Energy Usage Prediction

Prepared By: Rutvi Macwan

Table of Contents

Table of Figures

Abstract

In recent years, increasing use of various electrical and electronics devices has produced a major challenge for limited amount of reproducible energy sources. To overcome this challenge, multiple research studies have been conducted, where observations from low-energy households are analyzed to predict the energy consumption. These observations take the energy usage readings at the interval of every 10 minutes with other information such as temperature, humidity and atmosphere pressure etc. This project report contains a brief introduction of this household data and different approaches of Supervised Machine Learning for energy usage prediction.

Introduction

The dataset provided here has been collected from the UCI data repository. It contains the energy consumption readings from various low-energy households with other atmospheric parameters such as temperature, humidity, pressure, wind speed and visibility information. The goal of this project is to build the supervised model that can predict the energy usage based on the information provided in a given dataset.

I will start with data exploration and the feature engineering process. Next, I will use these engineered features to come up with the best suitable model for energy prediction. To decide upon the model, I will do various hyper parameter tuning and will compare their accuracies.

Data Exploration and Feature Engineering

This dataset contains the readings of the year 2016 for the first 5 months. These readings are noted at the interval of 10 minutes. Therefore, to predict the hourly usage, it is necessary to resample the dataset by hour. After resampling the dataset, there still exists a 'date' feature unique to all the observations. If not handled properly, it may create the leakage. Therefore, I extracted the 'Hour', 'Month' and 'Day' values from the date column and partitioned the dependent and independent variables.

Now after plotting all the independent variables with dependent one, it can be observed that there exists no linear relationship. In addition, there exists many features having similar pattern with dependent target variable. Below is shown the most relevant feature plots.

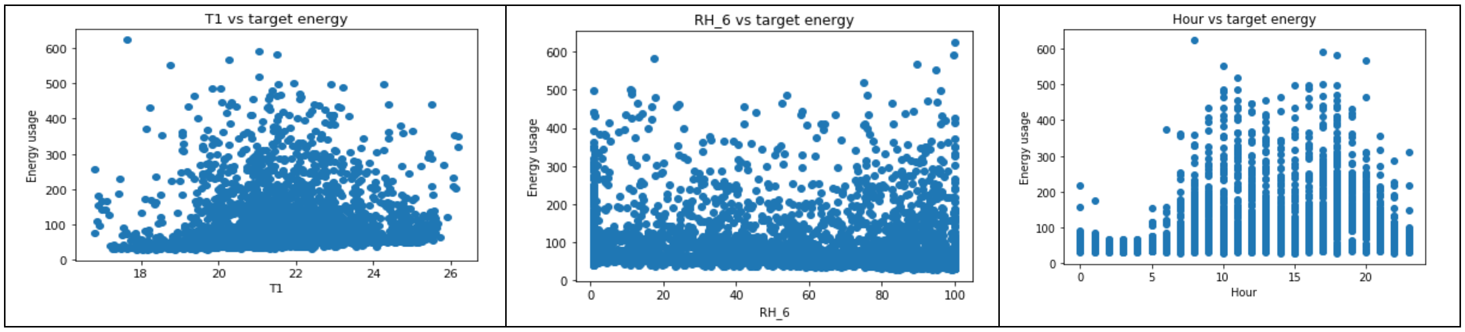


Figure 1: Feature plotting with target variable

So it becomes important to eliminate any redundant features to reduce the dimensionality of the data.

Apart from this, when we do box plotting of all the features as shown in figure2, we can see there are some outliers in the data. These outliers may be the result of any unusual condition and/or natural variation. Removing them may result in a loss of data variance information. If I remove them, it means the model to be built is forced to appear less variable than it is in reality. Therefore, I have decided to keep them as it is in the dataset.

Another thing noticed from the box-plot of features is that, there is a huge difference between some feature values. If this difference is not handled properly, the model can suffer 'high bias' issue. To deal with this, I have used 'MinMaxScaler' to scale all the features as it is robust to outliers.

