Institute of Architecture of Application Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Master Thesis

A framework for learning activities of office occupants

Prashant Gupta

Course of Study:	Master Infotech
Examiner:	Prof. Dr. Marco Aiello
Supervisor:	Dr. Ilche Georgievski
Commenced:	November 2, 2018
Completed:	May 2, 2019

Abstract

Energy consumption in buildings has a correlation with the activities of occupants. Buildings account for about 40% of the total energy consumption in many developed countries, making them the largest end-user consumer sector [1][2]. Non-residential buildings comprise 25% of the European building stock [2]. Average energy consumption in the non-residential sector is on average 280 Kwh/m^2 which is 40 % greater than an equivalent for the residential sector [2]. Commercial buildings like offices (23 %) and wholesale and retail shops (28 %) constitute the major part of total energy consumption in the non-residential sector[2]. Hence learning and understanding occupants' activities especially in a commercial landscape like offices is a complex process but with great benefits. The continuously changing patterns in data over time, ever generating new data types and dynamic streams of data make the task challenging as well as exciting. Understanding and analyzing the patterns in data produced as a result of human activities can lead to an increase in efficiency and higher performance resulting in lower energy consumption and more productivity of workers in the office environment [3]. The noisy raw data coming from multiple sensors installed at different locations in the office premises needs to be converted into some useful information and interpreted in an intelligent manner. This thesis investigates approaches to recognize occupant's activities in office premises and predict whether any of these activities will occur in some time window and how long these may last. The contextual information from raw sensor data can be used for learning different types of common office activities. ie., working or not working on a computer, reading a book using a lamp, presence, and absence from the room, preparing a meal using the microwave or making a hot coffee using a coffee machine in the kitchen. The sensors are selected keeping the privacy of the user in mind while deliberately not using cameras for video recording and capturing images. Our contributions can be grouped into a three-fold approach. Firstly, developing a framework which can address the problems of recognizing and predicting activities in real-time and off-line mode. Secondly, applying our framework on the real sensor data collected using a wireless sensor network in an office environment. Lastly, providing separate evaluating metrics for both problems of activity learning (recognition and prediction).

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

Total energy consumption of the building sector in Europe has increased by around 1% per year since 1990, mainly in non-residential buildings (1.5% per year for non-residential buildings compared to 0.6% per year for households) [1]. Meanwhile, the European Union (EU) has set an ambitious goal for new buildings to have almost zero energy status by 2050 [1]. The office sector is the largest in the commercial landscape both in terms of floor space and energy usage in many countries [3]. Heating, cooling, and lighting are the largest energy consumers in offices [3]. Traditional methods include asking participants to adopt measures voluntarily to conserve energy in buildings, and designing buildings in such a way that they include solutions like thermal insulation which can result in less thermal losses and better use of daylight to reduce lighting losses [3]. In addition, these methods involve companies adopting new ways for sustainable renewable energy like installing solar panels in their office premises [4]. Many other commercially available solutions require the use of occupancy sensors in offices with predefined timeout for turning off lights after no motion is detected without accounting for user preferences [3]. Often a low timeout value results in annoying the user and high value leads to energy losses. Although all these solutions offer energy conservation, they are far from being reliable and are often un-trusted and require a good amount of initial investment. These solutions leave a lot to be desired and this is where the role of modern technologies becomes important. Many modern solutions include the creation of smart grids in the commercial sector which can themselves generate surplus energy without impacting the environment [3]. But to make these grids truly smart we require the use of artificial intelligence due to the huge amount of data getting transferred along with electricity. Energy intelligent buildings refer to the buildings equipped with technology that allows for monitoring of the occupant's activities and/or facilities designed to automate and optimize control of appliances, with a goal of saving energy [3]. In all these cases adding artificial intelligence to help these buildings make decisions, interact with smart devices and wireless sensor networks based on user activities is of paramount importance for helping EU achieve its ambitious goals.

1.1 Problem Statement

Energy intelligent buildings, which are aware of occupants' activities through the use of modern techniques such as the machine learning approaches, need to be created. Understanding occupants' activities enables buildings to reason and react intelligently, and therefore, can improve the efficiency and effectiveness of the buildings. The occupants' activities are complex in nature as they require constant interaction with the environment which makes the sensor data noisy, imprecise, and corrupt. In addition, the order and duration of occurrence of the activities are always changing which makes the task of recognizing and predicting activities interesting as well as challenging. The problem of activity learning can be better understood in a sequential manner. The first step is to discover the meaningful patterns after studying the rich relational and temporal relationships from the noisy

wireless sensor data and interpreting them as activities. The second step is to use all the contextual information from the previous step to predict the next occurrence of activity and how long it may last using data-driven methods.

Conventional time series forecasting problems aim to predict future values based on the previous value of the variable to be predicted. These problems are useful in various scenarios like predicting economic, weather and stock prices trends. The problem of predicting the next occurrence of activities is different since we do not know about the time when activity is going to happen next until it happens [5]. We hypothesize that *buildings can be made more intelligent not just by recognizing the patterns in sensor data but also by predicting the next occurrence of the activity*. We further hypothesize that *having the information about when the activities have occurred in the past ie., contextual features will be valuable for the problem of activity prediction and indicating when the recognized activity will occur next* [5]. The challenge is how to use advanced techniques such as machine learning to learn activities (to recognize and predict) in a non-intrusive manner while also not discomforting user. The more precisely activities can be recognized and predicted, better services can be provided to the users.

For our problem at hand, we make the following assumptions while developing the system: the system is designed such that it monitors the activities of a single user (anonymous) and no user profiles are associated with the streaming sensors data. The analysis of sensor data is done keeping office infrastructure in mind and analysis of data for different building structure is beyond the scope of this thesis. The infrastructure is designed keeping current data needs in mind, please refer to the discussion in the Section 7.5 for how to make the present solution scalable and distributed (big data needs).

1.2 Practical Significance

The concept of the intelligent building offers many applications in fields of energy conservation, space optimization, developing location-aware services and healthcare. Monitoring the trends in user activities and forecasting their behavior through the use of machine learning can have a tremendous impact on space heating, cooling and ventilation demand, energy consumption of lighting appliances. For example, predicting activity patterns of the user can play an important role in smart thermostat since turning off the air conditioning device is not an instantaneous process [3]. The intelligent buildings can improve comfort, deal with medical rehabilitation, monitor physiological parameters and deliver therapy [3]. Therefore, the use of artificial intelligence to learn occupants activities can result in health benefits and increasing the overall productivity of the employee.

The space under-utilization is a common problem in offices which leads to lots of office area wastage. The opposite end of this problem is overcrowding which can lead to a noisy and stressful environment for the employee [6]. Thus maintaining a balance between the two problems offers crucial benefits. Nowadays, few companies offer visualization in the form of the dashboard to analyze the data and depending on the analysis give opinions about optimal space utilization in offices [7]. However, this is a time-consuming and not a real-time process.

Let's consider a simple example of how a system of learning activities can be useful in scheduling meeting [8]. Normally people use beacons and apps to report their presence [8]. However, this leaves out visitors or people who are not using these applications [8]. Instead of asking people to install

apps, and turn on services which can be inconvenient, unreliable and insecure, the better way would be if the system can automatically schedule meetings depending upon the presence of participant for long duration and unbook them also when the user has left depending on feedback from the wireless sensor network. The ability of the system to forecast depending upon the occupant's preferences can make the system even smarter. For example, if the model can predict whether the user is going to be absent during the day, it can cancel all his appointments and allocate the space to someone else.

1.3 Contributions

We summarize our contributions as follows:

- Study the novel problem of activity learning motivated by the real world applications while adopting non-obtrusive techniques for data collection and maintaining user privacy.
- Develop a framework consisting of techniques such as data collection, engineering, analysis, and visualization for handling the wireless sensor data collected in real-time or off-line in an office building environment.
- Formulate the problem of activity learning into a multi-step process and use of a single multi-output predictor for solving the problem of activity prediction.
- Reduce the annotation efforts of experts using manual annotations and pre-segmented data through automatic labeling using a Google calendar API for the pre-processed sensor data.
- Provide separate evaluation metrics to measure the performance of activity recognition and prediction models.

1.4 Document Structure

Chapter 2 gives essential background knowledge that discusses key concepts which are useful for understanding the terms used throughout the work. Chapter 3 gives insights into the current state of the art. Chapter 4 describes the system ie., gives insights into architectural design. The chapter further discusses the design choices and the functionalities of various components in the architecture. It also details the two important algorithms responsible for solving activity learning problem. Chapter 5 describes in detail the implementation of the system prototype. The chapter discusses various technologies utilized, how the data was collected, preprocessed and later explored. It also discusses various techniques for training recognition and prediction models. Chapter 6 provides a discussion on the results obtained. Chapter 7 gives concluding remarks and discusses how this system can be made better in the future.

2 Background Knowledge

2.1 Machine Learning

As stated by Mitchell in 1997," A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E " [9]. Machine Learning (ML) gives us methods to enable computers to learn from the data and improve themselves without the humans explicitly giving commands [10]. The way a machine learning model thinks is similar to human thinking and learning. It is an interdisciplinary field comprising of many fields like information theory, statistics, physics, optimization theory, neuroscience, artificial intelligence. In ML, data is a list of examples from which machine learning methods can learn the patterns and apply the knowledge to the new test data. The methods can be applied in many fields like speech recognition, financial trends predictions, natural language processing, image recognition, etc.

The ML field can be divided into two main branches namely unsupervised and supervised learning [11]. The primary difference between the two branches is the fact that labels are already available in supervised learning which is not in the case in unsupervised learning (clustering methods). The objective of clustering methods is to maintain high homogeneity within a cluster while high heterogeneity between clusters.

Supervised learning can be further broadly divided into classification and regression problems. The classification problems deal with discrete data ie., recognizing the categories or classes for the different examples present in the dataset while the regression problem deals with predicting the continuous real-valued data.

ML models can be further broadly classified into parametric models (like regression models) and non-parametric models (like k-nearest neighbors) [11]. Although parametric models have a fixed number of parameters which are faster to train, they make strong assumptions about the nature of data distributions (like the data available is already normalized). On the other hand, non-parametric methods do not have parameters and are more flexible but are often computationally intractable for larger datasets like big data. Hyperparameters are the settings that can be used to control the behavior of the algorithm [11]. They are not the trainable parameters which are estimated during the training. Hyperparameter tuning means searching for hyperparameters which result in the best performance on the validation set (unseen data).

The most important problem while training the model is called a generalization problem which deals with the performance on new unseen data [11]. The generalization is a result of overfitting the training results and showing poor results on unseen data or large test error ie., the training error continues to decrease while the test error starts increasing after a certain limit.

Given a vector of inputs $X^T = (X_1, X_2, ..., X_p)$, we can predict the output of the model (Y) given as [12]:

$$\hat{Y} = \hat{\beta_0} + \sum_{j=1}^{p} X_j \hat{\beta_j} \tag{2.1}$$

The term $\hat{\beta_0}$ is the intercept, also known as bias. The goal of linear regression, one of the popular machine learning algorithms is to reduce the sum of residual errors (RSS), also known as Least Squares [11].

$$RSS(\beta) = \sum_{i=1}^{N} (Y - \hat{Y})^2 \tag{2.2}$$

The difference in fits between training and the testing dataset is called variance. In machine learning, the ideal algorithm has low bias ie., can accurately model the true relationship and it has low variance by producing consistent predictions across different datasets. However, in the real world, there is always a trade-off between variance and bias. Three commonly used methods for finding the sweet spot between simple (low variance) and complicated model (low bias) are regularization, boosting and bagging.

Regularization

Regularization is any change we make to a learning algorithm that is intended to reduce its test set error (overfitting) but not its training error [13]. We add an extra term to the cost function which influences the parameters during training.

Ridge Regression in contrast to least squares line which might result in overfitting (high variance) adds a penalty term [14]. The main idea is to introduce a small amount of bias, leading to a significant drop in the variance. In other words, by starting with the slightly worse fit, Ridge Regression can provide better long term predictions. Lastly, even when there isn't enough data to find the Least Squares parameters estimates, Ridge Regression can find a solution with Cross-Validation and the penalty term.

$$RSS(\beta) = \sum_{i=1}^{N} (Y - \hat{Y})^2 + \lambda (theslope)^2 \tag{2.3}$$

where λ determines the severity of the penalty.

Lasso Regression is very similar to Ridge Regression, but there is one major difference. The big difference is that Ridge Regression can shrink the slope asymptotically close to 0, while Lasso Regression can shrink the slope all the way to 0 [15]. Since this technique can exclude useless variables, it is a little better than Ridge Regression at reducing variance in models that contain a lot of useless variables. Ridge Regression perform better when most variables are useful.

$$RSS(\beta) = \sum_{i=1}^{N} (Y - \hat{Y})^2 + \lambda \, |theslope| \tag{2.4}$$

Algorithm 2.1 Random Forest from [12]

procedure FOREST

1. For $b = 1$ to B:
   a) Draw a bootstrap sample Z^* of size N from the training data.
   b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable or split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees ${T_b}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{rf}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree.
Then $\hat{C}_{rf}^B(x)$ = majority vote ${\hat{C}_b(x)}_1^B$
end procedure

Elastic Net helps solve the dilemma if we have millions of parameters to estimate, how to decide if they are useful or useless. It combines Ridge Regression and Lasso Regression to shrink the parameters [14].

Bagging

Bagging is a technique which averages the prediction ($\hat{Y}$) over a collection of bootstrap samples (bootstrap dataset is the same size as original but we randomly select samples from original dataset allowing duplicates), thereby resulting in a lower variance.

Decision Tree are non-parametric supervised learning methods which are widely used for regression and classification tasks. The deeper the tree, the more complex decision tree and better the model can fit the data. Decision Tree resembles human-like thinking and is easier to interpret as compared to other machine learning methods. In a Decision Tree, each node represents a feature (attributes), the links represent the decision (rules) and the leaf represents the outcome[16]. Decision Trees are useful for all major types of input data like numeric, ranked, and categorical [16]. The depth of the tree is a factor playing an important role in deciding the occurrence of overfitting in the network (generalization problem) so it must be wisely chosen.

Random Forest is a bagging technique. As stated in Elements of Statistical Learning (also known as Bible of Machine Learning), " Trees have one aspect that prevents them from being an ideal tool for predictive learning, namely, inaccuracy" [12]. In other words, the trees work great with data used in training them but are not flexible when it comes to classifying new samples. Random Forest overcomes this weakness by combining the simplicity of decision trees with flexibility, resulting in a vast improvement of accuracy. Random Forest is also suitable for handling missing data in training as well as test data. Algorithm 2.1 shows a random forest algorithm.

Boosting

AdaBoost is one of the popular boosting techniques. In contrast, in a Forest of Trees made with AdaBoost, the trees are usually just a node and two leaves (stump). Stumps can only use the variable to make a decision. Thus, stumps are technically "weak learners"[12]. In Random Forest, each tree has an equal vote while deciding the outcome. In contrast, in a Forest of stumps made with AdaBoost, some stumps get more say in the final classification than others. Lastly, in Random Forest each decision tree is made independently of others. In contrast, in a Forest of stumps, the order is important. Each stump is made by taking previous stump's mistakes into account [17].

2.2 Deep Learning

Deep Learning is a special type of machine learning which has seen a lot of active research during recent times. Although the topic has a long research history since the 1940s, people stopped using it due to the resurgence of Support Vector Machine-based learning models. In 2006 in his paper on Deep Network, Hinton proposed to apply unsupervised learning to unsupervised pretraining to train neural networks with many hidden layers [18]. Nowadays, we can train neural networks without unsupervised pretraining using greedy layerwise pretraining, better initialization or non-regularities methods like RELU and Maxout and better regularization techniques like Dropout [19] [20]. Unsupervised pretraining still helps but only for small or rare datasets. A single neuron can only handle the task of binary classification. A single layer can only handle linear mappings or linear classification problems but generally for non-linear classification problems we need more hidden layers. We use the concept of activation function in order to translate a linear problem into a nonlinear problem. The important activation functions are sigmoid, softmax, hyperbolic tangent and RELU [21]. The learning process in a neural network can be divided into seven important steps [22] as seen in the Figure 2.1.

Model Initialization: The first step is the random initialization of parameters ie. weights and biases. The number of parameters indicates the capacity of the model. The network is more powerful if there are a higher number of parameters and more complex tasks can be solved. But in order to train a powerful model, we need to have enough training data to train. The basic rule says, that the number of training samples should be at least as the number of parameters in the model.

Forward Propagation: The activation of hypothesis function z^l for each individual unit (input) results in output vector. This calculation is repeated for each layer in the neural network. Since the flow of calculation is from the input towards the output, this is called forward propagation.

$$z^l = W^l a^{l-1} + b^l \tag{2.5}$$

$$a^l = \sigma(z^l) \tag{2.6}$$

where σ is an non linear activation function, a^{l-1} is the previous input vector and a^l is the output vector after activation and randomly initialized parameters $\theta_0 = [W^l, b^l]$.

Cost Function ($C(\theta_0)$): The obvious step after the forward propagation is to compare the performance. The loss function is an error metrics that gives an indication of how far the actual output values are from the desired output. The goal of any machine learning algorithm is to minimize this difference.

$$C(\theta_0) = \frac{\sum_{(x,y)} c(f(x), y)}{|D|} \tag{2.7}$$

where f(x) is the actual output and y is the true output.

Differentiation: The differentiation of the loss function w.r.t the parameters of the neural network is an optimization technique that gives an indication of the direction (gradient) and tells if we are moving closer or away from the desired output. The goal during this step is to descend to the global minima. This is known as gradient descent.

There are three variants of gradient descent. The trade-off between accuracy and optimization time is generally considered while deciding among the three variants.

- **Batch Gradient Descent**: We minimize the cost function over the complete dataset for each iteration hence it is a very slow process.
- **Mini-batch Gradient Descent**: Mini-batch Gradient Descent minimize the cost function over the samples of fixed batch size instead of going over all examples at once. Therefore, learning happens on each mini-batch of b examples
- **Stochastic Gradient Descent**: Stochastic Gradient Descent (SGD) performs the parameters update on each example instead of going over the entire dataset at once. Therefore, learning takes place in every example.

Mini-batch Gradient Descent is faster than Stochastic Gradient Descent due to lesser updates and better parallelization. Mini-batch is generally used for larger datasets. Each epoch is time taken for the network to observe the complete training data. It is important to shuffle the data after each epoch.

Back Propagation: Backpropagation is a way to understand how changing the parameters like weights and biases can affect the cost function. The direction of calculation is from output towards input and hence backpropagation. After application of chain rule

$$\frac{\partial C}{\partial w_{ij}^l} = \frac{\partial z_i^l}{\partial w_{ij}^l} \cdot \frac{\partial C}{\partial z_i^l} \tag{2.8}$$

where $\frac{\partial C}{\partial z_i^l} = \delta_i^l$ is the error of each layer

$$\delta^l = \sigma'(z^l)(W^{l+1})^T \delta^{l+1} \tag{2.9}$$

Parameters update: The learning rate is given as a very small magnitude constant, in order to force the parameters ie. weights and biases to get updated very smoothly and slowly so that there are no chaotic oscillations. If the learning rate is very small, it would take a long time duration to converge and become computationally expensive. However, if the learning rate is large, it may fail to converge and overshoot the minimum.

$$\theta_1 = \theta_0 - \eta \nabla C(\theta_0) \tag{2.10}$$

where $C(\theta_0)$ is the cost function , η is the learning rate.

Iterate until convergence: For each iteration, we update the parameters in the opposite direction of the gradient of the objective function where the gradient gives the direction of the steepest ascent so that the loss reduces. Since the objective function may contain one or many local minima, this procedure is repeated until we reach the global minima of the cost function.

