A PROJECT REPORT ON

MACHINE LEARNING ON REAL-TIME DATA TO ENHANCE HOME AUTOMATION

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE IN THE PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

BACHELOR OF ENGINEERING
(Computer Engineering)

BY

Mr. Ayyaj Attar B120634202
Mr. Sameer Shaikh B120634211
Mr. Siddik Pathan B120634216

Under The Guidance of

Prof. R. Shaikh

DEPARTMENT OF COMPUTER ENGINEERING
AAEMF's
College of engineering and management studies,
Koregaon Bhima
SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE
2016-17

CERTIFICATE

This is to certify that the Project Entitled

MACHINE LEARNING ON REAL-TIME DATA TO ENHANCE HOME AUTOMATION

Submitted by

Mr. AYYAJ ATTAR Exam Seat No: B120634202
Mr. SAMEER SHAIKH Exam Seat No: B120634211
Mr. SIDDIK PATHAN Exam Seat No: B120634216

is a bonafide work carried out by them under the supervision of Prof. R. Shaikh and it is approved for the partial fulfillment of the requirement of Savtribai Phule Pune university, Pune for the award of the degree of Bachelor of Engineering (Computer Engineering).

Prof. R. Shaikh
Guide
Dept. of Computer Engg.

Prof. R. Shaikh
Project Co-ordinator
Dept. of Computer Engg.

Prof. G. S. Kothawale
H.O.D
Dept. of Computer Engg.

Prof. S. Kapse
Principal
AAEMFs College of Engineering

Signature of Internal Examiner

Signature of External Examiner

PROJECT APPROVAL SHEET

A Project Report Titled as

Project Title

Is verified for its originality in documentation, problem statement, proposed work and implementation successfully completed by

Mr. AYYAJ ATTAR Exam Seat No: B120634202
Mr. SAMEER SHAIKH Exam Seat No: B120634211
Mr. SIDDIK PATHAN Exam Seat No: B120634216

at

DEPARTMENT OF COMPUTER ENGINEERING
AAEMFS
College of engineering and management studies,
Koregaon Bhima
SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE
ACADEMIC YEAR 2016-2017

Prof. R. Shaikh	Prof. G. S. Kothawale
Guide	H.O.D.
Dept. of Computer Engg.	Dept. of Computer Engg.

Acknowledgments

It is a great pleasure for us to acknowledge the assistance and contribution of a number of individuals and entities who helped us in presenting this project report on **"Machine Learning On Real-time Data To Enhance Home Automation"**.

First and foremost we wish to record our gratitude and thanks to - our Project Guide **Prof. Rukaiya Shaikh**, the Head of Department **Prof. G.S. Kothawale**, and the **Principal, Mrs. S. Kapse** - for their guidance, instruction and enthusiasm leading to the successful completion of this report.

This acknowledgement would be incomplete without expressing our special thanks to the authors and organizations maintaining the various open-source tools that we have used in our work - especially to Linus Torvalds for his college project that turned into what we know today as the Linux Kernel.

Last but not the least, we would like to thank our parents, friends and our colleagues who helped us directly or indirectly in successfully completing this project report.

Mr. Ayyaj Attar
Mr. Sameer Shaikh
Mr. Siddik Pathan

(B.E. Computer Engg.)

Abstract

Traditional home automation systems are mostly hard-coded or require manual automation plan generation by users. This requires interaction between a control system (an interface) and the user, requiring quite some effort and time to be put in manual planning of the home environment. This project intends to explore applying ML on real-time usage data to generate personalized and time-variant home automation plans. These plans will save the user time and effort, leading to a smoother ML driven home automation experience. We collect streaming usage statistics from smart-home occupants and store it on a centralized server. Simultaneously, we also collect external data (which may consist of environmental factors like natural light intensity, wind speed, et cetera) which may influence occupants usage behavior. These datasets are combined, with data timestamps as a unique identifying field, into a super-set. Its then fed into a Machine Learning system to correlate user habits with time of the day and the external factors. The correlation hence established will be updated in real time. This correlation will be in the form of a prediction model that will be used to predict near future values of target devices for the occupant. Hence, by combining real-time usage data from a conventional home automation system and Machine Learning, we will be able to provide smoother and more comfortable environment to the users, as the burden of plan generation will be greatly reduced.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Detailed Problem Definition

Most present-day smart homes use simple reflex agents for automation. Simple reflex agents are non-flexible and can work with limited percepts and hard-coded actuation rules. These rules may not suit all people. This rigidity in usability of present consumer automation systems forms the core of our problem.

We intend to develop a software solution to this problem, that is centered around machine learning. This solution will be in the form of a learning agent that learns user habits by observation and anomaly detection.

1.2 Brief Description

This project aims to enhance the home automation experience by collecting usage data from the user and applying prediction algorithms on it to predict the next step the user may take. Furthermore, external data will also be collected an correlated with the usage data in order to determine what external conditions may influence the user's behavior. This involves data like weather data and traffic data.

