Human Activity Recognition

1st Daksh Shah
Department of ECE
Stevens Institute of Technology
Jersey City, NJ, USA
dshah94@stevens.edu

2nd Parth Sharma
Department of ECE
Stevens Institute of Technology
Jersey City, NJ, USA
psharm28@stevens.edu

3rd Zheng Duan
Department of ECE
Stevens Institute of Technology
Jersey City, NJ, USA
zduan6@stevens.edu

***Abstract*—We establish a human activity recognition system utilizing various ML method/algorithm on ambient sensor data collected from 30 different homes. In our first iteration, we performed EDA (Exploratory Data Analysis) and base model fitting operation after Data Stitching and Data Pre-processing. Our EDA shows a strong correlation between between certain features in which human activities occur and our base model (Decision Tree) shows an accuracy of 71% on train and test samples.**

I. Introduction

Human Activity Recognition (HAR) is the process of interpreting human movement using ambient sensor readings. The objective is to build a ML model to classify these human movements into some of the daily activities. These activities include sleeping, washroom, eating, entering or leaving home, getting reading, prepping for meal, working, TV, entertaining guests or other activities. The quantity of the data and our multi-classification problem puts a burden on the computing power and designing of the models. In our study, we had to find the optimal way to process our data which was quite challenging. We have around 10M data points with 35 features in total.

This is a challenging problem because there is no obvious or direct way to relate the recorded sensor data to a specific human activity and each object may perform activities with significant variation, resulting in changes in the recorded sensor data. The goal is to record object-specific sensor data and corresponding activities, fit a model from that data, and generalize the model to classify the activities of new unseen objects based on their sensor data.

II. Our Solution

A. Description of Dataset

This dataset represents ambient data collected in homes with volunteer residents. Data are collected continuously while residents perform their normal routines. Ambient PIR motion sensors, door/temperature sensors, and light switch sensors are placed throughout the home of the volunteer. The sensors are placed in locations throughout the home that are related to specific activities of daily living that we wish to capture.

Following are the variables and their description:
1. lastSensorEventHours: Hour of the day
2. lastSensorEventSeconds: Seconds since midnight
3. lastSensorDayOfWeek: Integer day of the week
4. windowDuration: Time duration of the 30 event sliding window in seconds.
5. timeLastSensorEvent: Seconds since the last sensor event.
6. prevDominantSensor1: Dominant sensor ID from the previous window.
7. prevDominantSensor2: Dominant sensor ID from the second previous window.
8. lastSensorID: lastSensorID
9. lastSensorLocation: Last sensor location ID in the window.
10. lastMotionLocation: Last motion sensor location ID in the window, can be -1 if none within the sliding window.
11. complexity: Complexity or entropy in sensor counts.
12. activityChange: Change in activity levels between 2 halves of the sliding window, bisected temporally.
13. areaTransitions: Number of transitions between major sensor locations in the window.
14. sensorCount-Location: The weighted count of this sensor, starting at 1.0 for the most recent event each sensor event previous is worth n-0.01 the current event.
15. sensorElTime-Location: The number of seconds since this sensor was last seen, up to a maximum of 86400.

B. Data Compilation

The data for numerical and categorical features are captured separately. To merge the data sets, we have to find the data points which were not captured in the other data set and exclude them from our analysis. Since de didn't have any primary key to merge the data sets, we used the date-time columns to find the common data points in both the pool.It was challenging to do so since the Numerical dataset didn't have the date column to identify the day and we had to use a logic to extract it using the categorical dataset. Then selecting the ones whose date-time counts match in both the datasets.

C. Data Processing

We then carried out the standard data pre-processing which includes:
1. Checking for duplicates and missing values
2. Treating outliers of some numerical variables: Limiting the required variables at their 97.5% percentile value.
3. Label encoding the categorical features: The model does not read string formats.
4. MinMax scaling the numerical features: To bring all the variables in one scale.
5. Splitting the data into train, test and validation using stratified sampling to maintain the target proportion.

D. EDA

After doing our EDA on the dataset, we have found that certain target variables co-relate to specific areas of the house. For e.g., prepping for meal co-relates to Kitchen sensor count, Washroom activities (Bathing, Toilet, etc) co-relates to Washroom sensor count and so on.
We can also see co-relation of hour of the day vs the activities. For e.g., sleep activity is mostly observed from 10 pm to 10 am and the opposite for working. We can also we prepping for meal and eating, and getting ready and washroom go hand in hand.
We can also see some of the messages from a sensor type is also co-related to specific set of activities. For e.g., change in temperature reading is co-related to sleeping, increase in Light and Motion sensor count is co-related to washroom
All of the above analysis can be found in the plots at the end of this document.

