IntelLight+

PDEng THESIS REPORT

IntelLight+: Designing and Developing of Context-Aware Lighting Solution

Hossein Mahdian

October 2021

Department of Mathematics & Computer Science

PDEng SOFTWARE TECHNOLOGY

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

IntelLight+: Designing and Developing of Context-Aware Lighting Solution

Hossein Mahdian

October 2021

Eindhoven University of Technology
Stan Ackermans Institute – Software Technology

PDEng Report: 2021/076

Confidentiality Status: Public

Partners	Signify, Eindhoven Engine	Eindhoven University of Technology
Steering Group	Dr. Tanir Ozcelebi Dr. Fetze Pijlman Dr. Bernt Meerbeek PDEng Dr. Dzmitry Aliakseyeu	
Date	October 2021	

Composition of the Thesis Evaluation Committee:

Chair: Dr. Tanir Ozcelebi

Members: Dr. Fetze Pijlman

BSc Mohamed Elkady

Dr. Bernt Meerbeek PDEng

Dr. Dzmitry Aliakseyeu

Dr. Gijs Dubbelman

The design that is described in this report has been carried out in accordance with the rules of the TU/e Code of Scientific Conduct.

Contact Address Eindhoven University of Technology
Department of Mathematics and Computer Science
Software Technology
MF 5.080 A
P.O. Box 513
NL-5600 MB
Eindhoven, The Netherlands
+31 402744334

Partnership This project was supported by Eindhoven University of Technology and Eindhoven Engine.

Published by Eindhoven University of Technology
Stan Ackermans Institute

PDEng-report 2021/076

Preferred reference IntelLight+: Designing and Developing of Context-Aware Lighting Solution. The Eindhoven University of Technology, PDEng Report PDEng 2021/076, October 2021

Abstract The Intelligent Lighting Institute (ILI), part of Eindhoven University of Technology (TU/e), and Signify aim to address future challenges for intelligent lighting solutions. They aim to set the best light setting for different indoor activities using machine learning (ML) approaches. The Philips Hue is the product of Signify, an intelligent lighting pilot in this project. The result of this project enables adding the new context-aware lighting feature to the Hue system. The IntelLight+ system provides the infrastructure for developing ML algorithms needed to infer users' context to set appropriate light settings based on users' activities, needs, and preferences. IntelLight+ takes care of the ML life cycle for the intelligent lighting system.

This report elaborates on the context and technical needs for the IntelLight+ system by analyzing the problem domain, formulating the requirements, and describing the intended use cases. It explores the solution domain and proposes an architecture that emerges from the feasibility study and requirement analysis, followed by the identification of components and their integration. The project management, verification, and validation processes are also described in this document.

Keywords PDEng, Software Technology, TU/e, Context-aware lighting, Activity Detection, Intelligent Lighting, Machine Learning (ML), ML pipeline, MLOps, IoT

does it assume any legal liability, whether direct or indirect, or responsibility for the accuracy, completeness, or usefulness of any information.

Trademarks

Product and company names mentioned herein may be trademarks and/or service marks of their respective owners. We use these names without any particular endorsement or with the intent to infringe the copyright of the respective owners.

Foreword

In our drive to deliver the best lighting experience to our customers, the role of intelligent lighting is indisputable. Intelligent lighting simplifies user interactions with the system, and it enables continuous automatic adjustments to the customer's needs. Classification of activities is an essential ingredient in the intelligent lighting program, which lead to the birth of the IntelLight+ project.

Classification of activities that take place at home is a challenge as information from sensors is limited, and the diversity among homes is large. If one is able to solve that challenge, then another question is how classifications with finite accuracy can be used to deliver the right type of lighting. In order to address these challenges, there is a need for a platform on which experiments can be run, experiments that contain Machine Learning algorithms that translate a variety of sensor events into new lighting scenes. This platform we call a machine learning pipeline.

The assignment of Hossein was to develop this machine learning pipeline for experimentation of various algorithms. Developing a platform for experimentation of algorithms for use cases is complex when use cases and algorithms themselves are being developed. A new use case modifies inputs, outputs, and metrics, and a new algorithm may need to be trained with a different training strategy. Hossein has helped us in identifying the commonalities, and he has created a pipeline on which experiments are currently being executed. Results from these experiments will enable us to set the next step in delivering the best lighting experience.

Fetze Pijlman

September the 29th, 2021

Preface

The Professional Doctorate in Engineering (PDEng) in Software Technology (ST) at the Eindhoven University of Technology (TU/e) is a two-year technological designer program to prepare a candidate for proficiency in high-tech inter-disciplinary projects. At the final stage of the program, several design projects are proposed by various companies, and candidates are elected to take on a ten-month-long project based on their interest and fitting criteria.

This report describes the final PDEng project supervised by Dr. Fetze Pijlman on behalf of Signify and proposed and guided by Dr. Tanir Ozcelebi as the TU/e supervisor. The purpose of the project is to design a system that can meet the researchers' and data scientists' needs in having an infrastructure for building context recognition models for intelligent lighting. This project gives the researchers a tool for comparing and visualizing the performance of various solutions, deploying the selected models, monitoring the model's performance, and optimizing the models based on the received feedback.

The report covers the problem analysis, explores the domain where the problem is formulated, specifies the requirements of the system of interest, presents the design criteria and solution candidates, describes the implemented solution and its evolutionary development process, and summarizes the outcomes of using the implemented system.

Hossein Mahdian

October 2021

Acknowledgments

I would like to express my gratitude to everyone who supported and guided me through this project. Their contributions were essential for successfully completing this project. I want to especially thank my supervisor Fetze Pijlman who shared his ideas and guided me through this project. Your feedback regarding the implementation was essential for successfully completing this PDEng graduation project. Moreover, I absolutely enjoyed our discussions regarding various topics, and it was a pleasure hearing your thoughts as well as being able to share mine.

I am very thankful to my supervisor Tanir Ozcelebi. As an expert in the intelligent IoT domain, you have helped me understand the domain concepts and guided me through this unfamiliar terrain. I really appreciate your feedback and support not just in the technical but also with the non-technical aspects of the project. Your collaboration was absolutely essential for successfully completing this project.

I would like to express my heartfelt gratitude to my PDEng Software Technology program director, Yanja Dajsuren. I learned many things from your guidance and feedbacks throughout my PDEng years. I believe that all your advice will help me further in my career.

I am truly thankful to my other company supervisors Bernt Meerbeek and Dzmitry Aliakseyeu. I absolutely appreciate your critical attitude. You always asked the right questions that were essential to steer the project in the right direction. I really enjoyed and learned a lot from the discussions we had.

I am thankful to all my colleagues from PDEng program for your feedback and shared experiences during the last two years. Together we solved many challenges and learned a lot. I would like to especially thank PDEng ST secretary Desiree van Oorschot for her logistical support throughout the PDEng program.

Most importantly, I want to express my deepest and most sincere gratitude to my family. Throughout my PDEng, you remained by my side and provided unconditional support, encouragement, joy, as well as love. Moreover, I am eternally grateful to my parents and my sisters in Iran for their continuous support. Your trust, support, and encouragement is the blessing of my life.

Lastly, I like to express my gratitude to all the wonderful teachers who taught me a lesson—the teachers who brought the light of knowledge and wisdom to human lives. Behind the scenes, many different aspects of our lives are influenced by human teachers, researchers, explorers, and the ones yet to come.

Executive Summary

Scientists who research perception studies psychology, chronobiology, and lighting design, looking for the right light setting for different events and activities. In case of knowing the context of the environment, it is possible to study the appliance of related light settings to specific contexts. With the advances in data science and ML, it has become feasible to study the possibility of lighting devices that can setups the light setting based on context recognition. The added ML module to the lighting solution uses the data from the environment and user to detect the context.

Signify grew from a lighting equipment provider to a solution provider company in the lighting business. As a lighting solution provider, Signify delivers software-based solutions as a managed service to its customers. In this path, intelligent lighting plays an essential role in moving current solutions towards the next step.

As a step forward in enabling Signify to provide managed services for intelligent lighting, the IntelLight project co-initiated with TU/e. The focus of the IntelLight project is delivering high-resolution, robust, flexible, and privacy-preserving algorithms for context recognition. This project, IntelLight+, advances the IntelLight project by providing infrastructure to explore the ML approaches. IntelLight+ is designed and implemented to automize the process of ML by considering the extendability and flexibility in mind. By flexibility, it allows employing a wide range of training algorithms, libraries, and methods for context recognition. By extendability, it provides an API that can be implemented for adding new algorithms for context detection.

One of the constraints originates from the fact that the model training process is poorly reproducible. The training is an iterative process, which means data scientists launch several times the same run with slightly different parameters/data / preprocessing. If they rely on their memory to compare these runs, they will likely struggle to remember the best one. IntelLight+ visualizes the ML experiments with different parameters, data, and the code version that has been used to produce the model.

Besides making the experiments reproducible, IntelLight+ allows data scientists to develop models in local machines and cloud then deploy the models on-premise or cloud by a shared storage. After deployment, it allows applying the received feedback from users. Besides the main parts of the ML pipeline, a software component was developed to enable giving feedback regarding the quality of the deployed model.

During the IntelLight+ project, we validated the requirements with discussion with main stakeholders and the developed system verified by the direct stakeholders and running demos. The IntelLight+ infrastructure is ready to be used to help the data scientists manage and visualize the data, hyper-parameter tuning, and code resources used in experiments. It also covers deployment models in different target environments in an easy and convenient way.

Glossary

API	Application Programming Interface
AWS	Amazon Web Services
CI	Continuous Integration
CD	Continuous Delivery
CT	Continuous Training
DevOps	Development Operations
DVC	Data Version Control
FR	Functional Requirement
GCP	Google Cloud Platform
IDE	Integrated Development Environment
ILI	Intelligent Lighting Institute
IoT	Internet Of Things
IP	Internet Protocol
IRIS	Interconnected Resource-aware Intelligent Systems
JSON	JavaScript Object Notation
LED	Light Emitting Diode
ML	Machine Learning
MLOps	Machine learning operations
PEP	Python Enhancement Proposal
PhD	Doctor of Philosophy
PDEng	Professional Doctorate in Engineering
PIR	Passive infrared
PSG	Project Steering Group
QA	Quality Attribute
QAS	Quality Attribute Scenarios
R&D	Research and Development
REST	Representational State Transfer
SAI	Stan Ackermans Institute
ST	Software Technology
TU/e	Eindhoven University of Technology
YAML	Yet Another Markup Language

Table of Contents

List of Figures

List of Tables

Introduction

Over the past centuries, we created artificially illuminated environments where we live, work, rest, and refresh. People depend on artificial lighting in their environments to continue such activities. The lighting plays an essential role in the effects of an atmosphere on people's mood, well-being, and on their behavior. However, we lose natural light and its positive qualities for being exposed to the same light throughout our lives. Apart from that, light-sensing is related to visual perception, which varies from person to person.

In order to understand and evaluate how well the lighting system can support its users with varying interests, needs, and preferences, one needs to focus on specific scenarios. In this project, we focus on particular lighting scenarios that take place at home space. The lighting system's users may do different activities such as work, have dinner with family, refresh, etc., within the home environment.

This chapter presents the general context of this project. Section 1.1 gives a brief description of Signify and Intelligent Lighting Institute (ILI) of Eindhoven University of Technology (TU/e) as the initiators of this project. Section 1.2 describes the generic context of the project. In section 1.3, the objective and the motivation of this assignment are introduced. Finally, section 1.4 presents the overall structure of the document.

1.1 Project Partners

TU/e and Signify have a long history of collaboration in R&D and technological innovations for a considerable amount of time. During this time, the collaboration has already led to several transfers of the outcome of R&D, which successfully transformed into meaningful technological innovations that subsequently found their ways into new/modified methods for designing, developing, manufacturing products/services.

The assignment described in this report is part of a ten-month collaboration between TU/e and Signify under the auspices of the Software Technology design program. This Professional Doctorate in Engineering (PDEng) program is offered by Stan Ackermans Institute 4TU. School for Technological Design. Stan Ackermans Institute (SAI) is a federation of four leading Dutch technical universities: TU Delft, TU Eindhoven, University of Twente, and Wageningen University. The federation aims at maximizing innovation by concentrating the strengths in research, education, and knowledge transfer of all technical universities in the Netherlands. The SAI manages more than twenty post-graduate technical designer programs across the four technical universities. Each designer program is intended to teach the skills needed to design the complex systems needed in the high-tech industry to master's graduates starting or taking their careers to the next level.

1.1.1. Signify

Signify [1], previously known as Philips Lighting, is the world leader in lighting for professionals, consumers, and lighting for the Internet of Things (IoT). Signify manufactures lighting-related products under various brands, including Philips, Interact, Philips Hue, Color Kinetics, and WiZ. Signify's portfolio comprises electric lights, the IoT platform, and connected lighting systems aimed at consumers and professionals. Signify's energy-efficient lighting products, systems, and services enable their customers to enjoy a superior quality of light and make people's lives safer and more comfortable, businesses more productive, and cities more livable.

With a presence in over 70 countries, the company's purpose is to unlock the extraordinary potential of light for brighter lives and a better world. Signify achieves this through living its values, innovation, passion for sustainability, and desire to transform people's lives.

1.1.2. TU/e Intelligent Lighting Institute (ILI)

ILI [2] was founded on April 8, 2010, and is part of TU/e. ILI aims to create a scientific community of practice dedicated to intelligent lighting solutions with a scientific and application-based approach towards all aspects of light. The key is establishing partnerships with stakeholders in the public-private field: companies, knowledge institutes, and government bodies. ILI is taking a multi-disciplinary and multifunctional approach to achieve a breakthrough as the work in this field is concept-driven and evidence-based.

Members of the Interconnected Resource-aware Intelligent Systems (IRIS) research group of TU/e Computer Science are active contributors to ILI research, and they address challenges in (distributed embedded) systems performance such as timing behavior, dependability, programmability, reliability, robustness, scalability, accuracy, energy, and data computation efficiency, and trustworthiness. This project provides data and structure that is required for IRIS researchers.

1.2 Project Context

Intelligence is the capacity for logic, abstract thought, understanding, self-awareness, communication, learning, emotional knowledge, memory, planning, creativity, and problem-solving. Over the years, researchers have been trying to mimic some of these aspects of intelligence using machines to improve the quality of human life. Intelligent lighting is one such application where the objective is to learn and provide the most suitable lighting conditions in an environment.

Human-centric lighting should benefit users, but its implementation is no clear-cut process. Light affects human health and well-being in many ways. Among others, it powerfully regulates our internal circadian rhythm but also drives visual performance, comfort, and experience. Our biological clock requires very bright light for higher visual performance, although most of us are not consciously aware of this need; the comfort typically makes people dim the light. Moreover, lighting needs and preferences differ widely between individuals and within one person, for instance, with the time of day, task, or company [3]. Furthermore, we are all very much aware of the pressing need to save energy.

The current project builds on the latest insights from new sensors, IoT developments, and machine learning (ML) approaches. The Intelligent lighting system consists of two projects. The first one is the research project (IntelLight) that focuses on delivering high-resolution, robust, flexible, and privacy-preserving algorithms for context recognition technology. This is a four-year project that a Ph.D. student from the IRIS research group from TU/e works on. The second project, IntelLight+, is a design project that PDEng trainee performs on that. It provides the infrastructure for developing algorithms needed to infer, and even predict ahead of time, a user's context to accommodate appropriate light settings based on users' needs and preferences.

The focus of the IntelLight+ project is to automatize the manual works data scientists need to do to gather data, run experiments, building models, and deploying models for testing in different environments. The general pipeline and the deployment module of the IntelLight+ system can be used in different projects within the TU/e as Machine Learning Operations (MLOps). Our pilot intelligent lighting in this project is Philips Hue that contains numerous connected devices (bridge, sensors, and lightbulb) that communicate with each other. The challenge here is to design and develop a learning system that learns users' preferences through contextual data (the values of relevant features) gathered from the breakout area either implicitly via sensors or explicitly from the users. Later, this knowledge is used to predict suitable lighting conditions for user activities in the home environment.

1.3 Objective and Motivation

In the ten-month period of the PDEng project, we deliver a pipeline that covers steps in the ML lifecycle from data collection to inference. IntelLight+ automizes the process of ML that can employ a wide range of training algorithms from supervised learning to semi-supervised learning and online learning approaches for context recognition. The objective is to have a flexible system that manages the steps

needed in the life cycle of ML systems that are not necessarily part of a training algorithm. The IntelLight+ project, together with the IntelLight research project, is going to add the new context detection feature to the existing Hue lighting system platform.

ML is about manipulating data and developing models. These separate parts together form an ML project life cycle, which is the main motivation behind this project. IntelLight+ takes care of the ML life cycle for the intelligent lighting system. Models are produced in the lifecycle of the ML system that the training algorithm is a small part of it. The life cycle of an ML project can be represented as a multi-component flow, where each consecutive step affects the rest of the flow. The nature of the work of data scientists includes a lot of experiments. Being able to visualize the experiments with the version of data and configuration that has been used and automizing the deployment of produced models gives data scientists a real advantage.

1.4 Outline

In the remainder of this document, we first provide an in-depth problem analysis in chapter 2. After that, chapter 3 addresses the feasibility of the project, chapter 4 describes the use cases and provides an overview of the requirements. Then, in chapter 5, we describe the architecture and design of IntelLight+. Following that, we describe the implementation in chapter 6 and how we verified and validated IntelLight+ in chapter 7. Then, in chapter 8, we describe the project management process. Next, we provide a conclusion and directions for possible future work in chapter 9. Lastly, chapter 10 provides a retrospective of this project.

Problem Analysis

This chapter analyzes the problem this project is going to solve. Section 2.1 defines the problem, and section 2.2 describes the project goal. The project road map and general information about the Hue system are mentioned in section 2.3. Section 2.4 mentions the general technology roadmap of the project, the project scope, assumptions, and constraints follow in section 2.5. The stakeholders' interests, needs, and involvement in the project will be discussed at the end.

2.1 Problem Definition

This project builds an intelligent lighting system (IntelLight+) to predict the contexts relevant to the lighting within the home. The context recognition system can learn the user preferences by collecting data from multiple sources and offer automatic lighting that fits user needs and make lighting recommendations. An ML pipeline is required in order to recognize the context successfully. ML pipeline automates the life cycle of intelligent lighting. It focuses on the following questions:

1. How can data scientists use different data sources easily for context-aware lighting?
2. How can data scientists keep track of the code and data used for building the model?
3. How can data scientists try different models and reproduce a model?
4. What are the steps (in the system) for choosing the best parameters for the trained models?

2.2 Project Goals

As the possibilities for digital, connected lighting develop and expand, Signify's smart lighting infrastructure becomes the glue that connects the physical world to the users' well-being and digital realm, creating a true "Internet of Lights." In this brave new world of connected intelligence, lighting is an integral part of our everyday environments.

The integration of the lighting infrastructure with the IoT, including the integration of new functionalities based on the true capabilities of LEDs, will offer new validated user-centric value propositions. Lighting is the application domains in the physical environments (such as offices, industrial sites, homes, and smart cities) or industrial cases (such as automotive, horticulture, and healthcare). The challenge lies ahead in creating highly engaging, bright, and lit environments, which are also thoughtful of today's sustainability demands. This project addresses the challenge of new context awareness, enabling more intelligent light control.

For intelligent lighting control to accommodate user needs and preferences in different contexts, algorithms are needed to infer and predict a user's context ahead of time. Human activities and contexts need to be recognized accurately based on information collected by the surrounding sensing infrastructure and wearables. We aim to provide a system that enables context recognition technology framework and algorithms needed by home lighting solution providers by solving the issues arising from the typical lack of labeled training data, data imperfections, diversity of sensor types, models, topology, and user privacy concerns.

This PDEng software design project is part of the IntelLight+ project funded by the Eindhoven Engine. It aims to design and implement a system for context recognition, which will facilitate the work of a Ph.D. candidate in the IntelLight project funded via the TKI-Toeslag subsidy of Stichting TKI-HTSM (Holland High Tech). The Ph.D. researcher will focus on developing and testing the context recognition algorithms. The design should be flexible enough to let different types of algorithms be implemented and then compare the results of different learning approaches. However, the detailed algorithms are outside of the scope of this PDEng design project. The design project considers the system's life cycle

and deployment of different learning algorithms so that in the research project, data scientists (Ph.D. students) will develop the required algorithms easier.

2.3 Product Roadmap

Modern-day bulbs are not what people have been using traditionally. These light bulbs have gone through significant changes to get to where they are now.

In 1835, there was the first sign of constant electric light. Forty years later, scientists worked to develop what today we know as the incandescent light bulb. Then came a filament made of a rare metal called tungsten. These light bulbs became more luminous, consumed less energy, and lowered the cost of production.

Light Emitting Diodes (LEDs)[4] use semiconductors (generally Gallium Arsenide) to emit light. Basically, LEDs are tiny bulbs that emit light in a single direction. Many small bulbs are fit together in a LED bulb to emit light. Unlike incandescent light bulbs, LEDs don't have a filament bulb that gets hot and breaks after a certain amount of time; instead, they illuminate by the movement of semiconductor electrons.

2.3.1. Philips Hue

In 2012, Signify (previously known as Philips Lighting) launched the Philips Hue system. Hue is a connected home lighting system of linked bulbs that a smartphone or tablet can control via a ZigBee bridge that connects to the home router through the Ethernet.

Philips Hue is one of the leading and most installed innovative home / IoT products globally. Philips Hue enables color-tunable lights to be controlled from smartphones, web services, or other control logic and devices running in the system. Furthermore, it is an open system, i.e., via standardized or published interfaces, other suppliers can add smartphone apps, services, light switches, and lamps.

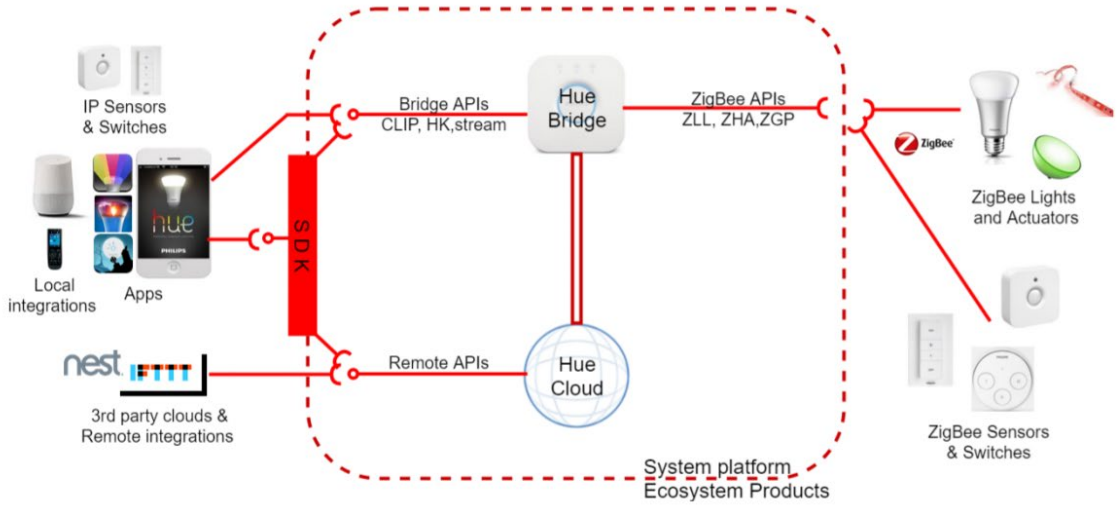


Figure 2-1: Hue system high-level overview [5]

Figure 2-1 shows a high-level overview of the Hue system and its main components. Hue lamps communicate via a standardized ZigBee protocol allowing integration with ZigBee-based devices, including sensors and light switches. The Hue bridge handles the home automation, and via the Hue bridge, the ZigBee network is connected to the home IP network and the Internet. On the IP network side, there are smartphones, web browsers, third-party services, and a Hue portal.

