.



We can see from the auto-correlogram (ACF) that the autocorrelations slowly decrease in magnitude and till lag 6 and also they exceed the significance bounds. But all other autocorrelations between lags 1-30 do not exceed the significance bounds.



The partial auto-correlogram shows that the partial autocorrelations at lags 1, 2, 3 and 4 continuously exceed the significance bounds and are slowly decreasing in magnitude with increasing lag.

Since the correlogram is zero after lag 6, and the partial correlogram tails off to zero after lag 4, this means that the following ARMA (Autoregressive Moving Average) models are possible for the time series:

• ARMA(4, 0) model, that is, an autoregressive model of order p=4, since the partial auto-correlogram is zero after lag 4, and the auto correlogram tails off to zero (although perhaps too abruptly for this model to be appropriate)

• ARMA(0,6) model, that is, a moving average model of order q=6, since the auto-correlogram is zero after lag 6 and the partial auto correlogram tails off to zero

• ARMA(4,6) model, that is, a mixed model with p and q greater than 0, since the auto-correlogram and partial correlogram tail off to zero (although the partial correlogram probably tails off to zero too abruptly for this model to be appropriate)

Principle of parsimony can be used to decide which model is best: that is, the model with the fewest parameters is best. The ARMA(4,0) model has 4 parameters, the ARMA(0,6) model has 6 parameters, and the ARMA(4,6) model has 10 parameters.

The auto.arima() function in R is used to find the appropriate ARIMA model. The output says an appropriate model is ARIMA(3,0,2). Akaike's Information Criteria (AIC) is used to select the model. Results after fitting the ARIMA model on the data are shown below.

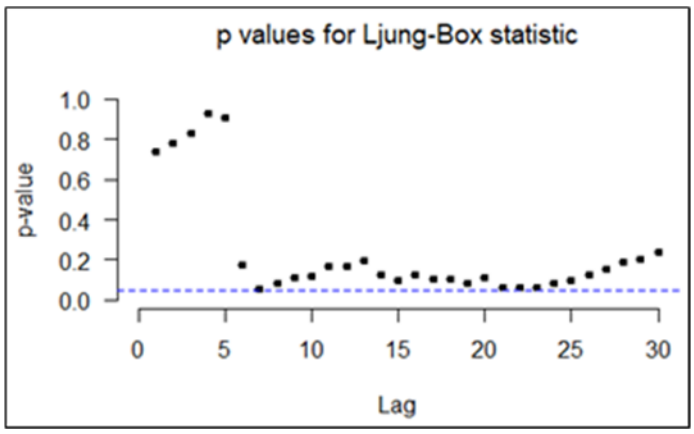
```
Series: residuals
ARIMA(3,0,2) with non-zero mean

Coefficients:
          ar1      ar2      ar3      ma1     ma2      mean
      1.578  -1.1189  0.3199  -1.2413  0.7571  1.1029
s.e.  0.074   0.0846  0.0352   0.0710  0.0546  0.0128

sigma^2 estimated as 0.04031:  log likelihood=250.46
AIC=-486.92   AICc=-486.84   BIC=-450.59
```

As discussed above, AIC is used in auto.arima() to select the best model based on principle of parsimony. Model with least AIC value is selected as the best model. The model ARIMA(3,0,2) selected above has an AIC value of -486.92. Now, diagnostic tests need to be run on the residuals to check if the characteristics of white noise are present in the residuals.

Diagnostic tests on residuals:

a. Ljung Box test:

This test is done to see if correlations exist between the residuals at various lags.

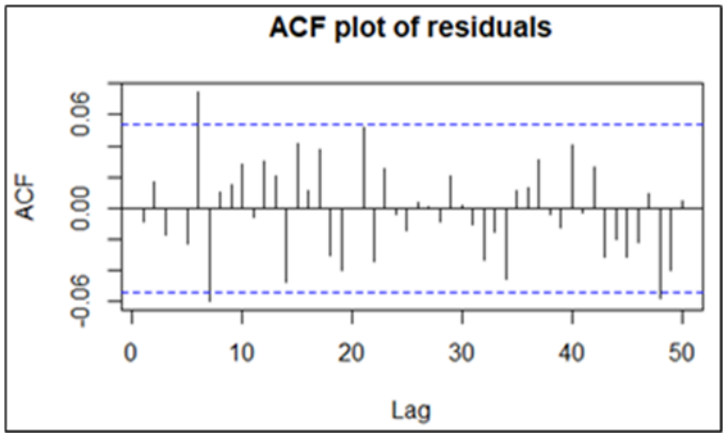



```
> Box.test(x = model$residuals, lag = 50)

        Box-Pierce test
 data:  model$residuals
X-squared = 54.905, df = 50, p-value = 0.2941
```