E. Machine Learning Algorithms

Following are the ML models we have used to solve our problem:
* Decision Tree: Tree based model would be better given the complexity of the features in data.
* Random Forest: Because its an Horizontal Ensemble of Decision Tree.
* OnevsOne: Used for Multi-label Classification.
* OnevsRest: Used for Multi-label Classification.
* XGBoost: Because its an Vertical Ensemble of Decision Tree.

III. Implementation Details

A. Decision Tree

A decision tree is a flowchart-like structure in which each internal node represents a test on a feature (e.g. whether a coin flip comes up heads or tails) , each leaf node represents a class label (decision taken after computing all features) and branches represent conjunctions of features that lead to those class labels. The paths from root to leaf represent classification rules. We have implemented Decision Tree as a base model to a random training sample. Following are the classification reports:

Training Report:

Eat	1.00	1.00	1.00	24980
Enter_Home	1.00	1.00	1.00	9148
Entertain_Guests	1.00	1.00	1.00	106237
Getting_ready	1.00	1.00	1.00	82266
Leave_Home	1.00	1.00	1.00	12054
Other	1.00	1.00	1.00	225404
Prep_for_meal	1.00	1.00	1.00	201787
Sleep	1.00	1.00	1.00	62950
TV	1.00	1.00	1.00	57305
Washroom	1.00	1.00	1.00	138908
Work	1.00	1.00	1.00	78961
accuracy			1.00	1000000
macro avg	1.00	1.00	1.00	1000000
weighted avg	1.00	1.00	1.00	1000000

Testing Report:

Eat	0.70	0.71	0.71	14347
Enter_Home	0.83	0.83	0.83	5211
Entertain_Guests	0.73	0.73	0.73	61063
Getting_ready	0.74	0.75	0.75	47057
Leave_Home	0.56	0.57	0.57	6875
Other	0.72	0.71	0.72	129221
Prep_for_meal	0.91	0.92	0.92	116183
Sleep	0.88	0.88	0.88	36237
TV	0.76	0.77	0.76	32950
Washroom	0.90	0.90	0.90	79624
Work	0.80	0.80	0.80	45421
accuracy			0.81	574189
macro avg	0.78	0.78	0.78	574189
weighted avg	0.80	0.81	0.80	574189

It seems that we have achieved an accuracy of 81% with our base model. We can also see an accuracy of 100% on training data-set which suggests the model is over-fitting. Lets use Random Forest and see if we get a similar result.

B. Random Forest

It utilizes ensemble learning, which is a technique that combines many classifiers to provide solutions to complex problems. A random forest algorithm consists of many decision trees. The ‘forest’ generated by the random forest algorithm is trained through bagging or bootstrap aggregating.It predicts by taking the average or mean of the output from various trees. Increasing the number of trees increases the precision of the outcome. Following are the classification reports:
Training Report:

Eat	1.00	1.00	1.00	24980
Enter_Home	1.00	1.00	1.00	9148
Entertain_Guests	1.00	1.00	1.00	106237
Getting_ready	1.00	1.00	1.00	82266
Leave_Home	1.00	1.00	1.00	12054
Other	1.00	1.00	1.00	225404
Prep_for_meal	1.00	1.00	1.00	201787
Sleep	1.00	1.00	1.00	62950
TV	1.00	1.00	1.00	57305
Washroom	1.00	1.00	1.00	138908
Work	1.00	1.00	1.00	78961
accuracy			1.00	1000000
macro avg	1.00	1.00	1.00	1000000
weighted avg	1.00	1.00	1.00	1000000

Testing Report:

	precision	recall	f1-score	support
Eat	0.90	0.73	0.81	14347
Enter_Home	0.90	0.89	0.89	5211
Entertain_Guests	0.90	0.79	0.84	61063
Getting_ready	0.88	0.78	0.82	47057
Leave_Home	0.75	0.67	0.71	6875
Other	0.81	0.81	0.81	129221
Prep_for_meal	0.91	0.97	0.94	116183
Sleep	0.93	0.92	0.93	36237
TV	0.84	0.85	0.84	32950
Washroom	0.90	0.98	0.93	79624
Work	0.88	0.86	0.87	45421
accuracy			0.87	574189
macro avg	0.87	0.84	0.85	574189
weighted avg	0.87	0.87	0.87	574189