The Hue API interface allows developers to make use of the functionality of the Philips Hue system. Using this interface, they can find information about the available devices in their local network, control them, and do much more. The Hue API is a RESTful JSON interface in which clients interact with resources in the Philips Hue system. Resources such as devices, groups, and lights in the Philips Hue system are represented by a unique URI that is interactable.

2.3.2. Lighting Evolution

To better understand the intelligent lighting roadmap, let us briefly look into the history of lighting control. Table 2-1 shows the evolution of lighting control. Initially, lighting systems could be controlled only using switches and dimmers. With the rapid development of sensors and semiconductor technologies, lighting systems could be controlled using sensors without switches. This has made it possible to emphasize user satisfaction and energy usage optimization by controlling lights' intensity and illumination patterns[6]. Revolution of memory devices enabled storing users' light preferences[7] or scenario-specific light settings[8] on the lighting systems to provide customized/adaptive lighting. The development of data mining techniques and ML algorithms has made learning possible from the data or through interactions. Intelligent lighting improves adaptive lighting where a predefined light setting is not stored for a context but is learned through the system's interaction with the environment. This means that a lighting condition need not be preset for a given context, and it may be adapted over a period based on the responses received from the environment.

Table 2-1: Lighting solution generations

Type of lighting	User Intervention	Behavior	Controlled By
Traditional	Maximum	User has to control each light manually separately	On/Off Switch
Autonomous	-	Users not needed for controlling lights	Sensor-based rules
Adaptive	Minimum	User gives his/her preferences to the system	Personal settings
Intelligent	Minimum	Lighting system learns user's preferences	ML model

2.4 Technology Roadmap

ML is the science of developing systems that can learn from data and act accordingly without programming the behavior of the application or specifying thresholds explicitly. The most widely accepted definition from Mitchell [9] is as follows.

"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."

In intelligent lighting, ML is used to determine the relationship between the context and the output lighting conditions. The input-output relationship is then used to predict suitable lighting conditions for new inputs. Applying Mitchell's definition to intelligent lighting, we have,

- Task T: predicting suitable lighting conditions for a given context
- Performance Measure P: quality of the predicted lighting conditions by the user survey
- Experience E: a sequence of observed contexts and the user response

2.4.1. Learning approaches

Supervised learning is the ML task of identifying the relationship between input and output from labeled training data[10]. In supervised learning, a prediction algorithm is trained using a dataset consisting of training examples. Each sample is a pair consisting of values for input features (context) and the corresponding class (output). During training, the algorithm analyzes the samples in the dataset and generates hypotheses. A hypothesis represents the relationship between the input and output that can take the form of functions, rules, or trees[11]. The hypothesis that provides the best prediction performance, *h*, is then used to make predictions for future inputs.

Figure 2-2 shows the framework for supervised learning for intelligent lighting. An intelligent lighting dataset consists of samples, i.e., input-output pairs collected from a pilot implementation. The input/context is a vector of features that may influence users' preference for particular lighting conditions. The output is the preferred lighting condition by the user for a given context. A supervised prediction algorithm is trained on the dataset to generate a hypothesis *h*, which is used to predict a suitable lighting condition for new incoming inputs.

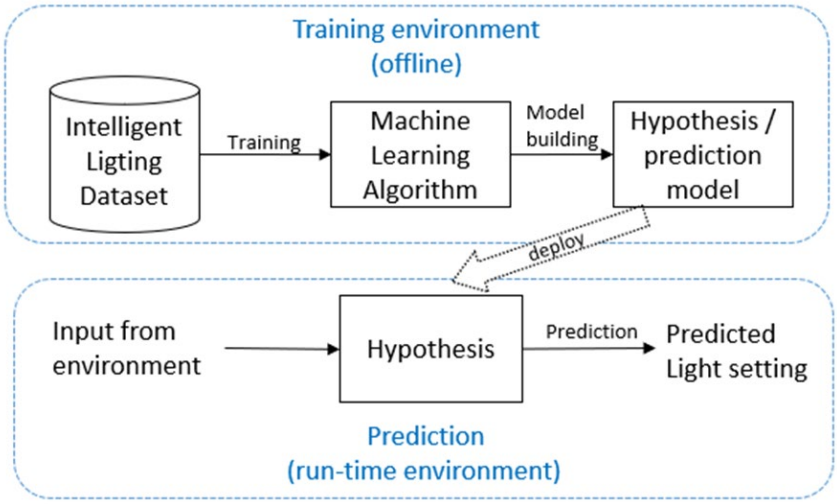


Figure 2-2: Intelligent lighting supervised learning framework

Instance-based learning

In this form of learning, the input-output relationship is not deduced when the training samples are provided, whereas it is deduced when a new input arrives that needs to be predicted. In other words, instance-based learning algorithms do not generate any valuable representations from the observed inputs[12]. The relationship to the already encountered samples is determined to assign a class label for a new instance. The main advantage of this kind of learning is that the class label is estimated every time a new instance is predicted rather than estimated at once[9]. This makes instance-based learning worthwhile when the input-output relationship changes over time. Figure 2-3 gives the framework for instance-based learning for intelligent lighting. For example, let us consider the popular instance-based algorithm, K-nearest Neighbor (KNN). In KNN, when a new input arrives, it computes the distance of the input to all other observed inputs in the intelligent lighting dataset. For a selected value of *k*, *k* samples that are the closest to the input distance are chosen. A lighting condition that appears the maximum number of times among the k selected samples is selected for prediction. A user may select a lighting condition that suits them if the user is not satisfied with the predicted lighting condition. The selected lighting condition, along with the input, is stored in the dataset so that the predictions can be improved for new inputs.

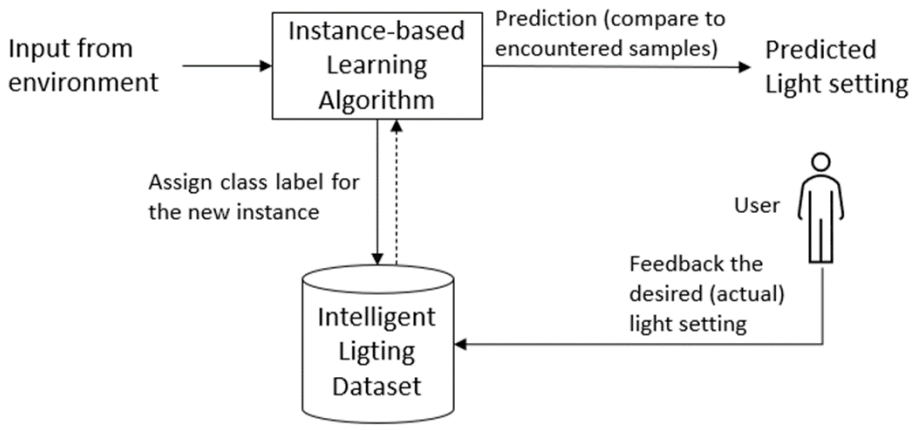


Figure 2-3: Intelligent lighting instance-based learning

Online learning

Online learning is an ML approach that learns from one sample at a time. In online learning, the input arrives as a sequence of samples in contrast to supervised learning, where the input is fed as a batch[13]. As an input arrives, the online learning model should predict an output. Immediately after the prediction is made, the actual output is made available to the model. This information can be used as feedback to update the prediction hypothesis used by the model. A key advantage of online learning over supervised learning is that the model adapts itself with every incoming new sample. Unlike instance-based learning, online learning models are relevant in applications where huge amounts of data need to be stored for training purposes. Figure 2-4 shows the framework for online learning for intelligent lighting. The input from a pilot implementation in the form of a feature vector is fed to the online prediction algorithm. The prediction algorithm predicts a suitable lighting condition for that input. A user may select a lighting condition that better suits them if the user is not satisfied with the predicted lighting condition. The lighting condition preferred by the user (actual output) is then revealed to the prediction algorithm in the form of feedback. The prediction algorithm uses this feedback to update its hypothesis in case the predicted lighting condition is not the same as the user preferred lighting condition.

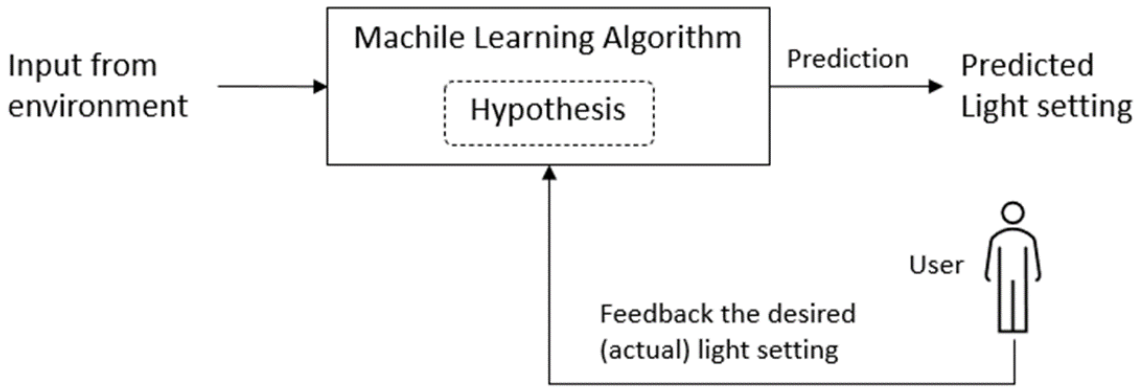


Figure 2-4: Intelligent lighting online learning framework

2.4.2. ML pipeline

When a data science team runs a few ML models, having a manual process for gathering data, preprocessing, training, and deploying is sufficient, especially when these models do not have to be retrained and reproduced frequently.

As the number of models increases in most organizations, we need to introduce the concept of Continuous Training (CT), which brings automation in the execution of an ML pipeline[14]. This is a vital capability to keep the models up to date when the world around them changes.

The ML pipeline, also called the model training pipeline, is the process that takes data and code as input and produces a trained ML model as the output. This process usually involves data cleaning and pre-processing, feature engineering, model and algorithm selection, model optimization, and evaluation.

There are several key differences between the software development process and the data science workflow, resulting in different processes and tools for the latter. The software building process delivers an artifact. The test/ quality assurance process approves the desired functionality, and the artifact is deployed in production. While operating the application, the discovered bugs are fed back to the development process, forming the circle of the continuous workflow.

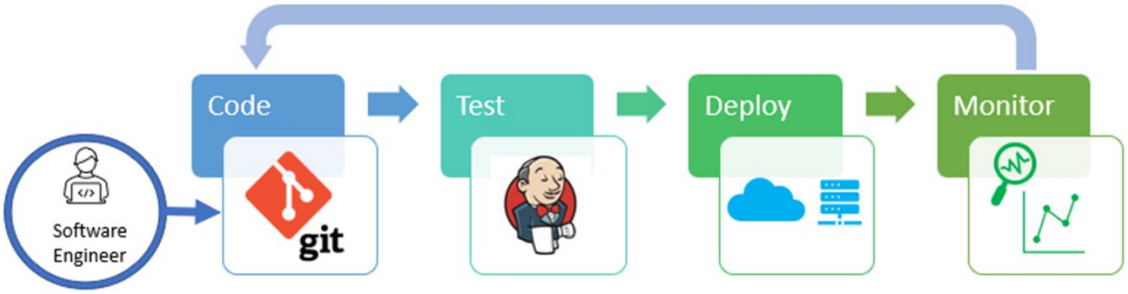


Figure 2-5: Software development workflow

The data science workflow delivers a model through a selection process of combinations of algorithms, parameters, and metrics (such as accuracy). It is then deployed in production to infer knowledge from the data. This inference process is prone to the input data. The dynamic relationship between the delivered model and the data reflects the difference between the software workflow shown in Figure 2-5 and the data science workflow shown in Figure 2-6.

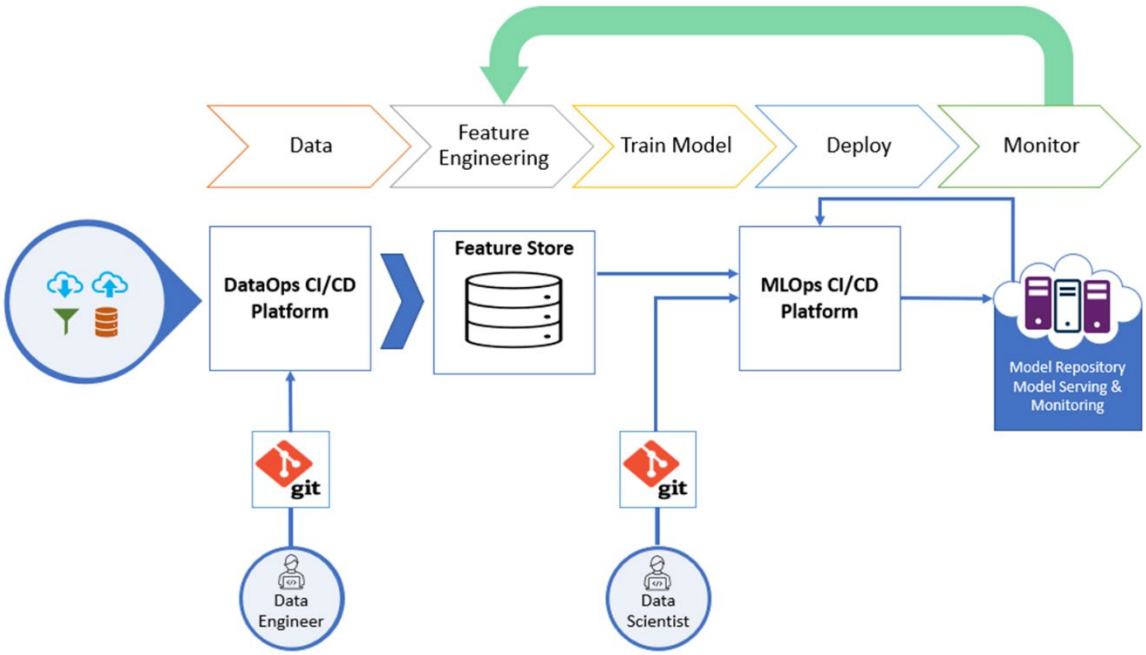


Figure 2-6: Data science workflow

Building and maintaining a platform to support building and serving ML models requires careful orchestration of many components. The fact that data are part of the workflow introduces the need for Continuous Training (CT), as an addition to Continuous Integration (CI) and Continuous Delivery (CD). This extra quality contributes to the step from DevOps to DataOps and MLOps.

Continuous Delivery is the ability to get changes of all types — including new features, configuration changes, bug fixes, and experiments — into production, or into the hands of users, safely and quickly in a sustainable way.

Besides the code, changes to ML models and the data used to train them are another type of change that needs to be managed and baked into the software delivery process (Figure 2-7).

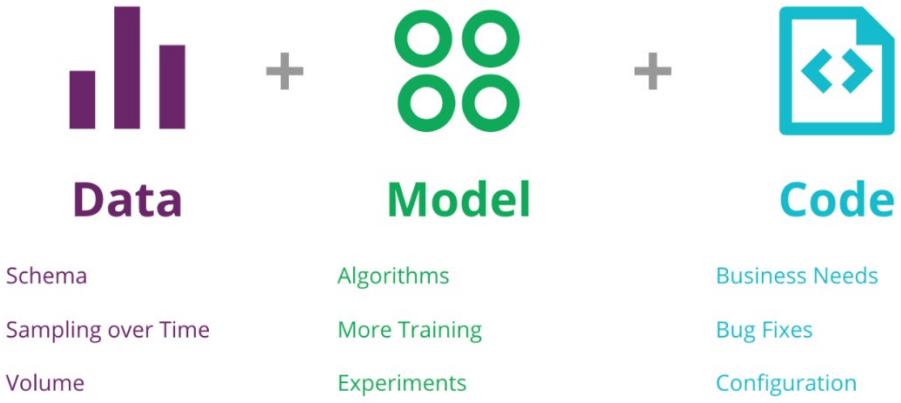


Figure 2-7: Data science project artifacts[15]

Reproducible Model Training

Once the data is available, we move into the iterative data science workflow of model building. This usually involves splitting the data into a training set, a validation set, and a test set. Then, trying different algorithms with tuning their parameters and hyper-parameters. That produces a model that can be evaluated against the validation set to assess the quality of its predictions. The step-by-step of this model training process becomes the ML pipeline.

We structured the ML pipeline for our problem, highlighting the different source code, data, and model components. The input data, the intermediate training and validation data sets, and the output model can potentially be large files, which we do not want to store in the source control repository. Also, the pipeline stages are usually in constant change, which makes it hard to reproduce them outside of the Data Scientist's local environment.

Data Versioning

The data versioning ability is a prerequisite to MLOps implementation, the same way that code versioning is required to practice DevOps. There are two main approaches to implement data versioning: the *git style* and the *time travel*.

DVC is a popular data versioning system using Git to store metadata about data files. It uses a git style method for versioning data that brings the power of Git and Git branches to try different ideas as an experiment management system.

2.5 Project Scope, Assumptions, and Constraints

The following section states the primary statements, which address the scope and assumptions for this project. The main stakeholders of the project agree with these statements:

- This graduation project's duration is ten months.

- The performance of the context detection algorithms is out of the scope of this project.
- The focus of this project is a system that provides an ML pipeline that automates the steps needed for experimenting with models, generating reports, and consequently deploying and testing the models.

Some assumptions about the project are as follows:

- For this project (design project), we focused on generating, preparing, and accessing the Hue data from Hue PIR sensors to make a prototype of a context recognition system. We also verified the ML pipeline on building a model from publicly available datasets for context recognition.
- Many of the features that have been used for context recognition in the literature (motion data and sound) are not available in the Hue system at the moment.
- The kind of data to be used for the pipeline prototype decided to be Hue PIR sensor data and what is available in public datasets. The public datasets for context detection have features, for example, smart devices motion sensors including accelerometer and gyroscope, as well as voice data.

2.6 Stakeholder Analysis

Stakeholder analysis is a valuable tool to identify stakeholders and describe the nature of their stake, roles, and interests. For the stakeholder analysis represented in this report, the focus is on the primary stakeholders for the sake of precision. Stakeholders are all individuals or parties who need to be considered in achieving project goals and whose participation and support are crucial to the project's success. The purpose of stakeholder analysis is to develop a strategic view of the relationships between different stakeholders and the concerns they care about most.

We conducted a stakeholder analysis to identify stakeholders and to understand their concerns. The main groups of people involved in the steering process of the project belong to two organizations: Eindhoven University of Technology (TU/e) and Signify. Table 2-2 provides an overview of the stakeholders in the project.

Table 2-2: The identified stakeholders for this project

Eindhoven University of Technology		
Name	**Role**	**Interest in**
Hossein Mahdian	PDEng Trainee	Successful management of the project process, regular schedule for meetings with stakeholders and supervisors Applying technological and soft skills to finish the project Learning the domain for building a career path Gaining architectural/design/development knowledge and experience Contribution to the proposed problem Successful graduation
Tanir Ozcelebi	TU/e supervisor	Successful project delivery, report quality, and IntelLight+ verification and validation
Yanja Dajsuren	PDEng ST manager	Quality of project results, relationship with clients, successful graduation of trainee
Ali Mahmoudi	PhD Researcher and IntelLight+ user	Usability, functionality, and maintainability of the IntelLight+ system
Signify		
Name	**Role**	**Interest in**

Fetze Pijlman	Signify supervisor / Data scientist and ML Expert	Usability and extensibility of IntelLight+, and guiding trainee throughout the project to reach project goals
Bernt Meerbeek	Group manager	The monitoring of the project progress and the giving feedback to the trainee
Dzmitry Aliakseyeu	Hue use-case expert	Successful integration of the IntelLight context detection system with the Hue system
Mohamed Elkady	Hue System Architect	Successful integration of the IntelLight context detection system with the Hue system
Ion Iuncu	Application Engineer	Development of IoT and lighting applications

Feasibility Analysis

During the first phase of the project, a feasibility analysis was performed that led to defining the project's scope, as well as determining the deliverables, outcomes, and requirements. This chapter covers the feasibility study that was done for this project.

Since the feasibility study assesses the practicality of a proposed plan for a project, it is best practice to produce a contingency plan in case of any unforeseeable circumstances or the unfeasibility of the project. The primary process was to get feedback about the project and the expected outcomes from the main stakeholders' perspectives. This contained asking questions and performing an analysis of the related information to make sure that the achieved results were reliable.

3.1 Challenges

This project addresses the challenge of lighting with new context awareness, enabling more intelligent control. Algorithms need to infer and perhaps even predict a user's context to provide customized lighting to accommodate house owners' needs and preferences.

Intelligent lighting is a very challenging application by itself. Besides that, context-awareness adds common challenges for ML projects such as managing model versions, managing data versions, and reproducing the models. The ML process is a constantly evolving process – systems and their features are changing at regular intervals. The machine-learning setup needs to incorporate the frequent changes in the complex implementation. When it is time to tweak the designs, teams often find that earlier models, features, and datasets were not documented appropriately.

Some of the challenges of ML implementations are listed here and should be kept in mind while designing the solution[16].

- **Selection of the suitable algorithm**: There are tens of widely popular algorithms available for ML implementation. Though algorithms can work in any generic conditions, specific guidelines are available about which algorithm would work best under which circumstances. Improper selection of algorithms can produce garbage output after months of effort, leading to massive loss of the entire effort and further pushing the target timelines.
- **Selection of the right set of data**: As they say – garbage in will produce garbage out; this is also applied to select the data set for ML. The quality, amount, preparation, and selection of data are critical to the success of an ML solution. It is important to avoid selection bias and select the data that is entirely representative of the cases.
- **Data Preprocessing**: Historical data is very messy and often consists of missing values, valueless values, and outliers. Parsing, cleaning, and preprocessing such data can be a tedious job. Feature properties and value ranges must be studied, and techniques like feature scaling must be applied to prevent certain features from dominating the entire model.
- **Data Labelling**: Easy and more appropriate models are the ones used in Supervised ML algorithms. Supervised ML algorithms require data labeling. Data labeling is a manually intensive task – but at the same time, it cannot be outsourced.

ML is usually used to realize intelligent behavior by learning from data or experience and providing relevant predictions. Several approaches to ML can be adopted based on factors such as the objectives to be achieved and the very nature of data. However, selecting an ML model is not straightforward without experimenting because there is no direct mapping from a specific problem to an ML model.

The IntelLight+ system must allow for the selection of different ML libraries, appropriate performance metrics, and deployment environments. The main challenge is to make the model training process

reproducible and auditable. Because data scientists use different tools, it becomes hard to automate it end-to-end. There are more artifacts to be managed beyond the code, and versioning them is not straightforward. Some of them can be really large, requiring more sophisticated tools to store and retrieve them efficiently.