Chapter 2

Analysis

2.1 Project Plan

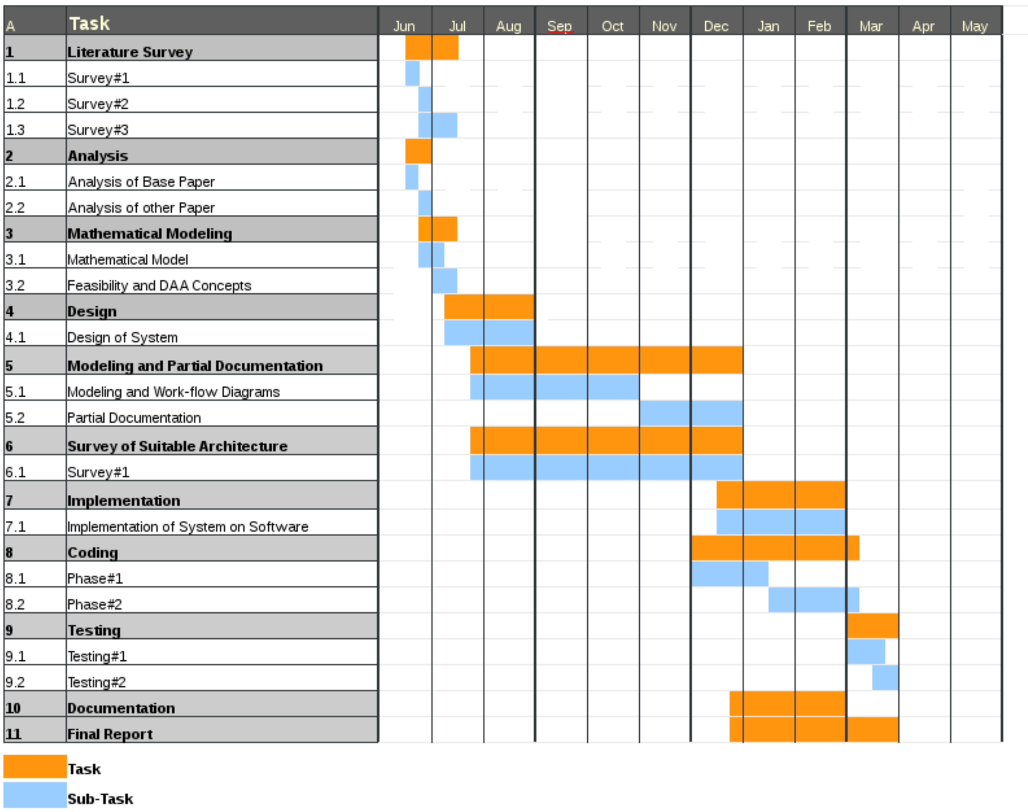

Figure 2.1: Project Plan

2.2 Requirement Analysis

¡Content¿

2.3 Team Structure

¡Content¿

Chapter 3

Design

3.1 Software Requirements

For the PC (LaaS server):

1. Any operating system with a Python interpreter available
2. Python 2.7
3. Flask micro-framework for implementing web service (package “flask”)
4. SciKit Learn ML library for Python (package “sklearn”)

For the Raspberry Pi (device controller):

1. Raspbian OS for the Pi
2. Python 2.7
3. Flask micro-framework for implementing controller’s web interface (package “flask”)

3.2 Hardware Requirements

1. A personal computer
2. A Raspberry Pi Single Board Computer (SBC)
3. Proper network infrastructure

3.3 Software Requirements Specification

3.3.1 Functionality

We intend the software to be divided over two tiers: the interface and the LaaS Server. The user is served up a web interface to the smart devices connected to the Raspberry Pi over a conventional TCP/IP network. It will be possible to view the states of individual devices and change them. Such usage of the system can happen even in the absence of the Machine Learning Service. That is to say, the first tier consists of an independent interfacing system that enables the user to control connected smart devices, and this may optionally utilize a usage-prediction-oriented machine learning service if it is available.

The second tier consists of a special case of SaaS (Software-as-a-Service), called LaaS (Learning-as-a-Service). This service is intended to provide proper WebAPI for the tier one application to communicate with. Data can be sent by the tier one application to the LaaS service using this API, which will then be processed continuously to generate predictions for individual devices. These predictions can also be requested by the tier one application so it can apply it in real-time to the smart environment it is providing an interface to. Non-availability of the services in this tier will not affect the user interfacing system implemented in the first tier, except for the unavailability of predictions.

Unavailability of the first tier, id est interface tier, is fatally deteriorating to the user experience. This tier must always be available and smoothly working, that is to say, it should be robust. The second tier provides a service which is not necessarily central to the functioning of the smart home by design. This means that unavailability of the service will not affect the core control system provided by the first tier.

3.3.2 External Interfaces

Interfacing between different components of this system is crucial to it's execution. We will use conventional communication mechanisms like RESTful WebAPIs for communication between both the client & first tier, and the first & second tier.

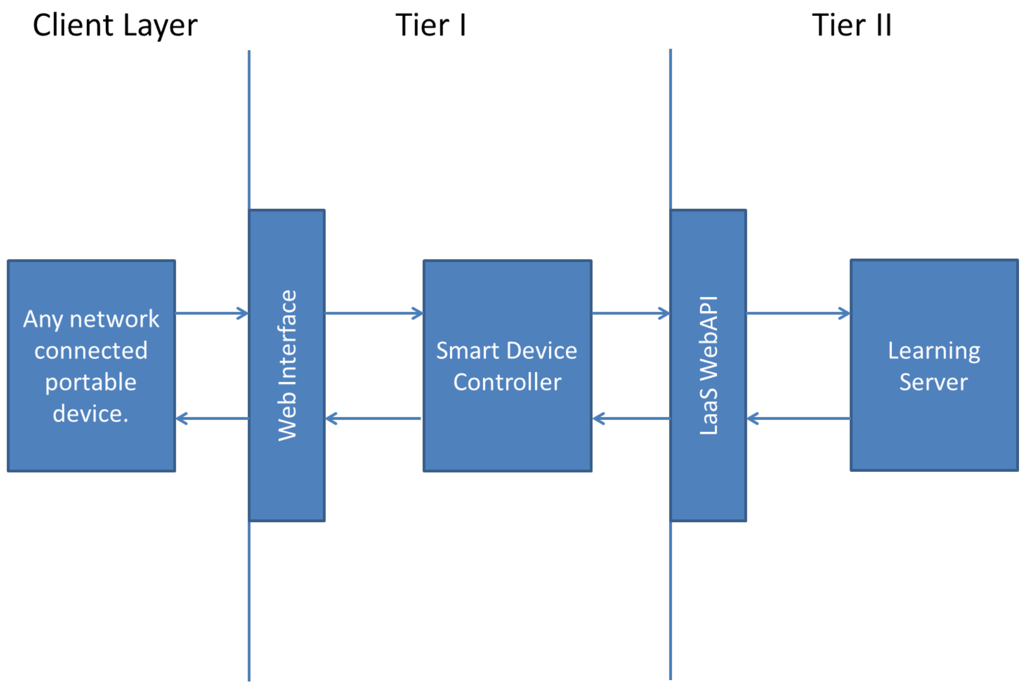

Figure 3.1: Overview of Tiered Components and their Interfacing

Client - Tier I Interface

The smart environment can be controlled via a web app. This app can be accessed by the user by connecting to the smart devices' controller (the Raspberry Pi) on the standard HTTP Port (port 80). The user can then view current states of the connected smart devices and change them according to their preference.