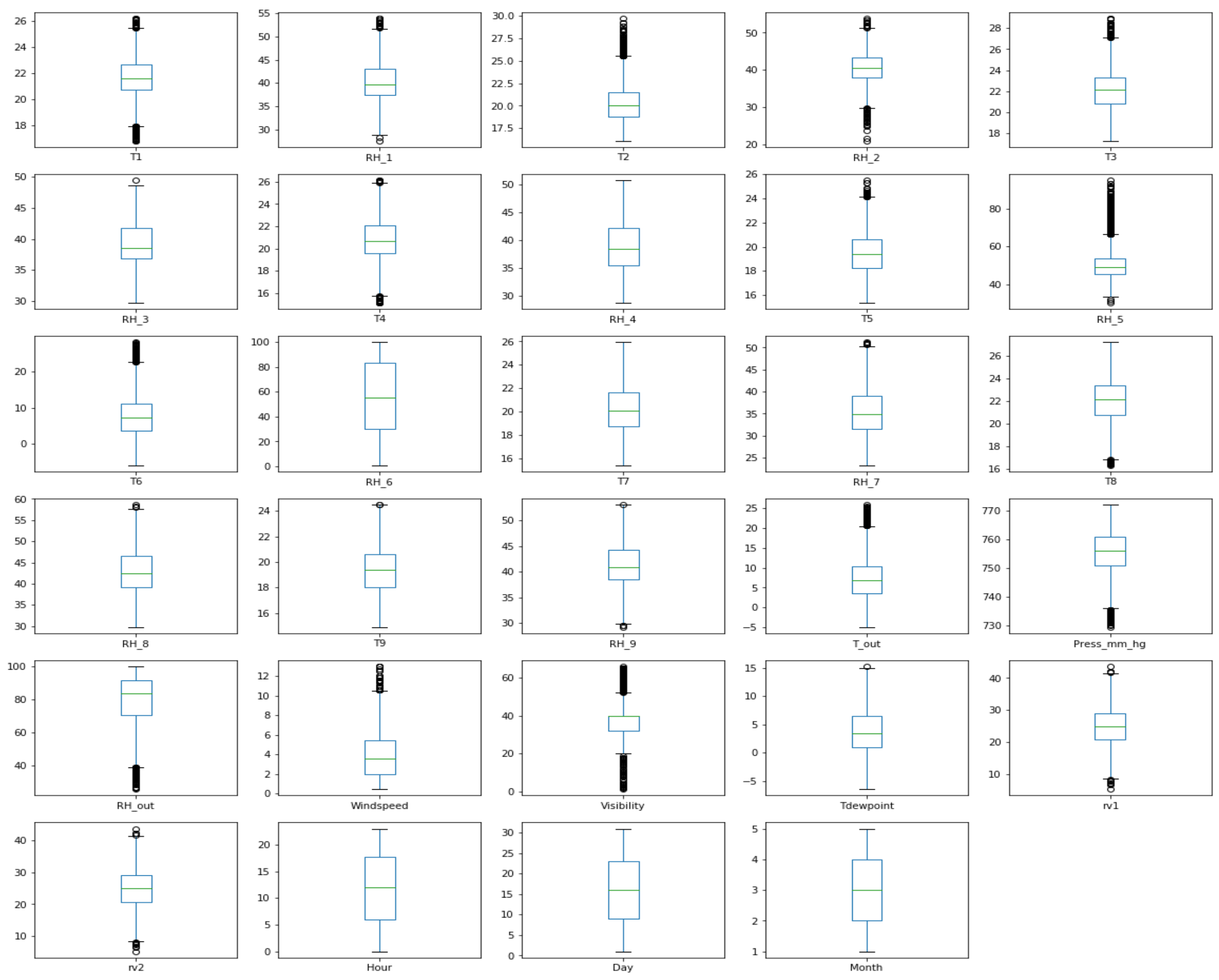