The second challenge is accomplishing inference in different environments, from the cloud with more computing power to more interactive on-premise devices.

3.2 Risks

The risk management process started by identifying an initial set of risks at the beginning of the Intel-Light (+) project. Each identified risk was assigned an impact and probability score. During the execution of the project, the initial set of risks were updated by adding newly identified risks and adjusting previously identified ones when they became obsolete or mitigated, or less relevant. Table 3-1 shows the list of different categories of risks (organizational, technical, external, and project management) combined with their potential impact, probabilities, and mitigation strategies.

Table 3-1: Risks analysis

Organizational					
Risk #	**Description**	**Impact (1-3)**	**Probability (1-3)**	**Criticality**	**Mitigation Action**
R1.1	Getting used to online meetings for progress meetings may lead to bad performance for the final on-site presentation	3	3	**High**	Share your concerns with stakeholders, if possible, arrange the final presentation online
R1.1	The domain is vast. It is possible to get lost in the sea of domain	3	2	**Medium**	Prioritize the requirements and start with requirements that have the highest priority.
R1.2	Working in isolation from home due to the corona situation can make stakeholders less accessible	3	2	**Medium**	Signal any impact on the project, discuss with the stakeholders for a possible solution, and set up more frequent virtual meetings and collaborations.
R1.3	Availability and reachability of direct stakeholders	3	1	**Low**	Plan all meetings ahead of time
Technical					
#	**Description**	**Impact (1-3)**	**Probability (1-3)**	**Criticality**	**Mitigation Action**
R2.1	Unable to verify the correctness of the developed system	3	3	**High**	Communicate with experts to verify the process for the correctness of the system
R2.2	Lack of knowledge in certain required technologies	3	2	**Medium**	Find online resources, ask for expert help if possible and needed
R2.3	The MLOps are very new technology, and not	3	2	**Medium**	Take workshops and online opensource meetings related to the

	yet a solid version is available				needed state of the art knowledge and technologies
External					
#	**Description**	**Impact (1-3)**	**Probability (1-3)**	**Criticality**	**Mitigation Action**
R3.1	Trainee finds residency problems	3	3	**High**	Share your situation with your supervisors
R3.2	Trainee becomes ill for a long period	3	2	**Medium**	Keep a buffer time at each task and negotiate requirements with stakeholders based on priority
R3.3	Psychological effects of living in quarantine and pandemic situation	2	2	**Low**	Include physical activities to stay more healthy
Project Management					
#	**Description**	**Impact (1-3)**	**Probability (1-3)**	**Criticality**	**Mitigation Action**
R4.1	Deliverable not being available in time	3	3	**High**	Make a minimum viable product and set a deadline to make sure the important deliverables are
R4.2	Misunderstanding of the project goals that result in a delayed, inefficient, and incorrect overall development process	3	3	**High**	Frequently communicate with stakeholders and confirm important decisions with stakeholders
R4.3	Not identifying all stakeholders	3	2	**Medium**	Ask main stakeholders if a potential stakeholder is forgotten
R4.4	Key stakeholders are not available for several reasons, such as sickness or holiday	3	2	**Medium**	Be aware of the stakeholders planned to leave and prepare replacement if necessary
R4.5	High priority requirements cannot be satisfied.	3	2	**Medium**	Raise the issue as early as possible, discuss with stakeholders for a possible workaround, and manage the stakeholder's expectations

Requirements and Use Cases

This chapter states the system requirements that the trainee found to be most relevant, such that the system complies with the user requirements. The requirements are the specifications that the IntelLight+ system needs to fulfill, and they are derived from the stakeholders' concerns. These requirements specify what IntelLight+ needs to do.

4.1 Introduction

IntelLight+ aims to facilitate the real-time context assessment of the homeowners during their daily activities. To achieve this goal, we designed the system containing two main sub-modules, the Ml pipeline with a deployment part and the software module. The main idea is to make an end-to-end ML process enabled so that different algorithms, features, and inference environments can be used for experimentation. The software sub-module provides a means for more interaction between users and the context recognition model that serves them. The software feedback module can be used for annotating data. In this way, the users specify when and near which sensors they had a particular context (having dinner, cooking, or watching TV). A combination of that feedback and sensor data for the user can be used as the data set.

IntelLight+ extracts the features from the Hue motion sensors and uses these features to predict the home context. IntelLight+ tracks and visualizes the code, data, metrics, and hyper-parameters used to build the models in the ML pipeline module. It enables data scientists to compare and choose between different models and then deploy them to different target machines. The software module allows data scientists to receive feedback from users and allows for more advanced ML approaches like customization based on user feedback. This feedback is given via an interface. The integration of the earlier mentioned modules in IntelLight+ lets data scientists focus on applying different approaches for context prediction.

4.2 System requirements

An early understanding of the customer requirements and their priorities in software engineering projects is the biggest challenge. A detailed list of requirements is essential to steer a project in the right direction. However, as the customer needs are subject to change, new requirements can emerge, and the old ones may need updating during the project lifetime. In this project, initial requirements were elicited using the use case analysis and interview techniques. These requirements were revisited and updated throughout the duration of the project.

The following sections of this chapter explain the identified use cases, corresponding functional and non-functional requirements, as well as the relationships among them.

4.3 Use cases

A use case is a description of potential interactions between a system and its users. In the context of this project, a homeowner who uses Hue lighting and data scientists who want to build and experiment with context recognition algorithms are the typical users of the IntelLight+ system. Figure 4-1 shows the potential actions that can be performed by the homeowner.

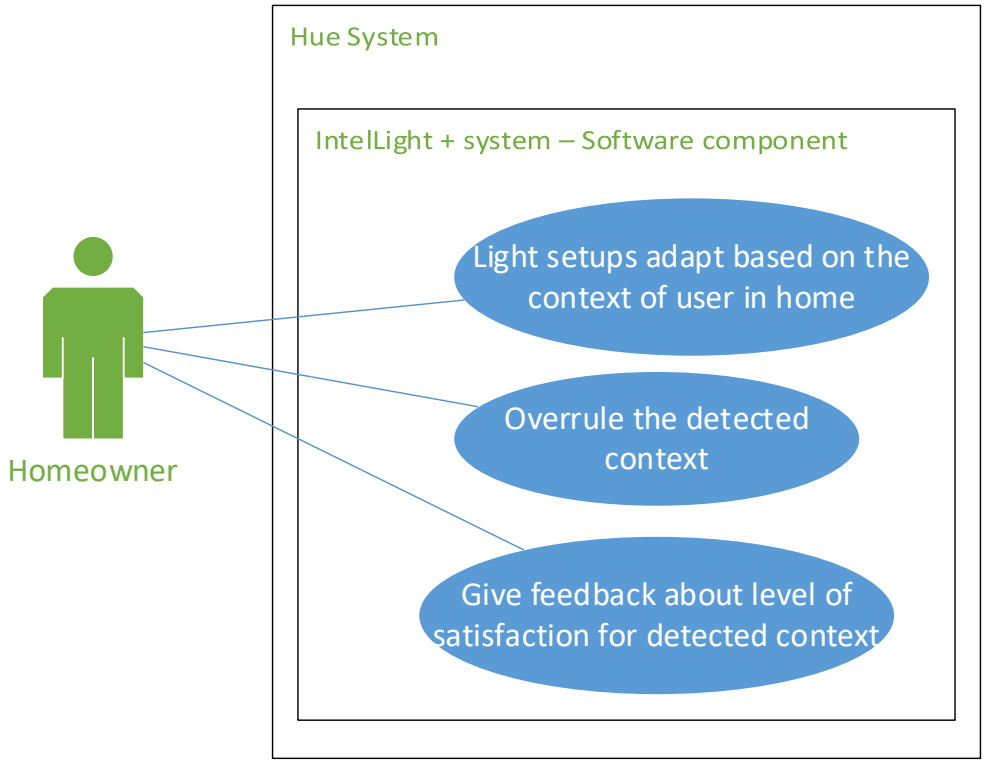


Figure 4-1: Use case for the homeowner

The MoSCoW method is used to prioritize the use cases and the elicited set of requirements. The word MoSCoW is an abbreviation of four words, each of which defines different priority levels. They are:
• **Must:** Critical product features that are essential for the current delivery timeline.
• **Should:** Important features that provide significant value.
• **Could:** Desirable but not necessary needs that can improve usability.
• **Won't:** Least critical and lowest priority needs.

Table 4-1: Use cases that the homeowner is involved in

UC #	Priority	Description
UC1.1	Must	The homeowners want the light scene to change based on what they are doing so that it matches their context and new activity.
UC1.2	Must	The homeowners want the system to recognize the predefined contexts for the user.
UC1.2.1	Must	The homeowners want the light scene to change to a specific light setting when they are doing **Office Work**.
UC1.2.2	Must	The homeowners want the light scene to change to a specific light setting when they are **Having Dinner**.

UC1.2.3	Must	The homeowners want the light scene to change to a specific light setting when they are **Working Out**.
UC1.2.4	Must	The homeowners want the light scene to change to a specific light setting when they are **Cooking**.
UC1.2.5	Must	The homeowners want the light scene to change to a specific light setting when they are **Lounging**.
UC1.3	Must	The homeowners can overrule the predicted context when they realize the detected context is not matching their context. This will be reported to the backend data scientist.
UC1.3.1	Should	The homeowners report via a yes/ no/ not sure question if they had a specific context(having dinner, watching TV, and cooking) at a particular time.
UC1.4	Could	The homeowners want to have the context recognition module working with available third party data (such as an entertainment API)
UC1.5	Should	The homeowners want when they use different types of sensors that are connected to their IntelLight+ system, it leads to the better setting of the light based on their context.

As the data scientists are the primary users of the IntelLight+ system, most user stories for the ML pipeline are related to them. Figure 4-2 shows the potential actions that the data scientist can perform.

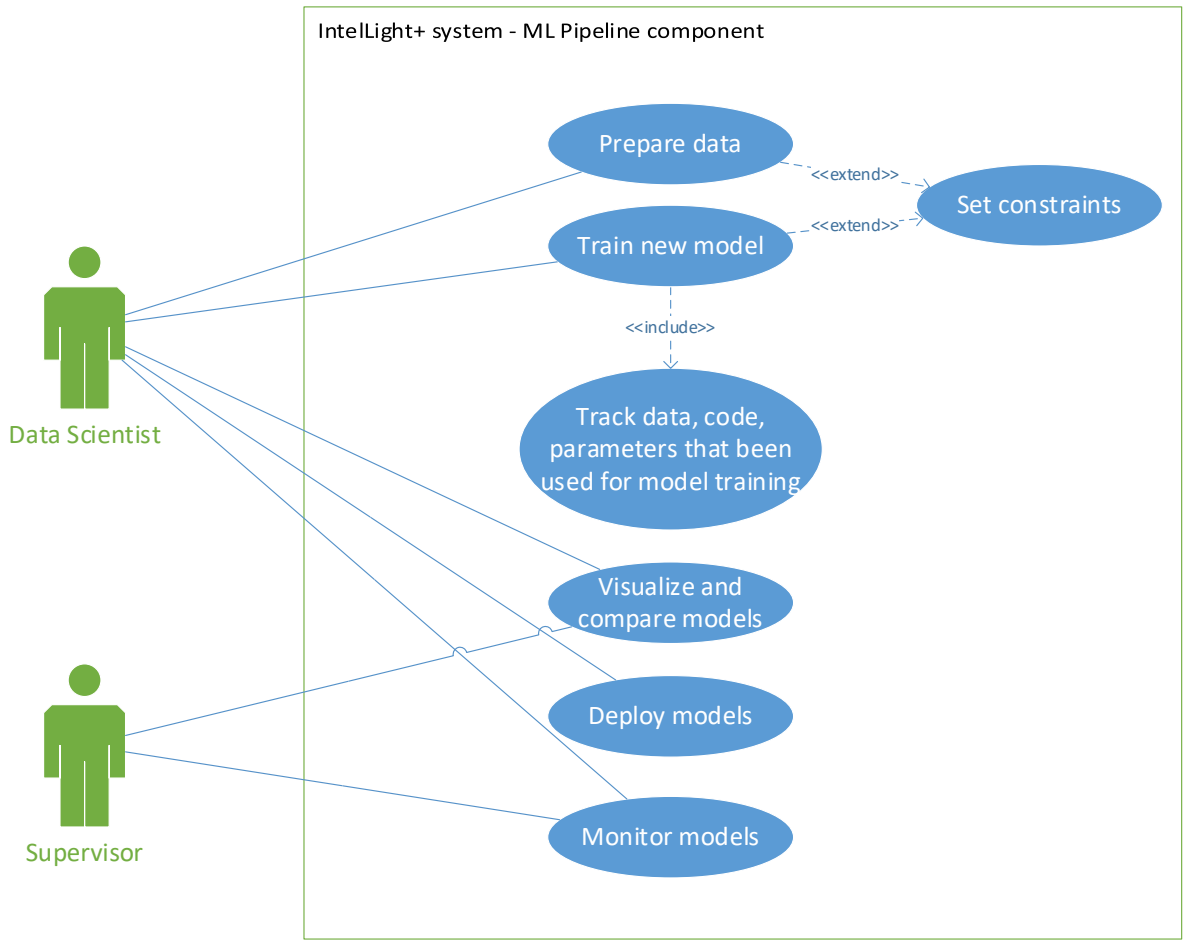


Figure 4-2: Use case for the data scientist

Table 4-2: ML pipeline use cases

UC #	Priority	Description
UC2.1	Must	Data scientist wants to trace the data, code, and configurations that been used for training models.
UC2.2	Must	Data Scientist wants to have a template that helps modularize the development of algorithms and separate the different concerns for his project.
UC2.3	Must	Data scientist wants to recreate previously experimented models.
UC2.4	Must	Data scientist wants to visualize and compare the models with different hyper-parameters.
UC2.5	Must	Data scientist wants to have the option to deploy the model on-premise, remote server, or both.
UC2.6	Must	Data Scientist wants to apply different preprocessing steps.
UC2.7	Must	Data Scientist wants to apply different metrics for model evaluation.

4.4 Functional Requirements

The initial requirements are elicited based on the identified use cases. These requirements are analyzed further and presented to the major stakeholders for validation. The requirements established through this process are the initial set of functional requirements (FR) and the established baseline for the system. The non-functional requirements (NFR) are connected to the quality characteristics and constraints of the system. The FRs are explained in three different categories. Section 4.4.1 describes the general list of requirements, section 4.4.2 describes the requirements related to the feedback software component, and section 4.4.3 lists the functional requirements related to the ML pipeline. Next, the NFRs are covered in section 4.5. Each requirement is given a unique identification (ID). However, these IDs do not imply any priority or order.

4.4.1. General Functional Requirements

This section lists descriptions of functional requirements related to IntelLight System alongside their priority and related use cases.

Table 4-3: General functional requirement

Req.#	Priority	Description	UC UID
FR1.1	Must	The system shall support running ML models to classify/ recognize the contexts of a single person in the home.	UC1.1, UC1.2
FR1.1.1	Must	The system shall support running ML models to recognize the contexts of multiple people in the home.	UC1.1, UC1.2
FR1.2	Could	The system shall support running ML models to predict the context based on different features from different sensors (accelerometer, gyroscope, and audio sensors) in a single prediction model.	UC1.1, UC1.2
FR1.2.1	Could	The system shall support running different ML models to predict the context based on different prediction models for features from each of the sensors (one model for accelerometer, one for gyroscope, and one for the audio sensor) and then combine the result.	UC1.5
FR1.3	Must	The system supports creating the ML model to recognize context when data from **Hue PIR sensors** are available.	UC1.5
FR1.3.1	Must	The system supports running the ML model to recognize context when **audio** data are available	UC1.5
FR1.3.2	Must	The system supports running the ML model to recognize context when motion and orientation data of **Accelerometer**, **Gyroscope**, and **magnetometers sensors** are available	UC1.5
FR1.3.3	Could	The system supports running the ML model to recognize context when any combination of the sensors mentioned in **FR1.3** is available.	UC1.4, UC1.5
FR1.3.4	Must	The system supports running the ML model to recognize context when only one of the sensors mentioned in FR1.3 is available.	UC1.5

FR1.4	Must	The system shall provide a template that allows the data scientist to add, activate, deactivate, and change orders of preprocessing steps.	UC2.6
FR1.5	Must	The system shall provide a template that allows the data scientist to add, activate, and deactivate metrics for model evaluation.	UC2.7
FR1.7	Must	The system allows having a separate model for each of the activities.	UC2.5

4.4.2. Software Component Functional Requirements

This section lists descriptions of functional requirements related to the feedback component module alongside their priority and related use cases.

Table 4-4: Feedback component module functional requirements

Req.#	Priority	Description	UC UID
FR2.1	Must	The system shall be able to ask for feedback from the homeowner for a specific detected context and report it to the backend data scientist.	UC1.3
FR2.1.1	Must	The system shall allow the homeowner to give feedback actively or passively.	UC1.3
FR2.2	Must	The system shall let the homeowner (test participant) manually report the time he had a specific context.	UC1.3
FR2.3	Must	The system shall allow the homeowner (test participant) to specify the period he wants to receive notification for each context.	UC1.3.1
FR2.4	Must	The notifications should have different options allowing the homeowners (test participants) to specify if they are doing an activity, not doing an activity, or don't know.	UC1.3.1
FR2.5	Should	The system should allow usage of the reported feedback for adjusting the training model.	UC1.5

4.4.3. ML Pipeline Functional Requirements

This section lists descriptions of functional requirements related to the ML pipeline alongside their priority and related use cases.

Table 4-5: ML pipeline functional requirements

Req.#	Priority	Description	UC UID
FR3.1	Must	The system shall keep track of the version of the code, data, and the parameters that have been used for each training experiment of the model.	UC2.1

FR3.2	Must	The system should provide the template that modularizes the project to different meaningful data science components.	UC2.2
FR3.3	Must	The system shall provide a pipeline that keeps track of all the (Python) libraries and their versions that have been used for training a model.	UC2.3
FR3.3.1	Must	The system shall provide an ML pipeline that works with different ML libraries.	UC2.3
FR3.4	Must	The system shall visualize the models with a range of different hyper-parameters and let data scientists compare them.	UC2.4
FR3.5	Must	The system shall be able to deploy and test the inference model on-premise and remote server environments in the same way.	UC2.5
FR3.6	Could	The system should support providing different models for users based on different settings in their homes.	UC2.5, UC2.6
FR3.7	Could	The system should let data scientists deploy and test different models for users based on the different categories of the settings of the users' homes.	UC2.5

4.5 Non-Functional Requirements

Non-Functional Requirements (NFRs) define the criteria that are used to evaluate the whole system, but not for a specific behavior, and are also called Quality Attributes (QAs). A Non-Functional Requirement defines the performance attribute of a software system. Where Functional Requirement is a verb, the Non-Functional Requirement is an attribute.

A situation in which the system has the desired combination of quality attributes, for example, interoperability and performance or reliability, shows the architecture's success and the software's quality. When designing to meet any requirements, it is essential to consider the impact on other attributes and find compromises between requirements. Along with this, the value or priority of each attribute differs from system to system. The covered qualities in this section are essential for designing the IntelLight+ system.

Table 4-6: Quality Attributes

QA#	Priority	Description	QA
QA1.1	Must	The system shall be able to reproduce the environment to run the selected training models on other platforms.	Reproducibility
QA1.2	Must	The system shall allow adding/delete/change the ML algorithm functionality, platform, attribute, or capacity	Modifiability
QA1.3	Should	The system shall be able to be extended for working with new sensors and other external systems (like an entertainment API) for context detection.	Extensibility
QA1.4	Could	The system shall be able to use different ML libraries, support various versions and adapt when external changes occur.	Flexibility

Reproducibility is a major principle for scientific methods. An ML model is reproducible if the results obtained by an experiment or in statistical analysis of the same data set can be achieved again with a high degree of reliability when the study is replicated. Only after one or several such successful replications should a result be recognized as a reliable approach.

Any results of experiments should be documented by making all data and code available so that if the reproduced model is executed again, it will achieve identical results.

Modifiability is important from different aspects. From the end-user point of view, users of the system need to be able to customize the context recognition component based on their own needs. The data scientist should be able to modify the context detection model by changing his code, data, and configurations.

The system should allow for using different sensors and data sources for future growth. **Extensibility** is essential for the system to gain data from different types of sensors or even external systems for context detection. Extensions can be through the addition of new functionality or through modification of existing functionality. Finally, **flexibility** is the ability of the system to respond to potential internal or external changes affecting its value delivery in a timely and cost-effective manner.

4.5.1. Quality Attribute Scenarios (QAS)

In this section, specific QAS for each of the quality attributes is specified. The quality attributes are expressed as QAS templates (stimulus-response). It consists of six parts.

- **Source:** This is an entity (a human, a computer system, or any other actuator) that generated the stimulus.
- **Stimulus:** The stimulus is a condition that needs to be considered when it arrives at a system.
- **Environment:** The stimulus occurs within certain conditions. The system may be overloaded or run when the stimulus occurs, or some other condition may be true.
- **Artifact:** Some artifact is stimulated. This may be the whole system or some pieces of it.
- **Response:** The response is the activity undertaken after the arrival of the stimulus.
- **Response measure:** When the response occurs, it should be measurable in some fashion so that the requirement can be tested.

The IntelLight+ project covers the QAs in Table 4-6 and enables them with QAS. Table 4-7 shows the QAS for reproducibility, Table 4-8 depicts the QAS for modifiability, extensibility QAS is shown in Table 4-9, and Table 4-10, the QAS for the flexibility is shown in Table 4-11.

Table 4-7: Reproducibility QAS

Source	Data scientist (algorithm developer)
Stimulus	Wishes to recreate previously experimented model
Artifact	Context recognition system
Environment	At runtime on cloud/raspberry
Response	Creates the model with same performance; deploys recreated model
Response Measure	Cost in terms of effort, time

Table 4-8: Modifiability QAS

Source	Data scientist (algorithm developer)
Stimulus	Wishes to modify/vary the code, data, and parameters for the detection model creation
Artifact	Context recognition system on user's home
Environment	At design time, at development time, at compile time, at deployment time
Response	Modifies without affecting other functionality; tests modification; deploys modification
Response Measure	Cost in terms of number of elements affected, effort, time

Table 4-9: Extensibility QAS for the data scientist

Source	Data scientist (algorithm developer)
Stimulus	Wishes to add a new algorithm for context detection
Artifact	Context recognition system on user's home
Environment	At design time, at development time, at compile time, at deployment time
Response	The new algorithm is added without affecting other functionality of the system; tests added algorithm
Response Measure	Cost in terms of number of elements affected, effort, time

Table 4-10: Extensibility QAS for the homeowner

Source	Homeowner (Hue user)
Stimulus	Wishes to add a new sensor to the context recognition system
Artifact	Context recognition system on user's home
Environment	At runtime on cloud/raspberry
Response	The new sensor will be added following the same design without affecting other functionality
Response Measure	Cost in terms of number of elements affected, effort, time

Table 4-11: Flexibility QAS

Source	Data scientist (algorithm developer)
Stimulus	Wishes to use various ML libraries
Artifact	Context recognition system on user's home
Environment	At design time, at development time, at compile time, at deployment time

Response	The different ML libraries are used without affecting other functionality
Response Measure	Cost in terms of number of elements affected, effort, time

4.6 Requirements traceability matrix

This section specifies how user stories are addressed, in other words, which requirements are satisfied in order to accomplish a user story. Table 4-12 depicts the requirement numbers that met to consider a user story as done.