It seems that we have achieved an accuracy of 87% which is greater than Decision Tree. We can also see an accuracy of 100% on training data-set which suggests the model is still over-fitting. But it has learned better rules than decision tree. Now lets tune the parameters of Random Forest to tackle our problem of overfitting

Training Report:

Eat	0.92	0.71	0.81	33199
Enter_Home	0.90	0.91	0.90	12137
Entertain_Guests	0.94	0.82	0.88	141105
Getting_ready	0.91	0.77	0.83	108727
Leave_Home	0.83	0.73	0.78	15990
Other	0.84	0.84	0.84	298445
Prep_for_meal	0.91	0.99	0.95	267746
Sleep	0.93	0.93	0.93	83520
TV	0.85	0.87	0.86	75925
Washroom	0.89	0.98	0.93	184062
Work	0.89	0.87	0.88	104612
accuracy			0.89	1325468
macro avg	0.89	0.86	0.87	1325468
weighted avg	0.89	0.89	0.89	1325468

Testing Report:

Eat	0.86	0.63	0.73	14347
Enter_Home	0.87	0.88	0.87	5211
Entertain_Guests	0.88	0.71	0.78	61063
Getting_ready	0.82	0.65	0.73	47057
Leave_Home	0.70	0.64	0.66	6875
Other	0.76	0.77	0.77	129221
Prep_for_meal	0.88	0.97	0.92	116183
Sleep	0.91	0.90	0.90	36237
TV	0.79	0.80	0.80	32950
Washroom	0.85	0.97	0.91	79624
Work	0.84	0.82	0.83	45421
accuracy			0.83	574189
macro avg	0.83	0.80	0.81	574189
weighted avg	0.83	0.83	0.83	574189

It seems that tuning the hyper-parameters have solved our problem of over-fitting but has a bit poorly performed on the test. This is the trade-off we were looking for. Next we use OneVsOne and OneVsRest approach to see if we improve the results.

C. OneVsRest

One-vs-rest involves splitting the multi-class data-set into multiple binary classification problems. A binary classifier is then trained on each binary classification problem and predictions are made using the model that is the most confident. We have used Random Forest as the classifier since it was a better algorithm then Decision Tree. Following are the classification reports:

Training Report:

Eat	0.92	0.71	0.81	33199
Enter_Home	0.90	0.91	0.90	12137
Entertain_Guests	0.94	0.82	0.88	141105
Getting_ready	0.91	0.77	0.83	108727
Leave_Home	0.83	0.73	0.78	15990
Other	0.84	0.84	0.84	298445
Prep_for_meal	0.91	0.99	0.95	267746
Sleep	0.93	0.93	0.93	83520
TV	0.85	0.87	0.86	75925
Washroom	0.89	0.98	0.93	184062
Work	0.89	0.87	0.88	104612
accuracy			0.89	1325468
macro avg	0.89	0.86	0.87	1325468
weighted avg	0.89	0.89	0.89	1325468

Testing Report:

Eat	0.85	0.60	0.70	14347
Enter_Home	0.88	0.86	0.87	5211
Entertain_Guests	0.88	0.70	0.78	61063
Getting_ready	0.79	0.61	0.69	47057
Leave_Home	0.69	0.60	0.64	6875
Other	0.75	0.77	0.76	129221
Prep_for_meal	0.87	0.97	0.92	116183
Sleep	0.90	0.89	0.90	36237
TV	0.78	0.79	0.78	32950
Washroom	0.84	0.97	0.90	79624
Work	0.82	0.82	0.82	45421
accuracy			0.82	574189
macro avg	0.82	0.78	0.80	574189
weighted avg	0.82	0.82	0.82	574189

It seems that the model is performing almost as same as compared to hyper-parameter tuned Random Forest on testing data set, which suggests no improvement. Now lets use OneVsOne classifier and see which approach is better.

