Machine Learning Enabled Smart Home

Deirdre Reilly

Abstract— Advances in technology are increasingly being incorporated into consumer's homes and changing people's lives. Users are expecting more from technology than just the simple automation of routine tasks, instead they are now envisaging devices that predict their next move. For home automation systems, users want their home and work environments proactively analysed managed as a human would. This paper describes a Smart Home automation system that uses Machine Learning to predict appropriate lighting settings for a home, based on past user action datasets as well as outside factors such as weather.

Included in this paper are analyses of data collection mechanism, home controller ecosystems, application data transmission protocols, cloud computing environments, and Machine Learning algorithms relevant to IoT systems.

I. Introduction

The Internet of Things (IoT) is enhancing the way people live, work, and play. Technology for the domestic home environment is being heavily invested in, resulting in many new IoT innovations. The current market is offering IoT applications such as Wi-Fi enabled light bulbs, Bluetooth speakers, smart televisions, and advanced automated home appliances.

With this onslaught of new technologies the consumers are increasingly seeking out intelligent management systems that convert the enormous data sets created by these devices to a series of simple actions that enhance their lives. Interactive extensible platforms such as Google Home and Amazon Alexa are at the forefront of this trend and simplistic decision coordination platforms such as IFTTT are beginning to demonstrate what the next generation of cross product coordination could look like.

However, simplistic decision tree logic (if this, then this) is not capable of handling the complexity of a modern home environment so a new approach is needed. Machine Learning and potentially Artificial Intelligence are the next frontiers in innovation for home automation. Advances in algorithms coupled with computing power gains, low cost storage (Big Data) all accessible on scalable low cost cloud based platforms offer foundational components for this next generation of Smart Home. This project examines how to put the intelligence in a Smart Home.

II. Prior Work

For this paper a Literature Review was performed to gather insights on previous research and implementation methods that have been evaluated in the field. In an IoT system there are multiple different architectural components that need assessing including: the data collection mechanisms, application communications protocols, data storage techniques, Machine Learning algorithms, and cloud computing hosting choices.

1. **IoT Data Gathering Devices –** The Raspberry Pi and the Arduino are the main experimental microcontrollers used in industry and academia. The main factors assessed included CPU performance characteristics, ability to store large datasets, and suitability of their hardware interfaces. The best solution discussed in [1], the Raspberry Pi, was selected due to its high-performance features and this was considered with the Arduino for its large selection of GPIO pins.
2. **Application Protocol Layers** – MQTT and CoAP are two popular protocols used to transfer data between IoT sub-systems. MQTT is the selected protocol for many engineers as it widely supported by cloud computing systems. MQTT also is a more robust solution as it utilizes a Publish/Subscribe communications model versus CoAP's less scalable asynchronous model.
3. **Data Storage –** The most common data storage mechanisms used in IoT project are based on NoSQL databases. IoT data can often be unstructured and frequently contains multimedia assets, thus favouring NoSQL storage mechanisms [2]. As this project offered the possibility of using structured data, an SQL database was selected thus easing future processing steps.
4. **Machine Learning Algorithms –** In Machine Learning (ML) there are two learning types called unsupervised and supervised**.** In this paper supervised learning algorithms are used to predict home control outputs including light state setting and light brightness values. The algorithm selected, Random Forest, is considered a robust learning algorithm as discussed in New method for accurate prediction of CO2 in the Smart Home [3].
5. **Cloud Computing –** In the market there are two dominating competitors Google Cloud and Amazon AWS. AWS supports MQTT, SQL and NoSQL databases, and has functional computing capabilities, all aspects that are needed to support a high-performance ML system.

III. TECHNICAL DESIGN

This paper is divided into multiple stages aligned to the overall system architecture; the data collection mechanism, data transmission & storage, data pre-processing, and Machine Learning. Figure 1 shows the high level architecture of the solution. For a more in-depth description, please see the reference architecture in Appendix C.