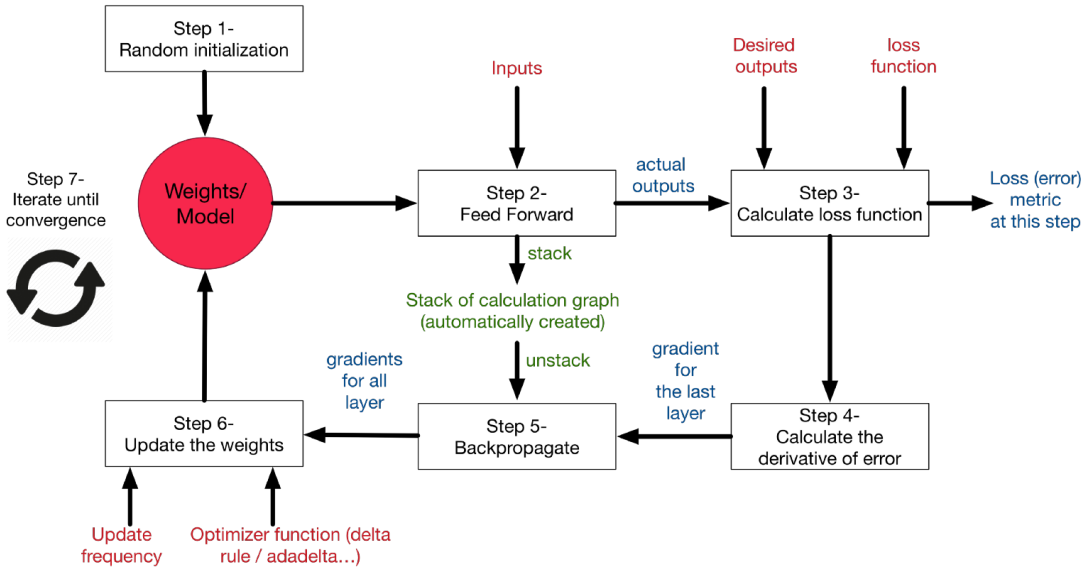


Figure 2.1: Summary of Training a Neural Network from [22]

Recurrent Neural Network (RNN)

RNN is a special type of neural network where prediction at each time step is influenced by previous time steps[23]. The recurrent structure seems to be very important which allows us to store information over time for a current prediction [23]. RNN based language model obtained great attention from speech and language research. It is commonly used for language modeling task, ie. predict the next word given the history and speech recognition. The chaining of many nonlinear functions leads to problems like gradient vanish or explode [23]. The gradient vanish occurs when multiplying values associated with tiny derivative and gradient explode occurs when multiplying values with the very large derivative (we can imagine multiplying a very small quantity weight w by itself many times).

Long-Short Term Memory (LSTM)

The challenge of keeping long-term dependencies and addressing problems like gradient vanish or explode give rise to another architecture style of RNN called LSTM networks by Hochreiter and Schmidhuber [24]. LSTM is a special neuron which allows us not only to store information over time but also to control which information should be stored and which should be forgotten.

The sigmoid layer output a value in the range [0,1] which indicates how much information is allowed to be passed through.

We can understand the LSTM network through the following steps [25]:

The first step is to decide how much information will be let through a 'forget'gate (where 1 denotes complete retaining and 0 denotes complete forgetting of information). It takes into consideration the output of previous timestamp and the input of current time step.

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \tag{2.11}$$

where f_t is forget gate, h_{t-1} is previous hidden state, x_t is input at current timestamp, W_f is the weight matrix and b_f is the bias.

The second step is to determine which new information should be stored. A sigmoid input layer decides which values should be updated. A tanh layer is used to create a vector consisting of new candidate values.

$$i_t = \sigma * (W_i.[h_{t-1}, x_t] + b_i) \tag{2.12}$$

$$\widetilde{C}_t = tanh(W_c.[h_{t-1}, x_t] + b_c) \tag{2.13}$$

where i_t is the input gate and $\widetilde{C}_t$ is the candidate vector.

At this step, the memory will be updated with the information saved in the previous time step and the new information.

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t \tag{2.14}$$

where C_t is the update gate.

Finally, the output can be computed and then passed to next step along with the memory. Again, the output of sigmoid layer is multiplied with the tanh of the memory cell to pass only the information which the network has decided to store.

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \tag{2.15}$$

$$\widetilde{C}_t = o_t * tanh(C_t) \tag{2.16}$$

where o_t is the output gate.

2.3 Activity Recognition

Activity recognition is a technique, where the aim is to recognize the actions and goals of one or more agents (human) from a series of observations on the actions of agents while accounting for environmental conditions [26]. Since the 1980s a lot of research has been undertaken in this branch of activity learning because of its multiple applications in the areas of medicine, sociology and human-computer interaction. The problem of recognizing activities can be solved through various machine learning approaches (such as a neural network, ensemble methods), probabilistic models and ontological rule-based approaches [27]. In our case, activity recognition is a type of multiple-class classification problem, which is solved using neural networks.

It has been referred to as goal recognition, intent recognition, behavior recognition, location estimation, and location-based services in various fields of studies [26]. Out of many types of activity recognition problems, sensor-based single user activity recognition is one of them. Researchers believe by monitoring the actions of an agent (with his consent) and analyzing the patterns in the noisy sensor data, one can estimate the consumption of energy in everyday life.

2.4 Activity Prediction

Activity prediction is a technique, where the goal is to predict the next occurrence of activity or the duration after which next activity is going to happen. In our scenario, activity prediction is a type of multiple output regression problem, which is solved using various machine learning approaches. It is a step that generally takes place after activity recognition so has some interdependence on the earlier mechanism. The problem becomes interesting due to challenges arising because of the error in the classification of labels provided by the experts and analyzing the patterns in noisy sensor data. The data contains rich temporal and relational relationships that must be explored in order to give correct predictions [28]. Activity prediction is valuable for studying various activity-aware services like intelligent home automation, prompting-based systems, and anomaly detection [28].

2.5 Activity Learning

Activity learning, as a combination of activity recognition and activity prediction, is a valuable means for buildings to be aware of occupant activities and make intelligent decisions. Activity learning aims to connect both problems in a sequential way so that the results of Activity recognition can help make a prediction of activities more accurate and precise. The challenge is how to use advanced techniques such as machine learning and data mining to learn activities from data coming from physical sensors.

2.6 Big Data

Big Data refers to data sets that are too large or have complex data structures structured or unstructured which makes it difficult for traditional computational methods to analyze or visualize them [29]. In this era of the Internet of Things, cloud-based systems and social media analytics, huge

amounts of sensor data (volume) are generated at high speed (velocity). They come in different types of data (variety), have different levels of trustworthiness (veracity) and should be used with a clear understanding of costs and benefits (value). From 2013 to 2020, it is estimated that the digital universe will grow 10 times from 4.4 trillion to 44 trillion gigabytes. It is expected to more than double every two years [30]. Due to the ever-growing number of servers, managing the data at zettabyte scale has become the need of the hour. The applications need to constantly interact with data storage and servers. In Google search (as an example of a simple application), 60 trillion web pages are indexed and 3 billion queries are processed every day. With the growth of data in such scales has given rise to many challenges such as managing heterogeneous data sources, scalability, real-time analysis, the privacy of user information [29].

2.7 Data Pipelines

The handling of an enormous amount of data (big datasets) has given rise to many data processing designs. Among the many, two architectures have become very popular.

- **Lambda Architecture**: It is a data processing technique that is designed to give increase throughput, reduce latency and give results with negligible errors. This design is useful for obtaining reliable results even if there is a small compromise with speed [31].

$$Query = \lambda(Completedata) = \lambda(Livestreamingdata) * \lambda(Storeddata)$$

  The architectural design consists of three layers in a sequence namely Batch, Speed and Service layer.

  1. Batch Layer: The data is stored on a weekly, monthly basis in the relational or NoSQL type of storage over a data lake. The data is processed using the Hadoop map-reduce algorithm or utilized by ML algorithms to make predictions.
  2. Speed Layer: The speed layer generally extracts fruits from the hard work of the batch layer. The speed layer uses the ML model generated by the batch layer to process the new data fed into it.
  3. Service Layer: The service layer is used to create a unified view of the outputs from the speed and batch layer. It serves to provide optimized responses to user queries.

- **Kappa Architecture**: It is a data processing technique where the goal is to get results as quickly as possible even if it that means some loss of accuracy. The architecture requires fewer resources due to the missing batch layer which means only one code base needs to be maintained ie., for the seed layer. This architectural design is useful for streaming real-time data and serves as online learners [31].

$$Query = \kappa(NewData) = \kappa(LiveStreamingData)$$

3 Related Work

In this chapter, we discuss the work that has already been done in the field of activity learning. It further discusses the types of sensors being used, the approaches authors have followed and how our work is similar as well as different from the current state of the art. The challenge of activity learning using multivariate sensor data due to dependencies make the problem more interesting. The field of activity learning can be investigated into two main branches namely activity recognition and activity prediction.

3.1 Activity Recognition

There has been a lot of work already done in the direction of activity recognition using different ML algorithms, probabilistic and rule-based models primarily for studying house activities.

3.1.1 Machine Learning and Data mining based Models

Fan et al. [32] used sensors like infrared sensors, force sensors, noise sensors placed at different locations in the house to collect data about house activities. They used several deep learning approaches like Recurrent Neural networks, LSTM and GRU to recognize different household activities and finally compared the different models based on seven-fold cross-validation using average precision and recall matrices. They used manual annotation of labels and interpolated data from different sensors based on the rate of sensors with the highest frequency. They concluded that LSTM and GRU outperformed standard RNN models. We have used automation techniques such as automatic indirect labeling of activities instead of manual annotation by experts.

In another paper by Singh et al. [33], the authors used the publicly available dataset from CASAS and concluded that there is a significant improvement in the performance of the LSTM model with the input raw sensor data in comparison with the results of probabilistic models based on Naive Bayes, Hidden Markov Model and Conditional Random Fields (CRFs). Similarly, we have used neural networks for activity recognition. However, we have collected our own dataset due to non-availability of a publicly available dataset that captures daily activities of office occupants.

A significant amount of work has been in the direction of activity recognition using smartphone sensors, wearable based sensors and using audio-video data. In one such study by Su et.al [34], the authors have compared core data mining techniques behind the stream activity recognition algorithms, analyze the challenges and presented some real-world applications. The smartphones are equipped with various sensors, including light sensors, temperature sensors, gyroscope, accelerometers, barometer, etc. Castro et al. [35] implemented C4.5 (decision tree) classifier to determine the activity done within four pre-established categories (lie, sit, walk and jog). The HAR-IoT system uses specialized hardware for vital signs monitoring including embedded heart, respiration and

body acceleration sensors. In general, C4.5 classifier suffers from generalization error (overfitting) much more than ensemble methods like Random Forest and AdaBoost used in our work. In our work, we have used wireless sensor network instead of wearable sensors so that office occupants do not feel discomfort while doing their day to day activities.

3.1.2 Probabilistic Models

Using the same testbed dataset from the CASAS smart home project, Ridi et al. [36] applied probabilistic approach Hidden Markov Model to perform activity recognition and then used simple neural networks to predict which activity will occur in the given time window. In another approach, Jiang et al. [37] used an improvised type of unsupervised learning technique using Hidden Markov Model to cluster the sensor events according to multiple activity class. The authors used similar sensors to study the problem of activity recognition. However, the authors used a generative approach (which focuses on finding the joint probability ie., model the distribution of individual classes) instead of discriminative approaches (which focuses on conditional probability ie., learning the boundaries between classes) used in our work. In general, discriminative models work better for larger datasets (more powerful) while generative models work better for smaller datasets. Moreover, many generative approaches make strong assumptions about the independent relationships between features instead in our case, we are exploiting the interdependence of features on each other for solving the problem of activity learning.

Another author Kelly et al. [38] investigated the idea of 24 hours time use diaries and wearable cameras for activity recognition. The activity sequence was reconstructed through the prompts from visual images. There was ambiguity with results as the diaries recorder 19.2 activities per day and image prompted interviews recorded 41.1 activities per day. The authors used daily time use diaries for annotation of activities where participants were asked to maintain notes and keep track of their activities every few minutes which can be inconvenient and unreliable. We have tried to reduce annotation efforts through the use of automatic labeling technique and Google calendar for incorporating the user preferences. The method allows participants to keep notes of activities only once instead of a constant reminder. The experience of adding new information in an online calendar is also more intuitive and convenient since the data is shared among various devices. We have also not used intrusive sensors such as a camera in our work keeping the privacy of users in mind.

3.1.3 Rule based Model

This next paper from the authors Georgievski et al. [27] focuses on ontology-based reasoning approach to address this problem of activity recognition and then uses Hierarchical Task Network Planning for decision making to control the environment. The authors apply this technique to the landscape of commercial buildings to make them energy efficient and intelligent. The authors first try to find the sequence of services that need to be executed after recognizing the activities and later implement techniques to execute these services efficiently in the building environment. The system was capable of sensing and recognizing occupant activities, planning for composing services and finally executing services upon devices. The authors also demonstrated that adopting such a system will result in energy savings that go beyond conventional lighting controls. Similar to our work,

the authors used a multi-step process and designed a framework which incorporates various data engineering principles. However, the authors have used the ontology-based approach instead of the machine learning approaches used in our work.

Nguyen et al. [39] proposed an activity recognition solution which can handle multi user, multi-area situations using the same ontology-based approach with an average accuracy of more than 92%. Although the rule-based approaches perform well for limited data the rigidity of rules pre-defined by domain and knowledge engineering experts makes it unsuitable for learning purposes.

3.2 Activity Prediction

The field of activity prediction is relatively new and emerging. Minor et al. [28] recently employed the study of activity learning motivated by real-world applications and formulated activity predictors in the framework of imitation learning to reduce the problem to simple regression learning. They used 24 hours smart home test-beds from CASAS multi-resident data set featuring motion, temperature, water, energy monitors, lighting controls and contact detectors. They also investigated activity prompting through a mobile application for evaluation. The authors assume that an activity recognition (AR) algorithm is available to annotate each sensor event with its corresponding activity category and also assume the availability of feature and loss function. The authors compared sensor windowing techniques ie. Recurrent Activity Predictor via Exact Imitation (RAP-EI) and Dagger Algorithm (RAP-DA) with the baseline approach Independent Predictors (IP). RAP-EI addresses the problem where IP ignores the temporal structural or the context features from previous sensor data. RAP-DA addresses the limitation of RAP-EI in addressing the propagation of errors to downstream decisions. The goal was to predict the relative next occurrence time of activity. The authors also used ensemble methods ie. random forest consisting of 100 decision trees to make the prediction about activity next time occurrence. The authors used multiple predictors for every activity instead of a single predictor like in our case for solving the problem of activity prediction. The authors also do not discuss any architectural design to solve the problem of activity prediction. In our case, we have developed a framework, which consists of data collection, pre-processing and engineering techniques for handling wireless sensor data.

In another interesting research, Xiao et al. [40] used Feed Forward Neural Network to predict the heart rate activity. Many popular time forecasting models based on the algorithms like autoregressive moving average (ARMA) and autoregressive integrated moving average (ARIMA) from Box et al. [41] [5] are based on the fact that the underlying time series is linear and stationary (ie., statistical properties like mean, variance and autocorrelation are constant over time). But these assumptions fail for the current scenario of activity prediction. These models are designed for uni-variate series where the previous values of the variable to be predicted are provided as an input to the model [5]. The current problem of activity prediction has a nonlinear multivariate relationship between the different variables.

Holder et al. [42] formulates the problem of activity prediction through a rule-based approach. They used the rules generated from the relationships between activities and their occurrence distribution to determine when the referenced activity is going to occur next. They used these rules to provide prompts in a smart home environment. This approach even though can work efficiently for a simple linear model but will fail for determining complex rich relational and temporal nonlinear relationships between input variables.

4 Framework for activity learning

In this chapter at first, we discuss the formal structure of the problem and give definitions for activities which are later used by indirect labeling mechanism, as described in the Section 5.7.1, to annotate the activities automatically. Later, we show insights into architectural design. We discuss the design choices and the functionalities of important components in the current architecture. Finally, we talk about the algorithms used to solve the problem of activity learning.

4.1 Formal Statement

Here, we formally define the problem of activity learning. First, we describe the problem of activity prediction and then use the results to solve the problem of activity prediction. Let A ={a_1,a_2,...,a_k} be a set of k activities where a_n belongs to n-th activity class. The training and test data consist of a sequence of sensor events (E_1, E_2,...,E_q) coming from the wireless sensor network, where E_i corresponds to the reading or the measurement recorded by the specific sensor at timestamp t_i. The features x $\in R^m$ (where m is the dimension of input features) are extracted from the sensor data as input. The labels a_k are extracted from using mapper functions and later through expert annotation.

The goal of activity recognizer is to map the features x $\in R^m$ to the corresponding activity label a_n. After labeling the raw sensor data, during the next stage output from Activity recognizer and new features x $\in R^l$ (where l is the modified dimension of input features after adding context features) are fed into the Activity Predictor as an input. The predictor generates $\hat{y}$ = ($\hat{y_1}$,$\hat{y_2}$,..,$\hat{y_k}$) as output, $\hat{y} \in R^k$ is the predicted next occurrence of activities a_k relative to the current timestamp t_j ie., $\forall$t: a $\in$A : $t_a = t_j$+r, r $\in$N. The ground truth labels ie., y = (y_1,y_2,..,y_k), y $\in R^k$ are later fed into the loss function L(x, y, $\hat{y}$). The goal of each activity predictor is to minimize the loss function ie., min$\sum_{c=1}^{\infty} L(y_c - \hat{y_c})^2$. The overview of the problem of activity learning for clearer understanding can be seen in the Figure 4.1 (modified from [28]).

4.2 Activities

The annotation of activities is based on the following rules as given in these definitions:

Definition 1 (Working on the Laptop): A person p is said to be working on the laptop if and only if d(p) in (x,y), where d is the distance of p from the desk in room(r), and e(p) in (z,w), where e is the electricity consumption of p.

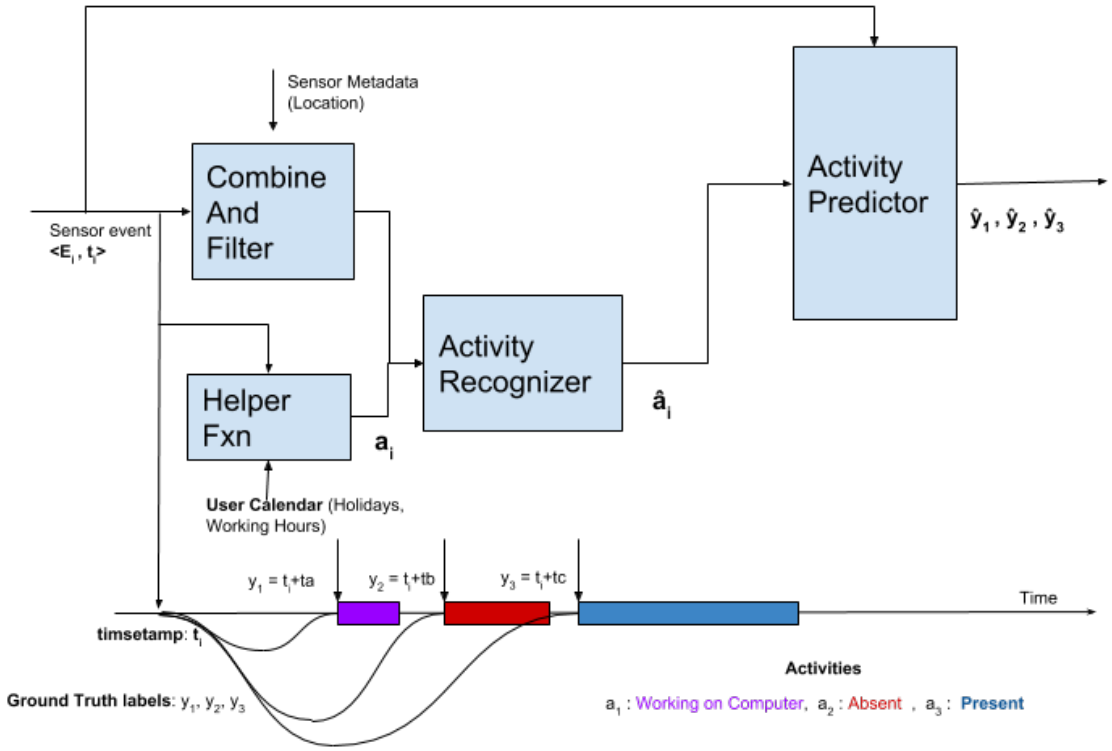


Figure 4.1: Overview of Problem

Definition 2 (Absent): A person *p* is said to be absent if and only if d(p) not in (x,y), where *d* is the distance of *p* from the desk in room(r), and e(p) not in (z,w), where *e* is the electricity consumption of *p* or the day is marked as a holiday in his/her calendar.

Definition 3 (Present): A person *p* is said to be absent, if and only if d(p) in (x,y), where *d* is the distance of *p* from the desk in room(r), and e(p) not in (z,w), where *e* is the electricity consumption of *p* and the day is not marked as a holiday in his/her calendar.