Notice that the p-values for the modified Box-Pierce all are above 0.05, indicating "non-significance." This is a desirable result. It can be seen from the above plot that the p-

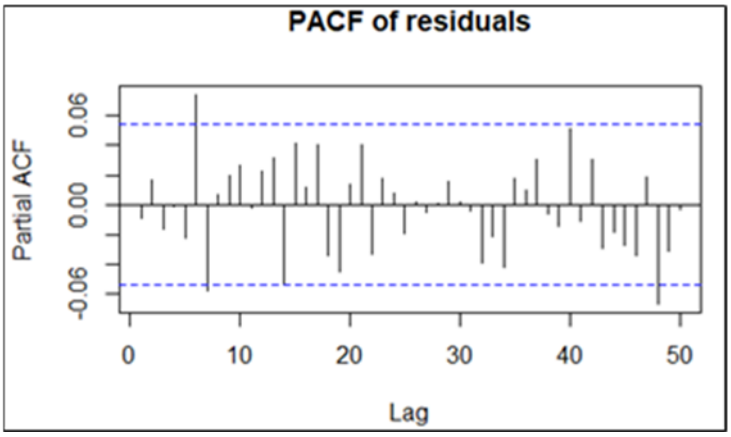
values at all lags are well above 0.05. Hence with 95% confidence, it can be said that there are no correlations between the residuals, which is one of the significant characteristic of white noise. Also, the p-value for the whole test is 0.29, which is well above 0.05. Hence we fail to reject null and accept the null hypothesis that there is no correlation between residuals.

b. ACF plot (auto-correlogram) of residuals:



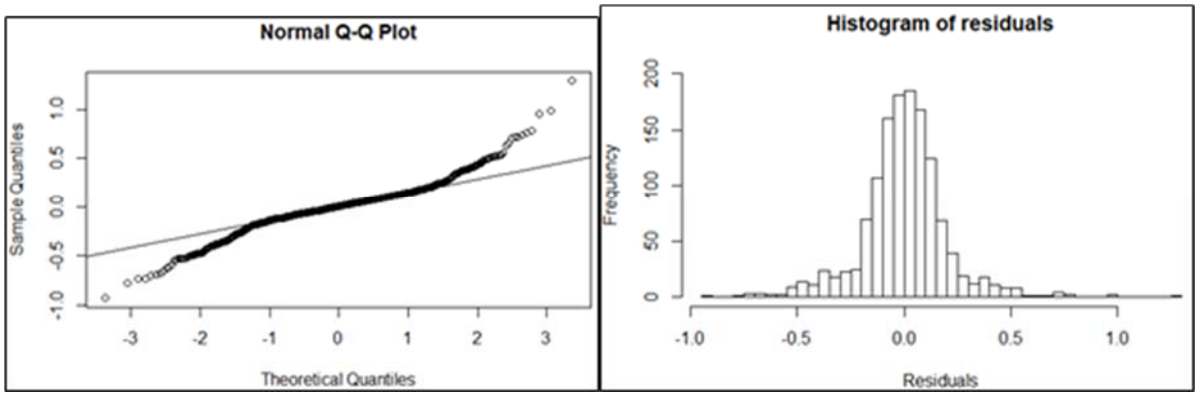
This auto-correlogram shows auto-correlogram of residuals for first 50 lags. It can be seen that correlation at almost all the lags till 50 is well within bounds, which is desirable and characteristic of white noise.

c. PACF plot (partial auto-correlogram) of residuals:



This partial auto-correlogram shows auto-correlogram of residuals for first 50 lags. It can be seen that the partial correlation at almost all the lags till 50 is well within bounds, which is the characteristic of white noise. The remaining residuals show that this model has captured the patterns in the data quite well, although there is a small amount of autocorrelation left in the residuals (seen in the significant spike in the ACF and PACF plots at lags 5, 6 and 48). This suggests that the model can be slightly improved, although it is unlikely to make much difference to the resulting forecasts.

d. QQplot and histogram of residuals to check normality:

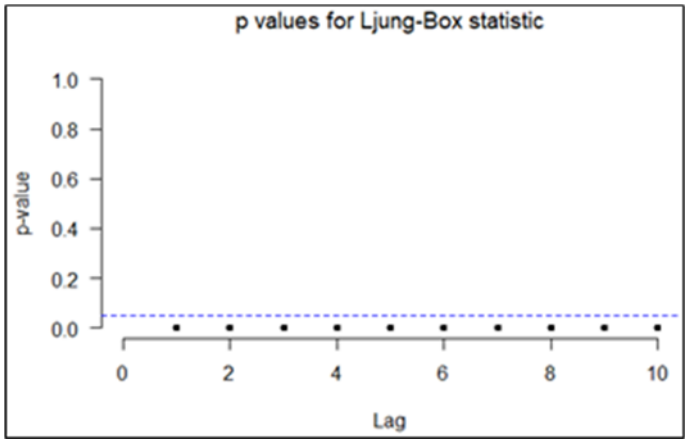


From the above QQ plot and histogram, it can be seen that the residuals are fairly normally distributed.