D. OneVsOne

Like one-vs-rest, one-vs-one splits a multi-class classification dataset into binary classification problems. Unlike one-vs-rest that splits it into one binary dataset for each class, the one-vs-one approach splits the dataset into one dataset for each class versus every other class. We have used Random Forest as the classifier since it was a better algorithm then Decision Tree. Following are the classification reports:

Training Report:

Eat	0.92	0.71	0.81	33199
Enter_Home	0.90	0.91	0.90	12137
Entertain_Guests	0.94	0.82	0.88	141105
Getting_ready	0.91	0.77	0.83	108727
Leave_Home	0.83	0.73	0.78	15990
Other	0.84	0.84	0.84	298445
Prep_for_meal	0.91	0.99	0.95	267746
Sleep	0.93	0.93	0.93	83520
TV	0.85	0.87	0.86	75925
Washroom	0.89	0.98	0.93	184062
Work	0.89	0.87	0.88	104612
accuracy			0.89	1325468
macro avg	0.89	0.86	0.87	1325468
weighted avg	0.89	0.89	0.89	1325468

Testing Report:

	precision	recall	f1-score	support
Eat	0.86	0.60	0.71	14347
Enter_Home	0.88	0.86	0.87	5211
Entertain_Guests	0.87	0.69	0.77	61063
Getting_ready	0.78	0.62	0.69	47057
Leave_Home	0.68	0.60	0.63	6875
Other	0.74	0.77	0.76	129221
Prep_for_meal	0.87	0.97	0.92	116183
Sleep	0.90	0.88	0.89	36237
TV	0.78	0.77	0.77	32950
Washroom	0.84	0.97	0.90	79624
Work	0.82	0.80	0.81	45421
accuracy			0.82	574189
macro avg	0.82	0.77	0.79	574189
weighted avg	0.82	0.82	0.82	574189

The results are similar as previous ones which suggests that using these approach didn't improve results that much. Now we use XGBoost to see if boosting improves our model.

E. XGBoost

Boosting is an ensemble learning technique to build a strong classifier from several weak classifiers in series. Boosting algorithms play a crucial role in dealing with bias-variance trade-off. Unlike bagging algorithms, which only controls for high variance in a model, boosting controls both the aspects (bias variance) and is considered to be more effective.
XGBoost stands for eXtreme Gradient Boosting. It became popular in the recent days and is dominating applied machine learning for structured data because of its scalability. Following are the classification reports:
Training Report @2k epoch:

Eat	0.98	0.97	0.97	33886
Enter_Home	1.00	1.00	1.00	12424
Entertain_Guests	0.97	0.98	0.98	143875
Getting_ready	0.95	0.92	0.94	110896
Leave_Home	0.99	0.97	0.98	16310
Other	0.96	0.91	0.94	304592
Prep_for_meal	0.96	1.00	0.98	273266
Sleep	0.98	0.98	0.98	85253
TV	0.95	0.96	0.95	77452
Washroom	0.97	0.99	0.98	187813
Work	0.96	0.96	0.96	106701
accuracy			0.96	1352468
macro avg	0.97	0.97	0.97	1352468
weighted avg	0.96	0.96	0.96	1352468

Testing Report @2k epoch:

	precision	recall	f1-score	support
Eat	0.91	0.82	0.86	14347
Enter_Home	0.92	0.92	0.92	5211
Entertain_Guests	0.93	0.92	0.93	61063
Getting_ready	0.88	0.80	0.84	47057
Leave_Home	0.83	0.74	0.78	6875
Other	0.85	0.82	0.84	129221
Prep_for_meal	0.93	0.98	0.95	116183
Sleep	0.93	0.93	0.93	36237
TV	0.85	0.87	0.86	32950
Washroom	0.92	0.98	0.95	79624
Work	0.89	0.88	0.89	45421
accuracy			0.90	574189
macro avg	0.89	0.88	0.89	574189
weighted avg	0.90	0.90	0.90	574189

It seems that XGBoost is by far the best model at not over-fitting and still proving better results on test than any other models. However this is that 2k epochs, we can run the model on 4k epoch for better results.