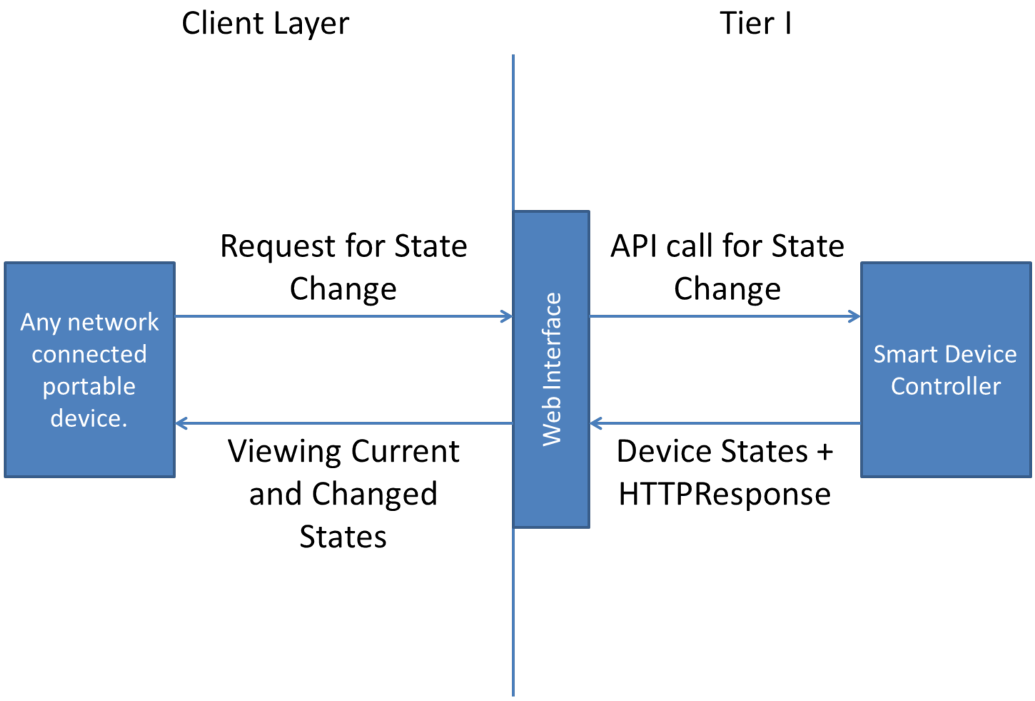

Figure 3.2: Interface between Client and Tier I

Besides collecting user generated actions, the controller may also collect external data (like current weather) in order to facilitate prediction of user actions based on regressive correlation with external influncers.

This interface is critical as it delivers the frontend for our system to the user. Network delay or inconsistencies within its implementation directly and drastically affect user experience.

Tier I - Tier II Interface

The LaaS server can be accessed by the Tier I application through a RESTful WebAPI. The API includes calls for collecting device state changes made by the user and for requesting the prediction based on recent usage data.

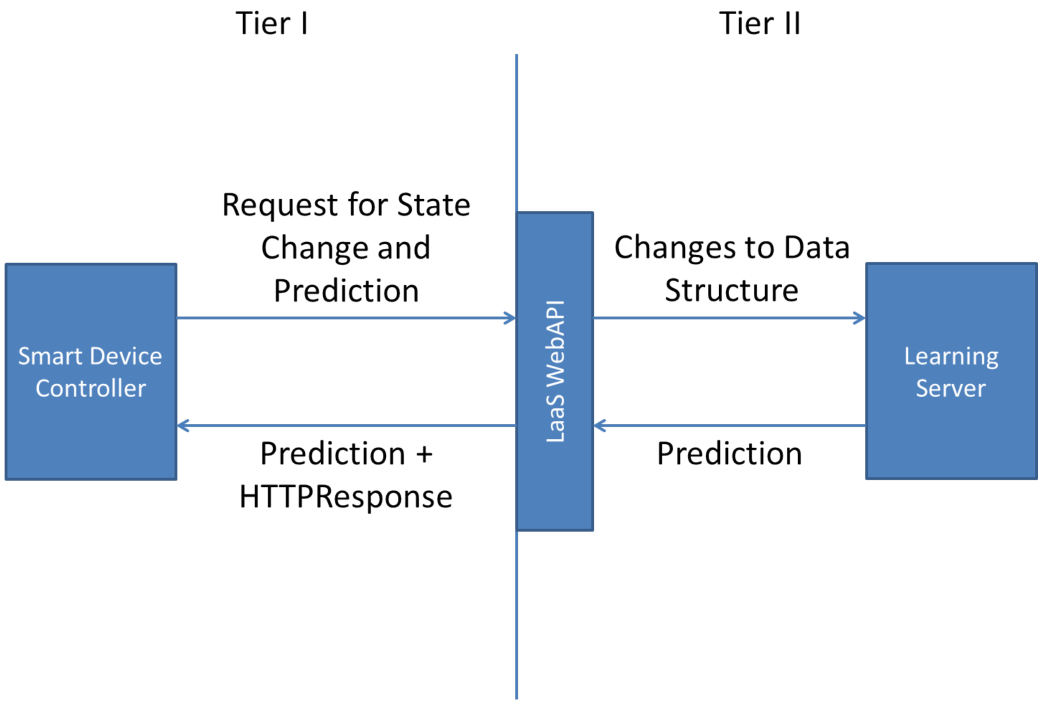

Figure 3.3: Interface between Tier I and Tier II

The only job of the LaaS server is to learn correlations between user actions and external influencing factors, and report the predictions thus generated back to the controller upon request. This server may be hosted locally within the controller's LAN, or publicly, as an on-cloud SaaS Service.

3.3.3 Required Performance

The performance of this system completely depends on the performance of individual machines and the speed their communication. This requires us to again describe the required performance with respect to individual tiers.

Tier I

This tier is performance critical and must always be available to the user. This is mainly because it serves the interface to the smart home environment, which is the most basic functional requirement of our system.

Besides, this tier should be able to work even with the unavailability of Tier II, which provides a supplementary, non-essential service. Also, the delay between the client and Tier I interface should be unnoticeable, as it can directly impact user experience as well as usability.

Tier II

This tier is not as critical to the performance of the system as Tier I. Although unavailability of the services provided by this system can deteriorate the user experience and usability, it will not render the system totally unusable, unlike Tier I.

While the system is running, it is expected to serve up predictions based on current actions quick enough that the user doesn't notice the delay. To achieve this, we need to make sure that the machine that hosts this server is fast enough to achieve this, and also that the network delay between the two tiers is the minimum possible.

3.3.4 Quality Attributes

This section describes the attributes that determine the quality of the software being written. The three attributes that we can identify within this system are Availability, Usability, and Functionality.

Availability

The system should provide at least a minimal device control interface at all times. Although, it should also provide proper and timely predictions for the user's actions most of the time. Then we can say that this system has an acceptable record of availability.

Usability

The system should be easy to handle for the user and provide a simple interface. Delay between an interface action (pressing a button) and response (changes in device states) should be minimal to the point of being unnoticeable. This requires the interfaces between client & Tier I, and Tier I & Tier II to be implemented with the main focus on speed. Such an implementation can be achieved using Responsive UI Design paradigms.