Definition 4 (Drinking): A person *p* is said to be making a cup of coffee, if and only if d(p) in (x,y), where *d* is the distance of *p* from the desk in room(r), and e(p) in (z,w), where *e* is the electricity consumption of *p*.

Definition 5 (Eating): A person *p* is said to be making a bread toast, if and only if d(p) in (x,y), where *d* is the distance of *p* from the desk in room(r), and e(p) in (z,w), where *e* is the electricity consumption of *p*.

Definition 6 (Reading): A person *p* is said to be reading a paper, if and only if d(p) in (x,y), where *d* is the distance of *p* from the desk in room(r), and e(p) in (z,w), where *e* is the electricity consumption of *p*.

Definition 7 (Not working on Laptop): A person p is said to be not working on a laptop if the person is not doing any of the above activities.

4.3 Architecture

The design of the living lab is based on a modified form of the Lambda architecture which ensures stream as well as batch processing of sensors big data. It is also highly reliable while performing real-time processing of distinct events. The architecture relies on both batch and speed layer for processing data. The architecture design offers very low latency for real-time streaming data and allows batch processing of entire dataset while creating models. This architecture also allows handling of historical data. The design offers a good balance between speed and reliability. The system architecture can be seen in Figure 4.2.

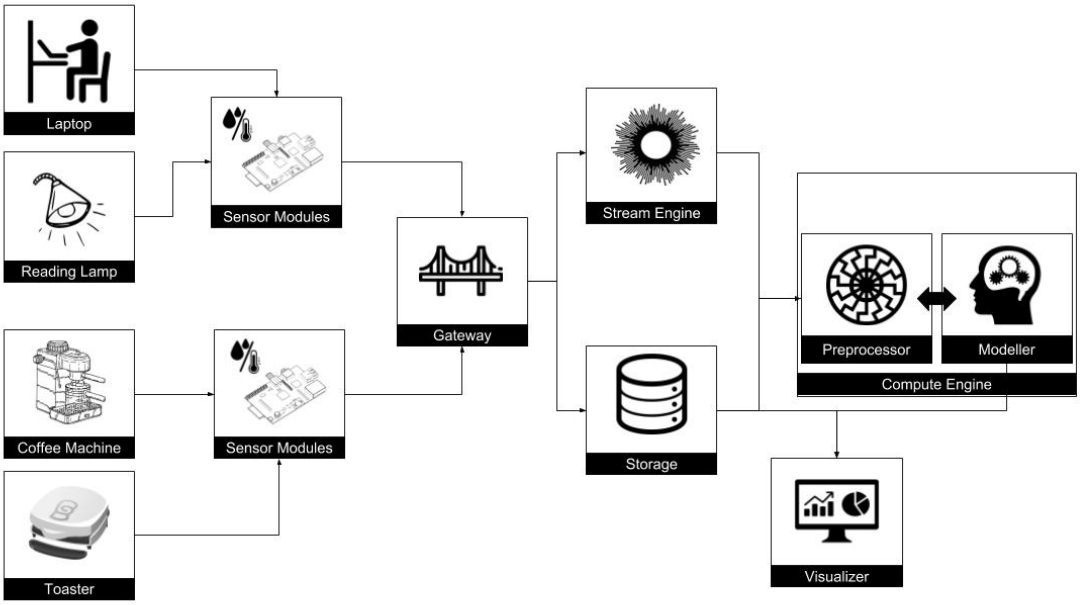


Figure 4.2: Basic Architecture

4.3.1 Sensor Modules

Sensor Modules form the most fundamental building block of the system design. The purpose of different sensor modules is to detect events or changes in its environment like changes in motion, light intensity, sound levels, temperature and humidity levels, range, and energy consumption. The sensor modules then further send the information to the gateway. The choice of sensor modules is based on the different types of activities and their possible correlation. The sensor modules act as a publisher of raw sensor data.

4.3.2 Gateway

Gateway collects data in a standardized format from different sensor modules under different topics and helps in interfacing with sensor networks using different protocols. The most important aspect of including a gateway is to ensure loose coupling. The loose coupling facilitates to increase efficiency for high-performance environments. The gateway decouples the producers and consumers and provides reference autonomy. The producers in our case sensor modules and the subscribers do not have to be available at the same time ensuring time autonomy. The producers and consumers do not need to understand the protocols and data formats exchanged ensuring format autonomy. The platform autonomy further decouples producers and consumers since do not need to be written in the same languages and environment. The gateway offers persistence [43] which stores data in case of network failures. Logging mechanism at each step is employed to monitor the changes in data sources incrementally.

4.3.3 Storage

Storage component is responsible for storing and retrieving data about the office building environment. The storage component acts as a subscriber or client. The choice of database is an important factor while designing the system. The sensor data can be divided into two main categories: one is the sensor module meta-data like physical location, id and other is time series sensor information like sensor type, timestamp, measure, id, and unit. The metadata about the devices and buildings is stored in a simple standardized format based file to be used later during data prepossessing for adding location information to the time series data. The file format is chosen because of its simplicity, less volume of meta-data information and maintaining low cardinality for faster access to the database. This also reduces messaging overhead during communication. The time series and interdependent sensor data are stored in a NoSQL database for faster performance in comparison to SQL databases. The choice is driven by characteristics of NoSQL like a schema-free design, data replication and focus on scalability for an enormous amount of sensor data generated at a very fast pace.

4.3.4 Stream Engine

Stream Engine provides a speed layer to the architecture. Stream Engine is responsible for accumulating multiple sensor events within a predetermined window interval and then creating a temporary view on which aggregation operations can be applied by the preprocessor component of compute engine later. The idea behind using an extra layer is to reduce latency in data generation while compromising on completeness of data in order to recognize and predict activities in almost real time. The design of such a system must also be robust and fault tolerant. It creates incremental updates rather than re computation updates thus providing a way to carry out analysis on most recent data. Stream Engine utilizes the model generated using the batch storage and applies ML algorithms to the new data fed into it.

4.3.5 Compute Engine

Compute engine is the brain of the system which is responsible for handling different stages of data processing until the model creation. Compute Engine must be able to handle and parallel process the huge volumes of sensor data on CPU and GPU units The compute engine can be further divided into two subcomponents namely Preprocessor and Modeler.

Preprocessor

The preprocessor is responsible for applying data scaling, data selection, data enrichment, data wrangling, and data exploration operations on the data accumulated by the batch storage and Stream Engine. The preprocessor is also responsible for coordinating the activities of different services like the compute engine and storage. It maintains a data flow between various components.

Modeler

The goal of Compute Engine is not just to prepare data for processing at later stages but to create the actual model for training and testing purposes. The modeler helps in preparing trained models for recognizing and predicting activities. The modeler also helps in communication between the different models. The preprocessed data is fed into the modeler to create trained models which are stored in disk storage for testing purposes in real time and offline mode. It is also responsible for maintaining information about the different versions of models created by the compute engine and evaluation metrics.

4.3.6 Visualizer

Visualizer is the eyes of the user through which the participant can interact with the system and understand the intrinsic complexity of data. The visualizer is responsible for showing the real-time streaming of sensor data from different sensor modules. This is done to explain the behavior of different sensor module to the participants. It also helps during the data analysis stage for the developer to explore the data through various plots or representations of data.

4.4 Algorithms

In this section, we describe the two algorithms used for solving the problem of activity learning ie., recognizing and predicting activities. Activity Recognition Algorithm which depends on local features, as described later in Table 5.5, to recognize the activity for the series of events. These local features are directly generated from streaming sensor data. Activity Prediction Algorithm uses the context features, as described later in Table 5.6, depends on previously sampled sensor data and output labels from recognizer to make final prediction for the next occurrence of all the activities simultaneously. In Activity Prediction Algorithm, baseline means the use of only local features for making predictions.

Algorithm 4.1 Activity Recognition Algorithm

Input: E = Training sequence of sampled sensor events data, L = Loss Function
Output: π = Recognizer

procedure CLASSIFICATION(E,L)
  Initialize C = $\emptyset$
  for each time step $i = 1$ to N **do**
    Compute Local Features $\theta_{local}(i) = \phi(\lambda_i)$
    Compute True Output $y(i) = \beta(\lambda_i)$
    Compute Predicted Output $\hat{y(i)}$ using π
    Compute Predicted Labels $\hat{\alpha(i)} = argmax(\hat{y(i)})$
    Add classification example $(\theta_{local}(i), y(i))$ to C
  end for
  π = Classification-Learner(C)
  return learned recognizer π
end procedure

4.4.1 Activity Recognition Algorithm

Algorithm 4.1 shows the pseudo-code for our Activity Recognizer. At each timestep i, we compute local features after preprocessing sampled sensor data where $\lambda_i = [E1, E2, ..., En]$ and ϕ is the preprocessing function. ϕ transforms, normalize and aggregates the sampled sensor data. Secondly, we compute the true outputs from the sensor data using the indirect labeling technique mapper function β. At last, the aggregate set of input features and true output labels ie., $(\theta_{local}(i), y(i))_i^N$ is given to the Classification Learner which computes the weights and bias parameters after minimizing the loss function L. The predicted output is a list of probabilities which is converted to the activity labels through argmax function which finds the index of the activity with the highest probability.

4.4.2 Activity Prediction Algorithm

Algorithm 4.2 shows the pseudo-code for our Activity Predictor (modified from [28]). Let A = $a_1, a_2, ..., a_k$ be the set of activities where a_j belongs to j-th activity. At each timestep i , we compute local features $\theta_{local}(i)$ as described in Algorithm 4.1. Then, for each of the activities, j we compute the contextual features $\theta_{contextual}(i, j)$ for each of the activities through multiple transformations such as masking, cumulative sum and time series shift using pandas library. Finally, we concatenate local and contextual features to get joint features $\theta(i, j)$. We compute the true outputs from the sensor data using another mapper function γ which uses forward fill, cumulative sum and count operations. At last, the aggregate set of input features and true output labels ie., $(\theta_{local}(i), y(i))_i^N$ is given to the Regression Learner which computes and learns the weights and bias parameters after minimizing the loss function L. Activity Predictor gives multiple outputs ie., the time for the next occurrence of each of the activities.

Algorithm 4.2 Activity Prediction Algorithm

Input: E = Training sequence of sampled sensor events data, L = Loss Function
Output: π = Predictor

procedure REGRESSION(E,L)
  Initialize P = $\emptyset$
  for each time step $i = 1$ to N **do**
    Initialize $\theta(i)$ = $\emptyset$
    Initialize $Y(i)$ = $\emptyset$
    for each activity $j = 1$ to K **do**
      Compute Local Features $\theta_{local}(i) = \phi(\lambda_i)$
      Compute Contextual Features $\theta_{contextual}(i, j)$
      Concatenated Features $\theta(i, j) = \theta_{local}(i) \oplus \theta_{contextual}(i, j)$
      Compute True Output $y(i, j) = \gamma(\lambda_i)$
      Compute Predicted Output $y(\hat{i}, j)$ using π
    end for
    Add $\theta(i, j)$ to $\theta(i)$
    Add $y(i, j)$ to $y(i)$
  end for
  for each time step $i = 1$ to N **do**
    Add regression example $(\theta(i), y(i))$ to P
  end for
  π = Regression-Learner(P)
  return learned predictor π
end procedure

5 Implementation

In this chapter we describe how we have implemented our system. We discuss the different technologies we have used for implementing various components in our architecture. We further discuss how we have collected, pre-processed data to make the problem fit for applying supervised learning methods, and later explored to get more insights into the patterns and structure of data. We have also discussed how we have trained the different models to solve the problem of activity recognition and prediction.

5.1 Living Lab

The living lab is the space equipped with sensors which we have used to perform our experiments and later to test our system. The lab is based on the user-centric research principles for prototyping, interacting and validating before releasing the final product [44].

5.1.1 Sensor Modules

The sensors we used in our implementation can be broadly classified into three main types as summarized in the Table 5.1.

Table 5.1: Sensor Modules

Sensor Modules	Sensor Types	Measurement Units
Grove Pi Sensor Box	Temperature sensor	celsius
	Relative Humidity sensor	%
	Light sensor	lux
	Ultrasonic range sensor	cms
	Sound sensor	db
Aeotec	Motion	no unit
Plugwise	Energy source	watts

We have employed five types of sensors in a sensor box as seen in Figure 5.1, as provided by Dexter Industries [1]. GrovePi is built on the Raspberry Pi, a powerful little computer that we can now use to control sensors. All the sensors are managed via an I2C interface. The temperature and humidity sensor [2] DHT11 relays digital data which is periodically sampled by the client application. The light

[1] Dexter Industries Homepage https://www.dexterindustries.com/

[2] GrovePi Temperature and Humidity sensor http://wiki.seeedstudio.com/Grove-TemperatureAndHumidity_Sensor/

Figure 5.1: Sensor Box in Student Room

Figure 5.2: Motion sensor in Kitchen

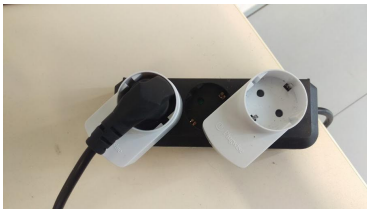


Figure 5.3: Plugwise energy sensors

sensor [3] integrates a photo register (light dependent resistor) to detect the intensity of light. It uses a dual OpAmp chip LM358 and is an analog device. Ultraviolet Range sensor [4] is a non-contact distance measurement device with a measuring range of 2 to 512 cms and is analog in nature. Grove Sound Sensor [5] can detect the sound intensity of the environment. The main component of the module is a simple microphone, which consists of the LM386 amplifier and an electret microphone. This module's output is analog. The data from the above five sensor types are sampled every 1 min to avoid polling issue. Each of the sensor's measurement is retrieved in parallel using thread mechanism.

The implementation is based on the factory pattern (class-based programming) to make the code more reusable. The factory pattern deals with the problem of creating objects without having to mention the exact object class that will be created. In our case, there are N classes which constitute the factory of classes and one can instantiate a class based on sensor type obtained from the topic dynamically. The pattern also follows inheritance since all the children sensor classes for the different sensor types inherit generic methods and properties from the base class.

Aeotec Multisensor 6 module [6] as seen in Figure 5.2 is used to collect motion, temperature and humidity data. With a 5 meter range and a 120-degree field of view, MultiSensor 6 captures motion data. Within each sensor is an incremental temperature sensor capable of accurately measuring between -10 to 50 °C. MultiSensor 6 offers incremental monitoring of humidity ranging from 0% to 100%. The module is chosen due to its easy wall mounted installation and ability to send data on battery. Although the module can send data while being operated on battery, we have used it in USB power source mode. The energy saving mode of module while on battery limited our ability to change the data sending frequency. The data is sampled every 10 seconds in our case. Each of the nodes in the Zwave network communicated with the gateway through an Aeotec Zwave stick Gen 5. The current source code is derived from an open source python implementation [45].

As for the energy monitoring device, we have used Plugwise Source [7] products as seen in Figure 5.3 consisting of ZigBee actuators which help in monitoring energy consumption of devices connected to these adapters. The data from these sensors is sampled every 10 seconds. The current source code is derived from an open source python implementation [46]. The data is relayed under three different topics: circle which reports state changes, power which reports energy changes per 10 second and

[3] Grove-Light sensor http://wiki.seeedstudio.com/Grove-Light_Sensor/
[4] Ultraviolet Range sensor http://wiki.seeedstudio.com/Grove-Ultrasonic_Ranger/
[5] GrovePi Sound sensor http://wiki.seeedstudio.com/Grove-Sound_Sensor/
[6] Aeotec Multisensor 6 https://aeotec.com/z-wave-sensor
[7] Plugwise Source https://www.plugwise.com/en_US/

energy which reports actual power consumption changes per hour. The devices communicate with ZigBee actuators connected in a mesh network with the help of ZigBee stick which serves as a gateway between devices and raspberry pi.

The data from different sensor modules based on different protocols are sent to the MQTT broker or gateway.

5.1.2 Visualization

The sensor modules were installed at two locations ie. at student assistant room and kitchen. The raw contextual data is collected from these sensor modules which form the lowest level infrastructure. For each of the installation at different locations, we first set up an interview with the participants and show the real-time streaming of sensor data from different sensor modules and visualization on a Grafana dashboard [47]. This is done to explain the behavior of different sensor modules. There are three dashboards each created for each of the rooms. The dashboards as seen in Figure 5.4 are customized according to the three sensor modules ie. Grove Pi, Aeotec, and Plugwise. Each of the dashboards shows the real-time streaming of sensors data. The participants can interact with the sensors in an intuitive manner, for example, they can move in front and away from sensors, make a sound, breathe and even turn on and off the electrical devices.

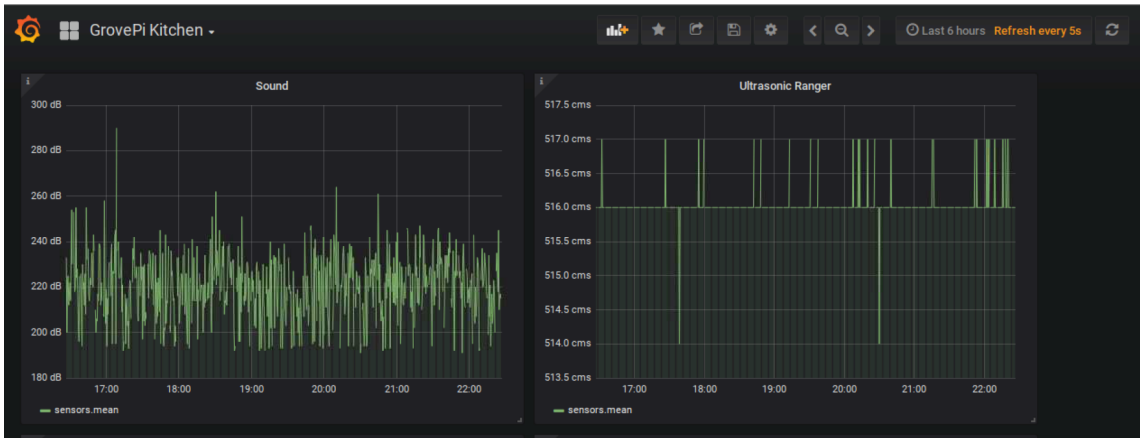


Figure 5.4: One Panel of Grafana Dashboard in Kitchen

5.1.3 Installation

The sensor modules were installed depending on the most comfortable locations for installing these sensor modules so these sensor modules do not interfere with their day to day work activities of the user. The different locations where the sensors are installed can be seen in Table 5.2.

Table 5.2: Installation of Sensors

Rooms	Grove Pi Sensor Box	Aeotec Multisensor 6	Plugwise Energy
Office Hiwi Room	Next to Computer	On the Wall	Laptop, Lamp
Kitchen Room	Food Preparation Table	On the Wall	Coffee Machine, Toaster

The choice of locations where the sensors were installed was also based on the following assumption that the activities have a possible correlation with changes in measurements as seen in the Table 5.3.

Table 5.3: Correlation between Activities and Measurements

Activities	Observed Measurements
Working on a Computer	Energy(Laptop),Light(Screen),Range(Computer Table), Noise
Working without a Computer	Energy(Laptop), Light(Screen),Range(Computer Table),Noise
Having a meeting	Motion(Meeting room),Noise
Preparing Coffee	Motion(Kitchen Room), Energy(Coffee Machine)
Making Bread Toast	Motion(Kitchen Room), Energy(Toaster)
Presence in the room	Motion(Hiwi Room), Energy(Laptop), Energy(Lamp)
Absence from the room	Motion(Hiwi Room), Energy(Laptop), Energy(Lamp)
Reading a paper	Energy(Lamp), Light

5.2 Gateway

Mosquitto is a lightweight open source gateway which uses MQ Telemetry Transport (MQTT) as a protocol. We have employed Mosquitto as our MQTT gateway which is kept running on the raspberry pi throughout the period of our experiment. The advantages of using MQTT over standard IP or TCP protocol in the Internet of Things world are primarily its ability to handle low constrained and unreliable networks [48]. It offers a low bandwidth network and minimizes device requirements [48]. It offers reasonable reliability and assures adequate delivery times which is suitable for our scenario. For adoption in enterprise purposes, RabbitMQ or HiveMQ may offer better reliability and persistence.