Table 4-12: User story - requirement relations

UC UID	REQ ID
UC1.1	FR1.1, FR1.1.1, FR1.2
UC1.2	FR1.1, FR1.1.1, FR1.2
UC1.3	FR2.1, FR2.1.1, FR2.2, FR2.3, FR2.4
UC1.4	FR1.3.3
UC1.5	FR1.2.1, FR1.3, FR1.3.1, FR1.3.2, FR1.3.3, FR1.3.4, FR2.5
UC2.1	FR3.1
UC2.2	FR3.2
UC2.3	FR3.3, FR3.3.1
UC2.4	FR3.4
UC2.5	FR3.5, FR3.6, FR3.7
UC2.6	FR1.4, FR2.6
UC2.7	FR1.7

System Architecture and Design

This chapter covers the system architecture and design for the project. The purpose of designing an architecture for a system is to solve a problem statement according to the system requirements. This chapter decomposes the architecture as well as the design of the envisioned solution into three main parts. Section 5.1 introduces design principles, which are taken to consideration in this project. The high-level architecture is covered in section 5.2, and section 5.3 describes the architectural views of the Intelight+ project.

5.1 Design Principles

In order to guarantee that the design of our system is good, we need to know the symptoms of poor design. This section demonstrates the smells that often accumulate in a software project and describes the design principles that can help us avoid them. Software designs often degrade because requirements change in ways that the initial design did not anticipate. Often, these changes need to be made quickly, and they may be made by developers who are not familiar with the original design philosophy. Although the change to the design works, it somehow violates the original design. Bit by bit, as the changes continue, these violations accumulate, and the design begins to smell.

The software system is rotting when it exhibits any of the following problems[17]:

- **Rigidity**: The tendency for software to be difficult to change, even in simple ways. A design is rigid if a single change causes a cascade of subsequent changes in dependent modules. The more modules that must be changed, the more rigid the design is.

- **Fragility:** Fragility is the tendency of a program to break in many places when a single change is made. Often, the new problems are in areas with no conceptual relationship with the area that was changed; fixing those problems leads to even more problems.

- **Immobility:** A design is immobile when it contains parts that could be useful in other systems, but the effort and risk involved with separating those parts from the original system are too great. This is unfortunate but widespread.

The design principles give practical guidance to software engineers by acting as strong guidelines that are applicable to any software design project. Here is the list of SOLID software design principles introduced by Robert C. Martin [18] and how they can be applied to an ML system:

Single Responsibility Principle (SRP): *A class should have only one reason to change*. If a class has more than one responsibility, then the responsibilities become coupled. Changes to one responsibility may impair or inhibit the ability of the class to meet the others. This kind of coupling leads to fragile designs that break in unexpected ways when changed.

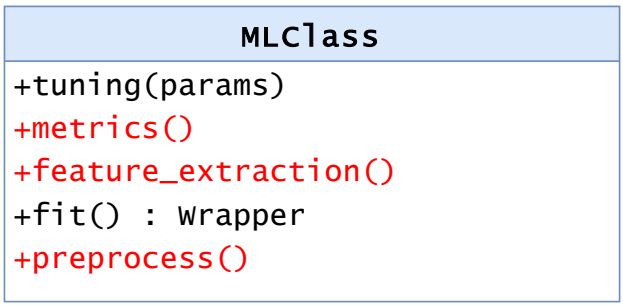


Figure 5-1: Violation of SRP

As shown in Figure 5-1 diagram, MLclass is going to change due to many causes. Activities of different origins like extracting features, preprocessing, defining metrics, fitting method, and tunning all are encapsulated in one class. This class reacts to changes in data processing, feature engineering, and model selection. This class needs to be divided into machine learning sub-components.

Open-Closed Principle (OCP): *Software entities (classes, modules, functions, etc.) should be open for extension but closed for modification.* When a single change to a program results in a cascade of changes to dependent modules, the design smells of Rigidity. The OCP advises us to refactor the system so that further changes of that kind will not cause more modifications. If the OCP is applied well, then further changes of that kind are achieved by adding new code, not by changing old code that already works.

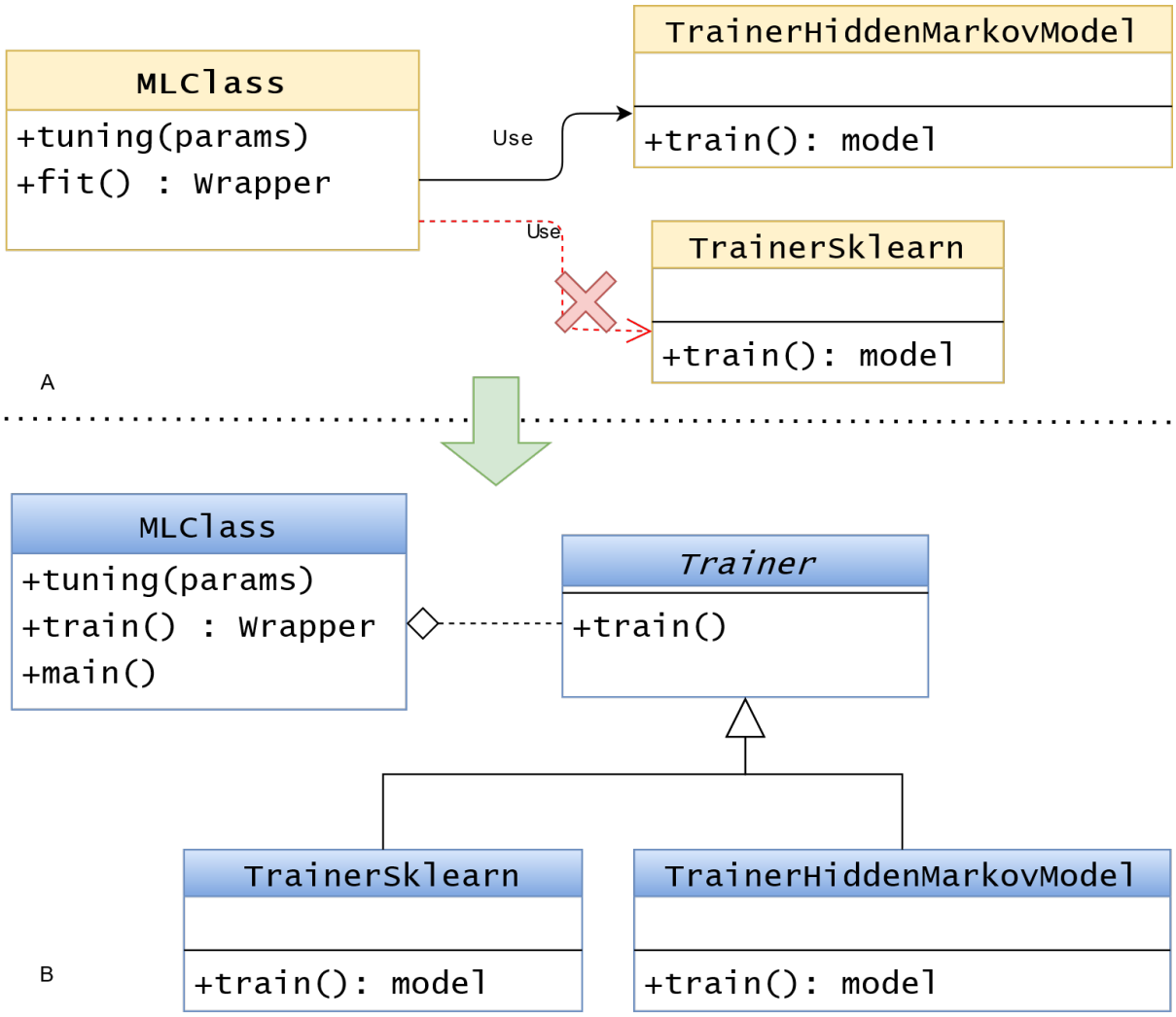


Figure 5-2: Refactoring the design following OCP

For example, as shown in Figure 5-2 A, the MLclass started by using the TrainerHiddenMarkovModel class. However, after a while, in the next version, it needed to use the TrainerSklearn. Due to the tight coupling, it was not possible to swap in the TrainerSklearn. To support the new TrainerSklearn would require changing the MLclass class. This is a violation of the open-closed principle: Old code should not have to change to add functionality.

The solution, as shown in Figure 5-2 B, is to abstract away the Trainer in an interface/abstract class. Then we make the dependency arrows point towards the abstract components. After the modification,

all of the *Trainer*s are interchangeable. MLclass can use either one without knowing the internals of each approach. Besides, the two Trainers do not need to know about the MLclass. This way, the developer can test them in isolation. This allows the developer to add a third Trainer (for example, TrainerCustom) without any changes to the MLclass or the other sub-classes of Trainer. This is true as long as the Trainers implement the shared interface.

Liskov Substitution Principle (LSP): *Subtypes must be substitutable for their base types.* When a method of a subtype class does something that the client of the base type class does not expect, an undefined behavior happens. This smell causes debugging of the program and finding the cause of a problem very hard. This problem is often the result of the wrong inheritance. If LSP follows, when the base class does something, the subclass should also do it in a way that does not violate the expectations of the callers.

Interface Segregation Principle (ISP): *Clients should not be forced to depend on methods that they do not use.* When clients are forced to depend on methods that they don't use, then those clients are subject to changes to those methods. This results in an accidental coupling between all the clients. In other words, when a client depends on a class that contains methods that the client does not use but that other clients do use, then that client will be affected by the changes that those other clients force upon the class. We would like to avoid such couplings where possible, and so we want to separate the interfaces.

Dependency Inversion Principle (DIP): *Abstractions should not depend on details. Details should depend on abstractions.* When high-level modules depend on low-level modules, it becomes very difficult to reuse those high-level modules in different contexts. However, when the high-level modules are independent of the low-level modules, then the high-level modules can be reused quite simply. This principle is at the very heart of API design.
DIP is also critically important for the construction of code that is resilient to change. Since the abstractions and details are all isolated from each other, the code is much easier to maintain.

5.2 High-level Architecture

The IntelLight+ project is a design project that designs a system for context recognition. The project's primary goal is to facilitate the work of the research project IntelLight (which focuses on the development of ML algorithms). This project automates the end-to-end lifecycle of ML in three different dimensions (code, data, and model). It allows different types of algorithms to be implemented. Although the development of ML algorithms is outside of this project's scope, the pipeline that covers different ML steps (Model Building, Model Evaluation, Model Deployment, Model Update) is part of this project.

The system consists of three main components:

- An ML pipeline that keeps track of data, code, and artifacts used in experiments and visualizing the results
- A feedback component that provides a means for communication between data scientists and user
- A deployment module that lets the data scientist deploy the model in different environments

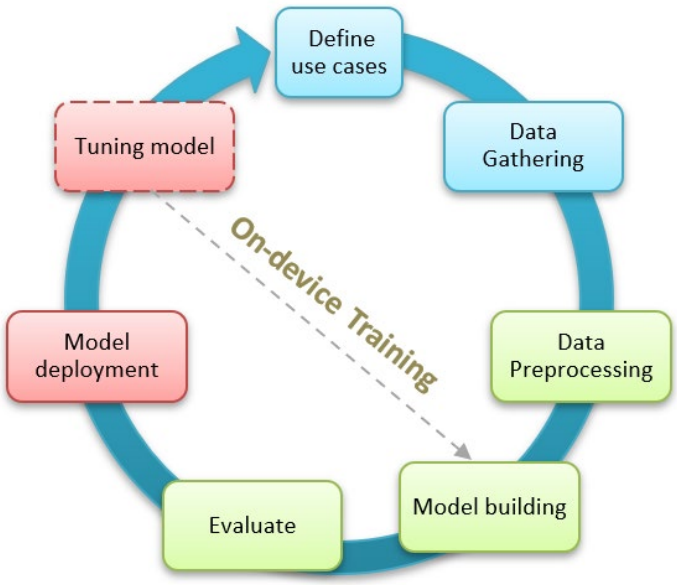


Figure 5-3: ML Lifecycle

5.2.1. ML Pipeline

An ML pipeline automates the ML workflow by enabling data to be transformed and correlated into a model that can then be analyzed to achieve outputs. This type of ML pipeline makes the process of inputting data into the ML model fully automated. The ML pipeline splits up ML workflows into independent, reusable, modular parts that can then be pipelined together to create models.

As shown in Figure 5-4, the ML Pipeline keeps track of data and code versions used in training the model. It visualizes the models with different parameters and lets the data scientist choose the model based on metrics. The technologies used to keep track of the data, code, and visualize the experiments are covered in more detail in section 6.2.

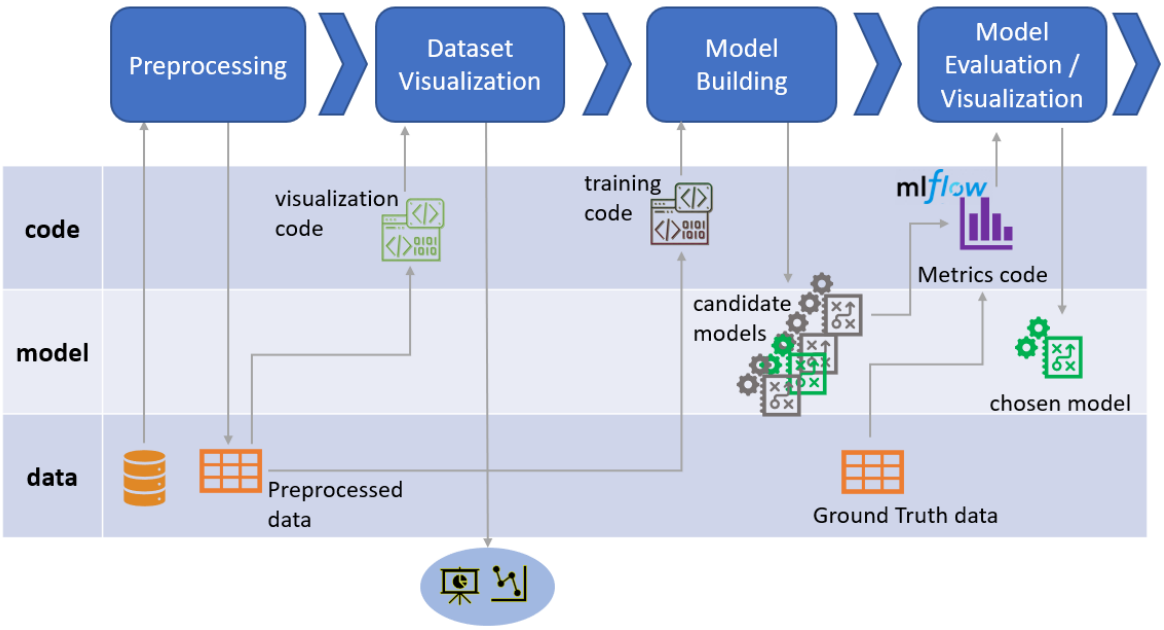


Figure 5-4: Management of different sources in ML pipeline

5.2.2. Feedback Component

The feedback component checks if the predicted activity and context were right by asking the users if they have the same context that the model predicted. The feedback component is also used for generating the data set. In this way, we can retrieve sensor outputs as features and user feedback as ground truth for labeling the data.

The initial design for the feedback receiving strategy is that the users can specify the usual time of activities they often do. If the probability is in a specified interval, the system sends a notification to the user to see if the user is doing the predicted activity. Suppose the probability of detected activity is more than a threshold and the current time is not in the interval specified by the user. In that case, the system can send a notification to the user and ask for his / her feedback.

The way the feedback component works in this project is that raspberry pi is used to read the sensors' outputs from the Hue bridge and save them in a remote server (amazon s3). Then this sensor data are used as the input for the initial trained model. The user receives feedback requests for the activities that are tracked based on mentioned feedback receiving strategy. What the data scientist is going to do with this feedback to improve the quality of the model is out of the scope of this project.

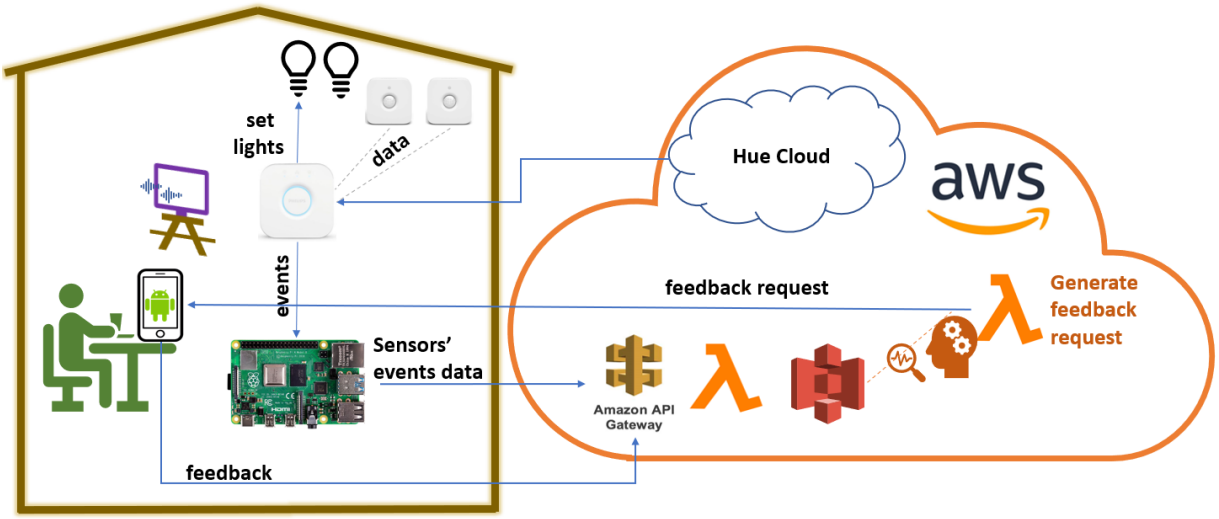


Figure 5-5: Feedback component design

5.2.3. Deployment

One of the main requirements of the IntelLight+ system is the ability to deploy and infer in a range of different environments from cloud to on-premise devices. Each approach of either inferring in the cloud or on-premise has its pros and cons.
As Figure 5-6 shows, when the model is deployed in the cloud environment, it can use more computational power, whereas, when the model is deployed on-premise, it is more interactive.

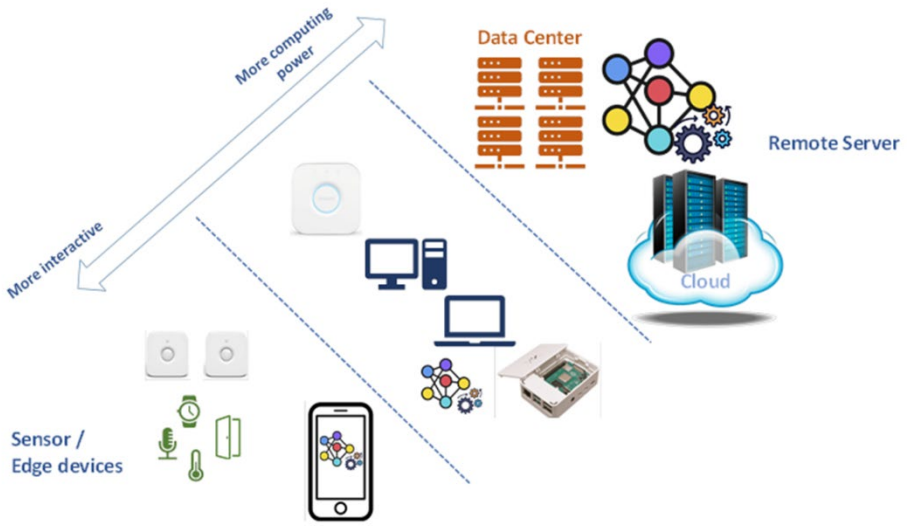


Figure 5-6: Deployment options

5.3 Architectural Views

In this section, we look at IntelLight+ software architecture. Software architecture is the fundamental design of an entire software system. It defines what elements are included in the system, what function each element has, and how each element relates to one another. It is the big picture or overall structure of the whole system—how everything works together. One important way software architecture is presented visually is through the UML diagrams. It is necessary to capture the complete behavior and development of a software system from multiple perspectives. Each of these perspectives is called a view.

A view is a representation of a set of system elements and relations among them—not all system elements, but those of a particular type[19]. Different views also expose different quality attributes to different degrees. Therefore, the quality attributes that are of most concern to the architect and the other stakeholders in the system's development will affect the choice of what views to consider. Thus, views let us divide the multidimensional software architecture entity into a number of interesting and manageable representations of the system.

One consideration is the functionality of the software. Functionality involves what a system does to satisfy the purpose the client desires. Focusing on this functionality and the needed objects leads to a perspective called the logical view.

5.3.1. Logical View

The logical view, which focuses on the functional requirements of a system, usually involves the objects of the system. From these objects, a UML class diagram can be created to illustrate the logical view. A class diagram establishes the vocabulary of the problem and the resulting system. Defining all of the classes, their attributes, and their behaviors makes it easy to understand the key abstractions and terminology.
Figure 5-7 defines the classes, their attributes, and their behaviors in the IntelLight+ pipeline.