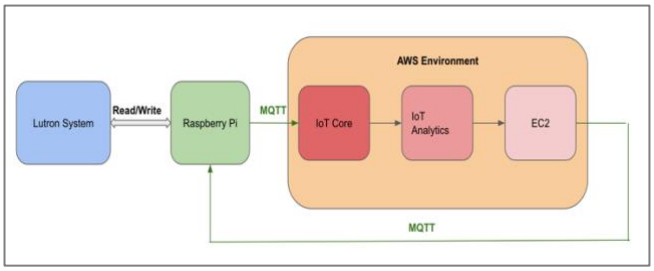

Figure 1: High Level Architecture Diagram

A. DATA COLLECTION

Lighting activity data can be secured from a user's house in multiple ways. For this paper a pre-installed home lighting control system was selected to capture room keypad actions and associated brightness levels. This paper used a Lutron system [4] to collect data as opposed to the use of discrete smart light bulbs. Using a lighting control system allows for coordinated full data collection for a building rather than the independent data gathered from smart light bulbs.

The Lutron system has a built in logging feature that enables an engineer to create a running log of real time lighting keypad actions triggered by the users. Using a Python program and a serial interface library on a Raspberry Pi the output of this log was captured reliably through a RS232 cable.

Once the user's real time actions was read into the Raspberry Pi, additional data elements needed to be added to the data structure. A timestamp was appended to ensure accurate time tracking, as well as a flag to indicate that this reading has come from the Lutron system.

B. DATA TRANSMISSION & STORAGE

Once the data from the Lutron lighting system was collected the message was transmit to the Amazon Web Services (AWS) IoT Core service for processing. MQTT is one of the simplest and most popular pub/sub models; a client subscribes to a topic through a TCP-IP service broker and another client publishes data on the topic through the broker. AWS supports secure MQTT connections between registered IoT devices. When data are published the broker ensures all subscribers receive the messages enabling distributed synchronicity between all devices.

The Raspberry Pi was set up as a publisher/subscriber client, with AWS being the broker. The Raspberry Pi then communicated to AWS using JSON formatted messages using MQTT. Once received in the IoT Core, Amazon has multiple services available to perform enrichment actions on the data received. One of these features is a SQL database storage option available via IoT Analytics. IoT Analytics created an SQL database to store MQTT messages received. These messages were then read into an EC2 Jupyter notebook instance to perform pre-processing on the raw dataset.

C. FEATURE ENGINEERING

Feature Engineering is used to create a strong and simple relationship between input and output fields. A feature is the ML term for a classical data field. The features identified and defined for the model will have a major influence over the outcomes of the model's predictions. The quality and quantity of features heavily influence the accuracy of the model. The model outcome required was a prediction of a i) room needing a lighting action and ii) a corresponding brightness level.

Many different factors influence when a room needs actioning in a household. The main features identified that have an impact on the room and lighting levels in the household are time of day, time of year, past actions, and external factors such as weather.

One of the most challenging features to utilize in a Machine Learning model is date and time. There are many ways to represent date and time features, such as:

- Binary feature (is weekend, is holiday)
- Categorical feature (Day of Week, Week of Year)
- Quantitative feature (Minutes elapsed for the day)

In this paper to represent date and time a mixture of categorical and quantitative features were selected. Each of these features was analytically tested for feature importance, to determine the ranking relevance the feature to the model's performance. The final time and date features selected were:

- Minutes elapsed for a day
- Week number (0-51)
- Day Number (0-6)
- Day Label (Morning, Evening, Night, Day)

These were chosen as they offered a mixture of categorical and quantitative parameters for the model. They represented the importance of time of day to a user's actions and addressed seasonality changes to preferences. In winter the number of hours of sunlight is shortened, hence there will be a change in output behaviour compared to summer time.

As well as date and time having an significant effect over the lighting in a user's home, external factors such as weather also have an impact on lighting outputs. Using a standard weather report the most available features that relate to lighting changes are temperature, cloud amount, sun duration, and rain fall. After feature importance testing, cloud amount and temperature were discovered to have the most significant impact on overall lighting outputs. The full testing of feature importance is explained in Appendix D.