5.2.1 Topics

MQTT topics follow a structure similar to the file systems folders hierarchy on a UNIX system starting with/as a delimiter. In our case, we have followed a simple hierarchy as shown below:

<building_ID>/ <room_ID>/sensors/sensor_type

A client can subscribe to an individual or multiple topics depending upon the wildcard selected. The are two types of wildcard frequently use while subscribing to MQTT topics:

- # (hash character) - multi-level wildcard
- + (plus character) - single level wildcard

5.3 Data Ingestion

Data ingestion is a technique where data from the one or more sources is stored which can be used for further analysis later. The data from multi-protocol sensor modules are ingested with the help of gateway which can be used for stream analysis or stored in a database to be used for batch analysis later. We have used InfluxDB for storage of sensor data which is collected from various topics fetched by MQTT client running on the same server [49]. InfluxDB is a time series no SQL database which offers excellent in memory features and retention policy. The following checklist (inspired by videos from influx data) was followed while designing the current schema.

- Low Cardinality which is based on the concept of indexable columns (tags) to the required minimum saves memory usage.
- Intersection of tags and fields should be null meaning no fields and tags should have a common name.
- Do not encode data in tags.
- Use queries to determine the design.
- Group by only works for tags which leads to the loss in functionality.
- Aggregation functions like sum, min and max only work on fields which further leads to a loss in functionality.
- One should be committed to data types since we cannot change data types later once a series is created.

A sample query runs on InfluxDB [49] (simple SQL like query):

SELECT mean(“temperature”) **FROM** “sensors”**WHERE** “sensor_id”= ‘000D6F0003562BE1’**AND** “time”> now()-1d **GROUP BY** time(20m)

5.3.1 InfluxDB Key Concepts

Each row written in InfluxDB follows a line protocol. A simple example showing the line protocol in our case is shown below:

measurement,tag1=string,tag2=string field1=any,field2=any Timestamp(default = ns)

The most fundamental unit is a point. Each point is a combination of fields and timestamps. A collection of many points forms a serie. The combination of measurements and tagset are mapped to get a unique key which is called a serie.

sensors,sensor_id=000D6F0003562BE1,sensor_type=power,unit=watts measure=4 timestamp=2018-12-12T11:10:20Z(ms)

The data can be written in InfluxDB in many precision forms: rfc3339 (YYYY-MM-DDTHH:MM:SS.nnnnnnnnnZ), h (hours), m (minutes), s (seconds), ms (milliseconds), u (microseconds), ns (nanoseconds). Though the default precision is in nanoseconds, for less granularity and easier indexing of search queries it is better to use higher precision. For our case, we have selected milliseconds. Every day at midnight a bash shell script is run through a CRON job to create a .csv file using an SQL query on InfluxDB which can be used for analysis later. The script is run on a daily basis to reduce computation since it is running on Raspberry Pi which has limited processing capabilities as compared to a regular computer.

5.4 Data Collection

In the data collection process, two sets of data were collected. Firstly, the data is collected from sensor modules placed at multiple locations and then from python google calendar API.

5.4.1 Sensor Collected Data

The data collected from different sensors modules is sent to the gateway in the following JSON object. For example:

{
“**SENSOR_ID**”: “000D6F0003562BE1”,
“**SENSOR_TYPE**”: “Temperature”,
“**TIMESTAMP**”: 1542559440,
“**MEASURE**”: 35,
“**UNIT**”: “Celsius”
}

There are six sensor types in our case temperature, humidity, motion, light, sound, distance, and energy. The sensor type is of string data type. The timestamp is recorded in Coordinated Universal Time (UTC) format because of easier calculation and independence from various time zones. UTC is the number of seconds that have elapsed since midnight on Thursday, 1 January 1970. The sensor id is of string data type and is dependent upon the sensor module. In case of grove pi sensors, sensor id comprises of mac address of attached raspberry pi and the pin number to which each sensor is attached. While in the case of Plugwise devices their mac address is selected as the sensor id. For Aeotec devices, the combination of product id and node id is selected as the sensor id. The naming convention for the various sensors is selected solely based on uniqueness.

5.4.2 Sensor Metadata

The sensor metadata consists of additional information such as the location of sensors, manufacturer, etc. which is helpful in exploring the data later. All fields are of string type. The metadata about different sensor modules, is stored in a file in JSON format and is described as follows:

{
"**SENSOR_ID**": "000D6F0003562BE1",
"**LOCATION**": "Kitchen",
"**MANUFACTURER**": "Plugwise"
}

5.4.3 Google Calendar Scheduler

The activity scheduler uses google calendar API [50]. The goal behind using a Scheduler was to include user preferences while deciding the nature of activities. Every user is allocated a token which has to be sent in every request to fetch data about specific calendar events for security purposes. The different calendars of the user can be distinguished with separate IDs ensuring separation between personal and work calendar for example in our case we could specifically query the work calendar of the user. The following Table 5.4 shows the top 4 events extracted from the participant work calendar. The activity scheduler will be used for generating labels and mapping these labels to features in prepossessed sensor data.

Table 5.4: Activity Scheduler

Start Date	End Date	Activity	Location
2018-11-11T09:00:00	2018-11-11T15:00:00	Data collection	Hiwi Room
2018-11-11T15:00:00	2018-11-11T15:30:00	Lunch Break	Kitchen
2018-11-11T16:00:00	2018-11-11T16:30:00	Meeting with Supervisor	Meeting Room
2018-11-11T17:00:00	2018-11-11T18:00:00	Dental Appointment	Hospital Bad Cannstatt

5.5 Data Prepossessing

5.5.1 Data Cleaning

Data cleaning is the process of detecting or removing outliers and inconsistencies (duplicates) from data to improve the quality of data for future analysis [51]. The process also involves replacing the inaccurate parts of data present in a table, record set or database.

In our case, the data stored in the InfluxDB database is in UTC format. The time stamp for each of the rows is converted using pandas [52] utility (pandas.Series.dt.tz_convert) which converts it from the UTC timezone format into user's timezone (European) which helps in exact mappings and visualization during the data exploration stage. The conversion further helps during the prediction to notify the user of the next occurrence of activity in their local timezone. The time-stamps are further parsed from a string into date time index which provides a lot of useful methods like slicing, time deltas, etc. initially using the pandas date-time utility.

5.5.2 Data Integration

Data integration is a process which involves combining data from various sources and provide users with an aggregated view of the data [53]. The problem of designing data integration systems involves challenges such as how to interpret and replace missing information (primary key).

The individual files containing entries for sensor data of each day are integrated within a date range into a unified data frame object on which further analysis can be done. The DataFrame object is enriched with additional information such as the location of sensor modules based on sensor metadata JSON file. The unified data frame object is further enriched with a unique id which is useful during the data wrangling stage. The primary purpose of adding this column is to create a column with unique categories for each of the sensors groups to distinguish them later while creating the pivot table.

5.5.3 Data Wrangling

Data wrangling is a process of transforming the data from one form to another in order to make it suitable for downstream data analysis [54]. In the context of our problem, data wrangling converts the data into a format suitable for applying supervised learning techniques later.

Resampling

The data from different sensor modules (sensor box, Aeotec and Plugwise sensors) are synchronized and aligned with the Google calendar scheduler in the uniform bins of 1 minute. Resampling is done using the maximum values for temperature, light, humidity, sound level, motion and the and average value of energy consumption and minimum values for distance (range sensor). Resampling results in 1440 data points each day. Please refer to Appendix A.1.1 to see the DataFrame information after re-sampling.

Pivot Transformation

The data is converted from long data (row-wise) into wide data (spreadsheet-like tabular format) through aggregation in order to apply operations like slice, filter, group data and represent the information in a visually appealing way. The pandas utility pandas.pivot_table is used here since it can be used for aggregation on numeric data [52]. This implementation is based on polymorphism (class-based programming) that allows creating different room objects which can be used for the same functions but with different behavior. This makes it easier to add more rooms later to this implementation for better reusability. Please refer to Appendix A.1.1 to see the transformation of DataFrame.

5.5.4 Outliers Detection

Redundant or noisy data is removed during this stage. The outliers are the observations which show large deviations from clusters of the same features [55]. Outliers are usually the unexpected spikes at certain interval of time. A dataset containing outliers have several characteristics given as follows [55]:

- There is a deterministic pattern in data but also a stochastic variation (different from the rest of data).
- Only a few data points are outliers.
- Outliers can be important but also redundant hence depends on the context of the problem.

Normally, for a regression problem if the outliers fall into the training set they may impact the regression slope and if they fall into the test set they may change the accuracy score. During each iteration, at first the model is trained with all data points and then the points with large residual error (less than 10 percent of total data distribution) are removed. After comparing the performance improvement or degradation the model is again retrained until the stability in the form of training error is achieved.

We have used the following three methods namely box plots, scatter plots and Z-score to find out uni-variate and multivariate outliers. Please refer to Appendix A.1.2 to get a better understanding of these outlier detection techniques [56].

Box Plots: Box plots show groups of numerical data through their quartiles and help to identify variability outside the upper and lower quartiles [56]. Outliers may be plotted as individual points. The points separated from the grouped collection (boxes) are commonly referred to as outliers. We have used box plots for identifying outliers in univariate continuous values.

Scatter Plots: We have used scatter plots in identifying the relationship between bi-variate variables and classifying anomalies as outliers.

Z-Score: Z-score is defined as the signed number of standard deviations by which the value of an observation is above the mean value of the target column. Normally, z-score for data points greater than a threshold equal to 3 is considered a criterion for outliers [56]. We have used z-score as a numerical tool to find outliers between multivariate variables.

$$Z - Score = (X - \mu)/\sigma$$

where X represents the observations, μ represents the mean and σ represents the standard deviation.

5.6 Data Exploration

Data exploration is an approach which helps a data analyst to understand the characteristics of data through visual means rather than traditional data management systems. Data exploration is also a way of querying and visualization of data to identify potential relationships that may be hidden in the data. We have used the Seaborn library extensively for plotting exploration graphs [57]. Please refer to Appendix A.2 to see a more detailed discussion on the exploration techniques described as follows:

Line Plots

Line plots are the easiest and sometimes the most effective way to visualize a relationship of variable with respect to another variable. We have used line plots to show the relationship of different variables (like distance, sound, motion, energy) with respect to time.

Correlation Maps

Correlations maps also known as heatmaps are ideal to plot matrices. We have used these heatmaps to visualize the correlation between input features or when some values or calculated values, such as averages, counts, etc. are more extreme. The positive correlation varies from 0 to 1 while the negative correlation from -1 to 0. In general, these correlated variables add unnecessary complexity and do not improve the performance of models. Therefore, one out of the two correlated input variables must be removed from features set before feeding the features into the model.

Distribution Plots

Distribution plots are histograms, which shows the bins (equal division of variables) and counts of data points in each bin on x and y-axis respectively [58]. In our case, the bins will be an interval of time at which certain activity is performed and the count will be the number of data points of variables (such as energy, sound, and range) falling into that interval [58]. We have used distribution plots to understand the most common occurrences of input variables.

Mean Shift Clustering

Mean shift clustering is a centroid based algorithm, which uses a flat kernel and works by shifting candidates for centroids to become the new mean of the points within a given region. Among the candidates, duplicates are filtered in the post-processing step to form the final set of centroids. We are using Sklearn function to cluster the data points into different levels and understand the data patterns between these clusters [59].

5.7 Activity Recognition

5.7.1 Indirect Labeling

The aim of indirect labeling is to reduce the efforts of experts for manual annotation by providing them with helper functions. Mapper class tries to extract class labels for different activities from patterns in data (distance and energy consumption) and also tries to collect work schedule information of the participant using google calendar API before feeding them into the network. This labeling mechanism annotates different activities by following the definitions in Section 4.2.

For example, a person is assumed to be working on a laptop in r=student assistant room if x=10, y=100 and the unit is cm, and z=8 and w=50 and the unit is W. Similarly, a person is assumed to be performing the reading activity in the r=student assistant room if x=10, y=100 and the unit is cm, and z=0 and w=4 and the unit is W. In another example, a person is assumed to be performing the drinking activity in the r=kitchen room if x=10, y=100 and the unit is cm, and z=1 and w=2000 and the unit is W. Similarly, a person is assumed to be performing an eating activity in the r=kitchen room if x=10, y=100 and the unit is cm, and z=1 and w=1000 and the unit is W.

This method does not take into account other features information like sound, motion which seems to have a good correlation with different activity classes thus is not completely reliable. For example, there can be times when the user is absent from the room but the mechanism classifies the activity as Present since the distance drops due to malfunctioning of range sensor. The information about movement in the room and the intensity of sound and light can be a deciding factor in these situations. However, since the model trained using deep learning techniques derives labels from patterns in all of the data rather than just two features ie., range (d) and energy consumption (e) and assign weights to all features accordingly, the above mistakes are self-corrected during the training phase of recognition step.

5.7.2 Data Reliability

The pairwise plots are another way of showing the reliability of data. A pairs plot allows us to see both the distribution of single variables and relationships between two variables [60]. The relationship can be useful for identifying patterns between features and class labels.

As we can see from Figure 5.5 when the energy consumption from a lamp is 3 watts, the data points belong to only one class ie., Reading. Another observation shows when the energy consumption of the laptop is in between 10 to 30 watts, there is a strong possibility that the person is working on the laptop. As sound intensity is in between 0 to 200 or when the distance is in between 300 to 500 centimeters, the person is mostly absent. Similarly, when there is motion or movement in the room, a person is present or doing some activity like working on the laptop or reading.

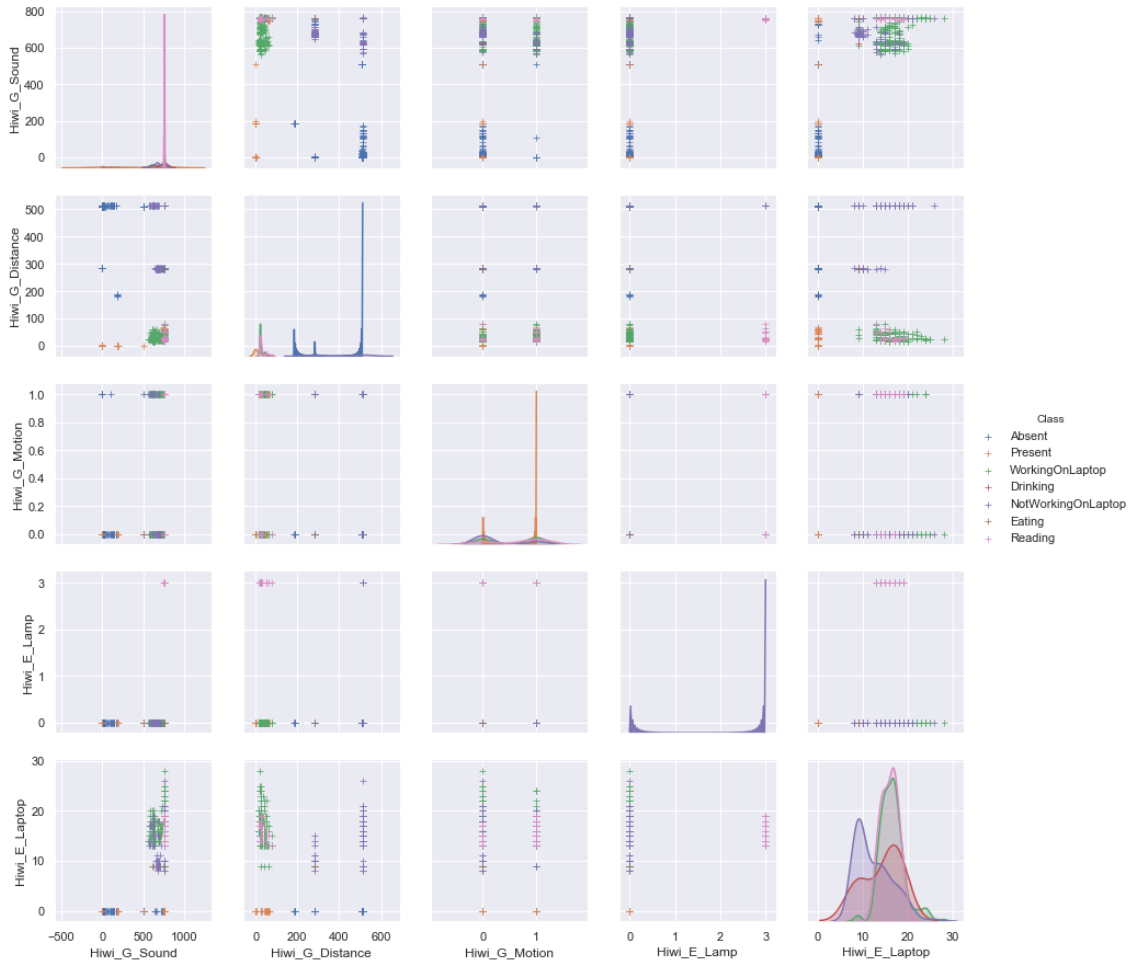


Figure 5.5: Pairwise Plot

5.7.3 One Hot Encoding

Labels are mostly converted into categorical data for classification purposes. The number of possible values is limited within a specified category. In our classification problem, there are seven activity categories. Most of the machine learning algorithms require the categorical data to be converted into numerical form. Similarly later at the prediction stage numerical data has to be transformed back into categorical data.

The simplest form is integer encoding where each category is assigned a value which can be used by machine learning algorithms due to their ordered nature [61]. However, for most cases where no ordinal relationship exists, one-hot encoding is used. In order to provide the categories as input to a statistical classifier, we need to map each category to a fixed-length vector. A very common approach is to represent each category in the input as a unit vector of the size of the vocabulary (dictionary of categories), where $e_i = 1 \Leftrightarrow c_i$ is the i_{th} category in the vocabulary. This is called a one-hot vector representation. We have used a helpful pandas [52] utility pandas.get_dummies to convert the categorical variable (activities) into their one hot encoding representation.

5.7.4 Feature Selection

As Einstein once said, ”Make everything as simple as possible but no simpler”. Feature Selection is a technique through which a subset of features are selected based on their importance to the final output (prediction) before the model formation. We want to take the minimal number of features that it takes to really capture the trends adequately. It is often said a machine learning algorithm is as good as the features that are fed into it. The goal is to select not only the best features but also add new features using our human intuition which can access the patterns in data [62]. Features and information are two different things. We want to draw a conclusion and have insights from information.

Dimensionality Reduction

Principal Component Analysis (PCA) is one such method of Dimensionality Reduction. As seen from Figure 5.6, a heatmap of important features with labels after applying PCA [62]. Working on the computer activity has a positive correlation with the energy consumption of computer and negative correlation with the distance of a person from the computer table indicating a person is sitting near the computer desk. Similarly, Drinking activity has a positive correlation with energy consumption of coffee machine and motion and negative correlation with the range of the person from the coffee table.

Figure 5.6: Dimensionality Reduction

The following features are selected based on correlation heatmap and different aggregation operations are applied with a sampling rate of 1 min depending upon the sensor type as shown in Table 5.5. These are known as Local Features.

Table 5.5: Local Features

Features	Description
Hiwi_G_Sound	Maximum value of sound sensor in student assistant room
Hiwi_A_Light	Maximum value of intensity of light in student assistant room
Hiwi_G_Distance	Minimum value of ultrasonic range sensor in student assistant room
Hiwi_G_Motion	Maximum value of motion sensor in student assistant room
Hiwi_E_Lamp	Average value of energy consumption of lamp in student assistant room
Hiwi_E_Laptop	Average value of energy consumption of laptop in student assistant room
Kitchen_G_Motion	Maximum value of motion sensor in kitchen
Kitchen_G_Distance	Minimum value of ultrasonic range sensor in kitchen
Kitchen_G_Sound	Maximum value of sound sensor kitchen
Kitchen_E_Toaster	Average value of energy consumption of toaster in kitchen
Kitchen_E_Coffee	Average value of energy consumption of coffee in kitchen

5.7.5 Feature Scaling

Feature scaling is a normalization technique which is used to standardize the range of independent variables or features of data. Most machine learning algorithms and neural networks depend on the magnitude of measurement while making the prediction and do not take into account the associated units with the measurement [63]. Let's say, we have two features (height of two persons) such as h1 = 182.88 centimeters and h2 = 6 feet. Human beings will recognize both persons to be of equal height, however, deep learning algorithms will give more weight to feature h1 since its magnitude is large and ignore the units while doing computations. This is the reason why one of the features having a greater magnitude can affect the prediction than an equally important feature.