If the squares of residuals (after fitting ARIMA model) are correlated, then there is a need to model variance/volatilities as well in the model. They can be modelled using GARCH (Generalised Autoregressive Conditional Heteroskedastic) modelling approach.

Test for squares of residuals using Ljung box test:

Code Snippet to run Ljung box test on the squares of residuals:
Box.test(x = model$residuals^2, lag = 50, type = "Ljung")



Box-Pierce test

data: model$residuals^2
X-squared = 265.33, df = 30, p-value < 2.2e-16

From the above plot and results, notice that the p-values for the Ljung Box test are all below .05, indicating “significant” correlation. Hence, with 95% confidence, it can be said that there are correlations between the square of residuals. This calls for modelling residual squares (variances/volatilities) using ARCH/GARCH approach, but that is not discussed in this report. As a following step, the accuracy of this method is attempted to improve using deep Neural Networks.

C). RNN using LSTM and Sequence to Sequence LSTM:

In this part, the effectiveness of a different deep learning technique is explored for performing building level forecasting. The presented methodology uses Long Short Term Memory (LSTM) algorithm. Presented work investigates using two variations of the LSTM:

1) Load forecasting using standard LSTM and
2) Load forecasting using LSTM based Sequence to Sequence (S2S) architecture.

Both methodologies are tested on a benchmark dataset which contained electricity consumption data for a single residential customer with time resolutions of one hour.

LONG SHORT TERM MEMORY:

An LSTM network is comprised of memory cells with self-loops as shown in Fig. 1.a). The self-loop allows it to store temporal information encoded on the cell's state. The flow of information through the network is handled by writing, erasing and reading from the cell's memory state. These operations are handled by three gates respectively: 1) input gate, 2) forget gate and 3) output gate. The equations (1.a) through (1.f) express a single LSTM cell's operation.

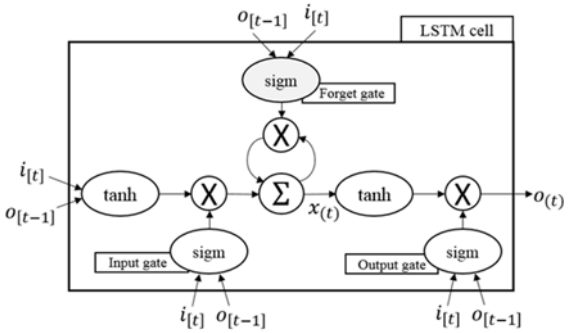


Figure: Architecture of single LSTM cell.

Standard LSTM:

The objective of the presented methodology is to accurately estimate the electricity load (active power) for a time step or multiple time steps in the future, given historical electricity load data. I.e. having M load measurements available

The output of the network, $\hat{y}[t] \in \mathbb{R}$ is an estimation of the active power for the next time step. With this model, the electricity load for the next time step is predicted given a set of load measurements of the past. To predict further into the future, the predictions made by the model can be used as additional inputs for the next time step.

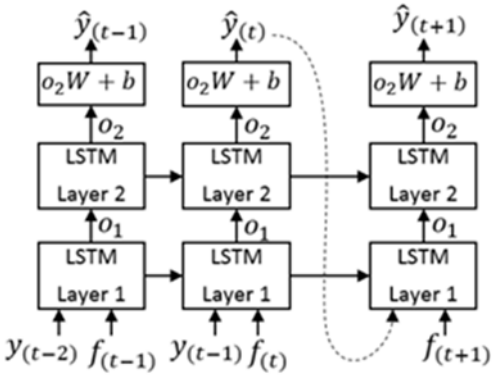


Figure: architecture of standard LSTM.

Sequence to Sequence LSTM:

In order to further improve the flexibility of the load forecasting methodology, a different architecture based on LSTM, called sequence to sequence (S2S), is explored. S2S is an architecture that was proposed to map sequences of different lengths.

The S2S architecture that is employed for load forecasting consists of two LSTM networks: encoder and a decoder. The task of the encoder is to convert input sequences of variable length and encode them in a fixed length vector, which is then used as the input state for the decoder. Then, the decoder generates an output sequence of length n. In this instance, that output sequence is the energy load forecast for the next n steps.