Functionality

Functionality of the system can be judged based on whether the right components and services are provided and work the way they're expected to. This involves observing how well the system responds to user actions and in what ways. Also, the responses generated should be in line with the developed tests. Each test case should be satisfied by the system at least under ideal condition, like availability of services and network connectivity.

Functionality of any system is subject to its usage, environment, et cetera, just as well as it is to its implementation. Nonetheless, we try to include as many possible test cases as are practically possible during the time of development.

3.3.5 Design Constraints

This system is entirely designed using Python (for the back-end on Tier I and II) and HTML/Javascript (for the front-end on Tier I). Tier I is implemented on a Raspberry Pi 2, model B+ - an ARM Cortex A7-based 32-bit quad-core Single Board Computer with 512 MB RAM and frequency of 900 MHz, and running the default Raspbian OS. Python is installed by default on these systems. With this, we need to install the python-flask package - a lightweight framework for implementing robust WSGI applications.

Tier II is targeted towards conventional PCs, and should execute smoothly on mid-level to performance-level PCs as well as workstations and servers. For our implementation, we've used a Linux distribution with Python 2, python-flask (Flask web framework library) and python-sklearn (SciKit Learn ML library) installed. Other operating systems that have a Python interpreter available for them can also be used.

The system can be initiated by running the following individual modules:

1. Raspberry Pi module, which implements the user interface to the smart devices and the connection and control logic for the connected devices. This module forms the Tier I of the system.

2. Server module, which can run on a conventional personal computer and implements the "brains" of the system. That is to say, it provides the system with usage-learning capabilities in the form of a conventional WebAPI. This module forms the Tier II of the system.

For communication, we use conventional TCP/IP network. This implementation is currently targeted towards LAN-based in-home usage and purposefully leaves out even common security features in order to manifest the core idea in a simple way.

Chapter 4

Modelling

4.1 Data Flow Diagram

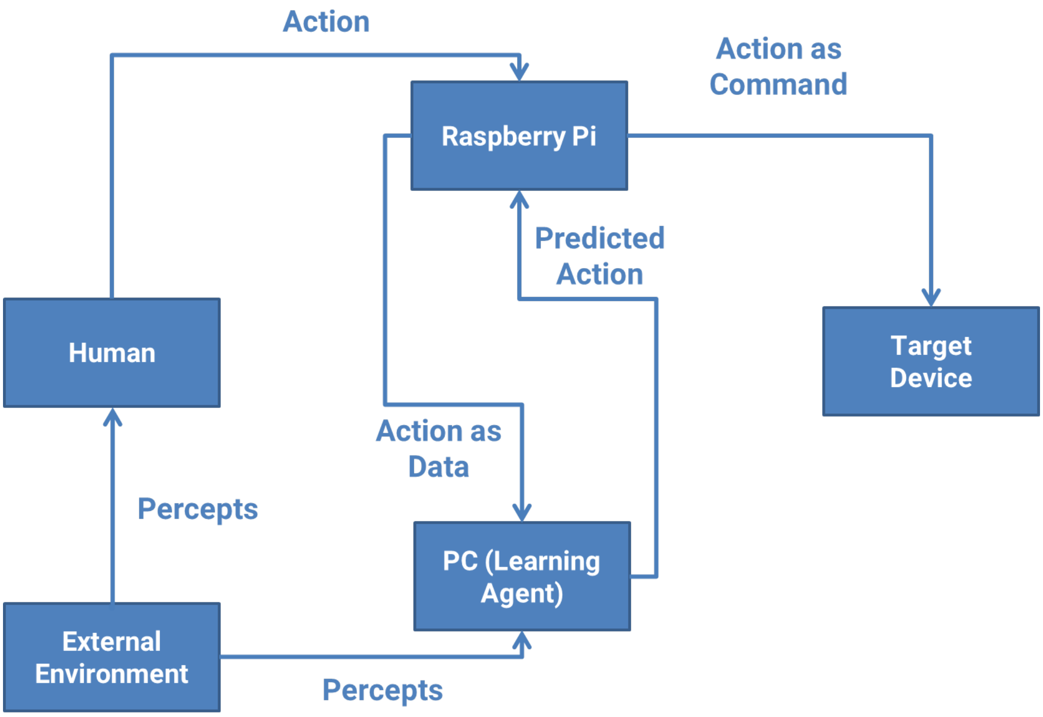

Figure 4.1: Data Flow Diagram

4.2 State Transition Diagram

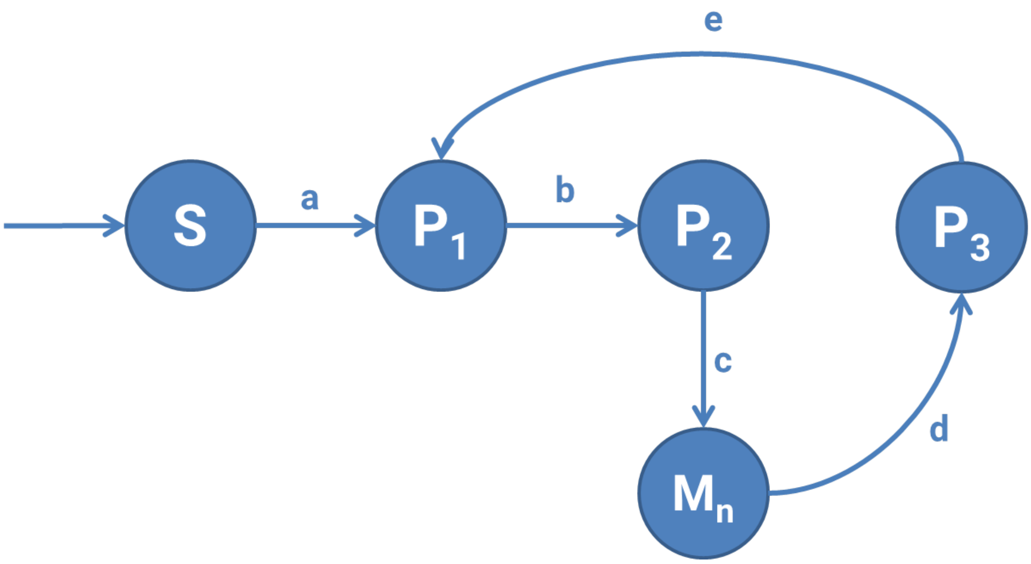

Figure 4.2: State Transition Diagram

Where,

S: Start state

P1: Phase 1- Observation and collection of usage data.

P2: Phase 2- Run machine learning algorithms on collected data to generate a prediction model.