Training Report @4k epoch:

Eat	1.00	1.00	1.00	33886
Enter_Home	1.00	1.00	1.00	12424
Entertain_Guests	0.99	1.00	1.00	143875
Getting_ready	0.98	0.98	0.98	110896
Leave_Home	1.00	1.00	1.00	16310
Other	0.99	0.97	0.98	304592
Prep_for_meal	0.98	1.00	0.99	273266
Sleep	0.99	1.00	0.99	85253
TV	0.99	0.99	0.99	77452
Washroom	0.99	1.00	1.00	187813
Work	0.99	0.99	0.99	106701
accuracy			0.99	1352468
macro avg	0.99	0.99	0.99	1352468
weighted avg	0.99	0.99	0.99	1352468

Testing Report @4k epoch:

	precision	recall	f1-score	support
Eat	0.93	0.84	0.88	14347
Enter_Home	0.92	0.92	0.92	5211
Entertain_Guests	0.95	0.95	0.95	61063
Getting_ready	0.91	0.85	0.88	47057
Leave_Home	0.85	0.75	0.80	6875
Other	0.87	0.85	0.86	129221
Prep_for_meal	0.94	0.98	0.96	116183
Sleep	0.94	0.94	0.94	36237
TV	0.88	0.89	0.88	32950
Washroom	0.94	0.98	0.96	79624
Work	0.91	0.90	0.91	45421
accuracy			0.92	574189
macro avg	0.91	0.90	0.90	574189
weighted avg	0.92	0.92	0.92	574189

IV. Conclusion

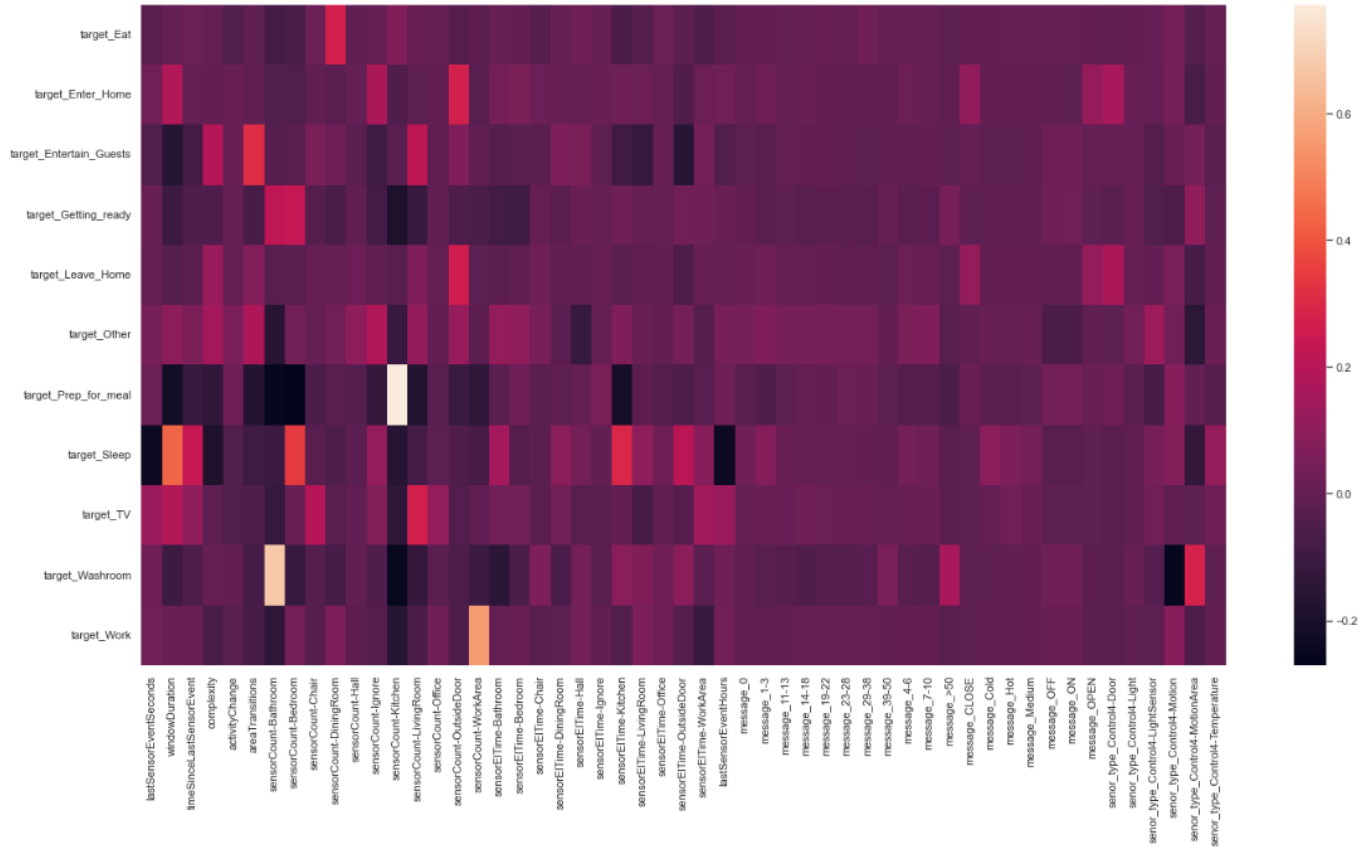
The amount of data and our multi-classification problem in this project burdened the computational power of our model. We performed EDA (exploratory data analysis) and base model fitting operations after data stitching and data preprocessing. Our EDA showed strong correlation between certain features of human activity occurrence and our base model (decision tree) showed 81% accuracy on the test samples. We used Random Forest algorithm to improve the results and achieved an accuracy of 87%. However, the models were over-fitting and so we tuned the parameters of Random Forest impacting over accuracy to 83%. We then tried OneVsOne and OneVsRest approach but achieved similar results. We then used XGBoost and found the best results. we achieved an accuracy of 96% on train and 90% on test at 2k epoch. It was computationally heavy to train the model. However we ran the model at 4k epoch and achieved an accuracy of 99% on train and 92% on test. The model size is double for the one at 4k epoch and the marginal improvement is accuracy is not that much. We accuracy is our highest priority, we select 4k epoch XGBoost as our final model but we select 2k epoch XGBoost model if are looking for an efficient model.