Many algorithms which do not exploit distances or similarities between data samples such as Decision Tree, Naive Bayes, etc. do not require feature scaling [63]. The gradient descent in case of neural networks takes a very long time to convert if unscaled features are fed into the model. We have used min max scaler function from Sklearn preprocessing library which standardizes features by scaling feature to a given range ie., between 0 and 1 [59]. The method can be used even if the input data is not normalized making it ideal for our case. The transformation is given by:

$$X\_std = (X - X.min(axis = 0))/(X.max(axis = 0) - X.min(axis = 0)) \tag{5.1}$$

$$X\_scaled = X\_std * (max - min) + min \tag{5.2}$$

where min, max = feature_range.

5.7.6 Activities Distribution

On an average 1400 samples per day were collected after preprocessing (assuming sampling rate = 1 min). Since we collected data samples for a month comprising of 31 days, overall 44,640 samples were collected. Out of which 38,834 samples were of absent activity and 5806 samples were of rest of the activities. Since the majority of samples (absent activity) result in a skewed representation, they are excluded from the Figure 5.7.

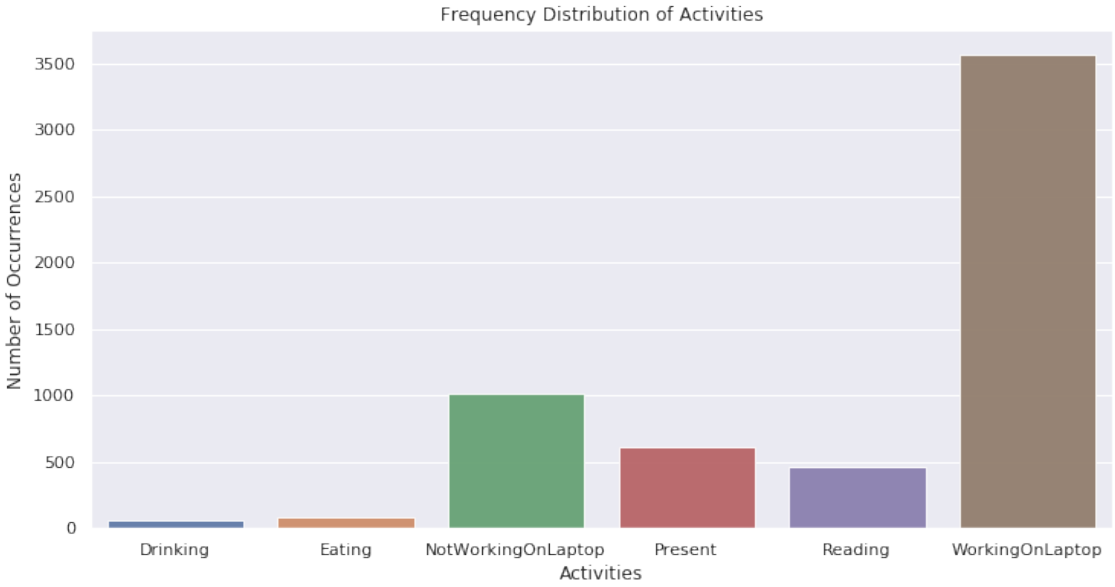


Figure 5.7: Distribution of Activities

5.7.7 Recognition Model

We have used a stacked LSTM network as our model with a stack of three hidden layers with 32 Neurons and finally adding a dense layer with Softmax activation function to map the non-normalized output of the network to the predicted output classes probability distribution. The other important hyperparameters used while training the network are learning rate = 0.001, batch size = 64, optimizer as RMSProp and loss function as categorical cross entropy. The model is trained for 20 epochs using the Keras [64] and TensorFlow [65] libraries for the recognition problem.

5.8 Activity Prediction

5.8.1 Data Enrichment

Data Enrichment is a technique of feature selection where we add new features using our human intuition which can access the patterns in data. This is based on our hypothesis that the information about when the activities have occurred in the past will be valuable for the problem of activity prediction. These new features are known as contextual features as seen in Table 5.6. The timestamp is normalized by a total number of minutes in a day. Events can be movement in the room (student assistant room, kitchen) or change in energy of devices like coffee machine, toaster, lamp, and laptop.

The contextual features are given higher importance while making predictions (root nodes) using a random forest approach and can be seen in Figure 5.8. The figure also proves our hypothesis that having the information about when these activities have occurred in the past ie., contextual features will be valuable for the problem of activity prediction.

Table 5.6: Contextual Features

Features	Description
Hour	The hour of the day when the event took place (0-24)
Day	The day of the week when current event happened (0-6)
Timestamp	Normalized time (minutes) since beginning of the day
TimeSinceLast{Activity}**	Time (minutes) since the last activity in the same category happened
TimeSinceLast{Event}**	Time (minutes) since the last event of same class happened

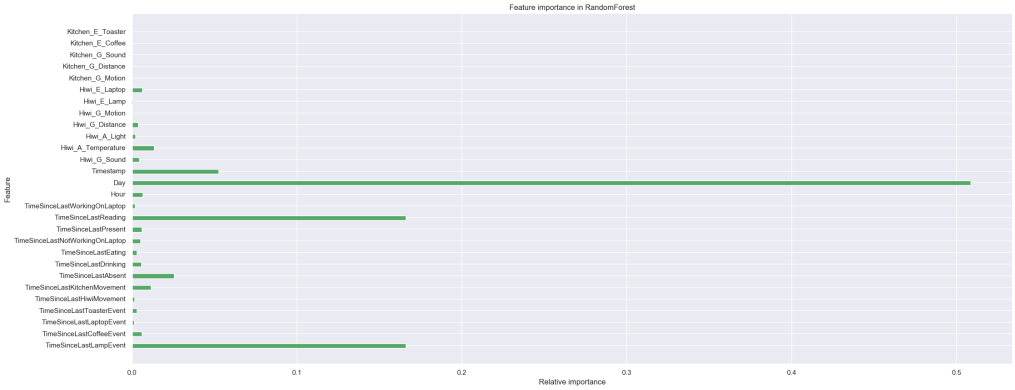


Figure 5.8: Relative Importance of Features in Random Forest

5.8.2 Prediction Model

We have employed various machine learning approaches for solving activity prediction as a multi-output regression problem. In the starting, we have used machine learning approaches involving neural networks such as a simple GRU and more advanced LSTM network using Keras [64] and TensorFlow libraries [65]. Later, we have employed traditional machine learning approaches such as Linear, Ridge, Lasso, Elastic-Net, KNN, Random Forest (estimators = 10 and max-depth = 20) and Adaboost. We implemented the traditional approaches using Sklearn Library. An example of one of the estimators in a Random Forest with evaluation criteria as MSE can be seen in Figure 5.9.

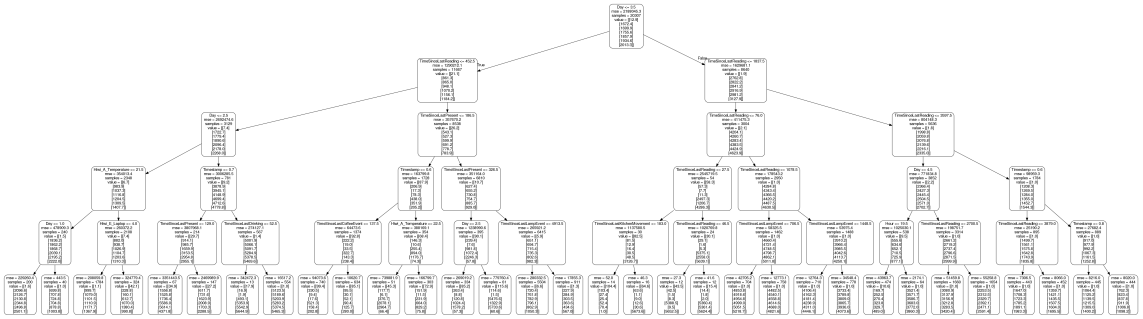


Figure 5.9: Sample Decision Tree in the Random Forest

6 Evaluation

We use different evaluation metrics to check the effectiveness of models developed using Activity Recognition and Prediction. Since recognition is a multi-class classification while prediction is a multi-output regression problem so evaluation methodology will also be different. In this chapter, we will discuss the different evaluation metrics for each of these problems in the context of test dataset and also identify the differences between Machine Learning and Deep Learning approaches through these evaluation metrics. We also used a Stream Engine using Apache Spark to create a temporary view of sensor events data for a window duration of 1 minute. We later applied aggregation and transformation operations before feeding the data into a model to see the recognition results in real time. The results can be viewed in an offset lag of 2-3 minutes depending upon the proper functioning of sensors in the living lab.

6.1 Recognition

6.1.1 Accuracy-score

Accuracy-score is the number of correct predictions divided by the total number of predictions made in the test dataset [59]. In our case, the accuracy-score is 0.9971. Though the result is pretty good accuracy-score must be taken with a grain of salt. The most important disadvantage of using accuracy-score metrics is that it doesn't take into account the total samples of each of the classes, ie., if we have a majority class (in our case Absent) that is bigger than the other classes and is correctly predicted most of the times then accuracy will be higher no matter even if the actual accuracy for other classes is worse. This is the reason we need better evaluation metrics to evaluate the performance of a multi-class classification problem.

6.1.2 Precision-score

Precision is the number of True Positives(TP) divided by the total number of True Positives (TP) and True Negatives(TN). Intuitively, Precision-score is the inability of the classifier not to label a positive sample that is negative. In our case, the overall Precision-score is 0.9849.

6.1.3 Recall-score

Recall-score is the ratio of TP / TP + FN where TP is the number of true positives and FN is the number of false negatives. Intuitively, recall-score is the inability of the classifier to find all positive samples. In our case, the overall Recall score is 0.9880.

6.1.4 F1-score

F1-score can be defined as an weighted average of precision and recall. F1-score varies between 0 to 1, where 0 denotes worse score and 1 denotes best score [59]. The relative contribution of both precision and recall is same towards the final f1-score. In our case, overall f1-score is 0.9868

$$F1 - score = 2 * (precision * recall)/(precision + recall)$$

6.1.5 Classification Report

The summary of important evaluation metrics for each of the activity classes can be seen in the Table 6.1.

Table 6.1: Classification Report

	Precision	Recall	F1-score	Support
Absent	1.00	1.00	1.00	7789
Present	0.91	0.96	0.93	134
WorkingOnLaptop	0.99	1.00	0.99	680
NotWorkingOnLaptop	1.00	0.95	0.98	200
Drinking	1.00	1.00	1.00	11
Eating	1.00	1.00	1.00	19
Reading	1.00	1.00	1.00	95
micro avg	1.00	1.00	1.00	8928
macro avg	0.98	0.99	0.99	8928
weighted avg	1.00	1.00	1.00	8928

6.1.6 Cross Entropy Loss

Cross-Entropy Loss function is generally used in neural networks or logistic regression. It is defined as the negative log-likelihood of the true labels given the probabilities of classifier's predictions [59]. It is generally used for multi-class classification. For a single sample with a true label y_t in {0,1} and estimated probability y_p

$$LogLoss = -(y_t log(y_p) + (1 - y_t) log(1 - y_p))$$

In our case, Cross Entropy Loss is 0.0110. The lower value of log loss indicates the better predictions.

6.1.7 Confusion Matrix

In a multi-class classification problem, the diagonal elements represent the number of points which are correctly classified ie., predicted label is equal to the true label while the non-diagonal elements are those mislabeled by the classifier. The higher the diagonal values of the confusion matrix means

better classification results. We have used a confusion matrix from the Sklearn to evaluate the quality of the classifier [59]. Figure 6.1 shows the confusion matrix without normalization while Figure 6.2 shows the confusion matrix normalized by the total number of elements in each class (class support size). The confusion matrix shows good results as the number of correct predictions for each of the classes is higher.

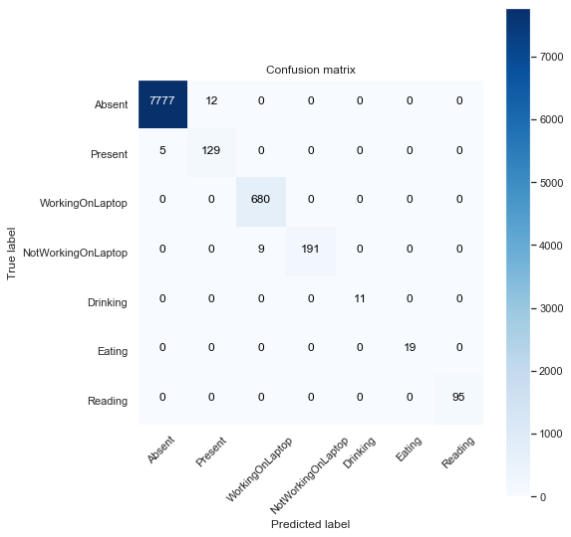


Figure 6.1: Confusion Matrix

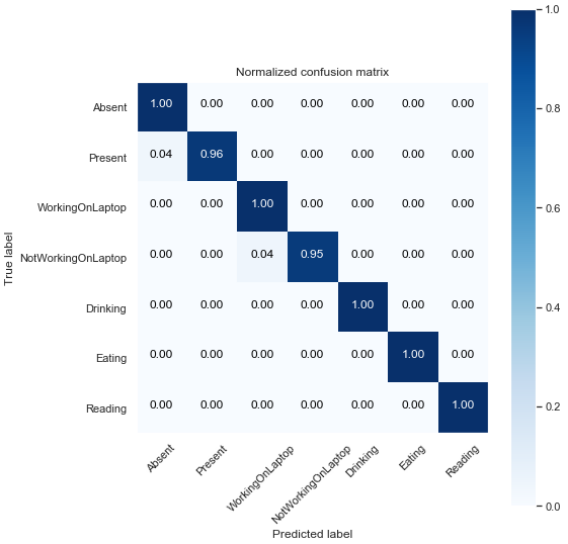


Figure 6.2: Normalized Confusion Matrix

6.2 Prediction

Since each machine learning algorithm is trying to optimize a different problem and dataset, the understanding of context is important before choosing the evaluation metrics. There is no single magic bullet to evaluate the performance of different problems. Every metrics have their pros and cons.

6.2.1 Mean Absolute Error

Mean Absolute Error (MAE) is defined as the sum of absolute differences between predicted ($\hat{y}$) and true values (y). MAE is a linear score that means all residuals are weighted evenly in the final error computation [66].

$$MAE = \frac{\sum_{j=1}^{n} |y - \hat{y}|}{n}$$

6.2.2 Root Mean Square Error

Root Mean Square Error (RMSE) is defined as the standard deviation of residuals ie., differences between predicted values ($\hat{y}$) and true observed values (y). If the RMSE is computed to be small, it is normally assumed that the model is a good estimator. The scale of the error depends upon the scale of target values.

$$RMSE = \sqrt{\frac{\sum_{j=1}^{n}(y - \hat{y})^2}{n}}$$

Since RMSE squares the large errors it weighs the large residuals more heavily than small ones thus effectively weighing errors unevenly [66]. This is the reason why MAE is robust to outliers whereas RMSE is not. Therefore, in problems with a large number of outliers MAE is obviously preferred. Generally, RMSE is more than MAE excluding the cases when the residuals are equal or zero resulting in RMSE equal to MAE. Both RMSE and MAE have a range [0,∞].

6.2.3 Explained Variance Score

As the name suggests, Explained Variance Score (EVS) is defined as the ratio between variance (square of standard deviation) of error and variance of true values [59]. It helps to identify how good your model can explain variation in the dataset.

$$EVS = 1 - \frac{Var\{y - \hat{y}\}}{Var\{y\}}$$

6.2.4 R2-Score

R2-Score also known as the coefficient of determination, provides a more consistent evaluation for predicting future values. If the activities have varying level of importance, RMSE or MAE will not give the correct indication of relative error [28]. Let's consider a case when one of the activity needs to be predicted more accurately than others [28]. For example, reminding a user to take medicine at the correct time has higher importance than housekeeping activity [28]. R2-score compares the fit of the chosen model with baseline horizontal straight line. It ignores the scale of output values (large or small), hence overcoming the disadvantage of RMSE. R2-Score varies in the range [-∞, 1] [67]. R2-Score is usually negative or 0 for a worse model and 1 for a perfect model [59]. R2-score can be zero in a case when the model always predicts the expected value of y, ignoring the input features [59]. R2-score can be negative when the chosen model is not able to understand the trends in data.

$$R2 - Score = 1 - \frac{\sum_{j=0}^{n_{samples}-1}(y - \hat{y})^2}{\sum_{j=0}^{n_{samples}-1}(y - \overline{y})^2}$$

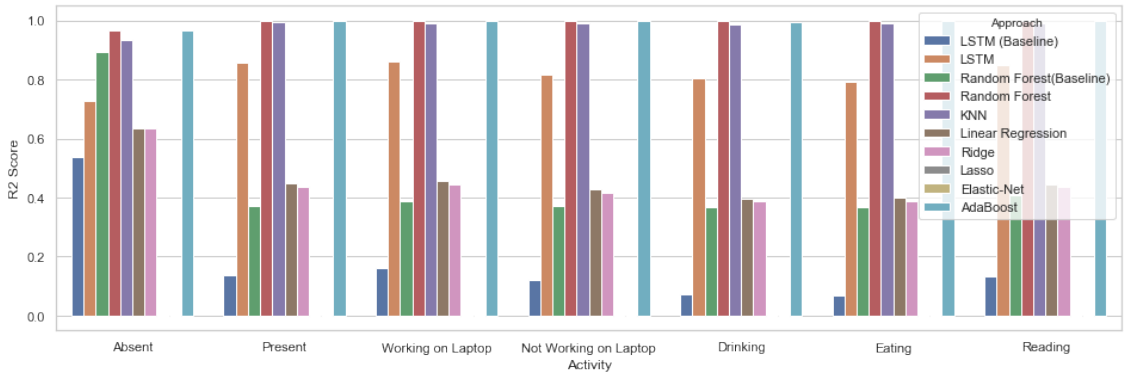


Figure 6.3: R2 Score using Different Prediction Approaches

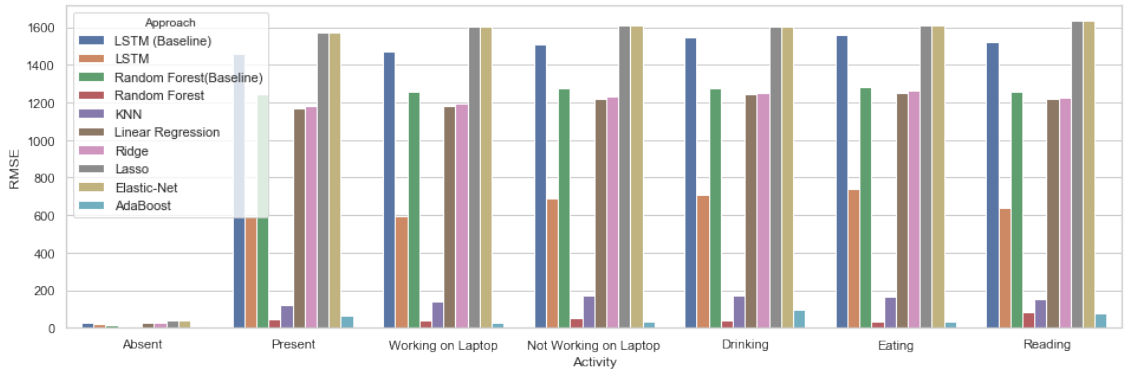


Figure 6.4: RMSE Score using Different Prediction Approaches

Here y is true value, $\hat{y}$ is the predicted value and $\overline{y} = \frac{\sum_{j=0}^{n_{samples}-1} y_j}{n_{samples}}$

Figure 6.3 shows the comparison between different models using R2-score evaluation metrics for the defined activities. Figure 6.4 shows the similar comparison using RMSE-score evaluation metrics. The models were created using machine learning approaches such as Linear Regression, Lasso Regression, Ridge Regression, Elastic Net, KNN, Random Forest, AdaBoost and advanced deep learning approach such as LSTM. The baseline approach is the one where only local features are used rather than using a combination of local features and context features like in other approaches. As we know, R2-score must be higher (close to 1) and RMSE-score must be lower(close to 0) for a perfect model. All approaches outperformed their baseline approach. We can further deduce that certain approaches like Elastic Net and Lasso Regression were the worst performers with a negative R2-score and a higher RMSE (magnitude of 1000s). The approaches like Linear Regression and Ridge Regression were comparatively better. The deep learning model trained using LSTM had an overall R2-score equal to 0.81 and a lower RMSE. However, ensemble techniques using Decision Trees like Random Forest and AdaBoost showed surprisingly great results with the R2-score close to 1 and a very low RMSE. The models trained using the Decision Tree outperformed other models as they require less training data as compared to deep learning models and are best suited for capturing

the non-linearity in data as opposed to linear regression approaches. Figure 6.5 and Figure 6.6 shows the model sizes and training times using different approaches. Though deep learning (LSTM) models have the smallest sizes, machine learning models were quicker to train.