Mn: Prediction Model- Current (nth) prediction model. n is the total number of discrete anomalies detected since start.

P3: Phase 3- Detect anomalies and modify data collected in Phase 1 to match the anomalies.

a: Create a mutable table for holding observed data for the learning agent to process.

b: Data collection threshold reached or data modification completed.

c: Prediction Model generated.

d: Anomaly detected. This implies change in user habit or detection of a new habit.

e: Make corrections in the table to avoid anomalies in the near future.

Our system does not have any state of absolute success. Success and failure are both temporary and the system is designed to learn from its mis-predictions. The canonical success in this project will be the usability of the machine learning system. The more data it is exposed to, the more successful the prediction model will be.

Chapter 5

System Design

5.1 System Architecture

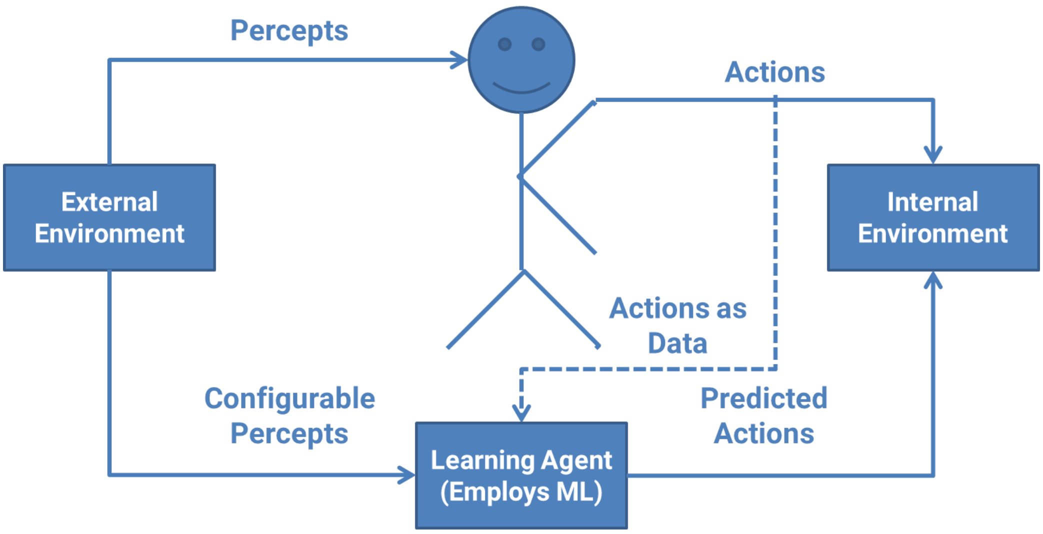

Figure 5.1: System Architecture

This defines the major elements within our system, their arrangement and interaction. It essentially represents an agent-environment model with the human and the Learning Agent as the two agents.

5.2 UML Diagrams

5.2.1 Use Case Diagram

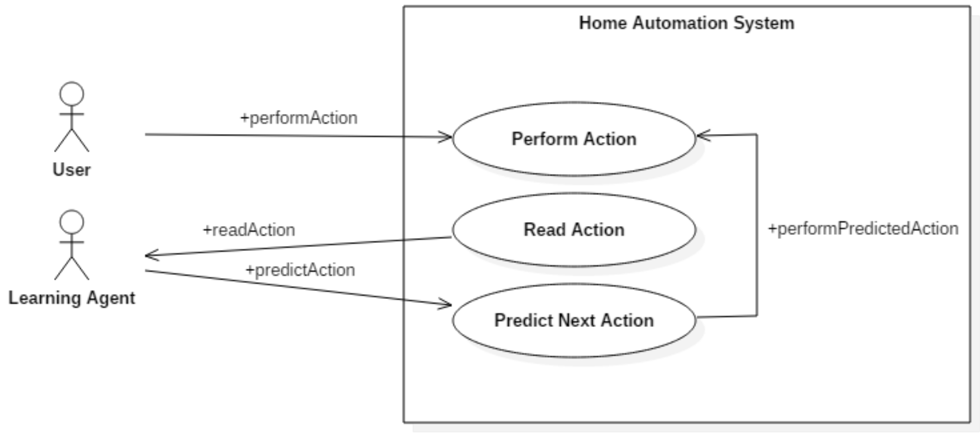

Figure 5.2: Use Case Diagram

This represents the various use cases that are possible in our system. The human user is the primary actor which may perform certain actions to initiate learning in a secondary learning agent that mimics the user's usage patterns.

5.2.2 Class Diagram

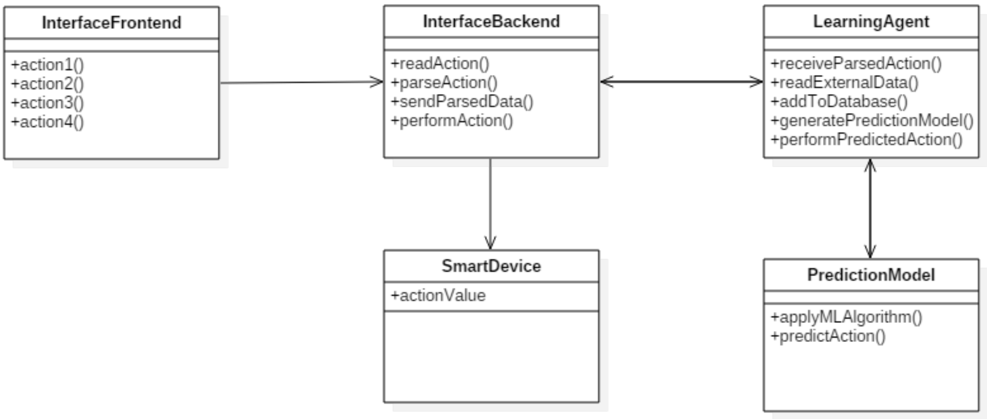

Figure 5.3: Class Diagram

This abstracts the major entities within our system as classes and shows the features they offer and the relationship between them.