The main advantage of this architecture is that it allows inputs of arbitrary length. I.e. an arbitrary number of available load measurements of previous time steps can be used as inputs, to predict the load for an arbitrary number of future time steps. To perform the prediction, y is used as the input for the encoder together with the corresponding date and time for and time is used as input.

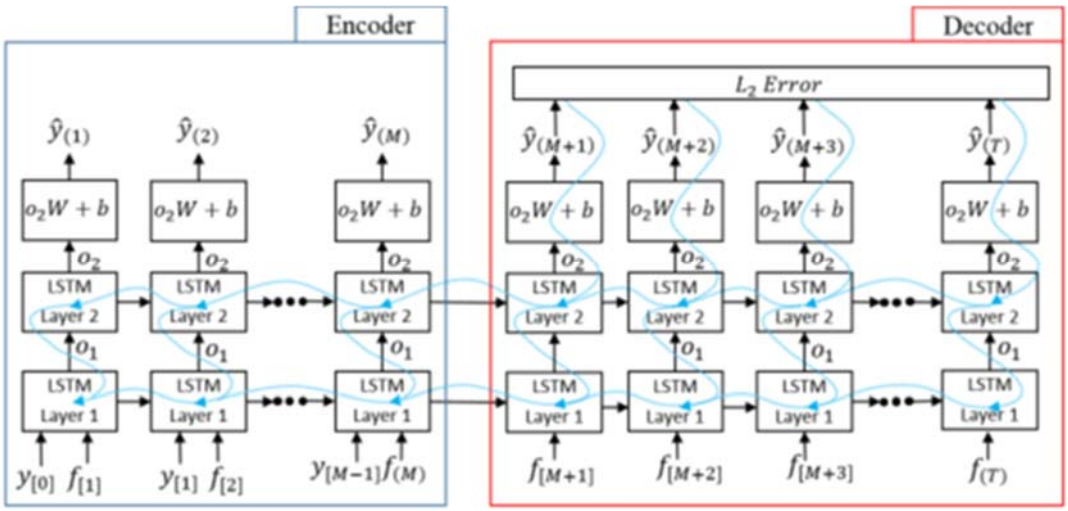


Figure: Encoder and Decoder architecture of Sequence to Sequence LSTM.

Results from LSTM Models:

The entire dataset is divided into train and test parts in 9:1 ratio. Both the models are train on train data and validated on test dataset. Prediction accuracy of S2S architecture is comparatively better than Standard architecture.

Actual Power vs. Predicted power for both models is plotted below.

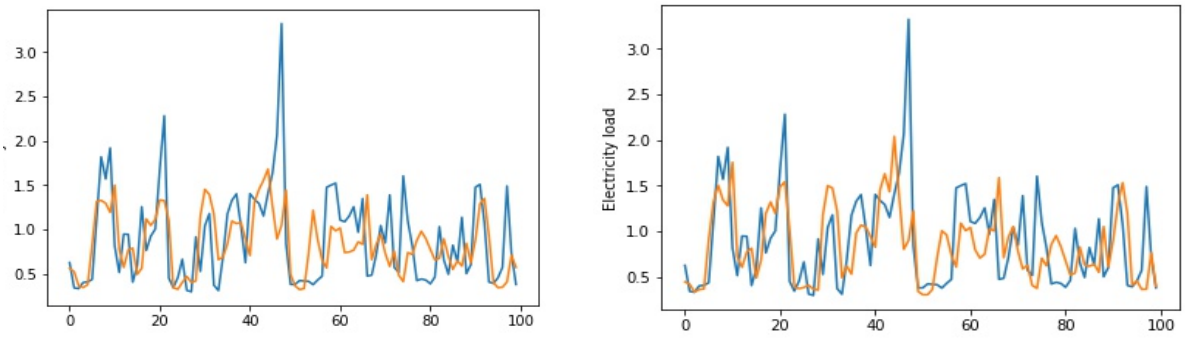


Figure: Actual (Blue) vs. Predicted (orange) Energy load of standard LSTM and Sequence to Sequence LSTM.

MSE from **Standard LSTM Model** is 0.241 and **S2S LSTM** is 0. 219. Hyperparameters like number of units and sequence length can be tuned to achieve better accuracy.

7. Code of data analysis:

Please find the code attached in the e-mail.

8. Conclusion of the Analysis:

Exploratory Data Analysis is done to gain insights into the data. Using this information, initial model is attempted using linear regression. Due to limitations in this model, time series features couldn't be captured with decent accuracy. Hence ARIMA model is successfully fit to the data after identifying and removing weekly and yearly seasonalities in the data.

But ARIMA model fails to capture seasonalities that change over time. It assumes that the seasonalities remain unchanged over the prediction interval. Hence Recurrent Neural Networks are used to improve the accuracy of the model even further.