D. PRE-PROCESSING

Without pre-processing there was potential for significant distortion of the insights due to parameter error or bias. The data originated from the Lutron system and was not expressed in a way that supported Machine Learning model building.

The first step in remediating this problem was manipulation of the raw data and filtering out irrelevant data.

	message	timestamp	unique_key
0	KBP, [01:05:08], 5	13/06/2018, 01:00:47	13/06/2018, 01:00:47180
1	DL, [01:01:00:03:04], 0	13/06/2018, 01:00:48	13/06/2018, 01:00:48181
2	DL, [01:01:00:03:03], 0	13/06/2018, 01:00:50	13/06/2018, 01:00:50182

Figure 2: Raw Input from Lutron

When sending data from the Raspberry Pi to AWS there were errors introduced by breaks in communication or hardware and software errors reading from the serial port. In the project data set of 20,237 entries, 2,104 entries contain null values (10%). Missing values in this case were discarded as many of these are initialization errors from the Lutron system, as well as split messages caused by transmission errors. As there was no way to impute the values they were removed from the data structure.

	Output	DateTime	Action	Location	Attribute
7	AT&F1E0F1Q1S0=1M0&C0&D0	16/05/2018, 03:02:35	AT&F1E0F1Q1S0=1M0&C0&D0	None	None
8	AT&F1E0F1Q1S0=1M0&C0&D0	16/05/2018, 03:02:39	AT&F1E0F1Q1S0=1M0&C0&D0	None	None
14	AT&F1E0F1Q1S0=1M0&C0&D0	18/06/2018, 12:38:55	AT&F1E0F1Q1S0=1M0&C0&D0	None	None
38	100	03/05/2018, 01:27:40	100	None	None
88	AT&F1E0F1Q1S0=1M0&C0&D0	15/06/2018, 01:23:45	AT&F1E0F1Q1S0=1M0&C0&D0	None	None
89	AT&F1E0F1Q1S0=1M0&C0&D0	15/06/2018, 01:23:48	AT&F1E0F1Q1S0=1M0&C0&D0	None	None

Figure 3: Sample Error Messages

The timestamp feature needed to be manipulated from the original DateTime python object into individual time features for processing. The DateTime object in python allows for simple date manipulation and this was used to extract weekday number as well as week of the year. Manipulating the string timestamp created a separation for hour and minute. Using these features the total minutes since midnight was calculated.

An additional feature was added to address data seasonality. By importing the sunrise and sunset times for each day of the year the time of day was split into four nominal categories (Day, Night, Morning, Evening).

Additional features were needed to create a robust model. A room location label and weather data were merged with the existing data structure. A user predefined data file containing the room labels and assigned unique identifier was merged into a DataFrame to give a categorical numerical feature for each room. In addition, a csv file containing historical weather feature data was merged with the DataFrame using the timestamp as the unique identifier. Figure 4 shows the full DataFrame ready to be fed into the model.

Attribute	Num_Week	weekday	Minutes	temp	cloud_amt	Floor	RoomLabel	Day_Label
0.0	18	0	1177	8.6	2.0	0.0	23.0	2
100.0	17	4	1042	9.4	2.0	1.0	8.0	1
100.0	18	2	430	6.7	7.0	1.0	1.0	0

Figure 4: Final DataFrame for Model

E. Random Forest Machine Learning Models

There are two supervised models used in this paper, the first predicts the room to be actioned, the second predicts the brightness level for the actioned room. Both models used a Random Forest Classification algorithm. A Random Forest is an algorithm that fits numerous decision tree classifiers on subsets of the dataset; this is then averaged to improve the predictive accuracy and control over-fitting. In Machine Learning overfitting is a major problem that occurs when the output states are so dependent on the input training values that the model becomes unreliable with unseen inputs. However, a Random Forest model will not overfit if enough trees are defined.

To generate the Random Forest model the number of trees (n_estimators) parameter needed to be determined. To select the number of trees, a series of tests were performed to select the optimal number of trees vs performance.