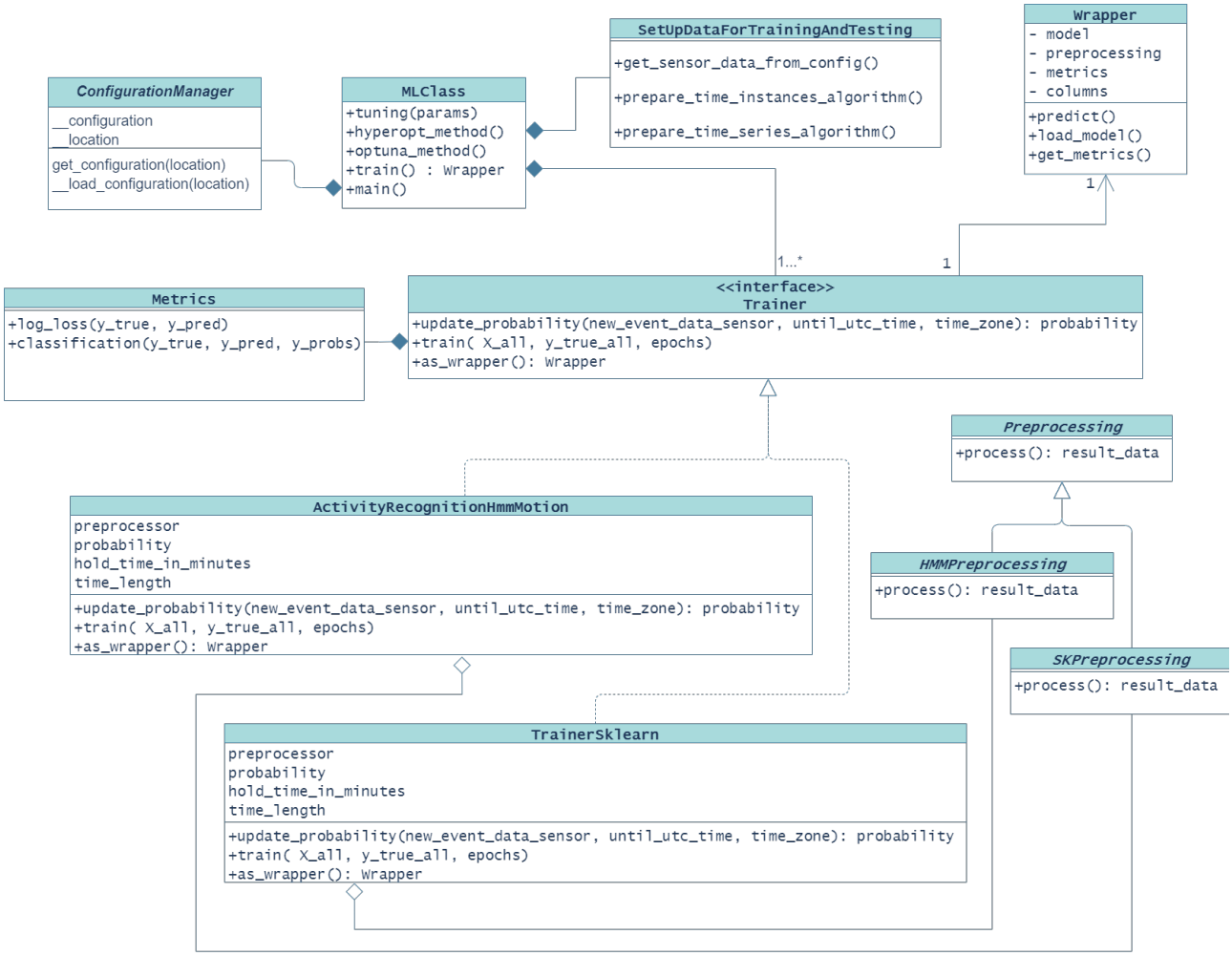


Figure 5-7: ML training UML class diagram

Some of the design practices like SRP, OCP, and DIP mentioned in section 5.1 can be seen in the ML training class diagram. Furthermore, as shown in Figure 5-8 in more detail, the *MLclass* class is the part of the IntelLight+ system that wants to use third-party libraries for training ML models. An adapter design pattern has been used to facilitates communication between *MLclass* and *ActivityThroughMotionAndHmm* by providing a compatible interface.

Figure 5-8: Adapter design pattern enables using third party training libraries for pipeline

The adapter design pattern (Figure 5-9) consists of several parts:
Client class: The part of the system that wants to use a third-party library or external system.
Adaptee class: This is the third-party library or external system that is desired to be used.
Adapter class: This class sits between the client and the adaptee. The adapter conforms to what the client is expecting to see by implementing a target interface. The adapter is a kind of wrapper class.
Target interface: This is used by the client to send a request to the adapter.

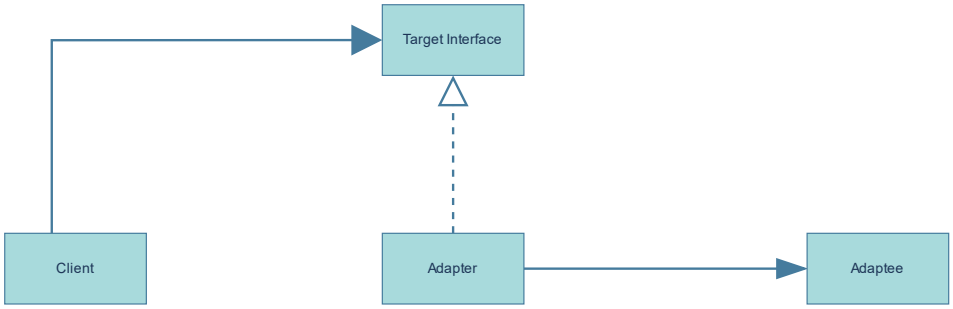


Figure 5-9: Adapter design pattern

As systems or parts of systems become larger, they also become more complex. This is not necessarily a bad thing – if the scope of a problem is large, it may require a complex solution. Client classes function better with a simpler interaction, however. In the IntelLight+ system, the ML pipeline consists of two different sub-components for training and building models on the one hand and using the trained model for prediction in another hand. In order to be able to infer from a trained model, we need to have a *Predict* class that can use the trained models without knowing the details about algorithms that are used in training. Figure 5-10 shows a means to hide the complexity of a subsystem by encapsulating it behind a unifying wrapper class. This is called the facade design pattern. It removes the need for *Predict* classes to manage the trainer submodule on its own, resulting in less coupling between the trainer submodule and the *Predict* class.

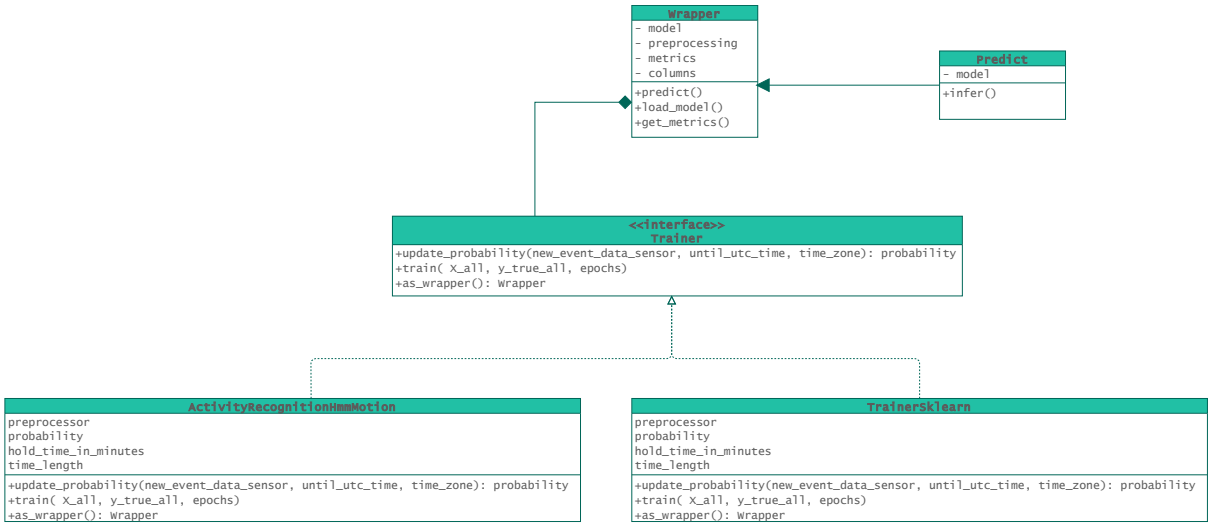


Figure 5-10: Facade pattern for prediction pipeline

Another consideration in software is how the software executes, dealing with the interaction of subprocesses. These characteristics affect the performance and scalability of the system. Focusing on the processes implemented by the objects in the logical view leads to the process view perspective.

5.3.2. Process View

The effective UML diagrams related to the process view of a system are the activity diagram and the sequence diagram. The activity diagram can illustrate the processes or activities for a system. Figure 5-11 presents the control flow of the IntelLight+ system from an activity to another. As you see, the activity diagram starts with retrieving data from shared storage. The data scientist then extracts features, preprocessing the data, extracting the ground truth, and running the experiment with parameters and hyperparameters specified in a configuration file. When the experiments for different algorithms on different activities are finished, the data scientist visualizes the results by *mlflow UI* API. If the model qualifies for inference, data scientist chooses the models for deployment. The selected models will be deployed to shared storage and then can be used in a virtual machine in the cloud or raspberry pi to predict the context. Finally, the lights are set based on the context that has been recognized.

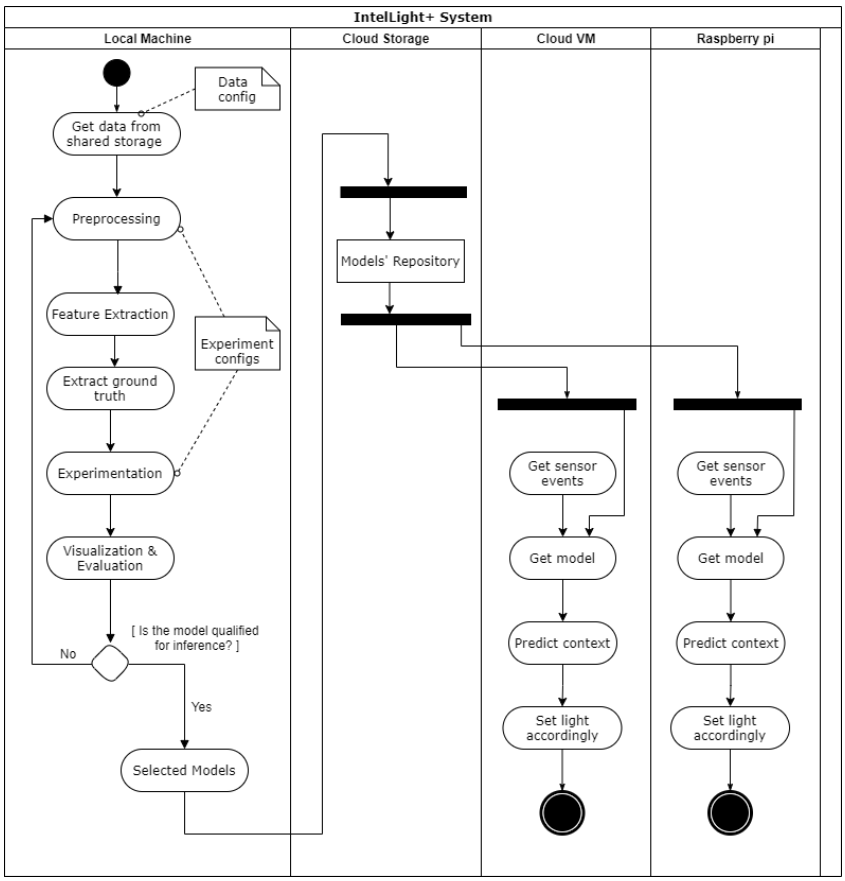


Figure 5-11: UML activity diagram

One crucial part of the design of the IntelLight+ system is the feedback component. The feedback component lets the homeowner specify if he is satisfied with the light set. It also can ask the users if they are doing an activity. The data scientist can realize if the trained model is working as expected by getting users' feedback regarding the predicted context.

The feedback component lets the IntelLight+ system use notifications to communicate with homeowners for annotating the data as well. In this way, the system will send notifications periodically to Hue users and asks about their current activity. Figure 5-12 shows the interaction between sensors and s3 storage in order to gather the data, as well as how the feedback is communicated to the users by notifications. In order to give feedback, users should install the context feedback app on their smartphones. After installing the app, participants log in with the AWS API key, then they use the phone to upload their feedback in the app to the cloud. Figure 5-12 shows the process of gathering sensor data and user feedback.

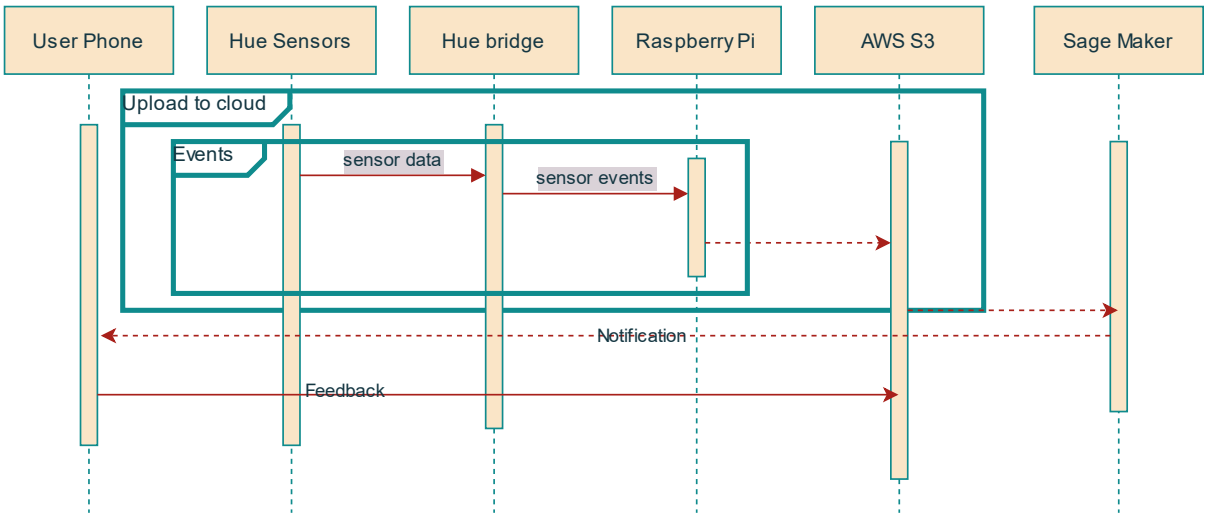


Figure 5-12: Data gathering and Notification sequence diagram

The software architecture can also involve the development view. This perspective focuses on implementation considerations such as the hierarchical structure of the software. The programming languages of the system will heavily influence this structure and therefore places constraints upon development.

5.3.3. Development View

The development view describes the hierarchical software structure. This view uses the UML Component diagram to describe system components. Figure 5-13 shows the components of the IntelLight+ system. Step 1 depicts the components interacting to build models for context recognition. Then in step 2, the qualified models will be placed in shared storage. Finally, in step 3, models are retrieved in the target environments and used for the prediction

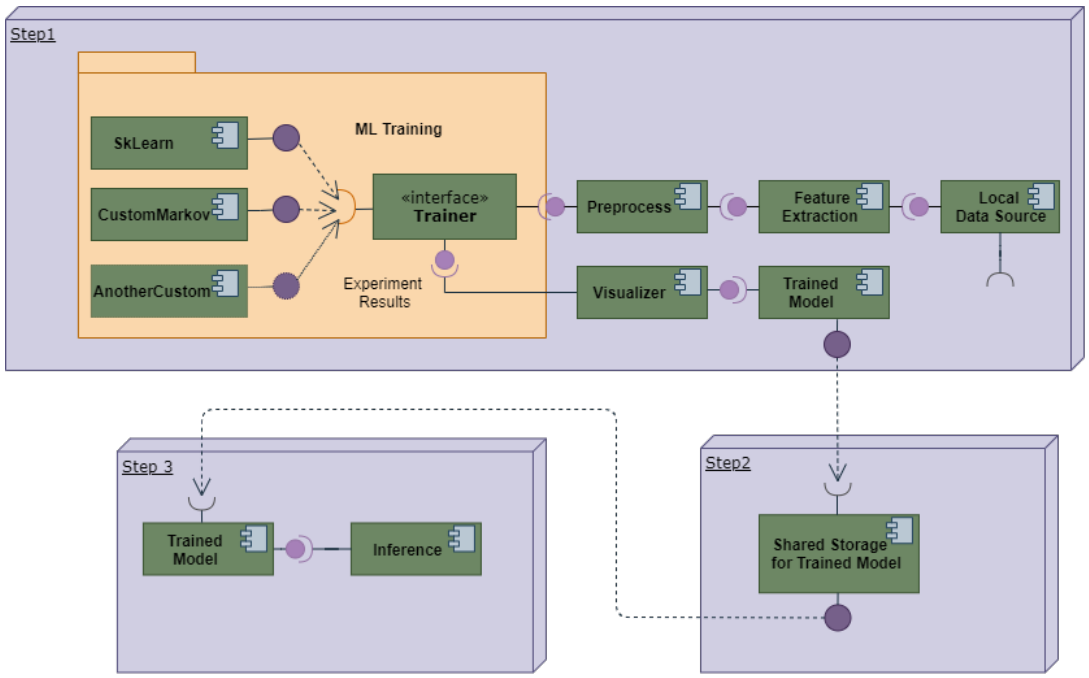


Figure 5-13: Component Diagram

Another perspective of the software can be seen through the physical view. The software will have physical components that interact and need to be deployed. The interaction between these different elements and their deployment will affect how the system works.

5.3.4. Physical View

The physical view handles how elements in the logical, process and development views must be mapped to different nodes or hardware for running the system. The deployment diagram for the IntelLight+ system shown in Figure 5-14 expresses how the pieces of a system are deployed onto hardware, and the modules in the execution environments interact.

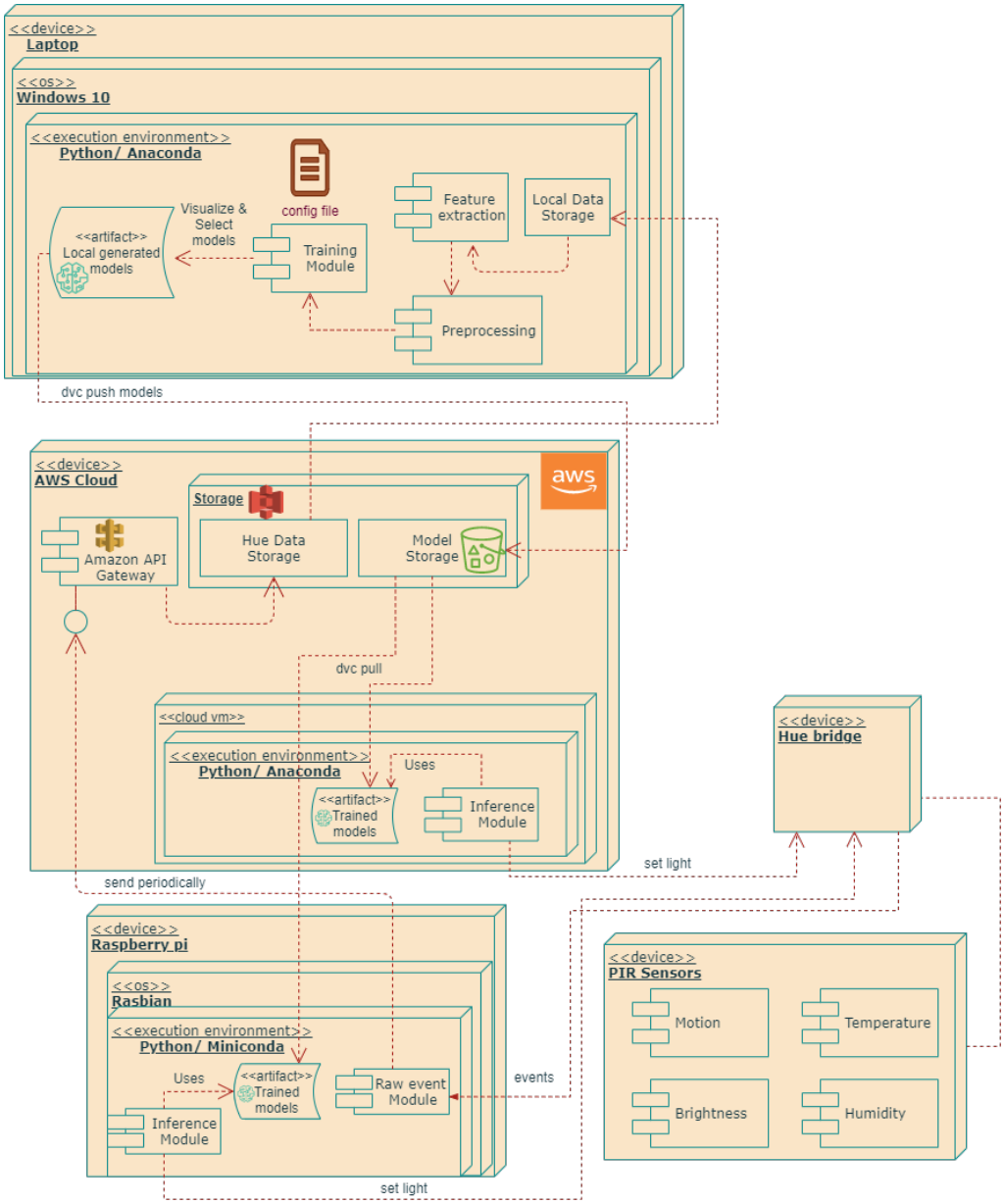


Figure 5-14: IntelLight+ deployment diagram

None of the architectural views are fully independent of each other, with elements of some views connected to others. The 4+1 view model makes the software system more versatile and helps to see a complex problem from many different perspectives.

Implementation

This chapter presents the details of the implementation phase. Chapter 5 discussed the high-level design of the IntelLight+ system, major design challenges, and architectural approaches to overcome them. The upcoming sections describe technology choices, the data gathering, and the model building pipeline phase of this project, according to the design in the last chapter.

6.1 Introduction

The IntelLight+ is an intelligent ML system that learns from different sensors and users' activities in the home context. The intelLight+ is an ML system containing components for gathering data, the ML pipeline that builds models, the feedback component for receiving feedback from the user, and the deployment module for deploying the trained model in different environments. For each of these components, different technology choices can be made.

In order to decide on the technology stack, it is important to use technologies that satisfy the requirements. In the intelLight (+) project, one of the main quality attributes was flexibility, meaning that the system should allow different ML approaches to be taken by data scientists. The system should also be modifiable in the sense that the data scientist is able to add or remove algorithms for building models.

6.2 Technology choices

Python was considered the language of choice to develop the IntelLight + system based on the previous experience of the PDEng trainee and also its popularity in data and ML domains. To choose the right ML pipeline, we compared the ML task coverage, library coverage, and level of support for a number of available libraries. We chose the pipeline that is compatible with the Signify platform.

The chosen tool should be flexible to work with different libraries and on different platforms. The main requirement for the ML pipeline we chose is that it must be cloud-agnostic. As data scientists from the university and Signify are not training their models using the cloud infrastructure for this project, the pipeline should not lock us into a single platform. It should be open-source and be supported by a well-known market player. The three options Kubeflow, MLflow, and DVC, as shown in Table 6-1, are open source and have the most number of members in their communities. Therefore, we chose to investigate them more.

Table 6-1: ML pipeline technologies survey

Pipeline	ML Task Coverage				ML Library Coverage					Platform Independent
	Data Pipeline and Versioning	Model and Experiment Versioning	Hyperparameter Tuning	Model Deployment and Monitoring	Scikit-learn	TensorFlow	Keras	PyTorch	Sagemaker	
Kubeflow		✓	✓	✓	✓	✓		✓		
ML flow		✓	✓	✓	✓	✓	✓	✓	✓	✓
DVC	✓				✓	✓	✓	✓	✓	✓

Given these requirements, we can use the MLflow for Model and experiment versioning because it manages to support a wide range of libraries while still requiring relatively little development effort to use. As an open-source project, it has a good amount of product support behind it. DVC is a good choice

for Data and Pipeline Versioning as it is very lightweight, open-source, and designed explicitly to work with Git.

Kubeflow is also a well-known pipeline backed by Google. It has a big community, but it requires a Kubernetes cluster as well as an installation of *kubectl* to work with, and it is mainly limited to the Google Cloud Platform (GCP). In this project (based on Signify preference), we do not have access to GCP, but if they want later to move to GCP, MLflow is integrated and works and scales with GCP.

6.2.1. Tracking the ML training

For tracking the ML training experiments, MLflow is used. The MLflow tracking component lets to log source properties, parameters, metrics, tags, and artifacts related to training an ML model.

MLflow tracking is based on two concepts: experiments and runs.

- An MLflow *experiment* is the primary unit of organization and access control for MLflow runs; all MLflow runs belong to an experiment. Experiments allow to visualize, search for, and compare runs, as well as download run artifacts and metadata for analysis in other tools.