References

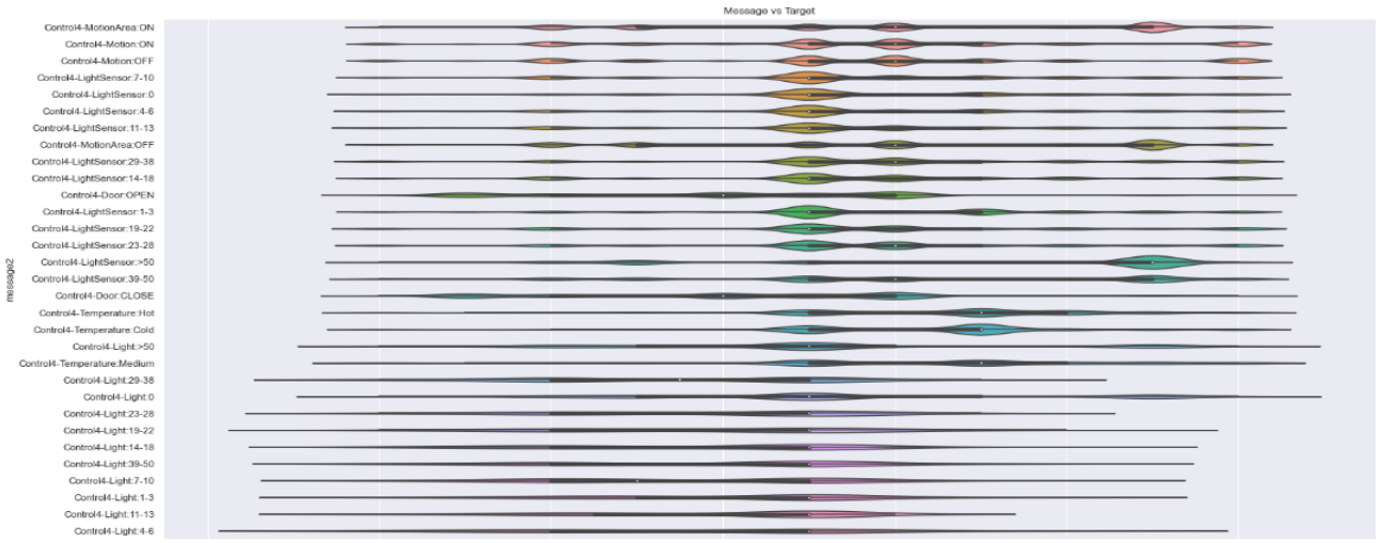
Data Source: https://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+from+Continuous+Ambient+Sensor+Data

EDA plots from next page.

Co-relation Heat map with target Variable .



Sensor Message vs Target Frequency



.

Hour of the week vs Target Frequency

.

.

Sensor count and time vs Target Bubble-plot