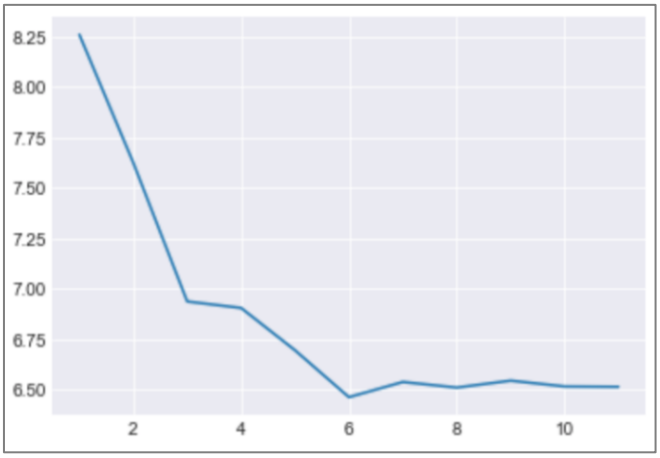

Figure 5: RMSE vs number of features

To ensure the model designed was performing accurately a series of accuracy metrics were developed. A simple accuracy score was used to measure the percentage of correct classifications against the test set. An Out of Bag (OOB) score was used to estimate the generalization accuracy of the model. A classification report from the python sklearn library was used to generate a text report detailing classification metrics such as model precision, recall, f1-score, and support (# tested). From this report an in-depth model analysis was performed and the results used for improvement of the existing model.

Following hundreds of test runs, a model was designed that produced high accuracy scores of 83% for room classification and 76% for brightness classification. Details are provided in Appendix D.

F. Real Time Implementation

To produce a real time prediction of both room and brightness settings real time data needed to be incorporated. Current date and time features were captured in a timing loop and generated every minute. The weather parameters were captured through the World Weather Online API [5]. These new real time parameters were then used to predict the room to action. The selected room was then fed into a new DataFrame that was used to predict the room brightness level.

In the AWS environment MQTT was used to publish these actions to the Raspberry Pi connected to the Lutron serial port. These messages were written to the Lutron system enabling a real-time lighting control system.

IV. Results

The results are divided into two sections; accuracy predicting actionable room and accuracy predicting room brightness level. Both models use Random Forest classification. To evaluate the performance of each model, the data was divided 70/30 into training and test data sets. The performance results were calculated from the test and train data sets to identify if there was over or underfitting.

For classification there are different industry metrics used to capture the accuracy of the model [8]. Six different methods were selected: out-of-bag score, accuracy, recall, precision, F1-score, and a confusion matrix.

"The out of bag is the mean prediction error for each training sample (x), using only the trees that did not have (x) in their bootstrap sample" [6]. In the room classification it produced a positive score of 81% which is highly accurate.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Recall = \frac{TP}{TP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$F_1 = \frac{2(\,Precision * Recall)}{Precision + Recall}$$

A. Room Classification

The room classification model results are shown in Figure 5. The training accuracy score was measured at 90% and the test accuracy at 82%. This show there was a slight overfitting in the final model, however this does not cause any concern.

The classification report breaks down the precision, recall, and f1-score for each of the 23 output categories. Showing the performance of each category visualizes if there are any external factor affecting the results.