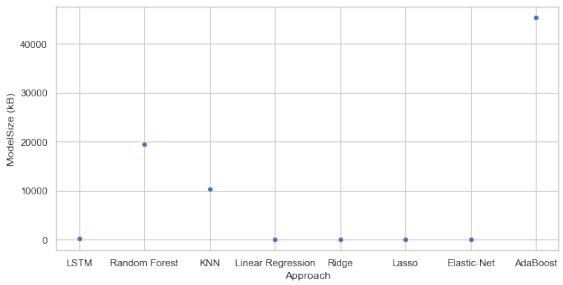


Figure 6.5: Model Size

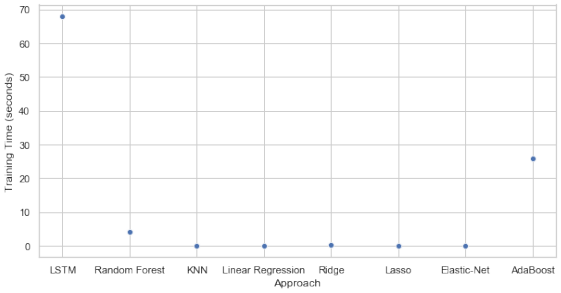


Figure 6.6: Training Time

6.2.5 Visualization

Evaluation metrics can indicate how well the model is performing but visualization enables us to see the actual behavior of models and compare the results. Since Random Forest gave the best evaluation results, it has been used as a base approach for visualization. Figure 6.7 shows the comparison between true (y) and predicted values ($\hat{y}$) for the Absent activity using Random Forest approach. Similarly, Figure 6.8 shows the scatter plot between predicted and true values using the two approaches for Absent activity.

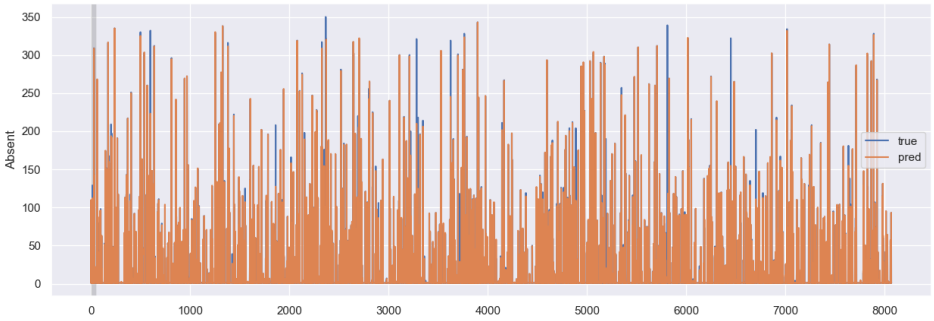


Figure 6.7: Comparison Plot for Absent activity using Random Forest

Figure 6.9 shows the comparison plots for the Working on Computer activity using Random Forest approach. Similarly, Figure 6.10 shows the scatter plot for the Working on Computer activity.

Please refer to Appendix A.3 to see the comparison and scatter plots using other approaches.

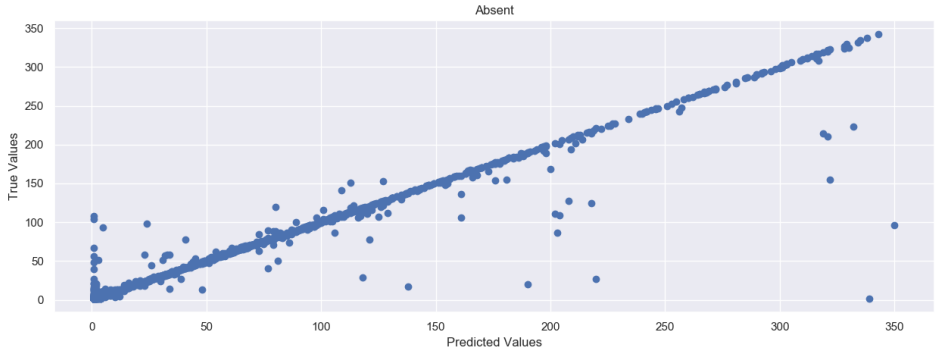


Figure 6.8: Scatter Plot for Absent activity using Random Forest

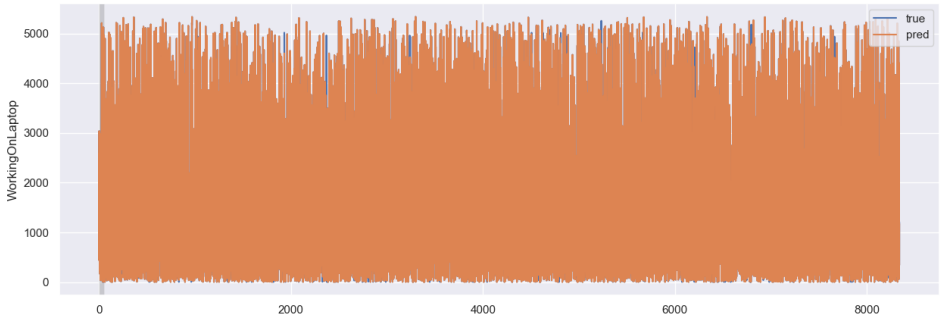


Figure 6.9: Comparison Plot for Working on Laptop activity using Random Forest

Figure 6.10: Scatter Plot for Working on Laptop activity using Random Forest

6.3 Evaluation Summary

We solved the problem of activity learning through a single multi-class classifier and multi-output regressor. During the recognition phase, machine learning approaches involving neural networks gave optimal results (ie., overall accuracy-score = 0.9971, precision score = 0.9849, recall-score = 0.9880, f1-score = 0.9868 and cross entropy loss = 0.0110) while during the prediction the results using neural networks were not satisfactory. After which we decided to approach the problem of activity prediction through traditional approaches of machine learning such as Linear, Ridge, Lasso, Elastic-Net, KNN, Random Forest and Adaboost. Though linear regression and regularization methods failed to give satisfactory results, we found surprisingly optimal results (ie., overall r2-score= 0.9989, mae = 3.5209, rmse = 44.208 and expected-variance-score = 0.9927) with ensemble decision tree based methods like Random Forest, AdaBoost. We found out that machine learning models involving neural networks as compared to models trained using traditional machine learning approaches were smaller in sizes but took significant time to train. Moreover, models trained using traditional approaches were easier to interpret and have a much smaller hardware dependency. Although it is inconclusive to state that traditional machine learning methods are more effective than neural network approaches for tackling the activity prediction problem due to the magnitude of data, tuning of thousands of parameters, for our case they did outcast them for the parameters we have selected.

In case of machine learning approaches involving decision trees, we deduce that increasing the depth of tree played a significant role in improving the accuracy of prediction model while increasing the number of estimators seem to have a marginal improvement in augmenting the accuracy of the model. For example, increasing the tree depth from 5 to 20 and keeping the number of estimators constant (ie., 10) increased the R2-score from 0.7721 to 0.9979. On the other hand, increasing the estimators from 1 to 10 and keeping the tree-depth constant (ie., 20) increased the R2-score from 0.9972 to 0.9979. Moreover, decreasing the number of estimators had a significant impact on the training time ie., it decreased from 134.82 to 3.91 seconds. This decrease in training time might be significant as the magnitude of data increases in the future. On the contrary, the depth of the tree seems to have an impact on the overall size of the model. Thus, the trade-off between depth of tree and size of the model is also one of the decisions one has to make before selecting parameters.

7 Conclusion and Future Work

7.1 Summary

We have used data-driven methods such as machine learning (involving neural networks and traditional techniques) to understand the problems of activity learning (ie., recognition and prediction) using non-invasive methods in the context of real-world applications for a single anonymous user in an office building environment.

At first, we studied current-state-of-the-art work and discussed how our work is similar/different from the previous work. Later, we formally discussed the structure of the problem of activity learning and designed the framework based on a modified form of lambda architecture consisting of stream and batch layer. We also discussed the two algorithms for solving the problem of activity learning.

During the implementation, we created a living lab and developed a framework consisting of techniques such as data collection, engineering, analysis and visualization for handling wireless sensor data collected in real-time and off-line. We collected our own dataset due to unavailability of publicly available office dataset. Our dataset consisting of 44,640 samples was collected for an entire month using our framework.

After the data was pre-processed, we generated the annotation for classes using the indirect labeling mechanism. Then, we applied different machine learning approaches to recognize activities using a multi-class classifier and predict the next occurrence of activities using a multi-output regressor. We decided to use separate evaluation metrics for comparing the performance of models trained using activity recognition and prediction algorithm.

7.2 Research Questions

The aim of the thesis was to address two important problems of activity learning (ie, recognition and prediction) in an office building environment. Firstly, we wanted to recognize the day-to-day activities of occupants in office. We formulated activity recognition as a multi-class classification problem and using our neural network, we were able to recognize the occupant's activities from the wireless sensor data at the current timestamp.

Secondly, we wanted to predict the next occurrences of activities and how long they may last. We formulated activity prediction as a multi-output regression problem and using ensemble machine learning methods, we predicted the next occurrence of multiple activities (minutes). We can add this next occurrence time to the current timestamp, and compute the time at which the next activity is going to happen. Since the activities are happening sequentially, we can also compute how long

each of the activity is going to last before the next activity starts. We also proved our hypothesis as seen in Figure 5.8 that having the information about when the activities have occurred in the past ie., contextual features will be valuable for the problem of activity prediction.

7.3 Findings

Our findings can be summarized as follows:

- Adding the annotations using the indirect labeling mechanism reduced the efforts of the experts, thus final recognition process is less costly and time-consuming.
- The models trained using ensemble machine learning approaches such as Random Forest, AdaBoost outperformed other approaches in solving the problem of activity prediction since they required less training data in contrast to neural networks, handle missing information well, and are best suited for capturing the non-linearity in data as opposed to linear regression approaches.
- The models trained using neural networks as compared to traditional machine learning approaches were smaller in sizes but took significant time to train.
- The ensemble methods predictions were much more impacted by the depth of the tree parameter rather than the number of estimators. The depth of the tree seems to have an impact on the size of the model while the number of estimators had an impact on the training time.

7.4 Limitations

Our approach has also certain limitations such as the small data set, possible improper position of sensors, possible bias because the person developing the framework is the same person performing the activities and collecting data. In order to incorporate user preferences, the current system depends on Google Calendar for annotation of activities especially Absent and Present activities since these depend upon user's working hours and holiday timings. Therefore the current system requires user's credential token, which needs to be passed in every code execution. The current system may not be suitable for distributed and scalable infrastructure. Moreover, the current system may not give similar results for other commercial buildings. We would like to address these limitations in future research work.

7.5 Future Work

The system is currently developed using a laboratory set up at the University of Stuttgart. In the future, we would like to make the framework more scalable to address the problem of Activity Learning for multiple users. Unsupervised learning methods such as clustering can be used to aggregate sensor data according to multiple user profiles. The framework can be made more robust and scalable in office scenarios by replacing the gateway Mosquitto with Kafka, data storage InfluxDB with distributed storage such as Cassandra and using Hadoop based technologies for

setting up multi-node clusters. We would like to build a central model management system that can address the magnitude of models generated due to high parameter tuning and different approaches. We would also like to apply our formalized approach to tackle a similar problem in other commercial buildings. We would also like to see how the current system can be used for addressing different real-world use cases such as energy conservation, space optimization.

Acknowledgments

I would first like to thank my thesis supervisor Dr. Ilche Georgievski for his advice during critical stages of my thesis, providing me resources and giving clear remarks and feedback during the implementation and report writing. I would also like to thank Professor Dr. Marco Aiello for supporting me and giving me an opportunity to write the thesis under his distinguished name. I would also like to thank the University for providing me background knowledge during the course of my studies. Finally, I would like to offer my gratitude and love to my parents, brother, and friends for supporting me during the important decisions of my life.

Bibliography

[1] U. Berardi, "Building energy consumption in us, eu, and bric countries", *Procedia engineering*, vol. 118, pp. 128–136, 2015 (cit. on pp. 3, 13).

[2] D. D'Agostino, B. Cuniberti, P. Bertoldi, "Energy consumption and efficiency technology measures in european non-residential buildings", *Energy and Buildings*, vol. 153, pp. 72–86, 2017 (cit. on p. 3).

[3] T. A. Nguyen, M. Aiello, "Energy intelligent buildings based on user activity: A survey", *Energy and buildings*, vol. 56, pp. 244–257, 2013 (cit. on pp. 3, 13, 14).

[4] N. R. Council *et al.*, *Carbon Management: Implications for R & D in the Chemical Sciences and Technology (A Workshop Report to the Chemical Sciences Roundtable)*. National Academies Press, 2001 (cit. on p. 13).

[5] B. Minor, D. J. Cook, "Forecasting occurrences of activities", *Pervasive and mobile computing*, vol. 38, pp. 77–91, 2017 (cit. on pp. 14, 29).

[6] M. T. Verklan, M. Walden, *et al.*, *Core Curriculum for neonatal intensive care nursing-e-book*. Elsevier Health Sciences, 2014 (cit. on p. 14).

[7] S. Inc, *Space analytics*, https://www.steelcase.com/asia-en/space-analytics/, Website, 2019 (cit. on p. 14).

[8] D. Zach, *Using motion sensors to automatically schedule meetings*, https://robinpowered.com/blog/using-motion-sensors-to-automatically-schedule-meetings-via-occupancy/, Blogs, 2015 (cit. on p. 14).

[9] T. Mitchell, B. Buchanan, G. DeJong, T. Dietterich, P. Rosenbloom, A. Waibel, "Machine learning", *Annual review of computer science*, vol. 4, no. 1, pp. 417–433, 1990 (cit. on p. 17).

[10] M. Awad, R. Khanna, "Efficient learning machines: Theories", *Concepts, and Applications for Engineers and System Designers, Apress, Berkely, CA*, 2015 (cit. on p. 17).

[11] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006 (cit. on pp. 17, 18).

[12] J. Friedman, T. Hastie, R. Tibshirani, *The elements of statistical learning*, 10. Springer series in statistics New York, 2001, vol. 1 (cit. on pp. 18–20).

[13] T. Gupta, *Deep learning: Regularization notes*, https://towardsdatascience.com/deep-learning-regularization-notes-29df9cb90779, Blogs, 2019 (cit. on p. 18).

[14] L. Fan, S. Chen, Q. Li, Z. Zhu, "Variable selection and model prediction based on lasso, adaptive lasso and elastic net", in *2015 4th International Conference on Computer Science and Network Technology (ICCSNT)*, IEEE, vol. 1, 2015, pp. 579–583 (cit. on pp. 18, 19).

[15] C. Maklin, *Machine learning algorithms*, https://medium.com/@corymaklin/machine-learning-algorithms-part-11-ridge-regression-7d5861c2bc76, Blogs, 2018 (cit. on p. 18).

[16] Y.-Y. Song, L. Ying, "Decision tree methods: Applications for classification and prediction", *Shanghai archives of psychiatry*, vol. 27, no. 2, p. 130, 2015 (cit. on p. 19).

[17] T. G. Dietterich, "Ensemble methods in machine learning", in *International workshop on multiple classifier systems*, Springer, 2000, pp. 1–15 (cit. on p. 20).

[18] G. E. Hinton, S. Osindero, Y.-W. Teh, "A fast learning algorithm for deep belief nets", *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006 (cit. on p. 20).

[19] Y. LeCun, Y. Bengio, G. Hinton, "Deep learning", *nature*, vol. 521, no. 7553, p. 436, 2015 (cit. on p. 20).

[20] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting", *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014 (cit. on p. 20).

[21] F. Ertam, G. Aydın, "Data classification with deep learning using tensorflow", in *2017 International Conference on Computer Science and Engineering (UBMK)*, Oct. 2017, pp. 755–758. DOI: 10.1109/UBMK.2017.8093521 (cit. on p. 20).

[22] A. Moawad, *Neural networks and back-propagation explained in a simple way*, [Feb 1, 2018], 2018. [Online]. Available: https://medium.com/datathings/neural-networks-and-backpropagation-explained-in-a-simple-way-f540a3611f5e (cit. on pp. 20, 22).

[23] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, S. Khudanpur, "Recurrent neural network based language model", in *Eleventh annual conference of the international speech communication association*, 2010 (cit. on p. 22).

[24] S. Hochreiter, J. Schmidhuber, "Long short-term memory", *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997 (cit. on p. 23).

[25] C. Olah, *Understanding lstm networks*, http://colah.github.io/posts/2015-08-Understanding-LSTMs/, Blogs, 2015 (cit. on p. 23).

[26] D. Hao Hu, S. J. Pan, V. W. Zheng, N. N. Liu, Q. Yang, "Real world activity recognition with multiple goals", in *Proceedings of the 10th international conference on Ubiquitous computing*, ACM, 2008, pp. 30–39 (cit. on p. 24).

[27] I. Georgievski, T. A. Nguyen, F. Nizamic, B. Setz, A. Lazovik, M. Aiello, "Planning meets activity recognition: Service coordination for intelligent buildings", *Pervasive and Mobile Computing*, vol. 38, pp. 110–139, 2017 (cit. on pp. 24, 28).

[28] B. D. Minor, J. R. Doppa, D. J. Cook, "Learning activity predictors from sensor data: Algorithms, evaluation, and applications", *IEEE transactions on knowledge and data engineering*, vol. 29, no. 12, pp. 2744–2757, 2017 (cit. on pp. 24, 29, 31, 36, 58).

[29] X. Wu, X. Zhu, G.-Q. Wu, W. Ding, "Data mining with big data", *IEEE transactions on knowledge and data engineering*, vol. 26, no. 1, pp. 97–107, 2014 (cit. on pp. 24, 25).

[30] Wikipedia contributors, *Big data — Wikipedia, the free encyclopedia*, https://en.wikipedia.org/w/index.php?title=Big_data&oldid=883455310, [Online; accessed 17-February-2019], 2019 (cit. on p. 25).

[31] I. Samizadeh, *A brief introduction to two data processing architectures—lambda and kappa for big data*, [Mar 15, 2018], 2019. [Online]. Available: https://towardsdatascience.com/a-brief-introduction-to-two-data-processing-architectures-lambda-and-kappa-for-big-data-4f35c28005bb (cit. on p. 25).

[32] X. Fan, H. Zhang, C. Leung, C. Miao, "Comparative study of machine learning algorithms for activity recognition with data sequence in home-like environment", in *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2016 IEEE International Conference on*, IEEE, 2016, pp. 168–173 (cit. on p. 27).

[33] D. Singh, E. Merdivan, I. Psychoula, J. Kropf, S. Hanke, M. Geist, A. Holzinger, "Human activity recognition using recurrent neural networks", in *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, Springer, 2017, pp. 267–274 (cit. on p. 27).

[34] X. Su, H. Tong, P. Ji, "Activity recognition with smartphone sensors", *Tsinghua science and technology*, vol. 19, no. 3, pp. 235–249, 2014 (cit. on p. 27).

[35] D. Castro, W. Coral, C. Rodriguez, J. Cabra, J. Colorado, "Wearable-based human activity recognition using an iot approach", *Journal of Sensor and Actuator Networks*, vol. 6, no. 4, p. 28, 2017 (cit. on p. 27).

[36] A. Ridi, N. Zarkadis, C. Gisler, J. Hennebert, "Duration models for activity recognition and prediction in buildings using hidden markov models", in *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, Oct. 2015, pp. 1–10. DOI: 10.1109/DSAA.2015.7344784 (cit. on p. 28).

[37] J. Jiang, R. Pozza, K. Gunnarsdóttir, N. Gilbert, K. Moessner, "Using sensors to study home activities", *Journal of Sensor and Actuator Networks*, vol. 6, no. 4, p. 32, 2017 (cit. on p. 28).

[38] P. Kelly, E. Thomas, A. Doherty, T. Harms, Ó. Burke, J. Gershuny, C. Foster, "Developing a method to test the validity of 24 hour time use diaries using wearable cameras: A feasibility pilot", *PloS one*, vol. 10, no. 12, e0142198, 2015 (cit. on p. 28).