- An MLflow *run* corresponds to a single execution of model code. Each run records the following information:
  - **Source**: Name of the script or notebook that launched the run, the project name, or an entry point for the run.
  - **Version**: Notebook revision if the run is from a notebook or Git commits hash if the run is from an MLflow Project.
  - **Start & end time**: Start and end time of the run.
  - **Parameters**: Model parameters saved as key-value pairs. Both keys and values are strings.
  - **Metrics**: Model evaluation metrics saved as key-value pairs. The value is numeric. Each metric can be updated throughout the course of the run (for example, to track how the model's loss function is converging), and MLflow records and allows to visualize the metric's history.
  - **Tags**: Run metadata saved as key-value pairs. The tags can be updated during and after a run completes. Both keys and values are strings.
  - **Artifacts**: Output files in any format. For example, it is possible to record images, models (for example, a pickled scikit-learn model), and data files (for example, a Parquet file) as an artifact.

The MLflow Tracking API is used for logging parameters, metrics, tags, and artifacts from running a model training experiment. The Tracking API communicates with an MLflow tracking server. The server can be configured to use a Databricks-hosted tracking server in the cloud to logs the data. The hosted MLflow tracking server has Python, Java, and R APIs.

6.2.2. Data Versioning

Data Version Control (DVC) is an open-source version control system for ML projects. It is a tool that lets the users define the pipeline regardless of the language they use.

When a problem in a previous version of the ML model is found, DVC saves time by leveraging data versioning to reproduce the model in order to identify the root of the problem. DVC can cope with the versioning and organization of considerable amounts of data and store them in a well-organized, accessible way. It focuses on data and pipeline versioning and management but also has some (limited) experiment tracking functionalities.

DVC keeps metafiles in Git instead of the big data files to describe and version control the data sets. DVC supports a variety of external storage types as a remote cache for large files. It fetches the complete data used in any experiment.

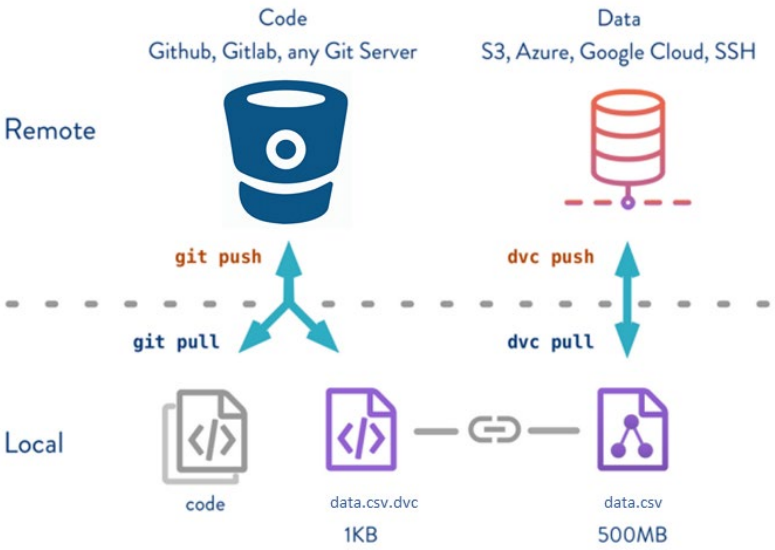


Figure 6-1: Data version control to version large data sets

6.3 Data gathering

This section explains the steps to installing the software feedback component implemented by Signify engineers as a part of the IntelLight+ platform. The feedback component is also used for gathering data for training purposes. The data gathering allows the test participants to either input their current activities manually (pro-active mode) or on-demand based on a notification. The feedback component is used for research projects that require user context labeling. In order to participate in gathering data, participants needed to:

- Set up the motion sensors in the house
- Get the bridge *API Key*, *bridgeId,* and *bridge IP* using the ready clip_debugger code
- Choose the list of sensors for each of the activities

In order to read sensor events, the raspberry reads the events from the bridge and sends them into the AWS s3 bucket. In order to set up the raspberry to send data to AWS:

- Connect the raspberry to the same wifi router that Hue bridge is connected to
- Open the python script “reading_bridge.py” in an editor and modify:
    - URL_bridge (using IP of the Hue bridge)
    - USER (using bridge API key)
    - BRIDGE_ID (using *bridgeid*)
    - API_KEY (using AWS API Key)
- Run the script (it will upload the sensors’ events to the cloud every 10 minutes)

In order to give feedback, users should install the context feedback app. After installing the app, participants log in with the AWS API key so that the app can upload their feedbacks to the cloud. Figure 6-2 shows the app environment.

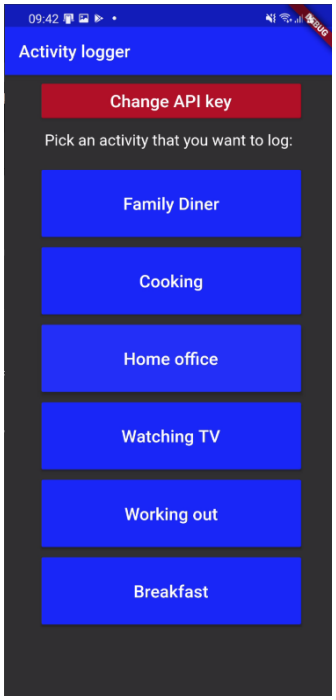


Figure 6-2: Context feedback app

Test participants can either input their current activities manually (pro-active mode) or on-demand based on a notification. By clicking on each of the listed activities, participants can set the activity manually, as shown in Figure 6-3. The system also sends notifications to users to check if they are doing a specific activity. The logic behind notification can be a distribution within the interval that the user-specified or based on model prediction. Figure 6-4 shows a sample notification that the user receives in the app.

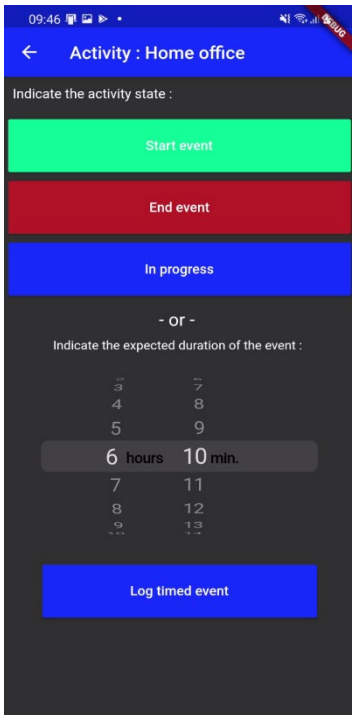


Figure 6-3: Pro-active mode feedback

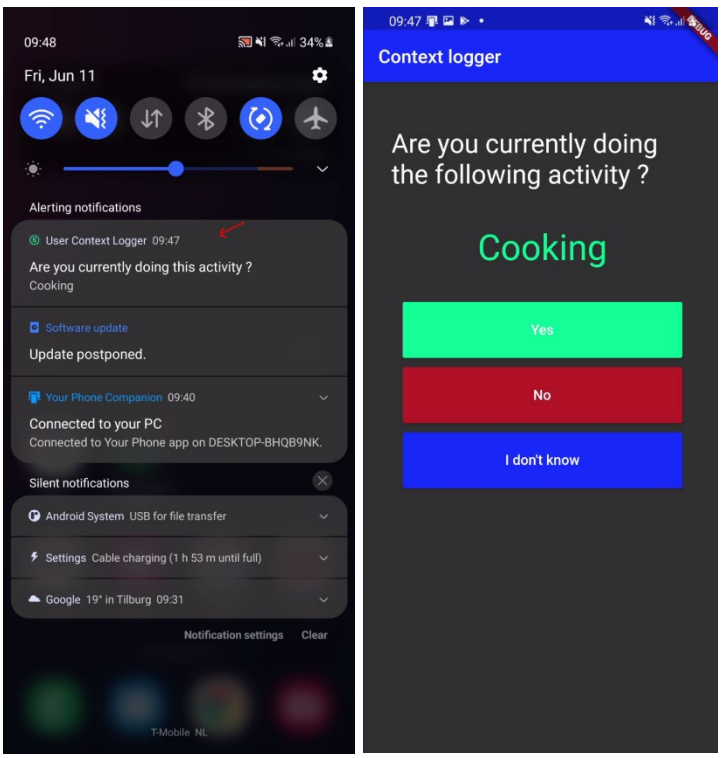


Figure 6-4: Feedback by notification

6.4 *Model building*

This section explains the details of the ML pipeline that builds models keeps track of the experiments and visualizes the result. The implementation code uses Python 3 for execution. The virtual environment decouples and isolates versions of Python and associated packages in the project implementation. Appendix A explains the steps to install and run the IntelLight+ ML pipeline to build models.

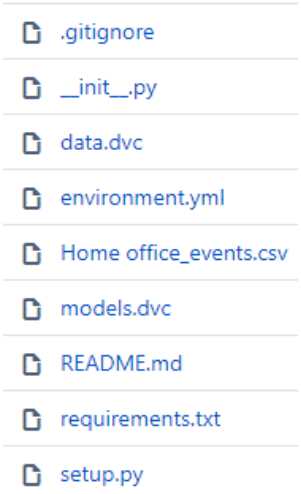


Figure 6-5: Repository root

Figure 6-5 shows the files in the root directory of the project's repository, *data.dvc* contains data version information, and *requirements.txt* includes Python libraries used in the project implementation. Figure 6-6 shows the *environment.yml* configuration file that helps to build the Anaconda virtual environment. Notice it uses *pip* to install python libraries from the *requirements.txt* file.

environment.yml

```
1  channels:
2    - anaconda
3    - conda-forge
4    - defaults
5  dependencies:
6    - python=3.8.8
7    - scipy
8    - pywin32
9    - pip
10   - pip:
11     - -r requirements.txt
12 name: mlflow-env
```

Figure 6-6: Configuration file to build anaconda virtual environment

In the root of the project, there are also the following folders:

- *.dvc:* configuration about data version control remote server
- *document:*
  - how to use git and version control
  - code standard and code template
- *src:* implementation code of project including:
  - *config:* configuration YAML file and configuration manager class
  - *ml:* ML model building code
  - *train script*
  - *predict script*

The *configuration.yml* file includes parameters to choose with which algorithms we want to run the experiment, hyperparameters, and parameters for creating training models for different activities. The configuration consists of *epochs*, *time_length, hold_time,* as well as *initial parameters* of models for each activity. The *train.py* script loads these parameters in the *MlClass* and uses them for training and building models.

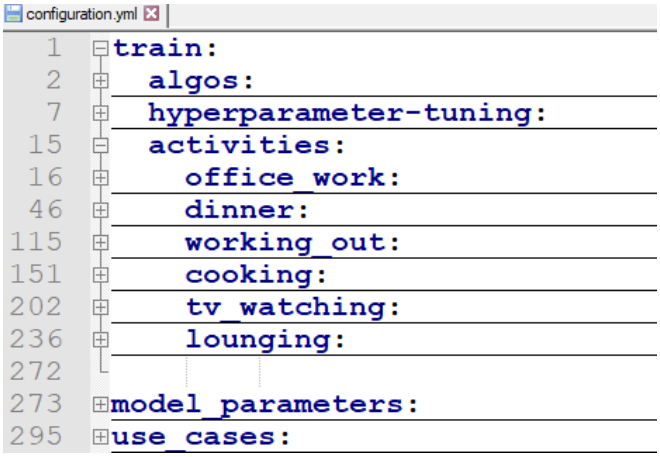


Figure 6-7: Anaconda configuration file

The ML codes for building models are in *src/model,* and the preprocessing code is in *src/preprocessing.* While the training experiments for building models are running, the results are logged and can be visualized by the web-based UI. As shown in Figure 6-8 for each activity, algorithm, and each trial in case it is doing the hyperparameters tuning. More images for the web-based experiment visualizer UI can be found in Appendix B.

		Parameters			Metrics					
Start Time	Run Name	Hold time	Time length	algorithm	accuracy	f1	loss	precision	recall	roc_auc
2021-09-08 13:57:	cooking_near	15	10	LogisticRegression	0.982	0	0.608	0	0	0.931
2021-09-08 13:55:	cooking_near	15	5	GradientBoostingClassifier	0.974	0.286	0.906	0.75	0.176	0.976
2021-09-08 13:54:	cooking_near	15	10	RandomForestClassifier	0.979	0.4	0.73	0.5	0.333	0.974

Figure 6-8: Web-based UI to visualize the ML experiments

As Figure 6-9 shows, for each experiment, the model is saved with configuration files that can be used to create a virtual environment to serve the model that is created during the experiment. The *requirement.txt* and *conda.yaml* files contain the libraries and their versions that are used to produce the model. Besides, for each experiment, the *Git commit*, and how to use the created model for prediction are also mentioned on the experiment page.

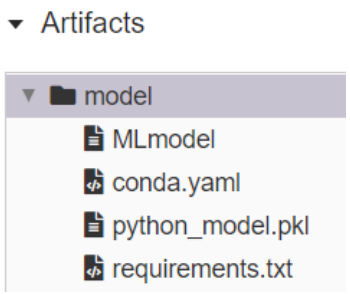


Figure 6-9: Resulted artifact for each of the experiments

6.5 *Model deployment*

In order to deploy the generated models, we use the DVC technology explained in section 6.2.2. DVC is used to make versions of chosen model experiments, and it creates a small file that will be versioned by Git, the models themselves can be shared in a remote storage place. Figure 6-10 illustrates how the git and DVC commands can be used to deploy models. More details on the steps of model deployment can be found in Appendix C.

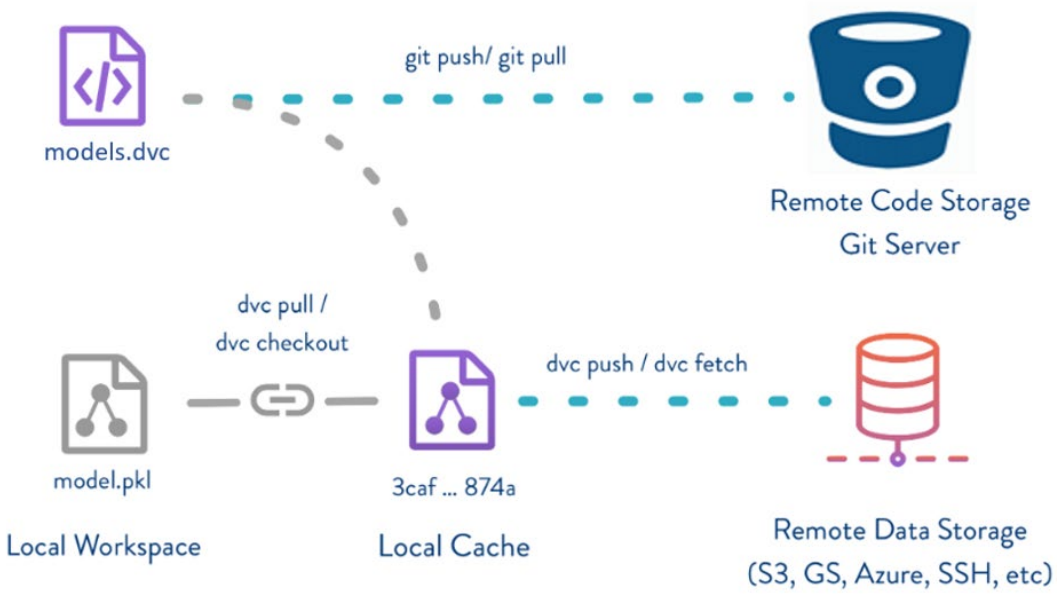


Figure 6-10: Model deployment by DVC

Models can be exchanged between any environments that support running git and python. Besides that, models can be categorized based on the environments that can be used for context recognition or even based on different interior settings of houses. Figure 6-11 shows how the shared storage is used to produce and serve models in different environments.

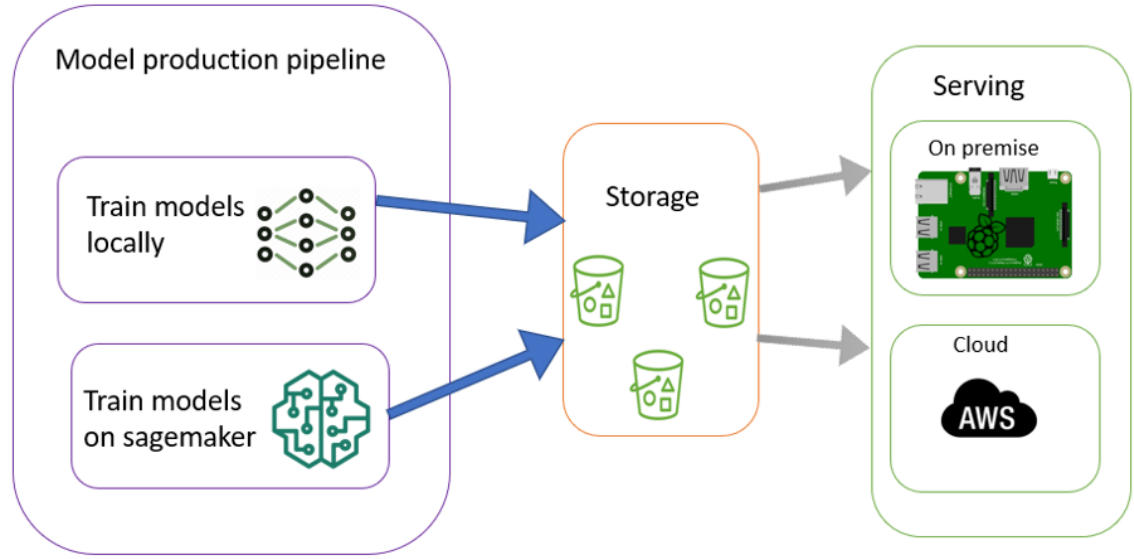


Figure 6-11: deploy models using a shared storage

Verification & Validation

This chapter aims to indicate the process of validating and verifying the steps taken in this project. The verification and validation process ensures that the proposed system has met the functional and non-functional requirements and works well.

7.1 Testing and quality assurance in ML systems

Testing and quality for ML systems are more complex than traditional software systems. Test Pyramid should be considered separately for each type of artifact (code, data, and model). There are different types of testing that can be introduced in the ML pipeline. While some aspects are inherently non-deterministic and hard to automate, many kinds of automated tests can add value and improve the overall quality of your ML system:

Validating data: Tests can validate input data against the expected schema or validate our assumptions about its valid values. For instance, they fall within expected ranges or are not null. For engineered features, we can write unit tests to check they are calculated correctly. It is possible to check if the numeric features are scaled or normalized, one-hot encoded vectors contain all zeroes and a single 1, or missing values are replaced appropriately.

Validating the model quality: While ML model performance is non-deterministic, Data Scientists usually collect and monitor a number of metrics to evaluate a model's performance, such as error rates, accuracy, AUC, ROC, confusion matrix, precision, recall, etc. They are also useful during parameter and hyper-parameter optimization. As a simple quality gate, we can use these metrics to introduce Threshold Tests in our pipeline to ensure that new models don't degrade against a known performance baseline.

As the data science process is very research-centric, it is common that data scientists will have multiple experiments to try, and many of them might not ever make it to be deployed for inference testing. IntelLight+ supports this governance process; it captures and displays the information that will allow data scientists and researchers to decide if and which model should be promoted.

7.2 Verification

Verification is the process of evaluating a system or component to determine whether the products of a development phase satisfy the conditions. It is a continuous process for checking the system that is being built to ensure it adheres to certain specifications.

During the development phase, the IntelLight + system was verified in two ways: unit testing and code consistency checking reviews. These processes are elaborated in the following sections.

7.2.1. Unit testing

Unit test is a software testing method that checks if the individual units of the corresponding software have expected behavior. The developer typically performs these tests by writing additional code that automatically tests the software. The unit tests are used to test the newly implemented features and ensure that the existing functionalities are not broken.

For unit testing, different parts of the IntelLight+ ML pipeline, which are explained in sections 5.2.1 and 5.3.1 are covered. These unit tests are listed in table 7.1. The *unittest* unit testing framework, initially inspired by *JUnit*, was used to write the unit tests. It supports test automation, setup and shutdown code for tests, aggregation of tests into collections, and independence of the tests from the reporting framework.

Table 7-1: Unit tests for testing the ML pipeline

Unit Test Name	Endpoint Type	Status
test_experiments_for_single_algorithm_tracked	Visualizing experiments	Passed
test_experiments_for_single_activity_tracked	Visualizing experiments	Passed
test_experiments_metrics_tracked	Visualizing experiments	Passed
test_experiments_configuration_params_tracked	Visualizing experiments	Passed
test_experiments_data_version_tracked	Visualizing experiments	Passed
test_model_created_for_deployment	Model building	Passed
test_model_with_conda_env_created_for_experiment	Model building	Passed
test_grand_truth_is_expanded	Data Preparation	Passed
test_each_activity_has_related_sensor_for_training	Data Preparation	Passed
test_loading_configuration_yaml_file	Data Preparation	Passed

7.2.2. Code consistency checking

As mentioned in section 6.2, the pipeline was developed with Python. The PEP8 style guide for Python was used to ensure that the coding style was consistent throughout the whole code repository. This style guide was developed by the creators of the Python language and has been widely accepted. PEP8 compliance options were enabled in the Integrated Development Environment (IDE) used in this project to check the coding style automatically. This provided basic guidelines (i.e., tabs vs. spaces for indenting, indent width, line length) for the code and made it easier to read.

7.3 Validation

Validation is the process of evaluating a system or component at the end of the development process to determine whether it satisfies specified requirements. Two aspects need to be taken into consideration in the validation process. First, does the built product address the functional requirements? Does it do what it says in the functional requirements? Second, does the product satisfy the non-functional requirements? For the non-functional requirements, the validation addresses the quality attribute scenarios. In the context of this project, the system under development was validated by the direct stakeholders at various stages, which are discussed in detail in the following sections.

7.3.1. Regular Stakeholder Feedback

Following the scrum iterative and incremental development approach, the results were continuously validated by the stakeholders. The product was accessible by direct stakeholders (data scientist and P.h.D researcher) on the code repository, and a meeting was scheduled with the main stakeholders, which included a progress report and a demo of the current version of the product. During these meetings, several architectural diagrams were used to explain to the stakeholders how the IntelLight+ infrastructure would be implemented. Based on these discussions, the stakeholders could identify whether the development activities were progressing in the right direction or not. The mentioned diagrams are presented and explained in chapter 5. The feedback was provided on the completeness and correctness of the product under development mainly by direct stakeholders (the data scientist and Ph.D. researcher).

7.3.2. Requirements status

The final prototype of the IntelLight+ infrastructure implemented all the requirements described in chapter 4. An overview of the functional requirements is shown in Table 7-2. The prototype was also demonstrated to the IntelLight+ stakeholders during the final prototype demonstration. The direct stakeholders accepted that the system met their expectations and satisfied all the elicited requirements. Moreover, several suggestions were provided by the PDEng trainee during the demonstration that indicated

the direction in which the stakeholders may further improve the IntelLight+ infrastructure. These feedbacks are listed in section 9.3.