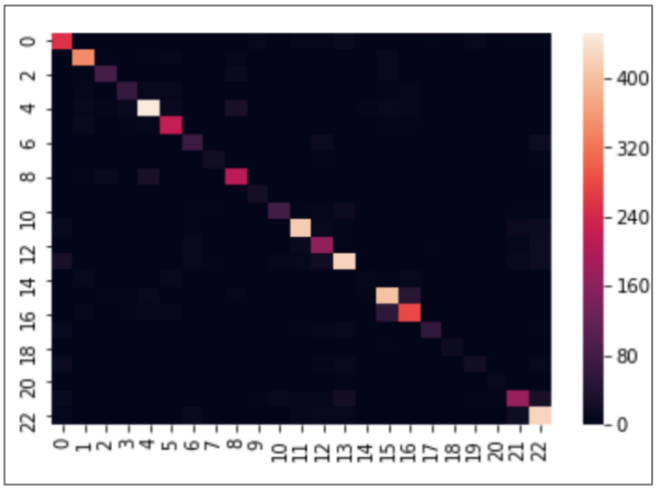

Figure 6: Room Classification Heat Map

```
Out-of-bag score estimate: 0.813
Train Accuracy:
0.9058122955631972
Test Accuracy:
0.824853228962818
             precision    recall  f1-score   support

        0.0       0.76      0.86      0.81       296
        1.0       0.89      0.94      0.92       365
        2.0       0.80      0.79      0.79       105
        3.0       0.79      0.68      0.73        92
        4.0       0.89      0.85      0.87       530
        5.0       0.85      0.90      0.88       242
        6.0       0.67      0.67      0.67       101
        7.0       0.72      0.84      0.78        25
        8.0       0.80      0.82      0.81       255
       10.0       0.67      1.00      0.80        24
       11.0       0.86      0.71      0.77       112
       12.0       0.92      0.89      0.90       468
       13.0       0.71      0.78      0.74       213
       14.0       0.84      0.82      0.83       516
       15.0       0.44      0.14      0.21        29
       16.0       0.83      0.85      0.84       476
       17.0       0.79      0.79      0.79       351
       18.0       0.88      0.67      0.76        88
       19.0       1.00      0.83      0.91        18
       20.0       0.74      0.43      0.55        53
       21.0       0.67      0.73      0.70        11
       22.0       0.79      0.68      0.73       258
       23.0       0.80      0.88      0.84       482

avg / total       0.83      0.82      0.82      5110
```

Figure 7: Room Classification Report

B. Brightness Classification

The brightness classification model results are shown in Figure 8 & 9. The brightness classification was divided into 5 categories (0, 30, 50, 70, 100). The 0 and 100 categories are the highest performing with 80% and 76% accuracy accordingly; these correspond to the lights fully off or on.

The intermediate levels of lighting have the weakest classification with around 30% accuracy.

The training accuracy score was measured at 88% and the test accuracy at 75%. This again shows there is a slight overfitting in the model, however this does not cause any concern

```
Out-of-bag score estimate: 0.755
Train Accuracy:
0.8806508429086639
Test Accuracy:
0.7585127201565558
             precision    recall  f1-score   support

        0.0       0.79      0.82      0.80      2590
       30.0       0.38      0.34      0.36       171
       50.0       0.30      0.23      0.26        71
       70.0       0.14      0.14      0.14        21
      100.0       0.77      0.74      0.75      2257

avg / total       0.76      0.76      0.76      5110
```

Figure 8: Brightness Classification Report

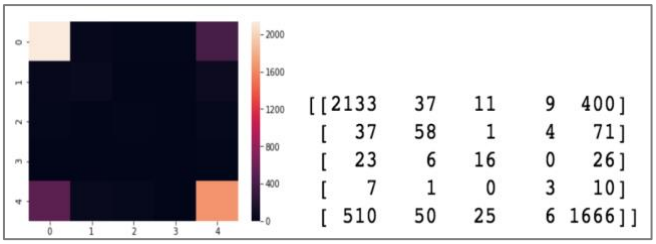

Figure 9: Brightness Classification Heat Map

V. Analysis

A. Feature Analysis

The room and brightness test results were positive, showing that the model was correctly designed given the input features. An in-depth analysis to find the correlation between input features was applied to explore the patterns that the Machine Learning algorithm identified. To visualize correlations between features a seaborn pair-plot library [7] was used. This library displays the 2-D relationship between features.