5.2.3 Activity Diagram

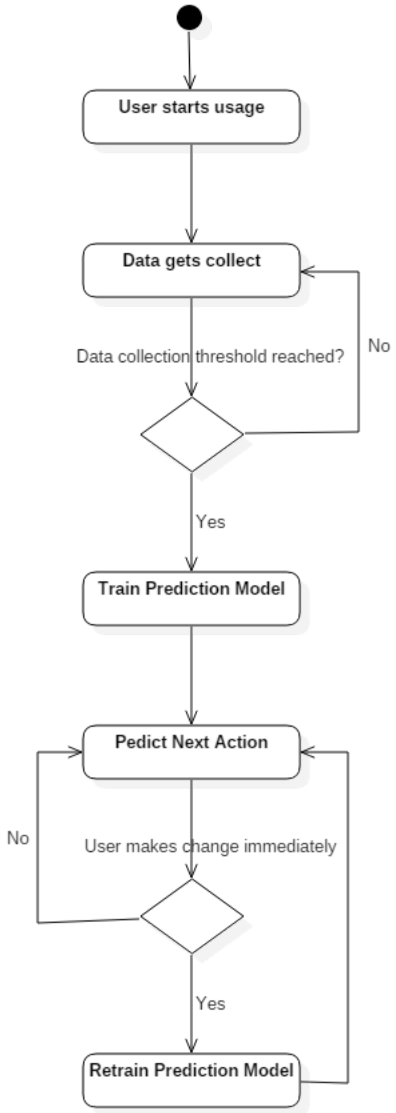

Figure 5.4: Activity Diagram

The activity diagram shows the flow of control through the process of our system. This system is designed to be non-terminating, so there is no halt in the flow. The system improves itself continually by realizing incorrect predictions.

5.2.4 Sequence Diagram

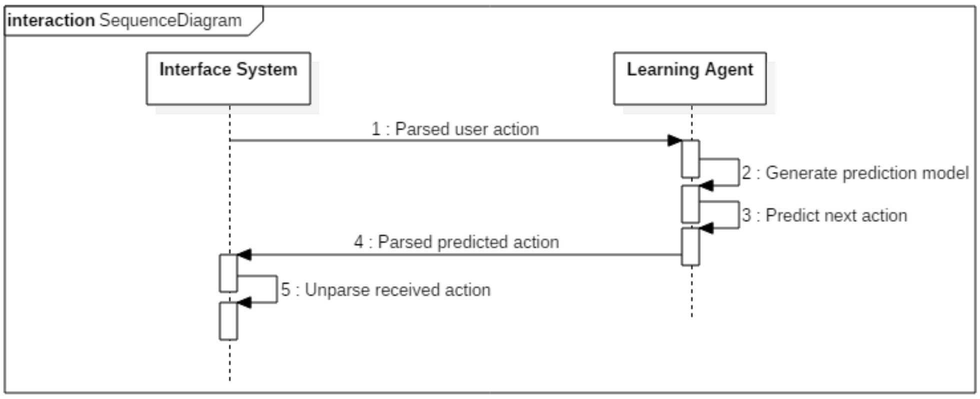

Figure 5.5: Sequence Diagram

It shows the flow of objective-critical data and signals between some major entities within the system.

Chapter 6

Coding

6.1 Algorithm

Logistic Regression Logistic regression is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable. Logistic regression was developed by statistician David Cox in 1958. The binary logistic model is used to estimate the probability of a binary response based on one or more predictor (or independent) variables (features).

6.1.1 Logistic Function

The logistic function is the heart of the logistic regression technique. The logistic function is defined as:

$$transformed = \frac{1}{1 + e^{-x}} \tag{6.1}$$

Where e is the numerical constant Eulers number and x is a input we plug into the function.

Inputs to the logistic function are transformed into the range [0, 1] and smaller numbers result in values close to zero and the larger positive numbers result in values close to one.

6.1.2 Logistic Regression Model

The logistic regression model takes real-valued inputs and makes a prediction as to the probability of the input belonging to the default class (class 0). If the probability is ¿ 0.5 we can take the output as a prediction for the default class (class 0), otherwise the prediction is for the other class (class 1).

Consider that a dataset has three coefficients, for example:

$$output = b0 + b1 * x1 + b2 * x2 \tag{6.2}$$

The job of the learning algorithm will be to discover the best values for the coefficients (b0, b1 and b2) based on the training data.

Unlike linear regression, the output is transformed into a probability using the logistic function:

$$p(class = 0) = \frac{1}{1 + e^{-output}} \tag{6.3}$$

This probability is then used to make a discrete-valued prediction that is then output.

6.2 References to Technology

6.2.1 LaaS: Learning-as-a-Service

This is a special case of Software-as-a-Service. In general, it refers to a range of services that offer machine learning tools as part of cloud computing services, as the name suggests. LaaS providers offer tools including data visualization, APIs, face recognition, natural language processing, predictive analytics and deep learning. But for our application, we will implement only the APIs and predictive analysis part. The server providing the service is responsible for creating and maintaining prediction models based on the data it receives from the clients through its APIs.

We are implementing our own server to provide LaaS. This server will be light-weight, as it is a Tier II component targeted towards conventional personal computers. Implementing this requires creating an API for the clients to interact via and writing back-end logic for the data accumulation and learning procedures. Besides, we need to make sure that the network connectivity is available and the machines can detect each other.

6.2.2 IoT: Internet of "Things"

The Internet of things (IoT) can be defined as the inter-networking of physical devices, vehicles (also referred to as "connected devices" and "smart devices"), buildings, and other itemsembedded with electronics, software, sensors, actuators, and network connectivity that enable these objects to collect and exchange data.

The Internet of Things (IoT) is a term coined by Kevin Ashton, a British technology pioneer working on radio-frequency identification (RFID) who conceived a system of ubiquitous sensors connecting the physical world to the Internet. Although things, Internet, and connectivity are the three core components of IoT, the value is in closing the gap between the physical and digital world in self-reinforcing and self-improving systems.

It can also be defined as an ecosystem of connected physical objects that are accessible through the internet. The thing in IoT could be a person with a heart monitor or an automobile with built-in-sensors, i.e. objects that have been assigned an IP address and have the ability to collect and transfer data over a network without manual assistance or intervention. The embedded technology in the objects helps them to interact with internal states or the external environment, which in turn affects the decisions taken. Simply put, the Internet of Things (IoT) is the network of physical objects that contain embedded technology to communicate and sense or interact with their internal states or the external environment.

6.2.3 Raspberry Pi

The Raspberry Pi is an ARM-based Single Board Computer. It was initially launched in the UK to teach programming to school children but being very cheap yet powerful, it became a favorite among computer hobbyists around the world. The latest revision available at the time of writing this report is revision 3. Raspberry Pi runs it's own flavor of Linux based on Debian called Raspbian OS.

This system is entirely designed using Python (for the back-end on Tier I and II) and HTML/Javascript (for the front-end on Tier I). Tier I is implemented on a Raspberry Pi 2, model B+ - an ARM Cortex A7-based 32-bit quad-core Single Board Computer with 512 MB RAM and frequency of 900 MHz, and running the default Raspbian OS. Python is installed by default on these systems. With this, we need to install the python-flask package - a lightweight framework for implementing robust WSGI applications.