Table 7-2: Statuses of functional requirements after implementation

ML pipeline				
Req.#	**Priority**	**Description**	**Verification**	**Status**
FR3.1	Must	The system shall keep track of the version of the code, data, and the parameters that have been used for each training experiment of the model.	The MLflow Tracking API visualizes parameters, code version, metrics, and output files when running the machine learning code.	Satisfied
FR3.2	Must	The system should provide the template that modularizes the project to different meaningful data science components.	The code template separates different concerns: data and source code and result models. The ml codes are in different packages like preprocessing, data source, model. Data is also separated at the highest level based on activities.	Satisfied
FR3.3	Must	The system shall provide a pipeline that keeps track of all the (python) libraries and their versions that have been used for training a model.	A requirement.txt file keeps all libraries that are used in the project.	Satisfied
FR3.3.1	Must	The system shall provide an ml pipeline that works with different ml libraries.	Currently, sklearn and tensorflow are used.	Satisfied
FR3.4	Must	The system shall visualize the models with a range of different hyper-parameters and let data scientists compare them.	The Mlflow UI shows the parameters for the optimal model used for training each algorithm.	Satisfied
FR3.4.1	Must	The system shall provide the same API for training different models	Same API used for training models of different libraries.	Satisfied
Deployment				
Req.#	**Priority**	**Description**	**Verification**	**Status**
FR3.5	Must	The system shall be able to deploy and test the inference model on-premise and remote server environments in the same way.	IntelLight+ uses python, git, shared storage, and DVC to deploy models on different environments.	Satisfied
FR3.6	Could	The system should support providing different models for users based on different settings in their homes.	Using a combination of git tag and DVC make it possible to make customized categories of deployment	Satisfied
FR3.7	Could	The system should let data scientists deploy and test different models for users based on the different categories of the settings of the users' homes.	Using a combination of git tag and DVC make it possible to make customized categories of deployment	Satisfied

Besides functional requirements, QAs also revisited, and the design of IntelLight+ took into consideration the QAS mentioned in section 4.5.1. Figure 7-1 shows how the designed system is modifiable to add new algorithms for making models with the lowest cost.

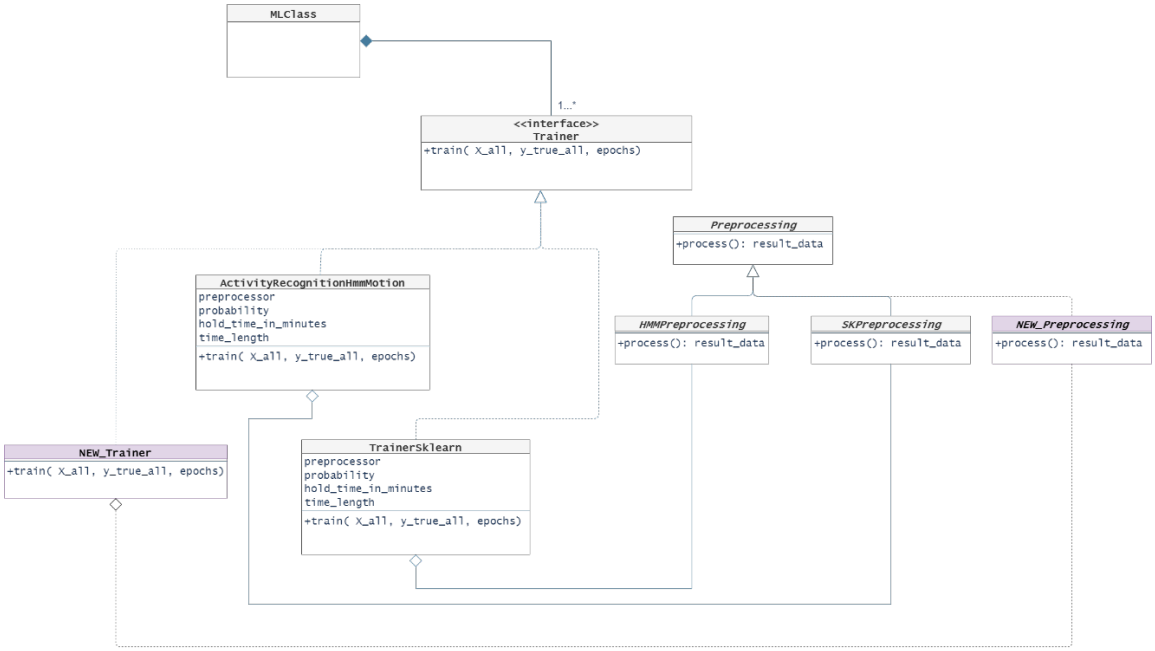


Figure 7-1: QA Modifiability revisited in design

7.3.3. Project Goal Evaluation

To validate the product, the initial questions in problem definition 2.1 are revisited to see if the system satisfies the main requirements recognized by this project. The system is validated if it has an apparent response to these questions that address the customers' actual needs.

How can data scientists use different data sources easily for context-aware lighting? In order to achieve a good result for detecting the context, the first step after collecting the data is to explore which data is useful for building models. In IntelLight+, with DVC, data scientists can tag different data sources for exploration. For this project, we used data from hue motion sensors to gather the needed features. The other data sources to experiment with are audio and motion sensors from smartphones or smartwatches. The feedback component helps to annotate the collected data from these sources.

How can data scientists keep track of the code and data used for building the model? This is one of the main requirements for the pipeline. In order to satisfy this need, we use *DVC* for versioning the data so that we know which data is used for building the model. We also use *MLflow* to keep track of the data version and git commit id. *Mlflow* also saves the configuration file used for setting the training parameters and other parameters needed.

How can data scientists try different models and reproduce a model? In the pipeline, we always use a virtual environment, and every time the *train* code is executed, the libraries that have been used for training are kept tracked and saved as an artifact. If a new ML library is going to be used, it can be easily added to the *requirement.txt* file in the repository's root that keeps a list of all the libraries used in the project. Besides, the code and data version for each experiment are logged by *MLflow*. In this way, models can be easily reproduced as each experiment's environment, code, and data are saved.

What are the steps (in the system) for choosing the best parameters for the trained models? In the training script, the train algorithms are called by a hyperparameter tuning method. Different algorithms work better with different hyperparameter optimization methods. To be able to choose a set of optimal hyperparameters for learning algorithms, two tunning methods *hyperopt*, and *optuna* are implemented. The *hyperparameter-tuning* method can be set in the *configuration.yml* file.

Project Management

The project management activities started from the very beginning of this PDEng project. This chapter elaborates on the management process of this project. It starts by describing the method deployed in this project. Then the project work breakdown is explained with the work breakdown structure. Next, project planning and scheduling are introduced.

8.1 Introduction

As described in chapter 1, the IntelLight (+) project was one of two separate joint projects of the Intelligent Lighting Institute ILI from TU/e and the Signify. Being part of a collaborative project, IntelLight had stakeholders from different organizations and with diverse backgrounds. This contributed greatly to the complexity of this project. Therefore, proper project management became one of the key activities to steer this project in the right direction.

8.2 Way of Working

The time duration of the project was ten months. This constraint limited the number of features or user stories that could be implemented. The requirement analysis process explained in chapter 4 made sure that these features or user stories were elicited as early as possible. However, in the case of real-life multi-disciplinary projects like this one, requirements can be rather dynamic in nature, and the used project management process should be able to deal with these uncertainties. To be able to do that, a hybrid approach was used. This approach bought some ideas from the Waterfall methodology [1] to deal with the time constraint and, to some extent, a fixed set of deliverables. Several concepts from the Agile methodology were also used to deal with the dynamic nature of the project.

In the whole period of the project, as is mentioned in section 8.6, we had weekly meetings. The system was designed and developed using Scrum methodology. Scrum [20] is a lightweight process framework that is a subset of the agile methodology. Scrum defines a set of roles, meetings, and fixed-length iterations known as sprints. Each sprint can have a duration of one to four weeks.

Figure 8-1 depicts the key activities and roles of a Scrum process. A product backlog is a prioritized list of work for the development team that is derived from the roadmap and its requirements. The most important items are shown at the top of the product backlog, so the team knows what to deliver first. Sprint planning aims to define what can be delivered in the sprint and how that work will be achieved. The sprint backlog is a list of tasks identified by the Scrum team to be completed during the Scrum sprint.

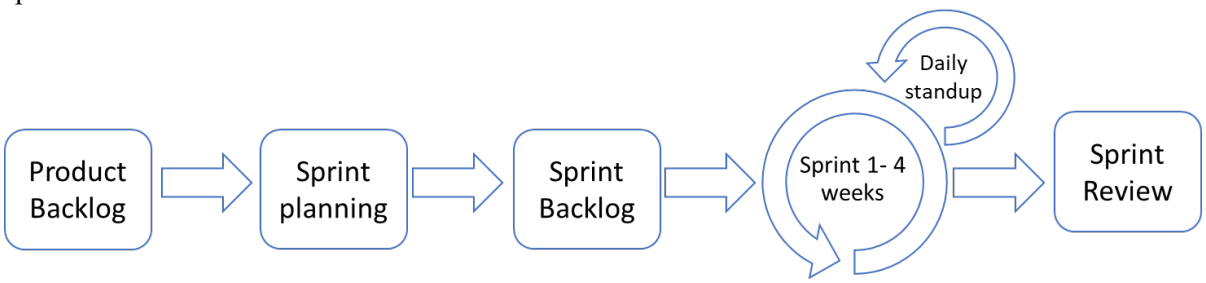


Figure 8-1: The scrum process

8.3 *Work-Breakdown Structure (WBS)*

In this section, the Work-Breakdown Structure of the project is discussed. The project is divided up into four major categories: Planning and Management, Analysis, Design & Implementation, and Documentation. Figure 8-2 shows the detailed activities conducted in each category.

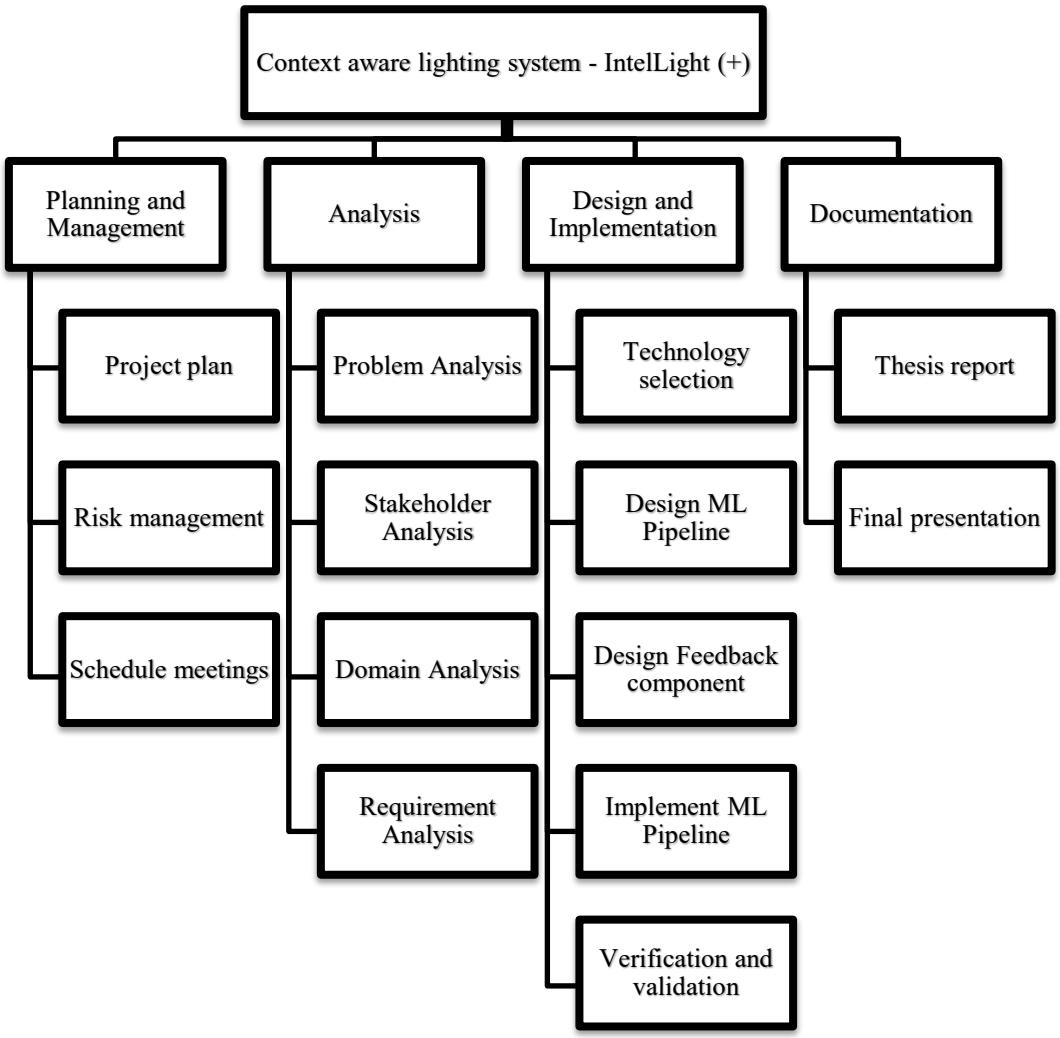


Figure 8-2: Work-breakdown structure of the project

8.4 *Project Planning and Scheduling*

In the whole period of the project, we had weekly meetings. The system was designed and developed using scrum methodology. Figure 8-3 shows the roadmap of the IntelLight + project with different phases.

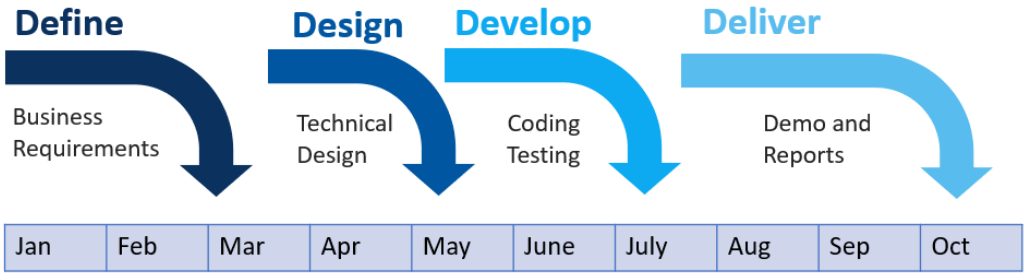
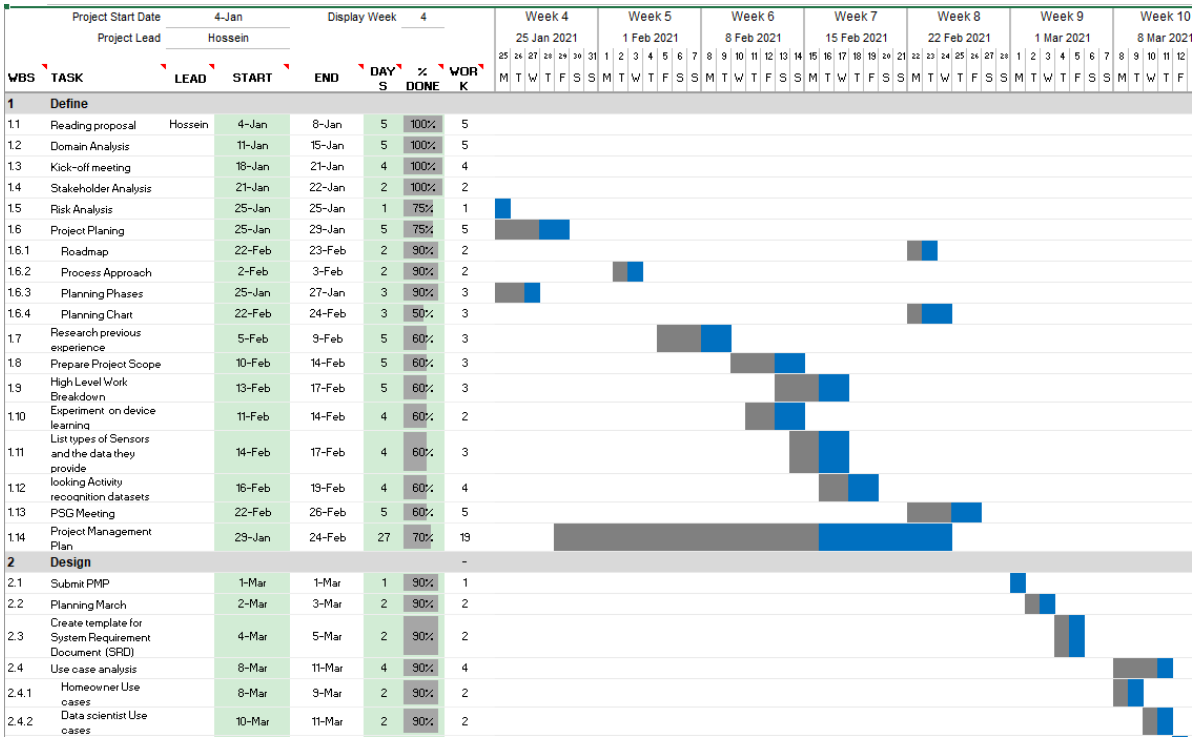


Figure 8-3: Project roadmap

8.5 *Project Timeline*

The project timeline helped not only to evaluate project progress but also to visualize the impact of one activity on others. The project timeline made sure that the domain analysis and requirement elicitation phases were scheduled before any implementation-related activity.

The project plan was tracked using Microsoft Excel and was modified iteratively. The final timeline is shown in Figure 8-4. It is divided into four primary activities: Define, Design, Develop, and Deliver.



Project Start Date 4-Jan | Project Lead Hossein | Display Week 4

WBS	TASK	LEAD	START	END	DAYS	% DONE	WORK
1	**Define**						
1.1	Reading proposal	Hossein	4-Jan	8-Jan	5	100%	5
1.2	Domain Analysis		11-Jan	15-Jan	5	100%	5
1.3	Kick-off meeting		18-Jan	21-Jan	4	100%	4
1.4	Stakeholder Analysis		21-Jan	22-Jan	2	100%	2
1.5	Risk Analysis		25-Jan	25-Jan	1	75%	1
1.6	Project Planing		25-Jan	29-Jan	5	75%	5
1.6.1	Roadmap		22-Feb	23-Feb	2	90%	2
1.6.2	Process Approach		2-Feb	3-Feb	2	90%	2
1.6.3	Planning Phases		25-Jan	27-Jan	3	90%	3
1.6.4	Planning Chart		22-Feb	24-Feb	3	50%	3
1.7	Research previous experience		5-Feb	9-Feb	5	60%	3
1.8	Prepare Project Scope		10-Feb	14-Feb	5	60%	3
1.9	High Level Work Breakdown		13-Feb	17-Feb	5	60%	3
1.10	Experiment on device learning		11-Feb	14-Feb	4	60%	2
1.11	List types of Sensors and the data they provide		14-Feb	17-Feb	4	60%	3
1.12	looking Activity recognition datasets		16-Feb	19-Feb	4	60%	4
1.13	PSG Meeting		22-Feb	26-Feb	5	60%	5
1.14	Project Management Plan		29-Jan	24-Feb	27	70%	19
2	**Design**						-
2.1	Submit PMP		1-Mar	1-Mar	1	90%	1
2.2	Planning March		2-Mar	3-Mar	2	90%	2
2.3	Create template for System Requirement Document (SRD)		4-Mar	5-Mar	2	90%	2
2.4	Use case analysis		8-Mar	11-Mar	4	90%	4
2.4.1	Homeowner Use cases		8-Mar	9-Mar	2	90%	2
2.4.2	Data scientist Use cases		10-Mar	11-Mar	2	90%	2

Week 4 (25 Jan 2021) | Week 5 (1 Feb 2021) | Week 6 (8 Feb 2021) | Week 7 (15 Feb 2021) | Week 8 (22 Feb 2021) | Week 9 (1 Mar 2021) | Week 10 (8 Mar 2021)

Figure 8-4: Project timeline in Microsoft Excel

8.6 *Communication Plan*

In order to monitor and regulate the project progress and direction, it is important to set up a clear and regular communication channel to avoid any misunderstanding and encourage early feedback. Most of the meetings that took place during the execution of this project can be divided into three different categories: weekly update meetings, monthly update meetings, and other meetings that were organized on demand.

8.6.1. Weekly Update Meetings

These meetings were scheduled at the very beginning of the IntelLight (+) project to occur every week on Friday until the end of the project. The usual participants were the company supervisors, the TU/e supervisor, and the PDEng trainee. The purpose of these meetings was to demonstrate incremental and small updates to the major stakeholders. These updates included the tasks that had been completed in the previous week, any blocking issues, and plans for the upcoming week. The weekly meetings made sure that the project was running in the right direction and provided the option to spot any misunderstanding as early as possible. These meetings played the role of the sprint review in the define and the delivery phase of the project. A demonstration was provided during the weekly meeting for the weekly task. Feedback was reviewed, evaluated, confirmed, and implemented when feasible.

8.6.2. Project Steering Group Meetings

The Project Steering Group (PSG) group consisted of major project stakeholders, including the PDEng trainee. During the project, the PSG had monthly meetings, typically on the last Friday of the respective month. The main purpose of these meetings was to encourage feedback from the stakeholders. Typically, these meetings started with a demonstration where the trainee explained the major features that were implemented since the previous PSG meeting. This was followed by a high-level discussion about the design and the architecture of the system, requirements as well as risks identified by the PDEng trainee. A high-level plan for the coming month was also discussed towards the end of these meetings.

8.6.3. On-Demand Meetings

Besides the regular weekly and monthly meetings, several other meetings took place during the execution of the IntelLight (+) project. Although these meetings were not regular and mostly scheduled on-demand, they played a key role in communicating with the key stakeholders and in understanding and implementing the project context.

8.6.4. Communication Medium

Due to the COVID-19 pandemic, everyone was advised to work from home if possible, and all academic activities were moved online. Microsoft Teams was chosen for all meetings as well as discussions remotely. The confluence was set as the preferred way for exchanging digital media (e.g., documents, images). The Jira was set for tracking the progress of user stories.

8.6.5. Performance Evaluation

The performance of the trainee was evaluated based on various criteria by the project supervisors, and feedback was provided. The feedback helped the trainee to identify improvement points and steer the project in the right direction.

Conclusions

This chapter elaborates on the results achieved, delivered artifacts, and recommendations for future work.

9.1 Results

The scope of this project included the automation of the parts of the ML lifecycle. The goal was to automate the process of producing models, getting feedback from users, and deploying the qualified models to target environments. During this project, an ML pipeline was developed where provided researchers and data scientists an infrastructure to build the models, investigate them visually on web-based UI, and use a shared repository to share the qualified ones to be deployed in different environments.

IntelLight+ provides a web-based UI for analyzing the result of experiments for different activities. It allows data scientists to use different algorithms with different sets of parameters and hyperparameters. For each of the activities, the web-based UI will list the best model of each algorithm based on hyperparameter tuning. Then the researcher can choose the models for each activity and deploy them based on the guidelines to the target environment(cloud, raspberry).

The major goal of this project was to develop a standardized platform that would allow data scientists to develop ML algorithms, compare them and test them in the target environments without the need to hassle all the works not related to developing the ML models manually.

9.2 Delivered Artifacts

The following artifacts are delivered to the stakeholders after successfully completing this PDEng graduation project. Artifacts are delivered to the clients digitally via the git repository, the confluence, and email attachments.