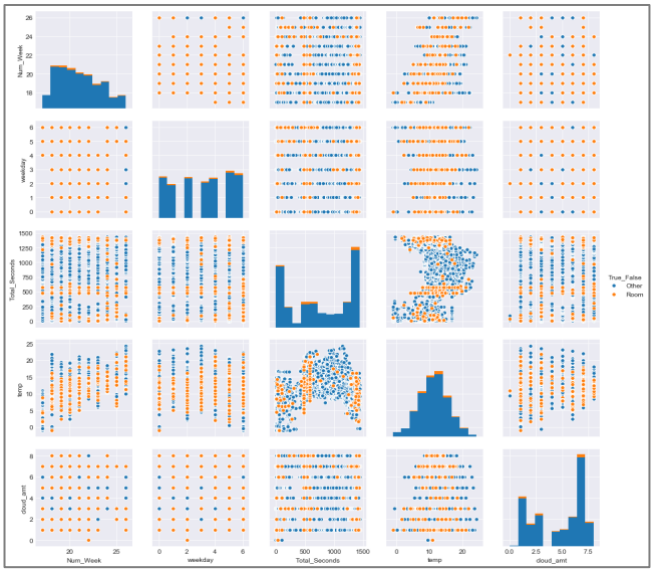

Figure 10: Room Classification Pair plot

The above pair plot shows the room selection analysis for the living room. In the plot above it shows a clustering between time of day and weekday features. It shows the lights triggered for this room happen every day around 8:30am and show a second cluster around midnight every night. The room and brightness pair plots can be seen in Appendix D in the paper.

B. Room Model Analysis

Predicting the room to action produced a test accuracy of 82%, which was excellent overall. However, four rooms produced disappointing results:

- Upstairs Hall (44%)
- Dining Room (67%)
- Outside Lighting (67%)
- Sun Lounge (67%)

Results for these rooms were poor because the rooms were rarely used and thus have minimal samples for training. In the test set each of these categories had less than 100 sample size out of 5110 (~2%).

Considering the data collected was only for a duration of ten weeks this model could be improved with a full year's data.

Feature analysis was run to determine the most impactful features driving the model. The output of this analysis can be seen in Figure 11.

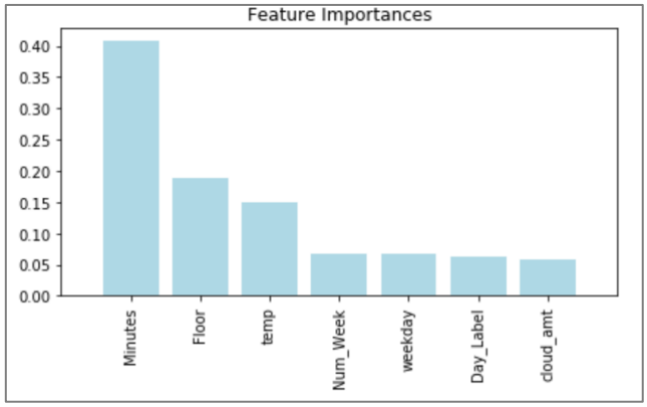

Figure 11: Room Feature Importance's

C. Brightness Model Analysis

Predicting the brightness level in a room, produced a test accuracy of 76%. The brightness categories 0 and 100 (OFF and ON) have the best performance with 80% and 76% accuracy. In previous testing a binary classification produced 80% accuracy. These results are documented in Appendix D.

The intermediate levels of lighting have the weakest performance with an average of 30% accuracy. This was due to the scarcity of samples for those categories. The overall sample size of 30%-70% brightness level contributes to 5% of the overall data. To increase the sample size, an energy saving policy could be written to decrease the lighting level in each room until the user intervenes. This would increase the sample size for the intermediate categories and the model would learn from these new inputs. These new outputs could be used to drive a home energy savings program.

The feature analysis for the brightness model is similar to the room model. The minutes feature is the leading feature in driving the brightness output. The room feature also attributes to 10% of the feature importance. Over time the week and day categories will increase in importance in response to seasonality changes in the models.