[39] T. A. Nguyen, A. Raspitzu, M. Aiello, "Ontology-based office activity recognition with applications for energy savings", *Journal of Ambient Intelligence and Humanized Computing*, vol. 5, no. 5, pp. 667–681, Oct. 2014, ISSN: 1868-5145. DOI: 10.1007/s12652-013-0206-7. [Online]. Available: https://doi.org/10.1007/s12652-013-0206-7 (cit. on p. 29).

[40] F. Xiao, Y. Chen, M. Yuchi, M. Ding, J. Jo, "Heart rate prediction model based on physical activities using evolutionary neural network", *2010 Fourth International Conference on Genetic and Evolutionary Computing*, pp. 198–201, 2010 (cit. on p. 29).

[41] G. E. Box, G. M. Jenkins, G. C. Reinsel, G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015 (cit. on p. 29).

[42] L. B. Holder, D. J. Cook, "Automated activity-aware prompting for activity initiation", *Gerontechnology: international journal on the fundamental aspects of technology to serve the ageing society*, vol. 11, no. 4, p. 534, 2013 (cit. on p. 29).

[43] HiveMQ, *Mqtt essentials*, https://www.hivemq.com/mqtt-essentials/, Blog, 2015 (cit. on p. 34).

[44] M. Pallot, K. Pawar, "A holistic model of user experience for living lab experiential design", in *2012 18th International ICE Conference on Engineering, Technology and Innovation*, IEEE, 2012, pp. 1–15 (cit. on p. 39).

[45] A. Peace, *Aeotec github source code*, https://pypi.org/project/zwave-mqtt-bridge/, Github, 2018 (cit. on p. 40).

[46] S. Watt, *Plugwise github source code*, https://github.com/SevenW/Plugwise-2-py, Github, 2016 (cit. on p. 40).

[47] Grafanalabs, *Grafana documentation*, https://grafana.com/docs/, Documentation, 2019 (cit. on p. 41).

[48] P. Sethi, S. R. Sarangi, "Internet of things: Architectures, protocols, and applications", *Journal of Electrical and Computer Engineering*, vol. 2017, 2017 (cit. on p. 42).

[49] Influxdata, *Influxdb technical documentation*, https://docs.influxdata.com/influxdb/v1.7/, Documentation, 2019 (cit. on p. 43).

[50] G. Inc., *Google calendar api documentation*, https://developers.google.com/calendar/quickstart/python, Documentation, 2019 (cit. on p. 45).

[51] E. Rahm, H. H. Do, "Data cleaning: Problems and current approaches", *IEEE Data Eng. Bull.*, vol. 23, no. 4, pp. 3–13, 2000 (cit. on p. 45).

[52] W. McKinney, "Data structures for statistical computing in python", in *Proceedings of the 9th Python in Science Conference*, S. van der Walt, J. Millman, Eds., 2010, pp. 51–56 (cit. on pp. 45, 46, 50).

[53] M. Lenzerini, "Data integration: A theoretical perspective", in *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, ACM, 2002, pp. 233–246 (cit. on p. 46).

[54] W. McKinney, *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. Ö'Reilly Media, Inc.", 2012 (cit. on p. 46).

[55] M. Alam, *Outlier detection*, https://www.datasciencecentral.com/profiles/blogs/outlier-detection-with-time-series-data-mining, Blogs, 2018 (cit. on p. 47).

[56] N. Sharma, *Outlier detection techniques*, https://towardsdatascience.com/ways-to-detect-and-remove-the-outliers-404d16608dba, Blogs, 2018 (cit. on p. 47).

[57] M. Waskom, O. Botvinnik, D. O'Kane, P. Hobson, S. Lukauskas, D. C. Gemperline, T. Augspurger, Y. Halchenko, J. B. Cole, J. Warmenhoven, J. de Ruiter, C. Pye, S. Hoyer, J. Vanderplas, S. Villalba, G. Kunter, E. Quintero, P. Bachant, M. Martin, K. Meyer, A. Miles, Y. Ram, T. Yarkoni, M. L. Williams, C. Evans, C. Fitzgerald, Brian, C. Fonnesbeck, A. Lee, A. Qalieh, *Mwaskom/seaborn: V0.8.1 (september 2017)*, Sep. 2017. DOI: 10.5281/zenodo.883859. [Online]. Available: https://doi.org/10.5281/zenodo.883859 (cit. on p. 48).

[58] W. Koehrsen, *Histograms and density plots*, https://towardsdatascience.com/histograms-and-density-plots-in-python-f6bda88f5ac0, Blogs, 2019 (cit. on p. 48).

[59] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, "Scikit-learn: Machine learning in Python", *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011 (cit. on pp. 48, 52, 55–58).

[60] W. Koehrsen, *Visualizing data with pairwise plots*, https://towardsdatascience.com/visualizing-data-with-pair-plots-in-python-f228cf529166, Blogs, 2019 (cit. on p. 49).

[61] J. Brownlee, *Why one hot encoding in machine learning*, https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/, Blogs, 2017 (cit. on p. 50).

[62] I. Guyon, A. Elisseeff, "An introduction to variable and feature selection", *Journal of machine learning research*, vol. 3, no. Mar, pp. 1157–1182, 2003 (cit. on p. 51).

[63] I. Dzindo, *Feature scaling*, https://medium.com/@ian.dzindo01/feature-scaling-in-python-a59cc72147c1, Blogs, 2019 (cit. on p. 52).

[64] F. Chollet *et al.*, *Keras*, https://github.com/fchollet/keras, 2015 (cit. on pp. 53, 54).

[65] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. Corrado, A. Davis, J. Dean, M. Devin, *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/ (cit. on pp. 53, 54).

[66] A. Swalin, *Choosing the right metric for evaluating machine learning models*, https://medium.com/usf-msds/choosing-the-right-metric-for-machine-learning-models-part-1-a99d7d7414e4, Blogs, 2018 (cit. on pp. 57, 58).

[67] G. Drakos, *Regression metrics*, https://towardsdatascience.com/how-to-select-the-right-evaluation-metric-for-machine-learning-models-part-1-regrression-metrics-3606e25beae0, Blogs, 2018 (cit. on p. 58).

A Appendix

> We cannot solve our problems with the same level of thinking that created them
>
> *(Albert Einstein)*

A.1 Data Preprocessing

A.1.1 Data Wrangling

Resampling

The summary of DataFrame as seen in the Figure A.1 give more insights to the underlying process.

	Hiwi_G_Sound	Hiwi_G_Distance	Hiwi_G_Motion	Hiwi_E_Lamp	Hiwi_E_Laptop	Kitchen_G_Distance	Kitchen_E_Coffee	Kitchen_E_Toaster
count	1440.000000	1440.000000	1441.000000	1441.000000	1441.000000	1440.000000	1441.000000	1441.000000
mean	485.857639	375.156944	0.138099	0.054129	3.886190	514.718750	3.580153	1.435115
std	3452.504948	202.377887	0.345123	0.399460	6.908898	24.222835	69.303484	31.431210
min	0.000000	0.000000	0.000000	0.000000	0.000000	47.000000	0.000000	0.000000
25%	2.000000	282.000000	0.000000	0.000000	0.000000	516.000000	0.000000	0.000000
50%	2.000000	512.000000	0.000000	0.000000	0.000000	516.000000	0.000000	0.000000
75%	760.000000	512.000000	0.000000	0.000000	9.000000	516.000000	0.000000	0.000000
max	65535.000000	512.000000	1.000000	3.000000	28.000000	516.000000	1499.000000	693.000000

Figure A.1: Summary of Data Frame after Resampling

Pivot Transformation

The dataframe after transformation from long data to wide data format. This transformation of dataframe can be seen in Figure A.2.

A.1.2 Outliers Detection

Box Plots

In the Figure A.3 we can easily observe one of the points ie value around 538 sound intensity is outside the normal continuous distribution (maxima and minima) thus can be safely removed from the distribution since our goal here is recognize the patterns rather than dissimilarity. In the other Figure A.4, there are no deviations of points outside the maxima and minima of the box indicating no outliers in this distribution.

datetime	MEASURE	SENSOR_ID	SENSOR_TYPE	UNIT	Location	ID:Type
2019-01-16 23:00:00	0	000D6F0000729939	power	watts	Kitchen	000D6F0000729939:power
2019-01-16 23:00:00	0	000D6F0005690BAB	power	watts	HIWI_Room	000D6F0005690BAB:power
2019-01-16 23:00:00	0	000D6F0003562BE1	power	watts	HIWI_Room	000D6F0003562BE1:power
2019-01-16 23:00:00	0	0f02310bc0_2	Motion		HIWI_Room	0f02310bc0_2:Motion
2019-01-16 23:00:01	22	0f02310bc1_3	RelativeHumidity	%	Kitchen	0f02310bc1_3:RelativeHumidity

datetime	Hiwi_G_Temperature	Hiwi_A_Temperature	Hiwi_G_Light	Hiwi_A_Light	Hiwi_G_Motion	Hiwi_A_Motion	Hiwi_G_Sound	Hiwi_G_Distance
2019-01-16 00:00:00+01:00	23.0	20.0	202.0	0.0	0.0	0.0	1.0	512.0
2019-01-16 00:01:00+01:00	23.0	20.0	190.0	0.0	0.0	0.0	1.0	512.0
2019-01-16 00:02:00+01:00	23.0	20.0	187.0	0.0	0.0	0.0	1.0	512.0
2019-01-16 00:03:00+01:00	23.0	20.0	187.0	0.0	0.0	0.0	1.0	512.0
2019-01-16 00:04:00+01:00	23.0	20.0	188.0	0.0	0.0	0.0	2.0	512.0

Figure A.2: Pivot Transformation of Data Frame

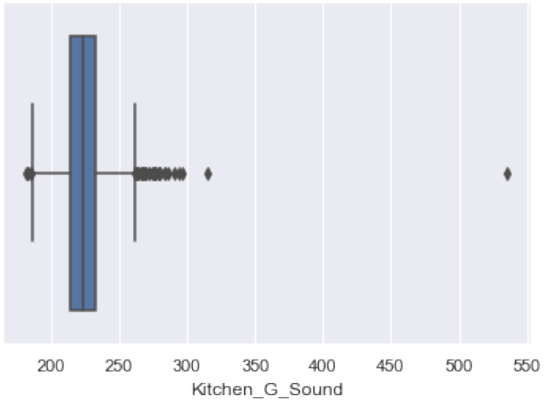


Figure A.3: Outliers in Box Plot

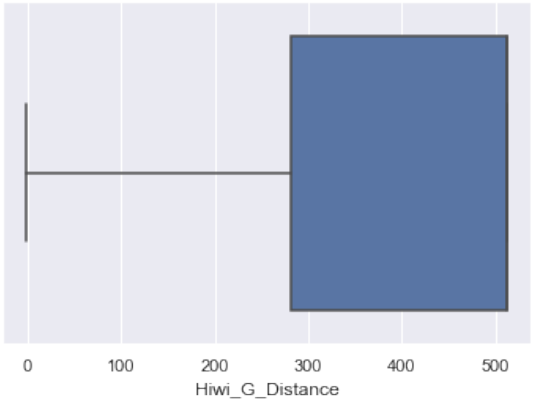


Figure A.4: No Outlier in Box Plot

Scatter Plots

As we can see from the Figure A.5, most of the points for sound intensity lie in range 0 to 200 for increasing energy consumption, however there are two points which have very unusual value ie 65500. These points can be safely considered as outliers.

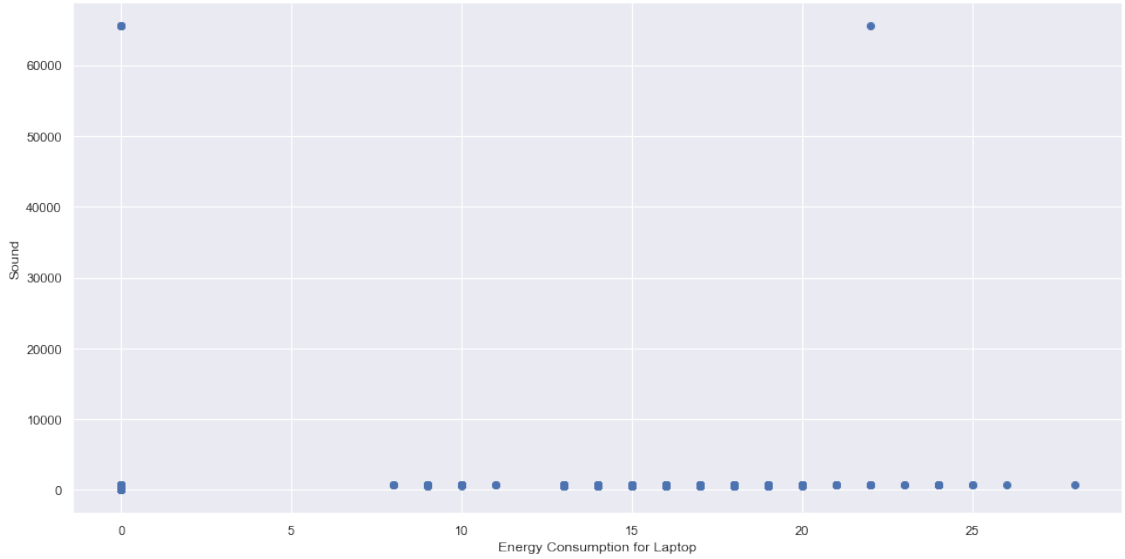


Figure A.5: Outliers in Scatter Plot

A.2 Data Exploration

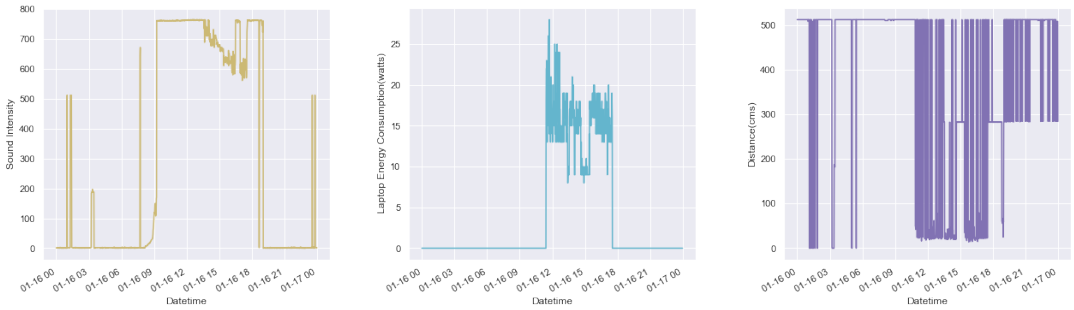
Data exploration is an approach which helps a data analyst to understand the characteristics of data through visual means rather than traditional data management systems. Data exploration is also a way of querying and visualization of data to identify potential relationships that may be hidden in the data.

A.2.1 Line Plots

As we can see from the Figure A.6, the sound intensity is changing rapidly during the working hours ie (9 am to 7 pm) as it depends on the activities of other coworkers too. The energy usages for computer varies during the hours 12 pm to 18 pm indicating the participant was working on his computer during this time. The range of the participant from the sensor box placed on the computer table decreases constantly during the same hours which further strengthens the argument. In other Figure A.7, the multiple motion movements during the working hours captures the activities of other coworkers around the kitchen area. The energy consumption for coffee machine show spikes at fixed intervals indicating the participant initiating drinking activity during those periods. The spikes in the range of participant from the kitchen table also highlight the other activities like eating in the kitchen area.

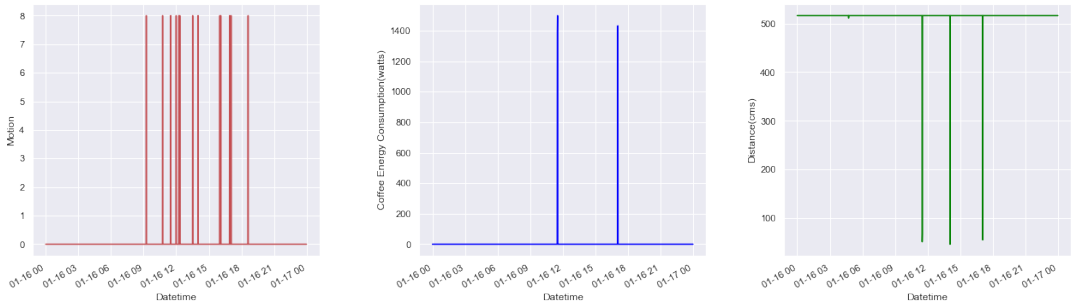
A.2.2 Correlation Maps

As we can see from the correlation plot in the Figure A.8, Energy consumption for laptop usages has a strong positive correlation with motion and sound sensor while a strong negative correlation with range sensor since the person while using laptop sits near the sensor box and also due to the activities sound level increases. Similarly, the range sensor has a strong correlation with intensity



(a) sound vs time **(b)** energy (laptop) vs time **(c)** range vs time

Figure A.6: Hiwi Room



(a) motion vs time **(b)** energy(coffee) vs time **(c)** range vs time

Figure A.7: Kitchen Room

from a sound sensor indicating some sort of activities done by the participant while sitting near the computer desk. As we can see from the Figure A.9 in the kitchen, there is a strong correlation between energy consumption by toaster and coffee machine indicating the participant might be doing both these activities together. Another interesting observation is the distance feature from the range sensor has a strong negative correlation with kitchen appliances energy consumption indicating the person may be standing closer to the kitchen desk while the appliances are running.

A.2.3 Distribution Plots

As we can see from the Figure A.10, the energy consumption from laptop varies. When the laptop is completely charged, the power consumption is around 10 watts which gradually increases to 30 watts per hour when it is charging. Similarly, sound levels vary from 600 to 800 when the person is working on the laptop. The range sensor level shows an inverse relationship and varies from 0 to 100 when the person is sitting closer to the computer desk and reaches to 512 when the person has moved away from the desk.

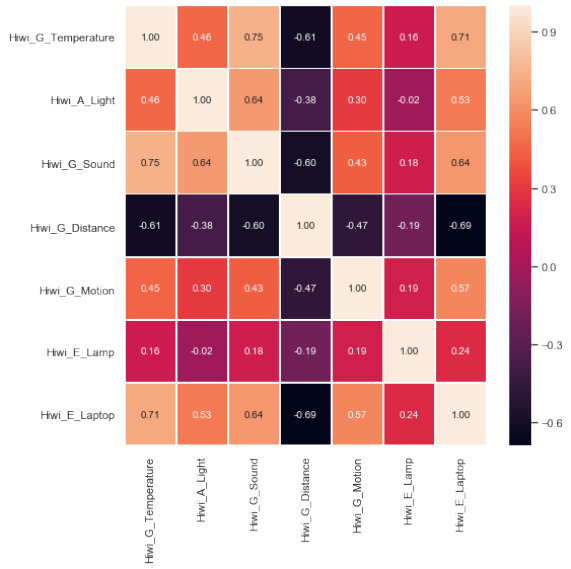
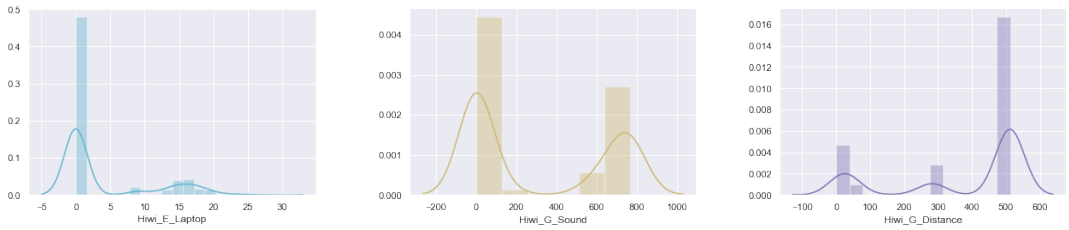


Figure A.8: Hiwi Room Heatmap

Figure A.9: Kitchen Heatmap



(a) Energy **(b)** Sound Intensity **(c)** Distance

Figure A.10: Distribution Plots

A.2.4 Mean Shift Clustering

The energy consumption for a computer as seen in the Figure A.11 is clustered into 2 levels with 0 indicating nobody is using the computer at that time while 1 indicating the various levels of charging and discharging levels of the laptop and the person working on the computer.