- **IntelLight infrastructure source code:** This refers to all the source codes produced during the PDEng project by the trainee and representatives from Signify. The codes can be found in the Bitbucket repository https://www.code.dtf.lighting.com/projects/INTLT/repos/activityrecognition/browse
- **Deployment instructions:** The instruction to deploy models on the different environments alongside a guideline for the repository, data versioning, code merge request process, and project code style are gathered in the documents directory of the project repository.
- **Video demonstration and presentation slides:** A video was produced and delivered to the customer for demonstration purposes. This video can be found on the SharePoint of the project on university servers. This screen recording shows how a data scientist would use the IntelLight+ infrastructure to experiment on the algorithms and set the parameters and hyperparameters for the training. The web-based visualization and deployment will also be covered. Besides the video, presentation slides for the final presentation and progress meeting are also included in the confluence and Sharepoint.
- **Project report:** It refers to this PDEng graduation report. This report not only explains the activities and designs involved in the development of the IntelLight+ system but also recommendations for improving and extending it.

9.3 Recommendations and future work

One of the main features of the IntelLight+ system is automating the ML pipeline by keeping track of the steps in the lifecycle of the ML process. In section 4.5 we saw that one of the important QAs of the IntelLight+ system is flexibility. This allows adding new features to the ML pipeline easily. Currently, the pipeline uses the *mlflow* library to keep track of the experiments in the local machines that are

running the ML pipeline. This can easily be configured to save the experiments in a shared cloud environment in case a team of data scientists is going to work on the project. The pipeline itself can also be integrated into all major cloud providers as the *mlflow* library is backed by Databricks. Databricks is fully integrated and supported in AWS, Azure, and GCP.

One other feature that can be added to the current pipeline is to deploy each of the created models with their specific virtual environment that contains all the libraries required to run that model for prediction. Currently, when running the training experiments for each of the trained models, a configuration file that includes the libraries that have been used for building the model is also created. The next improvement can be making a virtual environment for the model from the configuration file and deploy each model to an independent endpoint with its specific environment. This feature is especially useful for context recognition in the cloud.

Finally, IntelLight+ is an MLOps project that helps simplify the management, logistics, and deployment of machine learning models for machine learning researchers. In order to reach the desired outcome, it is necessary to use data pipelines to identify the data sources needed to solve a business problem. A data pipeline is needed to make that data ready for analytics, the analytics step(s), and the delivery of results to the ML scientists or apps that will use them. This means incorporating big data analytics tools like Spark and Hadoop.

Project Retrospective

This chapter discusses my gained experience during the period of the IntelLight+ project. This is a reflection on my technical and organizational lessons from the PDEng graduation project.

10.1 Introduction

This graduation project assignment was a very challenging and fulfilling experience for me. During the past ten months, my technical and non-technical skills improved. This project also provided me the perfect opportunity to practice the skills that we gained through various workshops, courses, and training projects during the first year of my PDEng.

This project gave me an opportunity to broaden my horizon with technical skills in the data domain. The IntelLight+ project deals with data as input and generates the ML models and their measurements. Also, other skills that are related to software development were exercised, such as managing and planning a project, communicating with stakeholders, and analyzing results effectively.

10.2 Technical lessons

Working in the lighting domain was a completely new experience for me. Therefore, understanding the domain was the first major challenge to deal with. Tips and feedback from my TU/e as well as company supervisors were crucial for me to be able to navigate through this unfamiliar landscape. To identify misconceptions and knowledge gaps, I used various diagrams to communicate my ideas. This way, I was able to gather enough information to move forward with the technical design and implementation.

At the beginning of implementing the project, I investigated some of the domain technologies to see how they could address the project needs. During that time, It became confusing as, on the one hand, there was a lot of tutorials that claimed to teach what we were looking for, such as ML pipeline, tracking experiments, and deployment with a short tutorial, and on the other hand, we could hear a lot that ML pipeline, model deployment is very complicated. In my idea, people often use the same terms while they mean different concepts or implementing different levels of features of a single idea.

Therefore, specifying the requirements and validating them based on the use cases was important to realize what technologies are really needed when we are talking about a concept. Based on meetings with stakeholders, the perception of actual requirements for the project and the required technology stack evolved gradually. One important lesson I learned is not to make assumptions about the infrastructure that will be available. First, talk to stakeholders to clarify the aim of a requirement. Then use the main requirements to choose your technology options.

My work was to design and implement a machine learning pipeline for building different models. On the other hand, It was needed to deploy these models in different environments. The technologies in this domain are still evolving radically, and I was/am inexperienced with many of them. In many instances, I spent a long time trying to figure out a simple thing that should have taken no time. I realized that I could do two things to encounter this: Ask my colleagues or learn myself. Both are valid, but as the domain is not yet very mature, it might be more efficient to search first and then look for options and then ask colleagues' opinions. I tried to survey the technologies we can use to build the ML Pipeline and then look up the pros and cons of available choices. I gathered the options together, and I called upon stakeholders from Signify who had more experience. Finally, based on the requirements of the project, we agreed on a technology.

My takeaway from this experience is when working within a new domain and encountering a problem that my experience is limited in, do the following in order:

- Gain basic knowledge in the domain.
- When encountering a problem, look up and use the documentation and forums to figure things out yourself.
- If this takes a long time, ask your local expert. This should not be limited to your team or group. Anyone with expertise can have an excellent benefit for a small amount of their time.

10.3 Organizational lessons

IntelLight+ project was my first experience in a system architect and designer role. In addition, it was also my first project in a large international company. When the project started, we had our first meeting three weeks after the project's start date due to restructuring in Signify. After the first meeting, my observation was that there was a gap between our understanding of the project's domain with Signify stakeholders. In the beginning, this was quite challenging because my position in this organization was not yet clear. Luckily, after some time, my relevant stakeholder became clear. I communicated with main stakeholders from Signify to make it clear that the PDEng design project is a separate project from the research project. In addition, my takeaway was that one could significantly benefit from the connections of colleagues. For example, I had to get in touch with privacy/security officers and IoT software engineers. Instead of going around looking for them myself, I asked my supervisor, who already knew whom to contact.

After knowing the main stakeholders, the project phases were defined to plan the progress of the project period. These phases included define, design, develop, and deliver. The first phase was more about reading the project proposal and talking with different stakeholders to realize their expectations and more details about the project's domain.

This project was a combination of the software design problem and intelligence system in practice. At first, like many other projects, which are a combination of research and industrial case, my contribution was uncertain and lacked domain expertise. I knew artificial intelligence before the start of this project. This project started with a research project. It was unclear that this project could answer the which of customer needs and which of them can be answered by the research project. Hence, formulating the requirements was not an easy process. The main point of reflection for me is that better preparation for the scope of each project was needed. Having a better view of the goal of the projects and how they complete each other is not only important but necessary. Of course, you can never predict all things, so a healthy open-minded attitude towards this fact is also necessary. On the other hand, persuading the stakeholders that in order to apply ML algorithms, an Ml pipeline can guide through a valuable result was challenging. It was also engaging in the sense that It challenged me to prove the expected results as soon as possible to be sure about the validity of the envisioned roadmap.

Being part of the scientists' team was an honor and challenging at the same time. During various team meetings and presentations, my role required me to find a common way of communicating that should work for everyone. Throughout the execution of the project, I shared my design ideas with the stakeholders and received their feedback. In other words, by mentioning the challenges and difficulties, my aim was to update them for the roadmap and plan and get feedback from them. In my opinion, this experience will help me become a better engineer and team player.

One organizational decision in this project was to use a combination of scrum sprints. We made a scrum team of me and direct stakeholders (data scientist and researcher). During the design and development phase, other engineers from Signify were occasionally added to the scrum team. In the last phase, we turned back to traditional software methodology as we were mainly focused on wrapping up the project and deliver the product rather than adding new features. My gained lesson learned from this experience is not to become biased to a specific software methodology. Instead, see which methodology serves you and your team better in your project.

References

[1] E. Rondolat, "Philips Lighting is now Signify," *Signify*. p. Press releases/2018, 2018, [Online]. Available: https://www.signify.com/en-nz/our-company/news/press-releases/2018/20180516-philips-lighting-is-now-signify.

[2] I. Heynderickx, "Intelligent Lighting Institute," 2010. https://www.tue.nl/en/research/research-institutes/top-research-groups/intelligent-lighting-institute/about-ili/.

[3] A. K. Gopalakrishna, T. Özçelebi, A. Liotta, and J. J. Lukkien, "Exploiting machine learning for intelligent room lighting applications," in *2012 6th IEEE International Conference Intelligent Systems*, 2012, pp. 406–411, doi: 10.1109/IS.2012.6335169.

[4] F. Wang, X. K. Liu, and F. Gao, "Fundamentals of solar cells and light-emitting diodes," in *Advanced Nanomaterials for Solar Cells and Light Emitting Diodes*, Elsevier, 2019, pp. 1–35.

[5] P. Samadi Khah, "Eventing in the Hue system," Technische Universiteit Eindhoven, 2018.

[6] V. Singhvi, A. Krause, C. Guestrin, J. H. Garrett, and H. Scott Matthews, "Intelligent light control using sensor networks," in *SenSys 2005 - Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, 2005, pp. 218–229, doi: 10.1145/1098918.1098942.

[7] A. Krioukov, S. Dawson-Haggerty, L. Lee, O. Rehmane, and D. Culler, "A living laboratory study in personalized automated lighting controls," in *Proceedings of the third ACM workshop on embedded sensing systems for energy-efficiency in buildings*, 2011, pp. 1–6.

[8] R. Magielse, P. Ross, S. Rao, T. Özçelebi, P. Jaramillo Garcia, and O. Amft, *An Interdisciplinary Approach to Designing Adaptive Lighting Environments*. 2011.

[9] T. Mitchell, "Machine Learning," McGraw Hill, 1997, pp. 239–258.

[10] P. Cunningham, M. Cord, and S. J. Delany, "Supervised Learning BT - Machine Learning Techniques for Multimedia: Case Studies on Organization and Retrieval," M. Cord and P. Cunningham, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 21–49.

[11] I. H. Witten, E. Frank, M. A. Hall, C. J. Pal, and M. DATA, "Practical machine learning tools and techniques," in *DATA MINING*, 2005, vol. 2, p. 4.

[12] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Mach. Learn.*, vol. 6, no. 1, pp. 37–66, 1991.

[13] A. K. Gopalakrishna, T. Ozcelebi, J. J. Lukkien, and A. Liotta, "Evaluating machine learning algorithms for applications with humans in the loop," in *2017 IEEE 14th International Conference on Networking, Sensing and Control (ICNSC)*, 2017, pp. 459–464.

[14] V. Lakshmanan, S. Robinson, and M. Munn, "Machine learning design patterns," O'Reilly Media, 2020, pp. 282–294.

[15] D. Sato, A. Wider, and C. Windheuser, "Continuous Delivery for Machine Learning," *Martin Fowler*, 2019. https://martinfowler.com/articles/cd4ml.html.

[16] Anil Gupta, "Machine Learning Challenges in the implementation of Industrial Internet of Things," 2017. https://www.iiot-world.com/artificial-intelligence-ml/artificial-intelligence/machine-learning-challenges-in-the-implementation-of-industrial-internet-of-things/.

[17] R. C. Martin, J. Newkirk, and R. S. Koss, *Agile software development: principles, patterns, and practices*, vol. 2. Prentice Hall Upper Saddle River, NJ, 2003.

[18] R. C. Martin, *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Boston, MA: Prentice Hall, 2017.

[19] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 3rd ed. Addison-Wesley Professional, 2012.

[20] M. Huo, J. Verner, L. Zhu, and M. Ali Babar, "Software Quality and Agile Methods," *Comput. Softw. Appl. Conf. Annu. Int.*, vol. 1, pp. 520–525, Oct. 2004, doi: 10.1109/CMPSAC.2004.1342889.

Appendix A. Installing and Running the IntelLight+ Project.

In order to install the ML pipeline, after getting the code by `git clone https://www.code.dtf.lighting.com/scm/intlt/activityrecognition.git` from Signify' Bitbucket server, it is needed to do the following steps.

Installation

1. Create & activate the conda environment using the environment.yml file **at** the root of the project. This takes quite some time to install all required libraries in conda environment. (you need **first to** install the anaconda)

```
conda env create -f environment.yml

activate mlflow-env
```

Or

```
conda activate mlflow-env
```

`mlflow-env` is the name of the virtual environment in the `environment.yml` file: `name: mlflow-env`

2. You can set the conda virtual environment as the default running environment in your IDE.

Pycharm: Go to `files > settings`, search for `project interpreter`, open it, click on the gear button and choose the conda environment you created.

Spyder: *) open command tool and activate virtual environment *) install spyder-kernel: 'pip install spyder-kernels==1.9.1' *) in spyder: `Tools -> Preferences -> Python interpreter`: select the python.exe that is present in venv (you can find the path by entering in command prompt: `which python`

3. Make importing files work together. In the root directory, run:

```
pip install -e .
```

After running this command `*.eff-info` folder will be built. `intellight_project.egg-info`

Show experiments

First, train models by running the `train.py` file**,** which is inside the `src` directory.

```
python src/train.py
```

The models and metrics are already logged via MLflow in `train.py` and will be saved in `mlruns` folder.

After that, a mlflow experiment is created. To visualize the experiment in mlflow web-based UI, run the following command:

```
mlflow ui
```

The web-based application will go up on `127.0.0.1:5000`.

Appendix B. IntelLight+ Frontend Pages.

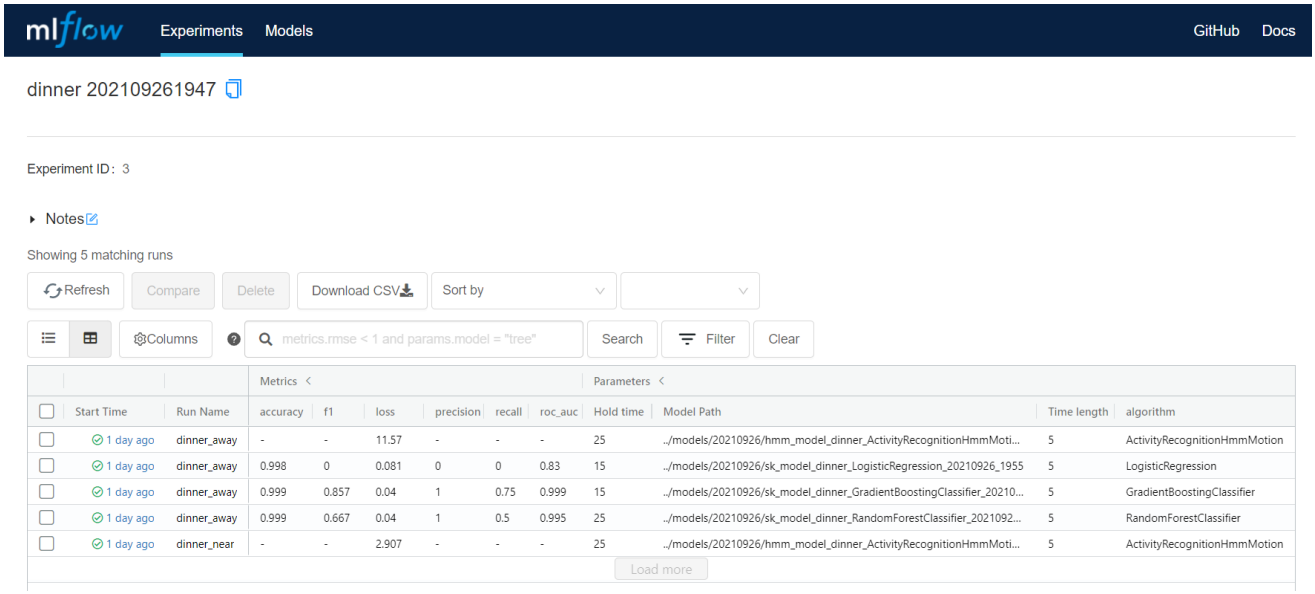


Figure B-1: List of experiments for dinner activity

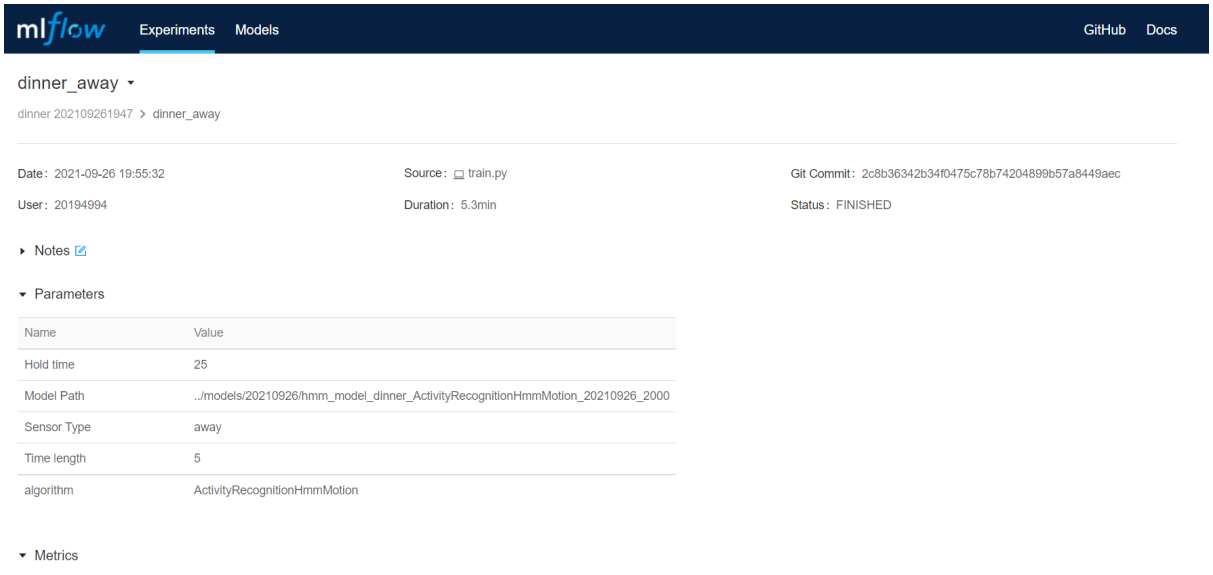


Figure B-2: Detail of parameters, metrics, and code version of each experiment

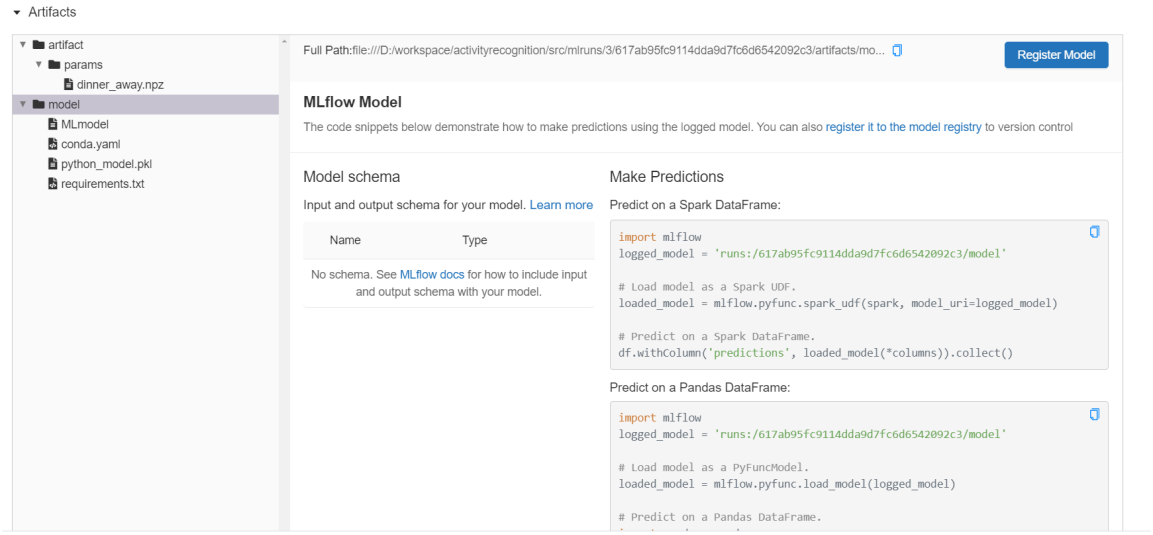


Figure 0-3: Artifacts, model, and environment files saved with each experiment

Appendix C. IntelLight+ Deployment Commands.

To deploy the models from the development environment to the inference environment, we need to do the following initial steps in both environments:

- Clone the project from the bitbucket repository:
  `git clone https://www.code.dtf.lighting.com/scm/intlt/activityrecognition.git`
- Install Python3.7+
- Install and configure `aws cli`. (we are using `aws s3` as shared storage.)

Make a model placeholder and put chosen models Remote Storage

1. Run the training experiments. This will generate new models in the `models/` directory.

2. Use `mlflow ui` command to see the result of experiments and keep the models that you want to deploy in the `models/` directory

3. Track the changes in the `models/` directory in the project, using:

```
dvc add models
```

DVC stores information about the added model files (or a directory) in a special `.dvc` file named `models.dvc`, a small text file with a human-readable format. This file can be easily versioned like source code with Git as a placeholder for the worth to deploy models (which will be added to `.gitignore`).

4. Create a placeholder of models with git tag:

```
git add models.dvc
git commit -m "New models are created"
git tag -a V1.3-ml -m "explain the models change"
git push origin V1.3-ml
```

5. Send the models to shared storage in `aws cli` (the shared storage is configurable and can be any storage place):

```
dvc push models
```

You can check the files that will be pushed before executing `dvc push models` and after running `dvc add models` by the following command:

```
dvc status -c
```

Retrieving and switching between models versions in the target environment

1. Check the available versions of models:

```
git tag -l "*-ml"
```

2. The regular workflow is to use *git checkout* first to switch the `.dvc` file, and then run `dvc fetch` and `dvc checkout` to sync data:

```
git checkout tags/V1.7-ml models.dvc
dvc fetch
dvc checkout
```

3. In case your dvc cache is updated, you can directly download the models in the project's root:

```
dvc pull models.dvc
```

*Raspberry Pi: you need to make a virtual environment and install the packages from `pi_requirements.txt` in order to deploy the models

```
pip3 install -r pi_requirements.txt
```

About the Author

Hossein Mahdian received his Bachelor's degree in Information Technology from Payam Noor University, Iran (2011) and a Master's degree in Information Technology Engineering from Mazandaran University of Science and Technology, Iran (2014). In his master's thesis, Hossein worked on a classification problem and proposed a machine learning algorithm for disease detection. After graduation, Hossein started his professional career by working as a software developer on an enterprise project. Then, he worked as a software engineer in financial software companies in the banking industry and capital market. His interests include software architecture, design, distributed systems, intelligent systems, IoT, and machine learning.

Hossein joined the Technical University of Eindhoven in October 2019 as a Professional Doctorate in Engineering (PDEng) trainee in the Software Technology program offered by the Stan Ackermans Institute. During his traineeship, he participated in the development of several data-intensive and software-intensive products in collaboration with global players like Bosch Security and the European Space Agency. In January 2021, he started working on his ten-month PDEng graduation project to design and develop the machine learning pipeline for intelligent lighting systems for Signify, the leading light solution provider company.

PO Box 513
5600 MB Eindhoven
The Netherlands
tue.nl

PDEng SOFTWARE TECHNOLOGY

EINDHOVEN
UNIVERSITY OF
TECHNOLOGY