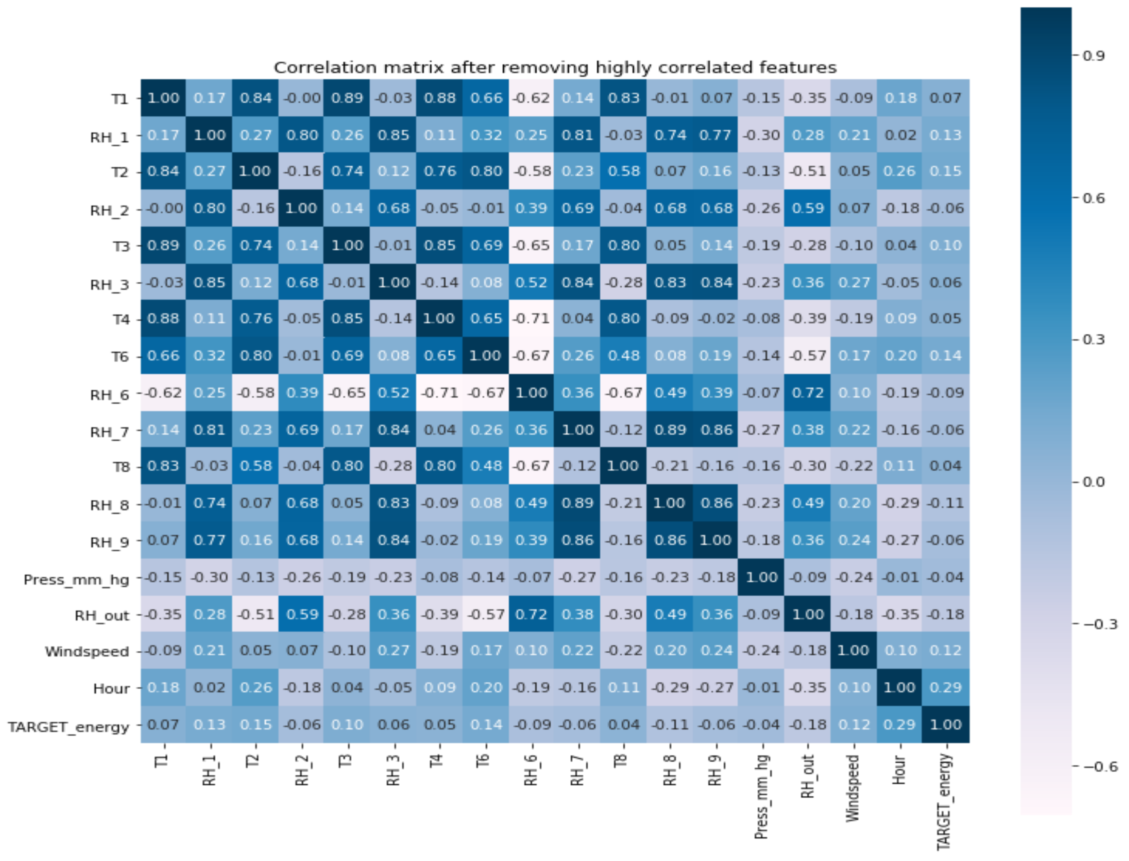


Figure 2: Boxplot of features

One more thing to explore while building a regression model is to check for correlation between all the features. If a very high correlation is found between two variables, removing one of them results in more robust model. To learn the correlations, I have plotted the heatmap of features as shown below.

It is observed from the correlation matrix of all features that there exists a very high correlation of upto 0.98 in the data (Refer jupyter notebook for this matrix). In addition, many features are inter correlated proving the existence of multi collinearity. Therefore, before jumping onto building the model, it is important to remove such highly correlated features to improve the model robustness.

As a part of feature selection, I have used SlectKBest method to determine 20 best features out of total 29. I have used f_regression statistic to derive the features having best relationship with target variable. Further, I removed the highly correlated features from the selected 20 and plotted their correlation matrix as shown in figure3. Figure3 shows the correlation between the features selected to build our model. As per the figure, there is no high correlation between these features. Therefore, it looks fine to proceed with these features to build our regression model.



	T1	RH_1	T2	RH_2	T3	RH_3	T4	T6	RH_6	RH_7	T8	RH_8	RH_9	Press_mm_hg	RH_out	Windspeed	Hour	TARGET_energy
T1	1.00	0.17	0.84	-0.00	0.89	-0.03	0.88	0.66	-0.62	0.14	0.83	-0.01	0.07	-0.15	-0.35	-0.09	0.18	0.07
RH_1	0.17	1.00	0.27	0.80	0.26	0.85	0.11	0.32	0.25	0.81	-0.03	0.74	0.77	-0.30	0.28	0.21	0.02	0.13
T2	0.84	0.27	1.00	-0.16	0.74	0.12	0.76	0.80	-0.58	0.23	0.58	0.07	0.16	-0.13	-0.51	0.05	0.26	0.15
RH_2	-0.00	0.80	-0.16	1.00	0.14	0.68	-0.05	-0.01	0.39	0.69	-0.04	0.68	0.68	-0.26	0.59	0.07	-0.18	-0.06
T3	0.89	0.26	0.74	0.14	1.00	-0.01	0.85	0.69	-0.65	0.17	0.80	0.05	0.14	-0.19	-0.28	-0.10	0.04	0.10
RH_3	-0.03	0.85	0.12	0.68	-0.01	1.00	-0.14	0.08	0.52	0.84	-0.28	0.83	0.84	-0.23	0.36	0.27	-0.05	0.06
T4	0.88	0.11	0.76	-0.05	0.85	-0.14	1.00	0.65	-0.71	0.04	0.80	-0.09	-0.02	-0.08	-0.39	-0.19	0.09	0.05
T6	0.66	0.32	0.80	-0.01	0.69	0.08	0.65	1.00	-0.67	0.26	0.48	0.08	0.19	-0.14	-0.57	0.17	0.20	0.14
RH_6	-0.62	0.25	-0.58	0.39	-0.65	0.52	-0.71	-0.67	1.00	0.36	-0.67	0.49	0.39	-0.07	0.72	0.10	-0.19	-0.09
RH_7	0.14	0.81	0.23	0.69	0.17	0.84	0.04	0.26	0.36	1.00	-0.12	0.89	0.86	-0.27	0.38	0.22	-0.16	-0.06
T8	0.83	-0.03	0.58	-0.04	0.80	-0.28	0.80	0.48	-0.67	-0.12	1.00	-0.21	-0.16	-0.16	-0.30	-0.22	0.11	0.04
RH_8	-0.01	0.74	0.07	0.68	0.05	0.83	-0.09	0.08	0.49	0.89	-0.21	1.00	0.86	-0.23	0.49	0.20	-0.29	-0.11
RH_9	0.07	0.77	0.16	0.68	0.14	0.84	-0.02	0.19	0.39	0.86	-0.16	0.86	1.00	-0.18	0.36	0.24	-0.27	-0.06
Press_mm_hg	-0.15	-0.30	-0.13	-0.26	-0.19	-0.23	-0.08	-0.14	-0.07	-0.27	-0.16	-0.23	-0.18	1.00	-0.09	-0.24	-0.01	-0.04
RH_out	-0.35	0.28	-0.51	0.59	-0.28	0.36	-0.39	-0.57	0.72	0.38	-0.30	0.49	0.36	-0.09	1.00	-0.18	-0.35	-0.18
Windspeed	-0.09	0.21	0.05	0.07	-0.10	0.27	-0.19	0.17	0.10	0.22	-0.22	0.20	0.24	-0.24	-0.18	1.00	0.10	0.12
Hour	0.18	0.02	0.26	-0.18	0.04	-0.05	0.09	0.20	-0.19	-0.16	0.11	-0.29	-0.27	-0.01	-0.35	0.10	1.00	0.29
TARGET_energy	0.07	0.13	0.15	-0.06	0.10	0.06	0.05	0.14	-0.09	-0.06	0.04	-0.11	-0.06	-0.04	-0.18	0.12	0.29	1.00

Figure 3: Correlation matrix

Model Building and Evaluation

For this project goal, I have implemented total 3 Supervised models:

1. **Random Forest Regression Model**

As shown in the figure1 of feature plots, there exists no linear relationship between dependent and independent variables. Therefore, it is a good idea to build a tree-based regressor for energy usage prediction.

Random forest model is an ensemble-learning model that works on the concept of bagging. It builds multiple decision trees and merge their results together to determine the final prediction that is more accurate and stable. While building the model, these decision trees do not interact with each other and provides their own independent result. Here, I have used various parameter tuning as shown below with cross validations to identify the best parameters:

- Bootstrap
  It is a Boolean value. If the value is TRUE, it uses different samples of the training data to build multiple trees. Else, it uses the whole training dataset for all trees.
- max_features
  It gives the number of features to look for while deciding on the best split. If the value is 'auto', it uses all available features. If value is 'sqrt', then it takes 'max_features' value as 'sqrt(total no. of features)'. If 'log2' then it takes 'max_features' value as 'log2(total no. of features)'.
- criterion
  Function given as a criterion measures the quality of a split. It takes two values either 'mse' or 'mae'.
- max_depth

This parameter takes integer values to limit the depth of the tree up to certain point.

To evaluate the performance of the model, I have calculated the train set and validation set accuracies. From the grid search cross-validation results, the train and validate accuracies received are 1.0 and 0.34 respectively. That shows the built model is 100% accurate for the train set data, while it performs poorly over validation set.

Testing this model with unseen data gives the MAE of 40.80 and R-squared value of 0.24. The validation curve and learning curve for this model are plotted below:

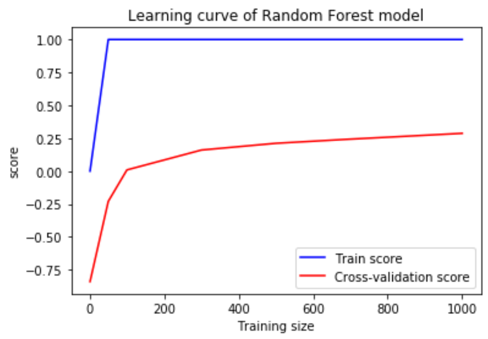


Figure 4: Random Forest learning curve

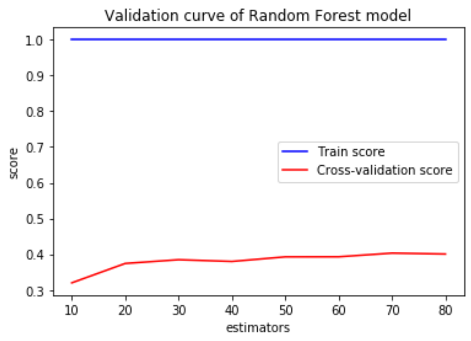


Figure 5: Random Forest validation curve

The learning curve shows that increasing train size can initially make validation set predict more general results but does not help much after a certain point. Adding more train data is unlikely to help here. Checking the validation curve confirms the huge gap between train and validation scores. This is the case of overfitting. In our scenario, it could be the result of white noise present in our time series data. Random Forest regressor is unlikely to help predict reliable energy usage values for our project.

So next, what if we try a model that interacts with other trees and learn from their mistakes? That is the concept of boosting model.

2. Gradient Boosting Regression Model

As mentioned above, Gradient boosting model builds new trees based on the previously built trees to improve the fit statistics. New trees correct any errors (residuals for regression) or mistakes from the previous trees. It continues tree-building process until it reaches the limit specified or the additional trees fail to improve the fit.

When a new tree is added into the model, the residual rate lowers for the test data points. Scaling the trees with a learning rate for deciding the final prediction prevents them from the issues of high bias.

I have performed following parameter tunings to derive the best model:

- loss
  The value of loss refers to the loss function to be optimized in each split. The default value 'ls' refers to the 'least squares' function. 'lad' refers to 'least absolute deviation' and is robust when input variables are ordered. 'huber' is a combination of these two.
- max_features
  This parameter is similar to the one in Random Forest model. Higher values of 'max_features' can lead to overfitting of the model.
- criterion
  This parameter is also similar to the 'criterion' in Random Forest. However, for Gradient boosting it accepts an additional value of 'friedman_mse', for the mse with improvement score developed by Friedman.
- learning_rate

The contribution of each tree is shrinked by the learning_rate. As mentioned above, the predicted value (for regression) of each tree is multiplied by a learning_rate to prevent the case of high bias.

After performing these parameter tuning and cross-validations for Gradient Boosting regressor, I received the train set and validation set accuracies of 0.59 and 0.34 respectively. It shows that there is no much difference between these two.

Testing this model with unseen data set, gives the MAE of 40.16 and R-squared value of 0.31. The validation and learning curve for this model are plotted here:

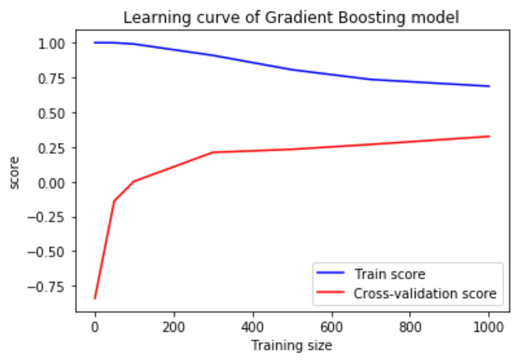


Figure 6: GBM learning curve

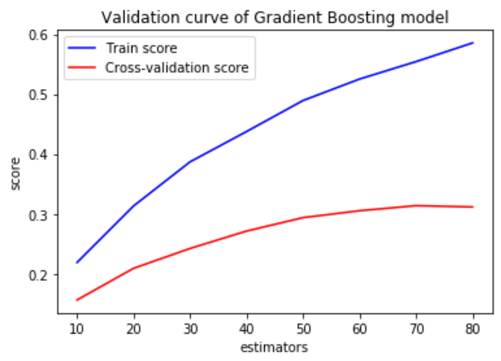


Figure 7: GBM validation curve

The learning curve shows that increasing train size can help converge both the curves resulting in a good fit. As observed from the validation curve, both training and validation score improves with increasing estimators. Also, the gap between train and validation score is not much higher as received in Random Forest model.

3. LSTM Recurrent Neural Network

I chose to work with RNN over ANN as the former has the structure where it can learn the previous input along with the current input while predicting the target variable at time 't'. RNN alone cannot hold onto the long-term memory. Therefore, I am using LSTM (Long Short Term Memory) with RNN.

LSTM is a variant of recurrent neural network that helps to predict sequence type of data such as time series prediction. Here I have used Bidirectional LSTM, as it looks a better approach when model can learn from both forward and backward sequences and can concatenate both interpretations to predict the energy usage. I have performed following parameter tunings to come up with the best neural network model:

- neurons
  It shows the number of nodes in a hidden layer. Higher no. of nodes makes the network more powerful. However, it requires more time to learn as the no. of parameters also increases.
- activation
  Activation functions are useful for neural networks as they help to learn complex pattern in a data.
- lr
  lr is a parameter of 'Adam' optimizer. It shows the learning rate or step size. Higher learning rate speeds up the learning process at initial stage while lower values slow down the training process. Default value is 0.001.

I have specifically used 'Adam' optimizer as it uses past gradients to speed up the learning process being suitable for Time Series prediction problems. An 'early stop' is also introduced to save time while running epochs.

The trained neural network model showed an MAE of 33.44, which is better than Random Forest and Gradient Boosting. It gave a prediction accuracy of 0.35 on unseen data.

Below loss curve gives the snapshot of model training process:

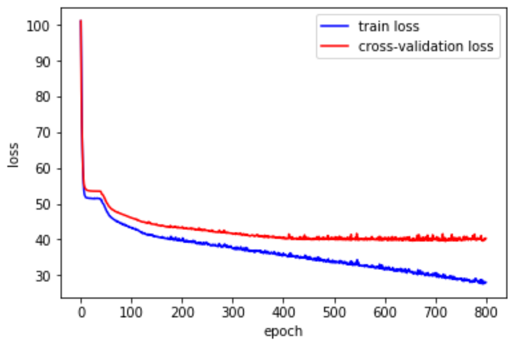


Figure 8: Loss curve for LSTM RNN model

It shows a good learning rate with increasing epochs. Also, the train and validation loss stays almost equal justifying the model is just right.

Conclusion

Below table summarizes the evaluation of all three models:

Model Name	Train Accuracy	Validation Accuracy	Mean Absolute Error
Random Forest Model	1.0	0.34	40.80
Gradient Boosting Model	0.59	0.34	40.16
LSTM RNN Model	0.52	0.35	33.44

As you can observe clearly from the table that Random Forest suffers overfitting while training the model. Therefore, it cannot be used to predict energy usage in real-time practice as it fails to generalize the prediction results. Gradient boosting model performs better in comparison to RF model, though both models give the similar MAE values, GBM overcomes an issue of overfitting by predicting more general results up to certain point. A scatter plot of predicted vs observed values for GBM model shows that up to some point it performs pretty well confessing low variance. As mentioned above, increasing the train size may overcome this issue. LSTM RNN model does really improve the MAE score for this data.

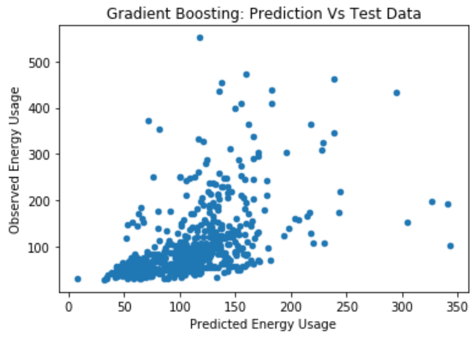


Figure 9: GBM predicted vs observed

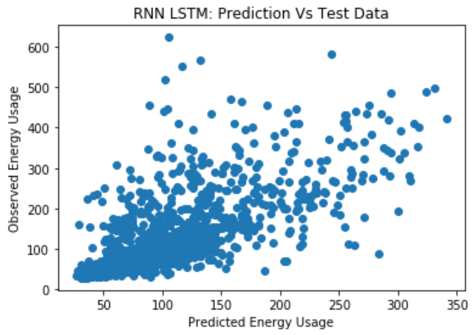


Figure 10: LSTM RNN: predicted vs observed

Also, the plotting confirms that it can predict many correct values over unseen data. Though there still exists some incorrect predicted values, it is more reliable than both the models created earlier as it carries high variance and comparatively low bias.

References

- Luis M. Candanedo, Veronique Feldheim, Dominique Deramaix, Data driven prediction models of energy use of appliances in a low-energy house, Energy and Buildings, Volume 140, 1 April 2017, Pages 81-97, ISSN 0378-7788
- ASHRAE -Start Here: A GENTLE Introduction. Kaggle.com. (2019). Retrieved 29 May 2020, from https://www.kaggle.com/caesarlupum/ashrae-start-here-a-gentle-introduction.
- Jain, A. (2016). Guide to Hyperparameter Tuning in Gradient Boosting (GBM) in Python. Analytics Vidhya. Retrieved 29 May 2020, from https://www.analyticsvidhya.com/blog/2016/02/complete-guide-parameter-tuning-gradient-boosting-gbm-python/.
- Brownlee, J. (2020). How to Develop LSTM Models for Time Series Forecasting. Machine Learning Mastery. Retrieved 29 May 2020, from https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/.
- Wierenga, R. (2019). An Empirical Comparison of Optimizers for Machine Learning Models. Medium. Retrieved 29 May 2020, from https://heartbeat.fritz.ai/an-empirical-comparison-of-optimizers-for-machine-learning-models-b86f29957050.
- 3.2.4.3.6. sklearn.ensemble.GradientBoostingRegressor — scikit-learn 0.23.1 documentation. Scikit-learn.org. Retrieved 29 May 2020, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html.
- Optimizing Hyperparameters for Random Forest Algorithms in scikit-learn. Medium. (2019). Retrieved 29 May 2020, from https://medium.com/@ODSC/optimizing-hyperparameters-for-random-forest-algorithms-in-scikit-learn-d60b7aa07ead.