The range sensor data points can be clustered into 3 groups (excluding 3 level being an outlier) as shown in the Figure A.12 with 0 level indicating no person is sitting near the computer table while 1 and 2 levels indicating the different sitting positions of person (bending posture towards the desk and leaning against the chair).

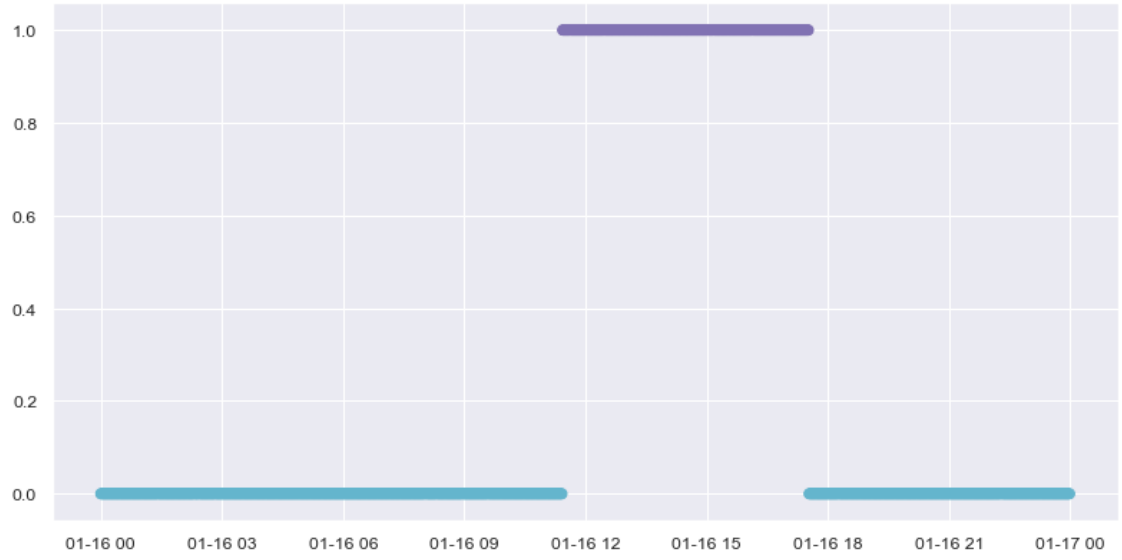


Figure A.11: Energy Mean Shift Clustering

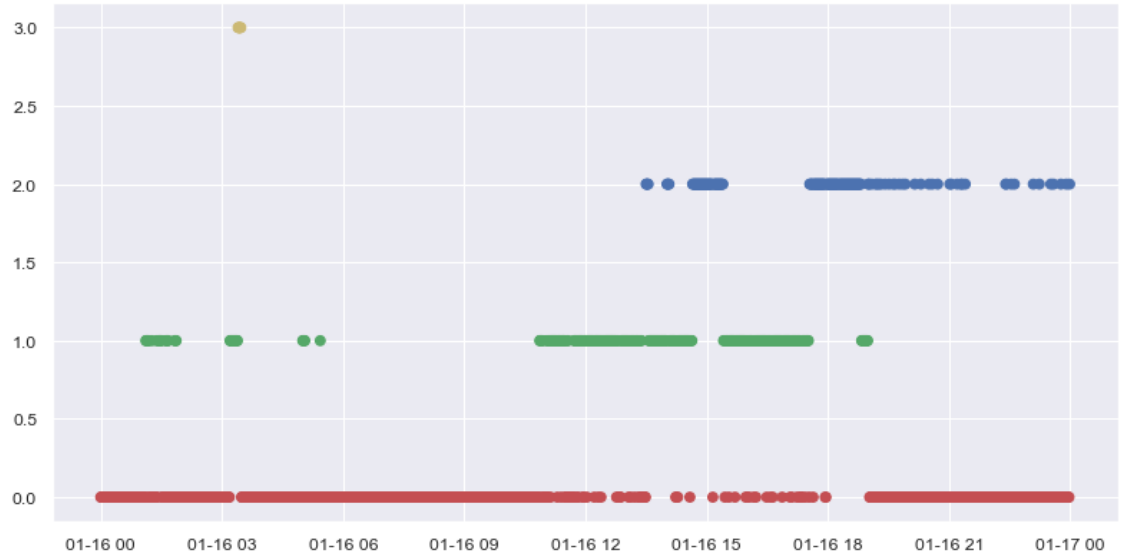


Figure A.12: Distance Mean Shift Clustering

A.3 Evaluation

A.3.1 Visualization

Comparison Plots

Comparison Plots as shown in Figure A.13, Figure A.14, Figure A.15, Figure A.16, Figure A.17, and Figure A.18 between true (y) and predicted values ($\hat{y}$) for the Absent activity using other approaches.

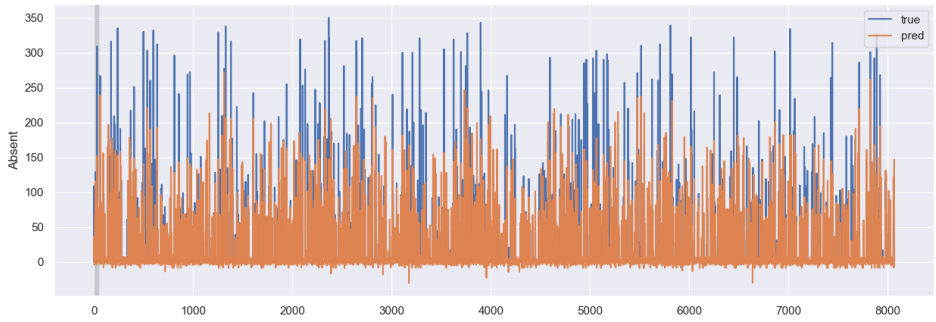


Figure A.13: Comparison Plot for Absent activity using Linear Regression

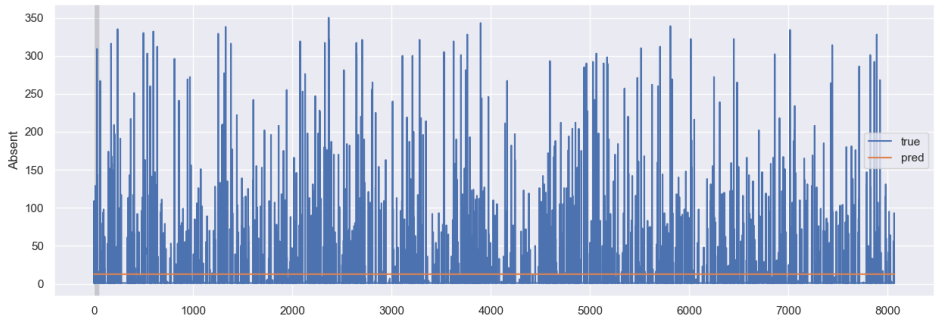


Figure A.14: Comparison Plot for Absent activity using Lasso Regression

Scatter Plots

Scatter plots as shown in Figure A.19, Figure A.20, Figure A.21, Figure A.22, Figure A.23, and Figure A.24 between true (y) and predicted values ($\hat{y}$) for the Absent activity using other approaches.

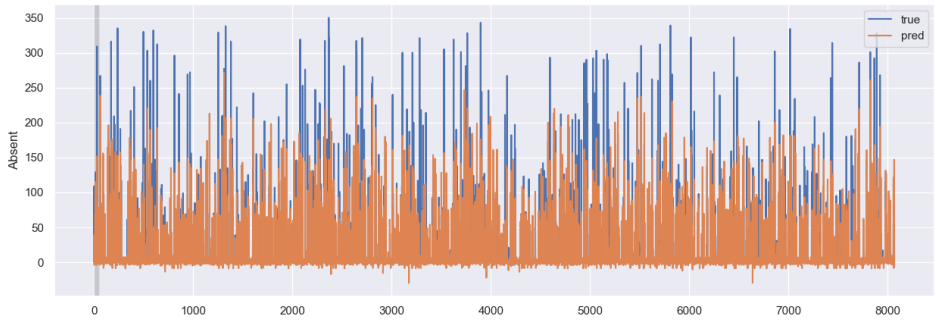


Figure A.15: Comparison Plot for Absent activity using Ridge Regression

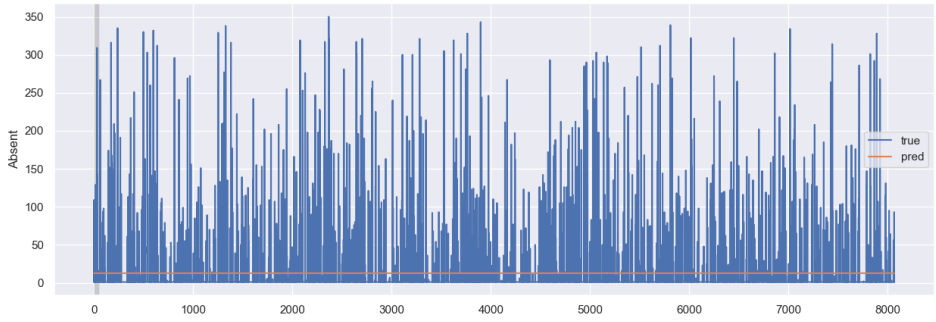


Figure A.16: Comparison Plot for Absent activity using Elastic Net

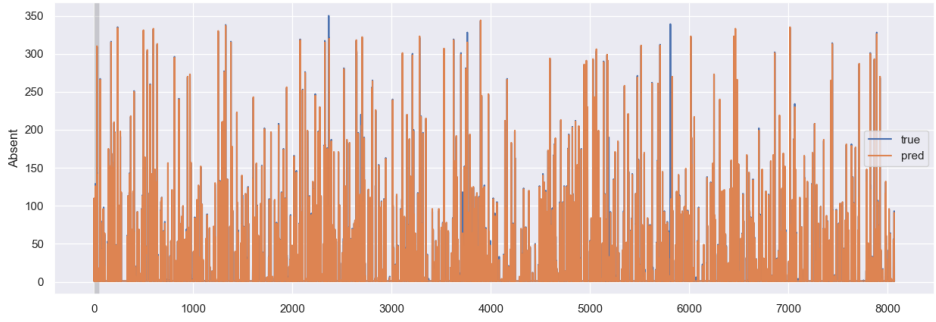


Figure A.17: Comparison Plot for Absent activity using AdaBoost

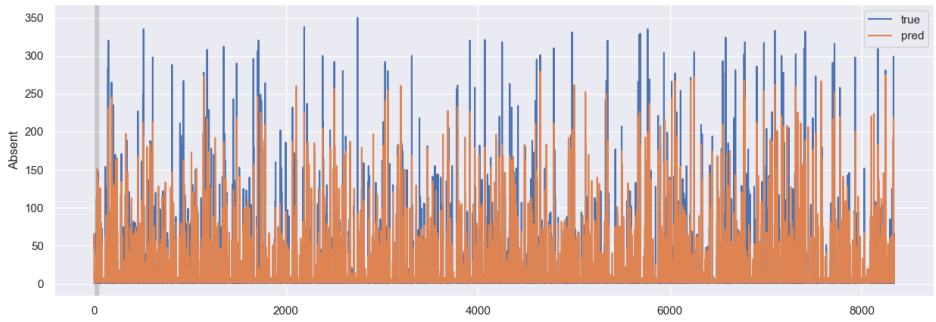


Figure A.18: Comparison Plot for Absent activity using LSTM

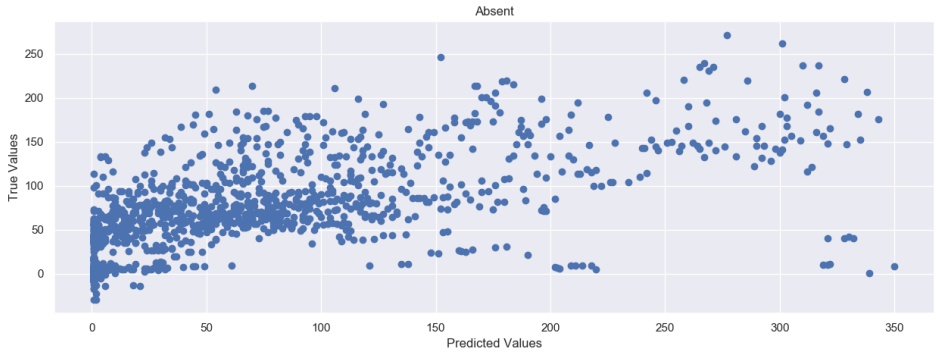


Figure A.19: Scatter Plot for Absent activity using Linear Regression

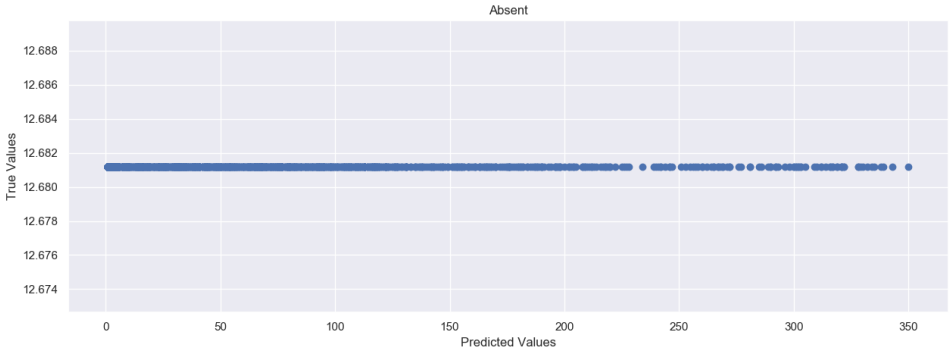


Figure A.20: Scatter Plot for Absent activity using Lasso Regression

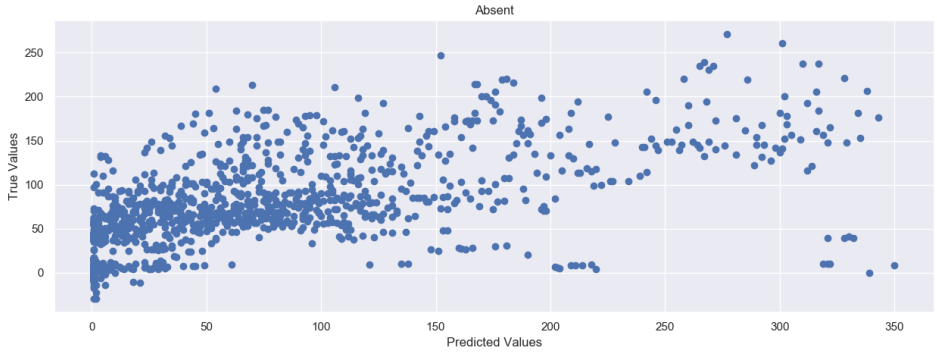


Figure A.21: Scatter Plot for Absent activity using Ridge Regression

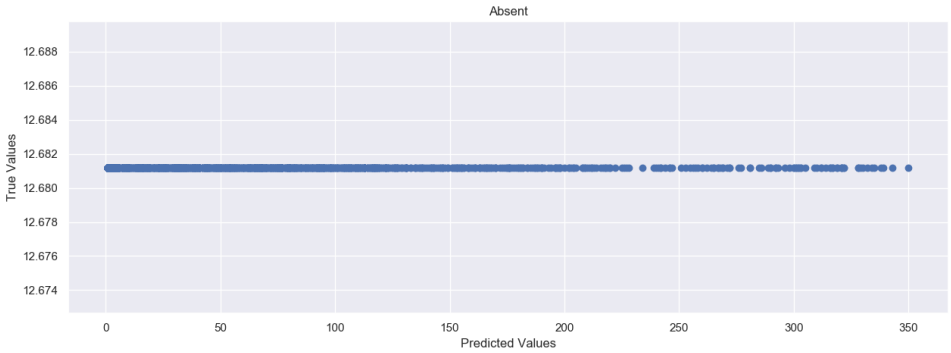


Figure A.22: Scatter Plot for Absent activity using Elastic Net

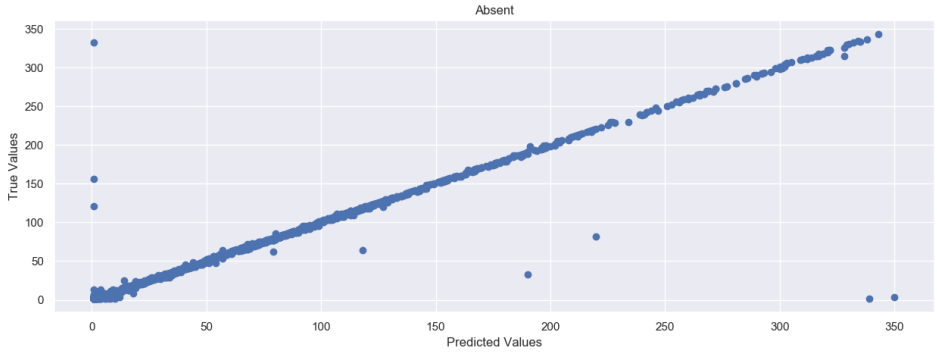


Figure A.23: Scatter Plot for Absent activity using AdaBoost

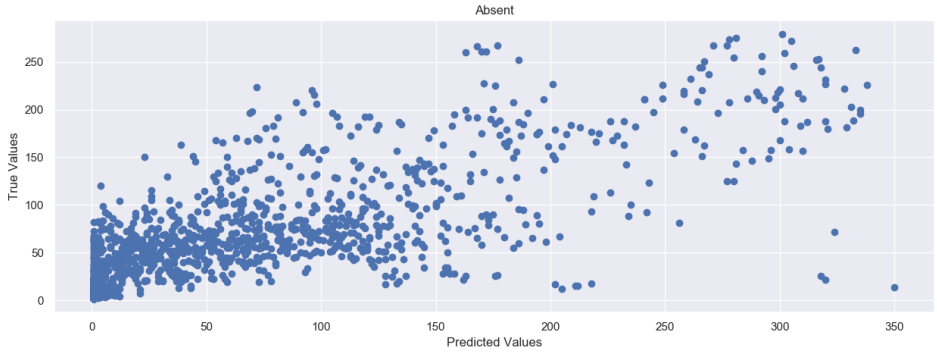


Figure A.24: Scatter Plot for Absent activity using LSTM

A.3.2 scores

Figure A.25 and Figure A.26 shows the comparison between different models using MAE-score and Expected variance score metrics for the defined activities.

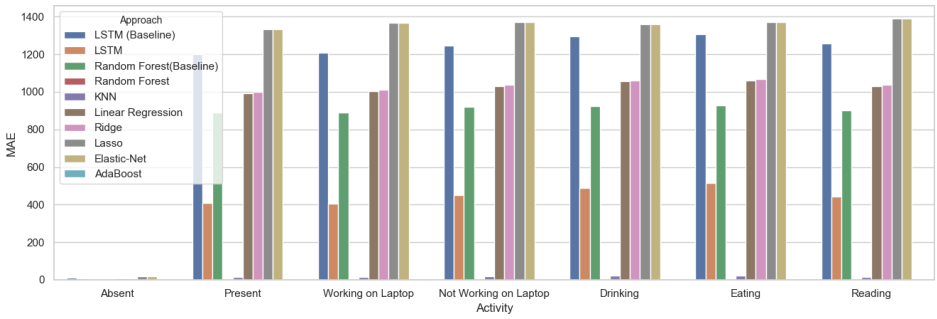


Figure A.25: MAE Score using Different Prediction Approaches

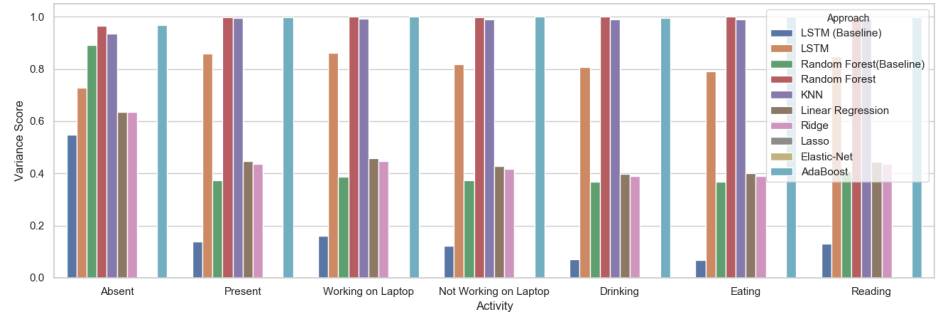


Figure A.26: Expected variance score using Different Prediction Approaches

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature