AMS 559 – Assignment 1

Name : Viraj Kamat
SBU-ID : 112818603

Objective

Given our dataset containing the energy load for three homes along with their weather information, we need to predict the load for the houses either for the next hour or the next day.

We intend to use machine learning in order to achieve this objective.

Data Preparation

In order to load and prepare the data I used the Python Pandas module. The weather data and electricity consumption data for each of the home was split into two files. I had to combine this data into one file for each of the homes.

The weather data for each of the files had two categorical variables (icon, summary), which were in string format which were converted to numbers using the pandas **pd.factorize** function. Next, to convert timesteps in hours and days respectively in order to perform predictions for the next hour and day I found the following issues with the data :-

1. The data provided had inconsistent timesteps, either present in steps of every minute or every hour
2. These timesteps were inconsistent across the weather data and meter data for each of the homes which made the merging of the two datasets inconsistent.

In order to overcome the issue, I used the pandas resample function that would merge minute by minute or half-hourly data into an hourly data. This was applied to both weather and meter data for the files.

Next, with consistent time-steps across both weather data and meter data, I checked the format of the datetime fields, converted them to an appropriate format (unix timestamp) and merged the two files into a single dataframe over the date-time field using the pandas **pd.merge** function.

The prediction would also be improved if we were able to identify the seasons. To do so the calendar module was used, given the date and the time I identified the holidays, if a given day was a holiday I created an **is_a_holiday** field with a value of 0 else with a value of 10 .

Similarly, it would be better to know the season that a particular record belongs too, since a summer or a winter season is more likely to account for higher air conditioning usage that spring or autumn, thus all days that fell in this season range and additional record called season was added that gave a value of 10 if it was spring or summer, else I gave a value to that field as 0.

To identify whether a family was at home or not was not straight forward, the only way to infer this is to compare the current total energy usage with the over all energy usage across all hours/days, any values that were less than 10 percentile of the total energy usage was marked with a field of **is_family_out** which was given a value of 0, others were given a value of 10.

In each of the above cases the additional fields for season, family out, holiday were created so that those rows where the family was indeed at home was given more weightage, thus implying a higher power usage.

Since the original data now has been resampled for hourly predictions, we also need data for daily load prediction. In order to do this, I resampled the rows into a daily data with the Pandas resample function. While resampling however a custom aggregation function was used, i,e all power consumption units were summed for the hourly data whereas the weather data for a day was computed by taking a mean of the weather data values across all hours of the day. Finally, to create the column for the predicted load for the next day I copied the total energy prediction for the next day into a new column called **load_next_day**.

Correlation and HeatMaps

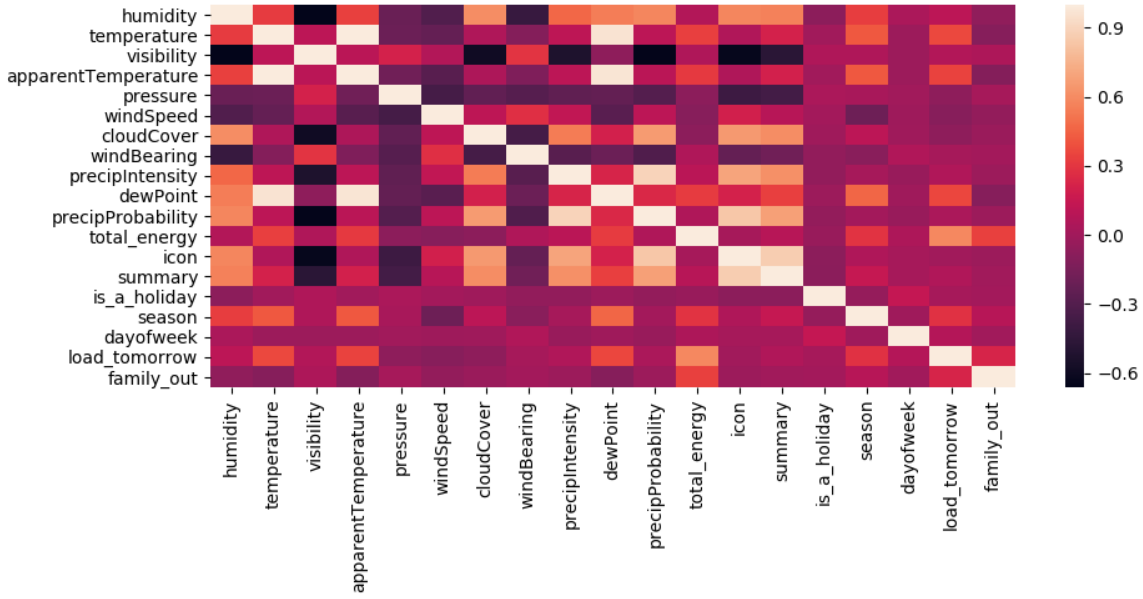

Figure 1 : Heatmap of data for daily electricity load (House B)

As we can see from the correlation matrix above, the energy prediction for the next day is dependent on the temperature/apparent-temperature which means hotter a day more is the air conditioning use; the energy consumption for the current day; and lastly dewpoint along with the season. The importance of the season field is straight forward as it means that in colder seasons people are more likely to stay indoors resulting in a more power consumption.

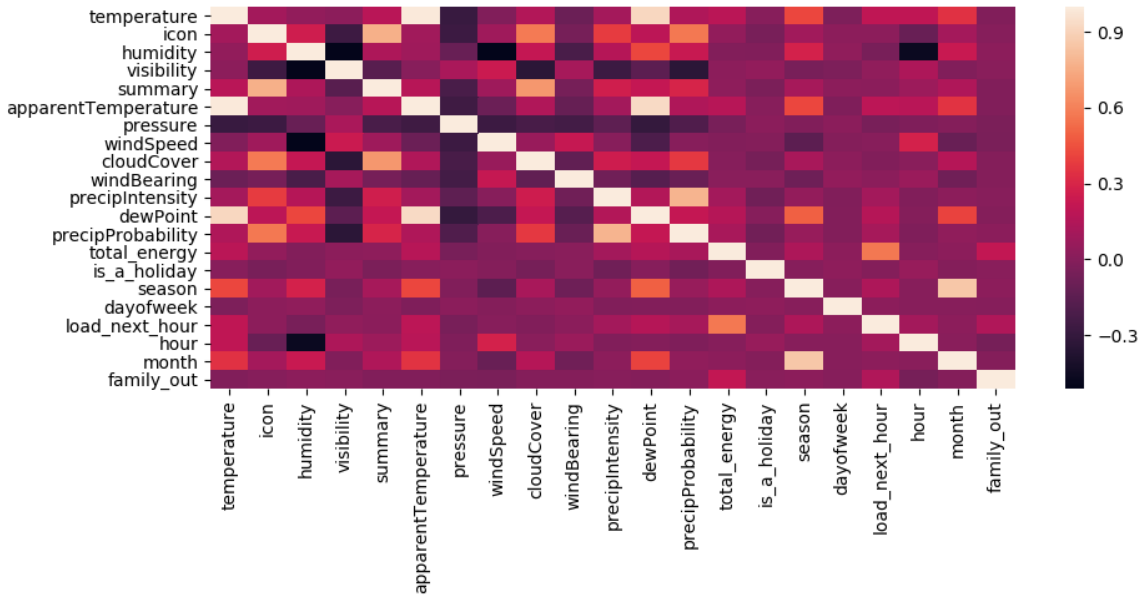

Figure 2 : Heatmap of data for hourly electricity load (House B)

In case of hourly demand of the electricity we can see that it is dependent on the load demand for the previous hour, which would make sense since appliance keep running across hours. We can also see the the apparent-temperature, temperature, precipitation probability also affects the load for the next hour – rains or higher temperatures would mean people are more likely to stay indoors and turn on appliances.

Choice of Models and Traning the Models

The following models were used for traning and predicting the load values :

1. **Linear Regression** – It tries to output the predicted value as a function of the dependent variables . It attempts to construct a best fitting line in an N-dimensional space that explains the dependency between the variables and the predicted value – in our case the weather data and the load demand.
2. **XGB (Extreme gradient boost)** – A decision tree classifier that has multiple decision trees (learners) in sequence that help each one better predict/learn the output values to train the model. Each decision learns from the errors/biases from the previous trees.
3. **RandomForestRegressor** - Uses multiple decision trees in parallel trained on multiple samples of data picked with replacement, it is different from XGB in that the decision tress are not employed in sequence but rather the average of the decisions from each tree is taken in training the model. This is known as bagging.

4**. GradientBoost** - Also a boosting algorithm where in the multiple learners in sequence learn from the mistakes of the previous weaker models, the current model hopes to improve my fitting itself to the derivation of the loss function of the previous learners. It uses an additive approach where each tree depends of the results of all the previous trees.
5. **LSTM (Long short term memory)** - Using a gated recurrent neural network mostly used in time series data wherein the values of the current timestep is based on those of the previous timesteps. This is important as in our training we are predicting energy load based on the observations of weather in the previous timesteps.

In each case there was a training set created which accounted for 80% of the data and the rest the testing set. The sklearn train_test_split function was used to split the data and dataset was not shuffled to perform our predictions.

In both hourly and daily load predictions, the input values in order to train the model was the weather data but the predicted values was the load for the next hour and load for the next day respectively.

The training dataset was used to train the model and testing the perform our predictions on each of the trained model, the MAE was used to get an estimate of how well our model performed.

Implementation of the Naïve Model

The implementation of the Naïve model was straight forward. For each row in the dataset for which the total energy consumption was known, I simply randomly selected as the total energy consumption from another row and set is as the predicted value for that row.

While this approach would seem too simple, another technique that was attempted computed the mean of the total energy consumption of all the rows up until that row and set it as the naïve predicted value, this approach however got too close to the real value – this naïve model was fairly accurate but not so useful in comparing it against the predictions of our machine learning models.

Obtaining the Mean Absolute Error

The Mean Absolute Error was obtained as following :-

1. An absolute value of the difference of the true value and the predicted value was taken.
2. The differences for all the rows was summed up.
3. The mean of the difference was taken by dividing it with the total number of the rows.

This was done for both, each of the machine learning models and the Naïve model.

Mean Absolute Error Results

House B

Model Name	Daily data MAE	Hourly data MAE	Naïve daily data MAE	Naïve hourly data MAE
Linear Regression	10.3609	0.7568	9.85888	1.1024
XGB Boost	8.5346	0.5807	8.6018	1.1187
RandomForest Regressor	7.8602	0.7401	10.0963	1.0878
Gradient Boost	7.8897	0.6198	8.6594	1.0910
Long Short Term Memory	11.2564	0.60871	10.2378	1.0797

House C

Model Name	Daily data MAE	Hourly data MAE	Naïve daily data MAE	Naïve hourly data MAE
Linear Regression	22.2450	1.3082	32.0079	2.3266
XGB Boost	19.3249	1.2544	26.6421	2.5634
RandomForest Regressor	19.1483	1.3466	29.8200	2.5152
Gradient Boost	20.2483	1.273	37.2548	2.5296
Long Short Term Memory	18.1816	1.1564	33.1595	2.5161

House F

Model Name	Daily data MAE	Hourly data MAE	Naïve daily data MAE	Naïve hourly data MAE
Linear Regression	2893.7281	198.1985	3547.4605	421.5855
XGB Boost	2712.7848	164.5159	3910.0500	437.3853
RandomForest Regressor	2693.8215	175.8957	4233.5789	446.2692
Gradient Boost	2858.4364	165.8347	3362.0524	440.2245
Long Short Term Memory	N/A	196.880	N/A	454.556

Diagnosing the Models

The **Linear Regression** model's performance as can be seen from the MAE charts above is relatively good. It happens to be the simplest of all models and provides performance at par with advanced machine learning models such as XGB and RandomForest Regressor.Since the goal of the prediction was the output (energy demand) calculated as a function of input values in our case the weather data information, linear regression seemed to be an appropriate model to begin with.

XGB boost and **GradientBoost** seemed to perform relatively same and on some occasions better than the Linear Regression model. Both models employ decision tree classifiers used in sequence to learn from the errors of the previous to provide a better fitting model. Each tree would pick up randomly sample data, train itself and forward the trees parameters to the next tree, in our case weather data would be randomly sampled to each tree, a prediction model would be trained upon it and the next tree would receive the paramaters of this prediction model – it would then randomly sample a new set of weather data points which would then again adjust its prediction model accordingly for predicting the energy demand either hourly or daily. This implies that the load demand for a timestep is also dependent on the load demand and weather factors at previous timesteps; which would enable a training a more robust model.

RandomForestRegressor performed better than the Linear regression model but performed slightly worse than any of the Boosting model, which makes sense since it employs multiple decision tress trained upon randomly sampled datapoints whose prediction model is an average of those decision

trees, no to datasets are the same and since each decision tree receives a different set of data to train itself it can introduce unwanted biases into the final output of the model.

Long Short Term Memory seemed to be promising but did not deliver as expected possibly as the hyperparametes need to be tuned further. However, its predictions were in the expected range, LSTM for short, it is gated recurrent neural network that takes in values from the previous timesteps in order to help training upon the data at the current timestep, thus the use of LSTM is ideal and preferred since it would predict load demand depending on the values of previous timesteps – in our case the weather data seen at previous timesteps would definitely help in improving the prediction for the load demand.

Prediction vs True Value Plots (Best 2 models per home)

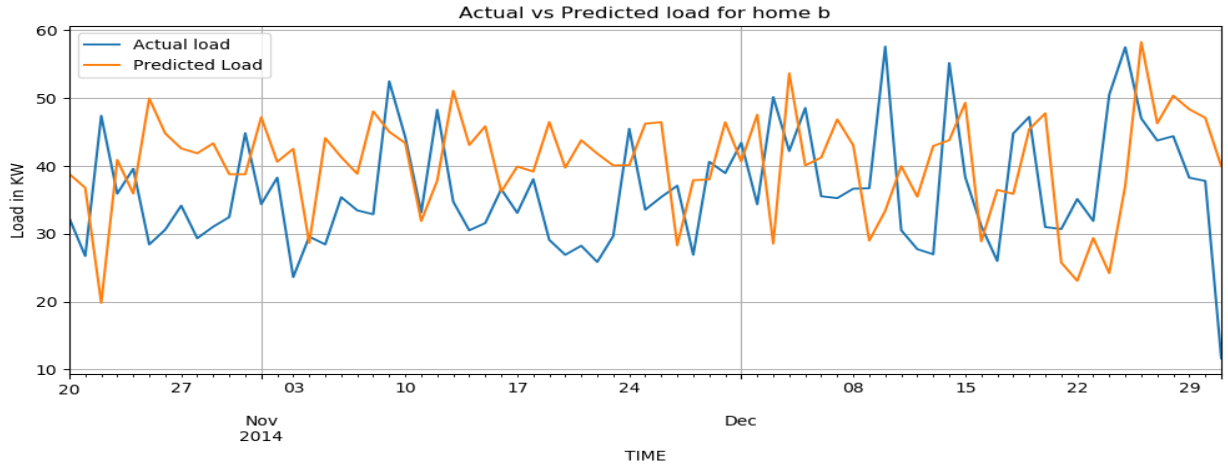

Fig 4 : Daily predictions using Linear Regression (House B)

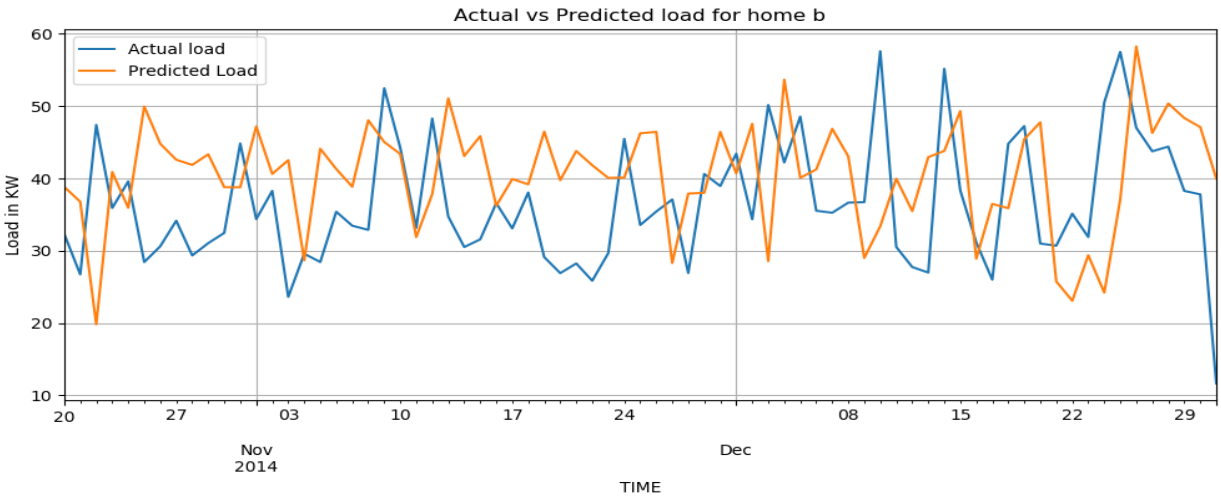

Fig 5 : Hourly predictions using Linear Regression (House B)

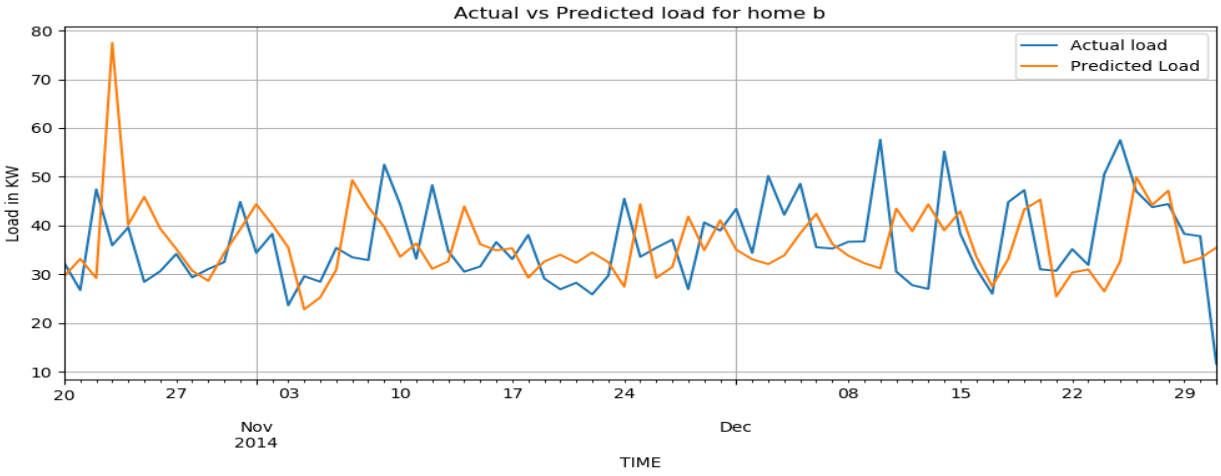

Figure 6 : Daily Predictions using LSTM (House B)

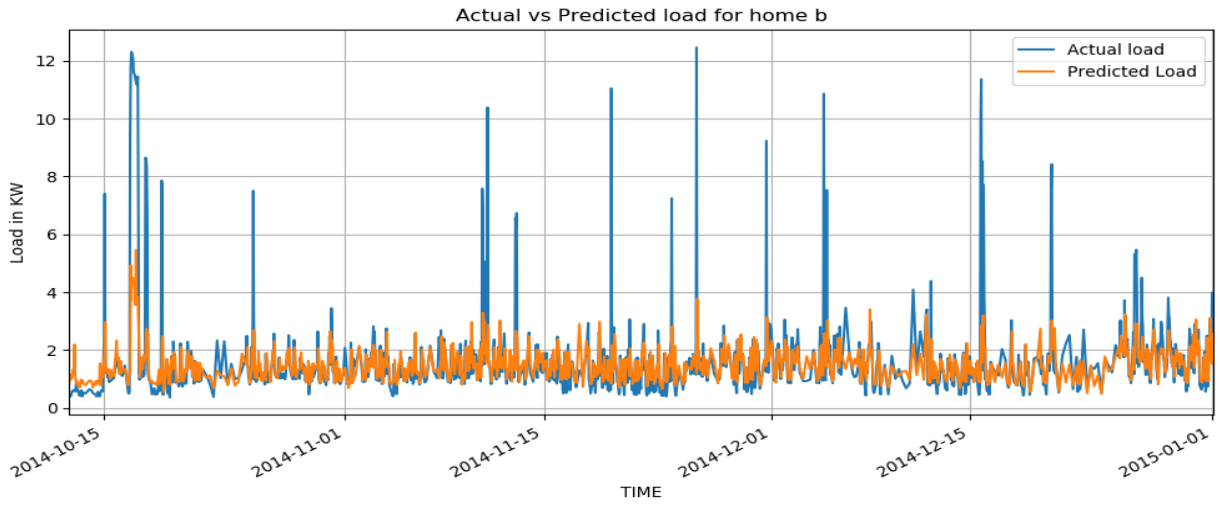

Figure 7 : Hourly Predictions using LSTM (House B)

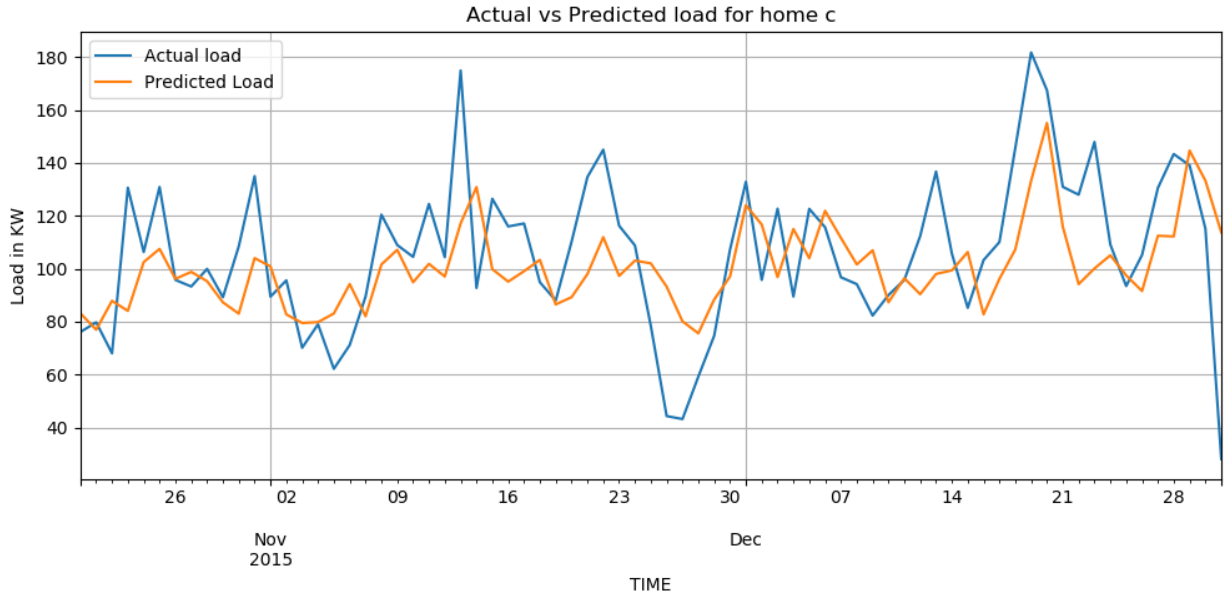

Figure 8 : Daily predictions using XGB (House C)

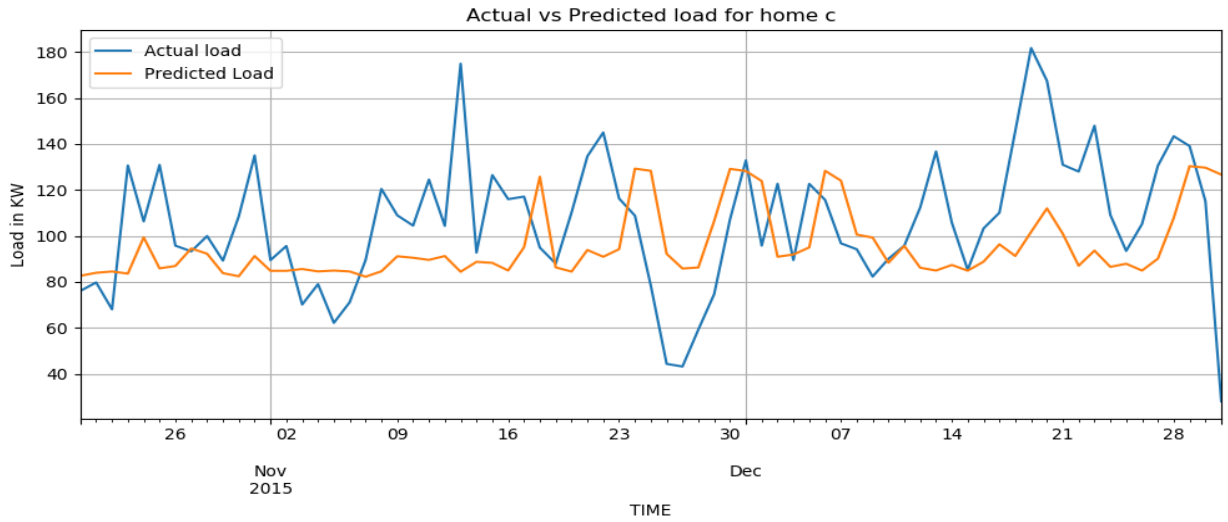

Figure 9 : Daily predictions using LSTM (House C)

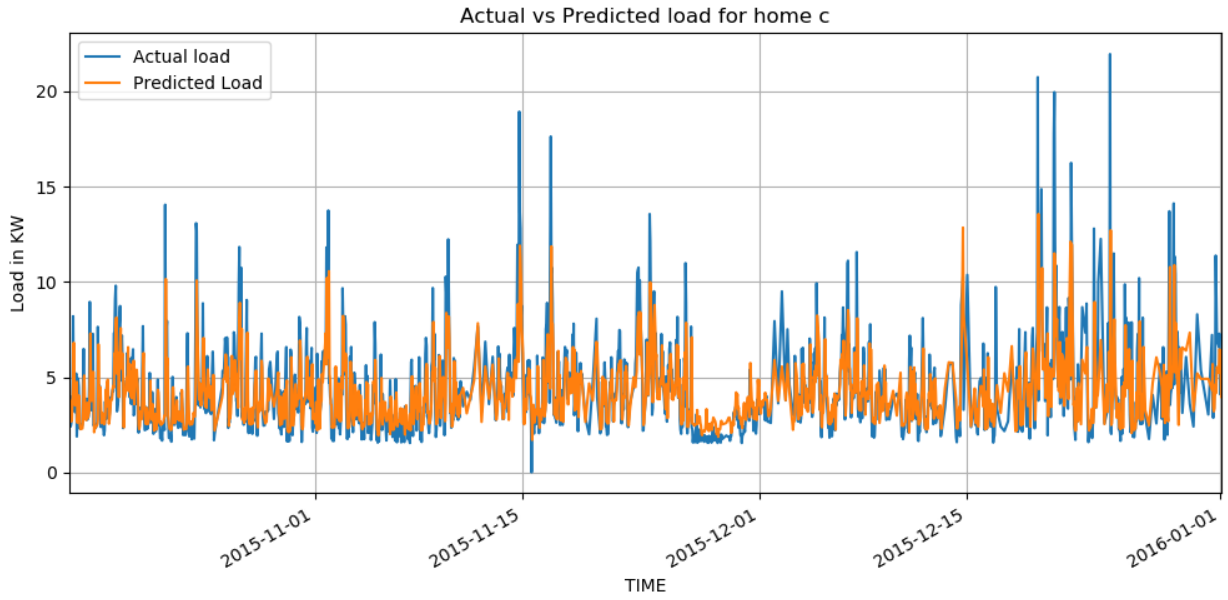

Figure 10 : Hourly predictions using XGB (House C)

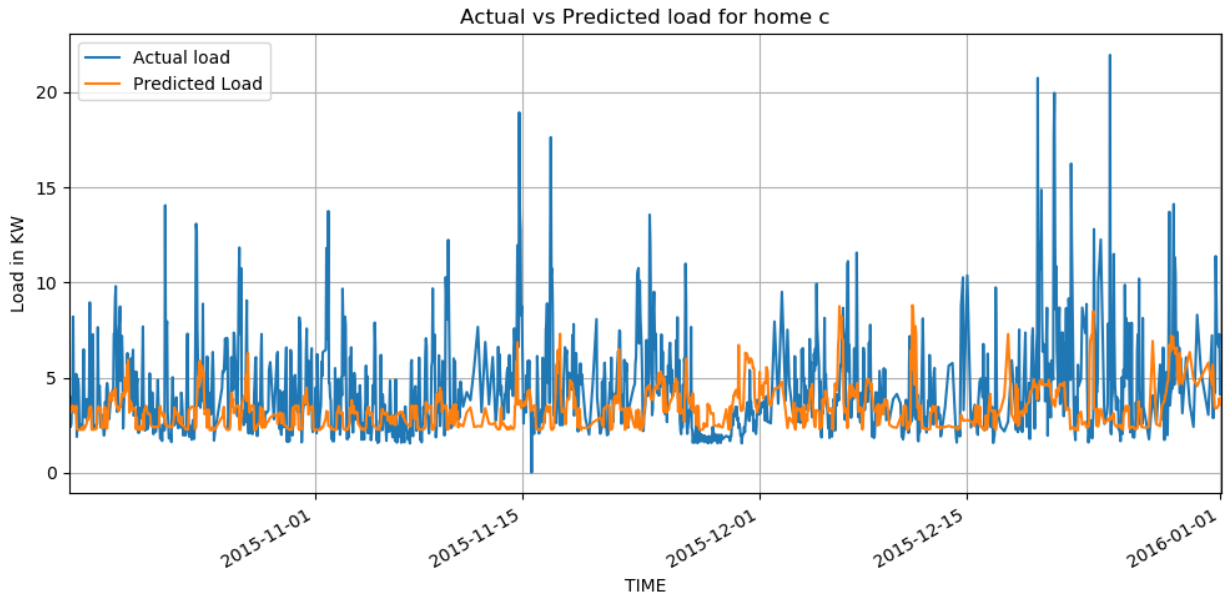

Figure 11 : Hourly predictions using LSTM (House C)

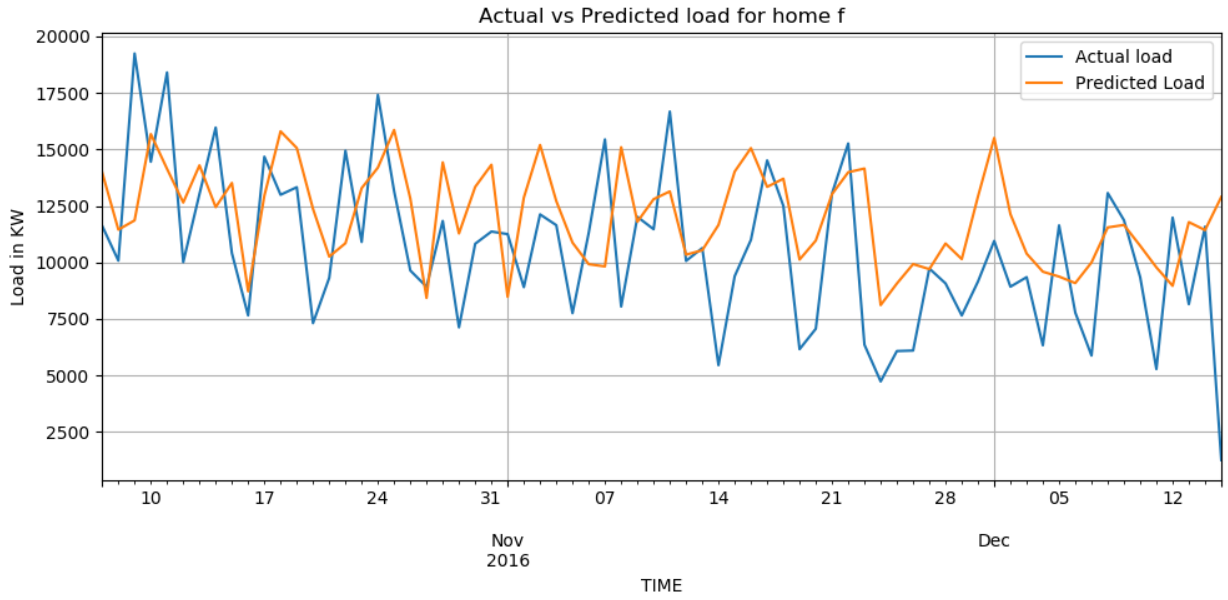

Figure 12 : Daily predictions using RandomForestRegressor (House F)

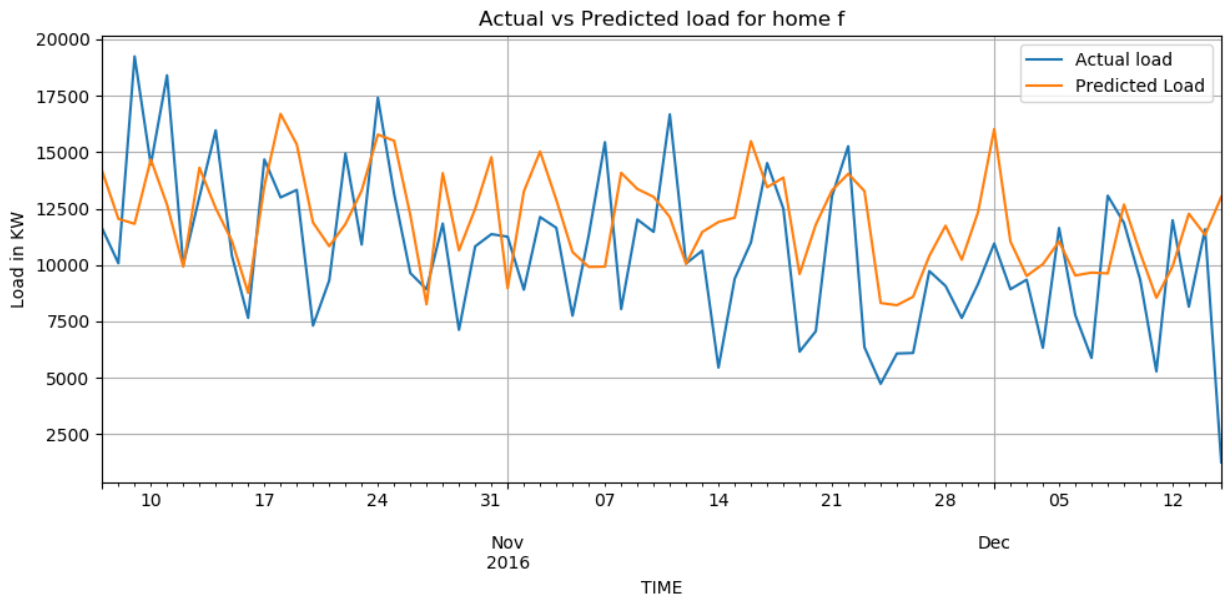

Figure 13 : Daily predictions using XGB (House F)

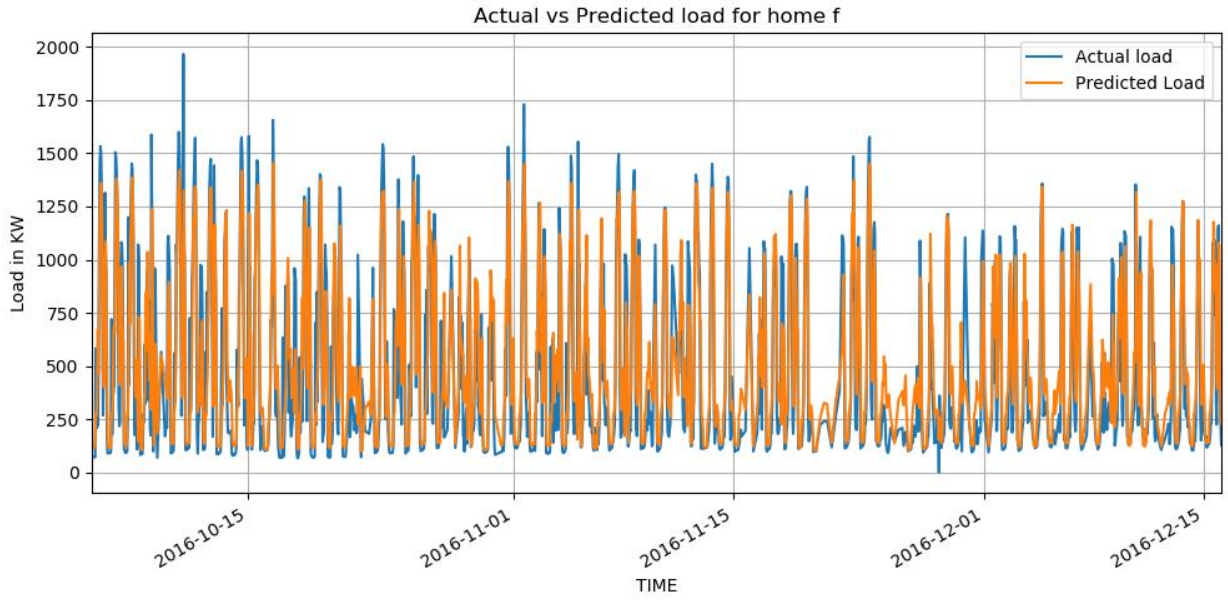

Figure 14 : Hourly predictions using RandomForestRegressor (House F)

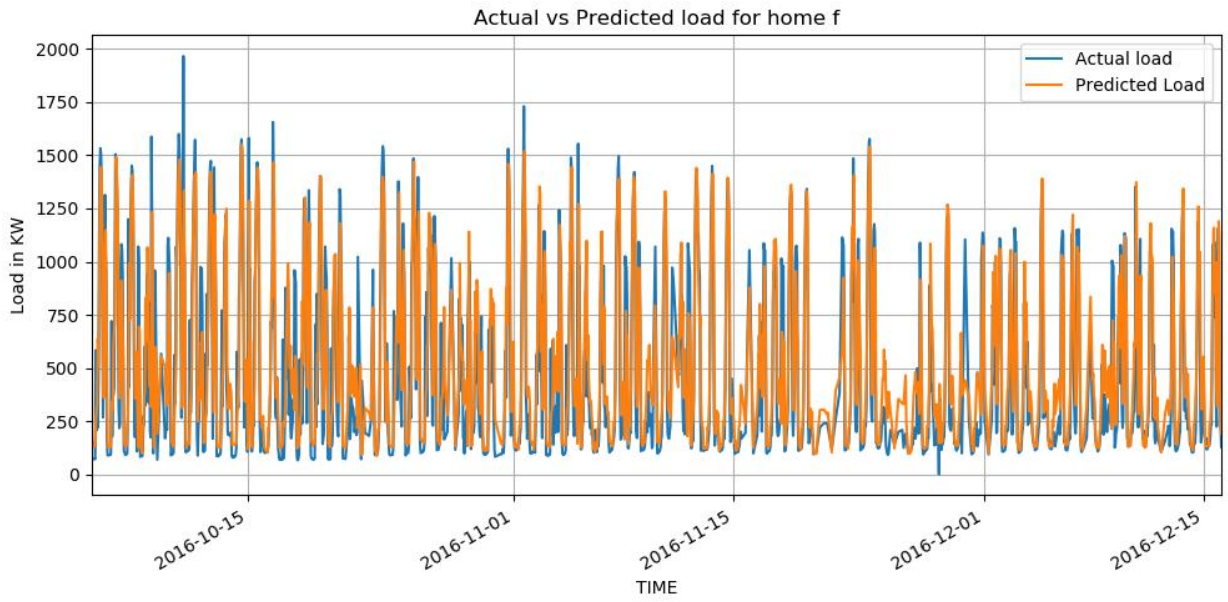

Figure 15 : Hourly predictions using XGB (House F)

Conclusion

In the above observation it has been demonstrated that we can indeed predict load demand with Machine learning provided appropriate models are used and the dataset in properly cleaned and modified to help with training the models.

The boosting models namely XGB and GradientBoost performed well as they captured the information from multiple datapoints to build a prediction model for the data.