6.2.4 Python

Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991. An interpreted language, Python has a design philosophy which emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly braces or keywords), and a syntax which allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java. The language provides constructs intended to enable writing clear programs on both a small and large scale. Python is an interpreted language with object oriented features.

It features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object-oriented, imperative, functional programming, and procedural styles. It has a large and comprehensive standard library. Interpreters for Python are available for many operating systems, allowing Python code to run on a wide variety of systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation.

Python is a multi-paradigm programming language: object-oriented programming and structured programming are fully supported, and many language features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming. Python uses dynamic typing and a mix of reference counting and a cycle-detecting garbage collector for memory management. An important feature of Python is dynamic name resolution (late binding), which binds method and variable names during program execution.

The design of Python offers some support for functional programming in the Lisp tradition. The language has map(), reduce() and filter() functions; list comprehensions, dictionaries, and sets; and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML. Rather than requiring all desired functionality to be built into the language's core, Python was designed to be highly extensible. Python can also be embedded in existing applications that need a programmable interface. This design of a small core language with a large standard library and an easily extensible interpreter was intended by Van Rossum from the start because of his frustrations with ABC, which espoused the opposite mindset.

6.2.5 Flask (Python Library)

Flask is a micro web framework written in Python and based on the Werkzeug toolkit and Jinja2 template engine. It is BSD licensed. The latest stable version of Flask is 0.12 as of December 2016. Applications that use the Flask framework include Pinterest, LinkedIn, and the community web page for Flask itself.

Flask is called a micro framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Extensions are updated far more regularly than the core Flask program.

The following code shows a simple web application that prints "Hello World!":

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

if __name__ == "__main__":
    app.run('',80)
```

Once this is running, we can simply visit the root of the server (something like "<host_ip>:80/") to see the generated document.

Given below, is a simple web service which performs addition on two numbers provided in the url:

```
from flask import Flask
app = Flask(__name__)

@app.route("/<num1>/<num2>")
def hello():
    return str(num1+num2)

if __name__ == "__main__":
    app.run('0.0.0.0',80)
```

Obviously, this doesn't perform any type checks on provided arguments for validity, but works well as an example, nonetheless. To use the service, we can perform an invocation from the commandline of the same machine as follows:

```
$ curl localhost/4/3
[...]
7
$ _
```

6.2.6 SciKit Learn (Python Library)

Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. Scikit-learn is largely written in Python, with some core algorithms written in Cython to achieve performance. Support vector machines are implemented by a Cython wrapper around LIBSVM; logistic regression and linear support vector machines by a similar wrapper around LIBLINEAR.

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. The library is built upon the SciPy (Scientific Python) that must be installed before you can use scikit-learn.

Extensions or modules for SciPy care conventionally named SciKits. As such, the module provides learning algorithms and is named scikit-learn. The vision for the library is a level of robustness and support required for use in production systems. This means a deep focus on concerns such as easy of use, code quality, collaboration, documentation and performance. Although the interface is Python, c-libraries are leverage for performance such as numpy for arrays and matrix operations, LAPACK, LibSVM and the careful use of cython.

6.2.7 HTML

It stands for Hypertext Markup Language. HTML is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript it forms a triad of cornerstone technologies for the World Wide Web. Web browsers receive HTML documents from a webserver or from local storage and render them into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects, such as interactive forms, may be embedded into the rendered page. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets.

HTML can embed programs written in a scripting language such as JavaScript which affect the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. HTML markup consists of several key components, including those called tags (and their attributes), character-based data types, character references and entity references. HTML tags most commonly come in pairs like ¡h1¿ and ¡/h1¿, although some represent empty elements and so are unpaired, for example ¡img¿. The first tag in such a pair is the start tag, and the second is the end tag (they are also called opening tags and closing tags).

6.2.8 JavaScript

JavaScript is a high-level, dynamic, untyped, and interpreted run-time language. It has been standardized in the ECMAScript language specification.

Alongside HTML and CSS, JavaScript is one of the three core technologies of World Wide Web content production; the majority of websites employ it, and all modern Web browsers support it without the need for plug-ins. JavaScript is prototype-based with first-class functions, making it a multi-paradigm language, supporting object-oriented, imperative, and functional programming styles. It has an API for working with text, arrays, dates and regular expressions, but does not include any I/O, such as networking, storage, or graphics facilities, relying for these upon the host environment in which it is embedded.

JavaScript is also used in environments that are not Web-based, such as PDF documents, site-specific browsers, and desktop widgets. Newer and faster JavaScript virtual machines (VMs) and platforms built upon them have also increased the popularity of JavaScript for server-side Web applications. On the client side, developers have traditionally implemented JavaScript as an interpreted language, but more recent browsers perform just-in-time compilation. Programmers also use JavaScript in video-game development, in crafting desktop and mobile applications, and in server-side network programming with run-time environments such as Node.js.

6.2.9 RESTful Web APIs

Web services that use REST architecture are called RESTful APIs. Representational state transfer (REST) or RESTful Web services are one way of providing interoperability between computer systems on the Internet. REST-compliant Web services allow requesting systems to access and manipulate textual representations of Web resources using a uniform and predefined set of stateless operations. Other forms of Web service exist, which expose their own arbitrary sets of operations such as WSDL and SOAP.

REST is often used in mobile applications, social networking Web sites, mashup tools, and automated business processes. The REST style emphasizes that interactions between clients and services is enhanced by having a limited number of operations (verbs). Flexibility is provided by assigning resources (nouns) their own unique Universal Resource Identifiers. Because each verb has a specific meaning (GET, POST, PUT and DELETE), REST avoids ambiguity.

6.3 Advantages

1. **Raspberry Pi**

   It's a cheap and yet complete, Linux-powered computer that can be used in lieu of many things like conventional microcontrollers, data sinks, and under-utilized workstations to name a few.

   It's size and cost also make it suitable to be used as a device interface in our project. Having a Python library to control its GPIO pins also makes it easier to program.

2. **Python**

   There are many third party modules for it that expand its capabilities.

   Provides a large standard library which includes areas like internet protocols, string operations, web services tools and operating system interfaces.

   Python has built-in list and dictionary data structures which can be used to construct fast runtime data structures.

   Python has clean object-oriented design, provides enhanced process control capabilities, and possesses strong integration and text processing capabilities and its own unit testing framework.

   Python is considered a viable option for building complex multi-protocol network applications.

3. **Flask**

   Flask uses the Django-inspired Jinja2 templating language by default but can be configured to use another language.

   It is light weight and easy to program.

   small web applications as well as light weight services can be rapidly built using Flask.

   Consumes less machine resources - be it memory, processing, or disk space.

   Ideal for small web project that don't expect to have too many users at a time.

4. **SciKit Learn**

By using this library, we are spared the effort of rewriting machine learning algorithms that we intend to use.

Being in Python, importing and using this library becomes as easy as writing conventional Python code.

Sklearn is far richer in terms of decent implementations of a large number of commonly used algorithms as compared to most other libraries.

It is ideal for small scale machine learning applications, like this project.

Does not require users to be an expert at ML, as it provides conveniently packaged modules for a wide variety of models.

5. **JavaScript**

Javascript is executed on the client side, thus saving bandwidth and strain on the web server.

Javascript is relatively fast to the end user as it does not need to be processed in the site's web server and sent back to the user consuming local as well as server bandwidth.

JavaScript can be used to generate dynamic contents within webpages, thus providing dynamism in interactivity.

Due to its flexibility, ease of learning, and speed, it enables developers to design richer interfaces with commendable convenience.

6. **REST Web APIs**

The REST protocol totally separates the user interface from the server and the data storage thus improving the portability of the interface to other types of platforms, increasing the scalability of the projects, and allowing different components of developments to be evolve independently.

Separation of client and server makes it easier to have the front and the back on different servers, and this makes the apps more flexible to work with.

The REST API is always independent of the type of platform or languages - it always adapts to the type of syntax or platforms being used.

6.4 Applications

1. **Raspberry Pi** The Pi is being used to design a smart device enumerator-cum-controller. For the purpose of this demo, we've hardcoded some devices in our interface itself. The Pi will provide the user with a convenient web interface for controlling the smart devices. It will also use the machine learning services from Tier II to perform predictive analysis of the user's actions and influencing factors. This computer will be deployed at Tier I, which means that it will directly be interacted with by the user from the client layer.

2. **Python** Python is used to develop both the tiers. It is being used to implement the Tier I as it is easy to program the Pi using Pi. Besides this, we are using it to implement a Learning Service at Tier II, as the language has support for various platforms. Also, there are a variety of modules available for the language, from low-level hardware control to web services frameworks.

3. **Flask** Flask is being used by the Tier I to implement an API for the controller's interface, which will be utilized by the client to control the smart devices. Due to the flexibility and vastness of the REST architecture, we're using Flask at Tier II too. At Tier II, Flask is used to implement a web service API for the Learning Service. At Tier II, Flask will have to be used with SciKit Learn library to provide ML services to Tier I applications.

4. **SciKit Learn** This library is being used to implement learning capabilities for the system. It will be used at Tier II with the web services framework to provide Learning services for Tier I application. The best part about using this library is the ease of use. We do not need to be experts at ML to be able to utilize the many useful learning models that have been implemented in the library. Besides, being meant for Python, we've had to write less amount of code to implement even sophisticated learning models using SciKit Learn.

5. **RESTful WebAPIs** These are APIs for web services that are designed using the REST paradigm. We are using these for both client-Tier I and Tier I-Tier II interfaces. At the client-Tier I interface, we use the this type of API for use by the client to control smart devices. To make any changes to the state of a device, the client device performs a call to a pre-crafted REST API that contains the device identifier and the state the user intends to switch it to. At the Tier I-Tier II

interface, the API is used by Tier II to provide the Learning Service to Tier I application. The Tier I application will perform API calls to send usage data periodically, and each time one of the device states of external parameters gets modified, it will request for prediction of the states of dependent devices.

Chapter 7

Result Sets

¡MORE INFO NEEDED¿

Chapter 8

Testing

8.1 Test Plan

¡Content¿

8.2 Test Cases

¡Content¿

8.3 Test Results

¡Content¿

Chapter 9

Conclusion

With the rise of Internet of Things and more powerful computers, we will be able to achieve Utopian homes using Smart Automation powered by Machine Learning Algorithms of higher complexity than Temporal Difference based Reinforcement Learning running on current data. More data will lead to better prediction of potential user action which will help us lead more comfortable lives. Physically challenged people can also benefit from such systems which will eventually make them more independent, as more and more data gets collected to predict such users' habits. The impact of this technology on human lives will be deep and possibly every human-machine interface in the future will have some form of machine learning powered intelligent assistance. Changes in user habits can also be used to predict a lot of other things about the user such as the user's physical and mental health. In fact, many current technologies aim to provide basic level medical assistance using machine learning systems.

References

[1] Panos Luridas, Christof Ebert,"Machine Learning", IEEE Computer Society, 2016

[2] Jeremie Saives, Clement Pianon, and Gregory Faraut, "Activity Discovery and Detection of Behavioral Deviations", in IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING, 2016

[3] Christopher Osiegbu, Seifemichael B. Amsalu, Fatemeh Afghah, Daniel Limbrick and Abdollah Homaifar, "Design and Implementation of an Autonomous Wireless Sensor-based Smart Home", IEEE, 2015

[4] Wesllen S. Lima, Eduardo Souto, Richard W. Pazzi, Ferry Pramudianto, "User Activity Recognition for Energy Saving in Smart Home Environment", IEEE Symposium on Computers and Communication, 2015

[5] Vahid Ghasemi, Ali Akbar Pouyan, "Activity Recognition in Smart Homes Using Absolute Temporal Information in Dynamic Graphical Models", IEEE Computer Society, 2015

[6] Beichen Chen, Zhong Fan, and Fengming Cao, "Activity Recognition Based on Streaming Sensor Data for Assisted Living in Smart Homes", IEEE Computer Society, 2015

[7] K.S.Gayathri, Susan Elias, S.Shivashankar, "Composite activity recognition in smart homes using Markov Logic Network", IEEE Computer Society, 2015

[8] Labiba Gillani Fahad & Muttukrishnan Rajarajan,"Anomalies detection in smart-home activities", in IEEE 14th International Conference on Machine Learning and Applications, 2015

[9] Z. Meng, and J. Lu, "A Rule-based Service Customization Strategy Context- aware Automation for Smart Home", in IEEE TRANSACTIONS ON MOBILE COMPUTING, 2014

[10] Chiming Chang, Paul-Armand Verhaegen, Joost R. Duflou, “A Comparison of Classifiers for Intelligent Machine Usage Prediction”, IEEE Compuer Society, 2014

[11] Parisa Rashidi, Student Member, IEEE, and Diane J. Cook, Fellow, “Keeping the Resident in the Loop: Adapting the Smart Home to the User”, in IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, 2009

Plagiarism Check Report

Paper Published Details