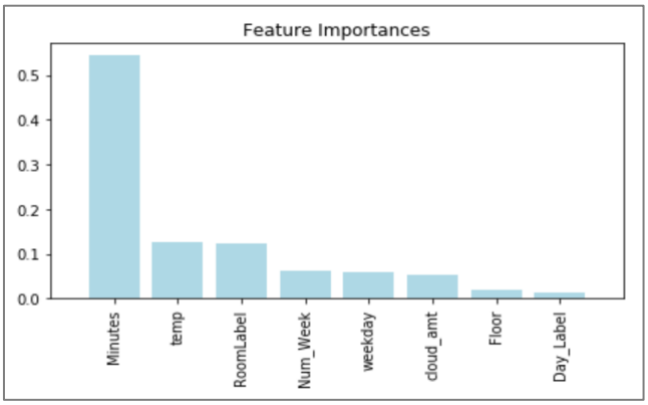

Figure 12: Brightness Feature Importance's

VI. Future Work

The project has shown that the various sub components necessary to make an effective Machine Learning enabled home lighting management system viable. Possible future stages to the project could include:

.

1. **Full year dataset**: The data was collected from April to June, leaving three quarters of the data not collected. The full years data would reflect the patterns in seasonality.
2. **Added user location**: With the advancements in geolocation in mobile devices, it has become easier to track user's location. Adding a feature to report which users are in the building, could help in deciding which rooms to activate. However, this can also be considered an invasion of privacy. With data privacy becoming a more prominent topic in society, an engineer has an obligation to keep the anonymity of a user's data.
3. **Energy Saving:** This paper focuses on predicting the lighting settings in a house based on lighting system past performance. To add a level of energy saving to the project a post actuation program can be implemented after the prediction has been made. The lighting brightness can be decreased by 10% until the user intervenes. The Machine Learning system would then learn appropriate adjustment levels.
4. **Abstraction**: Genericize the input and output data processing so the system works with environments other than Lutron.
5. **Web Service**: Offer the Machine Learning home controller as a web service available to anyone with smart lighting.

As the core model is not dependent on the lighting system used the above possibilities are relatively less complex to deliver now that the core Machine Learning model is accurate.

VII. Conclusion

This paper has shown that Machine Learning based home lighting automation is possible, accurate, and reliable. The paper has further shown that Random Forest supervised learning models can generate high (>80%) accuracy decisions with training data sets available from user actions and public data. The models explored in this paper are enabled by data from an entire building and take consideration of external variables such as weather in forming their predictions.

Building a complex closed loop system is possible using Open Source libraries and cloud based computing services even though the end devices are behind firewalls in a user's home. In summary the project demonstrates that users can look forward to increasingly complex home automations based on Machine Learning enabled controllers.

VIII. References

[1] Kumar, P. and Umesh Chandra Pati (2016). Arduino and Raspberry Pi based smart communication and control of home appliance system. 2016 Online International Conference on Green Engineering and Technologies (IC-GET).

[2] Rautmare, S. and Bhalerao, D. (2016). MySQL and NoSQL database comparison for IoT application. 2016 IEEE International Conference on Advances in Computer Applications (ICACA).

[3] Vanus, J., Martinek, R., Bilik, P., Zidek, J., Dohnalek, P. and Gajdos, P. (2016). New method for accurate prediction of CO2 in the Smart Home. 2016 IEEE International Instrumentation and Measurement Technology Conference Proceedings

[4] Lutron.com. (2018). [online] Available at: http://www.lutron.com/europe/Pages/default.aspx [Accessed 10 Apr. 2018

[5] WorldWeatherOnline.com. (2018). World Weather Online. [online] Available at: https://www.worldweatheronline.com/. [Accessed 17 Jul. 2018].

[6] Scikit-learn.org. (2018). OOB Errors for Random Forests — scikit-learn 0.19.2 documentation. [online] Available at: http://scikit-learn.org/stable/auto_examples/ensemble/plot_ensemble_oob.html [Accessed 14 Jul. 2018].

[7] Seaborn.pydata.org. (2018). seaborn.pairplot — seaborn 0.9.0 documentation. [online] Available at: https://seaborn.pydata.org/generated/seaborn.pairplot.html [Accessed 18 Aug. 2018].

[8] Raschka, S. and Olson, R. (2016). Python machine learning. Birmingham: Packt Publishin